Criptoanálisis: factor clave para la ingeniería de seguridad de redes de ordenadores

EDICIONES TÉCNICAS REDE

Criptoanálisis: factor clave para la ingeniería de seguridad de redes de ordenadores

Dr. Javier Areitio Bertolín

El presente artículo analiza la Tecnología de Criptoanálisis aplicada al campo de las Redes de Ordenadores. Se identifican, describen y clasifican los diferentes métodos de ataque más utilizados. También se sintetiza e implementa un módulo software para la ruptura criptoanalítica de PRNGs lineales.

En la última década se ha producido un crecimiento especialmente importante en investigación en todos los aspectos relativos a la Criptología y el Criptoanálisis ha sido una de las áreas más activas. Muchos sistemas criptográficos (o criptosistemas, por ejemplo DES) que se creían seguros se han visto rotos y se ha desarrollado un elevado conjunto de herramientas matemáticas para poder llevar a cabo de la forma más adecuada el criptoanálisis.

El criptoanálisis suele ser un proceso duro, a menudo tedioso, repetitivo y de gran costo económico. El éxito nunca está asegurado y los recursos son siempre limitados. Consecuentemente, también se deben considerar otros enfoques, a veces, más efectivos para obtener información oculta o las claves secretas (esto plantea un compromiso entre el criptoanálisis y la subversión). Cuando la fuerza de un desarrollo criptográfico (por ejemplo, un cifrador) excede por mucho el esfuerzo requerido para obtener la misma información de otra forma, el cifrador probablemente es lo suficiente fuerte para resistir los ataques.

El criptoanálisis implica una combinación de razonamiento analítico, aplicación de herramientas matemáticas/informáticas, encuentro de patrones, paciencia, determinación, suerte, encontrar esquemas de factorización de enteros, aplicar técnicas de inteligencia artificial (redes neuronales, algoritmos genéticos, sistemas expertos, etc.), utilizar algoritmos discretos, aplicar métodos estadísticos sofisticados, ordenación y reordenación de los datos para revelar características o manifestaciones no aleatorias (por ejemplo, contadores de frecuencia, repeticiones, patrones, fenómenos simétricos), etc.

Los ataques a un sistema criptográfico intentan explotar una debilidad del mismo y pueden ser de dos tipos:

– Ataques al propio sistema, aquí se pretende deducir una debilidad en el diseño del sistema.

– Ataques a la implementación del mismo, aquí la debilidad que se pretende obtener es de la implementación particular del mismo; este tipo de ataques suele tener más éxito, un ejemplo son las debilidades encontradas en las primeras implementaciones del protocolo de seguridad SSL del navegador Netscape.

Criptoanálisis y tipos de ataques a los criptosistemas

El Criptoanálisis (es uno de lo dos componentes que conforman la Criptología junto a la Criptografía que se define como la ciencia que trata de escribir mensajes que nadie salvo el receptor deseado lo pueda leer) puede definirse como la ciencia que engloba un conjunto de técnicas tendentes a descifrar comunicaciones cifradas sin conocer las claves correctas, obtener el polinomio de realimentación de un PRNG a partir de la salida conocida, obtener el esquema interno de un determinado mecanismo criptográfico, etc. Existen muchas técnicas, las más importantes para un diseñador/implementador práctico de sistemas son:

Ataques sólo a Texto Cifrado

El atacante no conoce nada acerca de los contenidos del mensaje y debe trabajar a partir sólo del texto cifrado. En la práctica a menudo es posible adivinar algo de texto sin cifrar ya que muchos tipos de mensaje tienen cabeceras de formato fijo. Incluso las cartas y documentos ordinarios empiezan de una forma muy predecible. También puede ser posible adivinar que algun bloque de texto cifrado contiene una palabra común.

Ataque a Texto Sin Cifrar Conocido

El atacante (criptoanalista, adversario o espía) conoce o puede adivinar/inferir el texto sin cifrar para algunas partes del texto cifrado. La tarea consiste en descifrar el resto de los bloques de texto cifrado utilizando esta información. Esto se puede hacer determinando la clave utilizada para cifrar los datos o utilizando algún "atajo".

Ataque a Texto Sin Cifrar Elegido

El atacante puede tener algo de texto que quiere, cifrado con la clave desconocida. La tarea consiste en determinar la clave utilizada para el cifrado. Algunos métodos de cifrado (por ejemplo, el algoritmo de clave pública o asimétrico RSA) son extremadamente vulnerables a los ataques de texto sin cifrar elegido. Cuando tales algoritmos se utilizan debe tenerse extremo cuidado para diseñar todo el sistema de modo que un atacante nunca pueda conseguir texto sin cifrar elegido cifrado.

Ataque "man-in-the-middle"

Atacante mediante intromisión. El adversario se coloca en medio de las partes legítimas que se comunican o "meet-in-the-middle". Este ataque es relevante para protocolos de intercambio de claves y comunicación criptográfica. La idea es que cuando dos partes se intercambian claves para comunicaciones seguras (por ejemplo utilizando Diffie-Hellman) un adversario se coloca entre las partes en la línea de comunicaciones. El adversario entonces realiza un intercambio de clave separado con cada parte. Las partes finalizarán utilizando una clave diferente cada una de las cuales es conocida por el adversario. El adversario entonces descifrará las comunicaciones con la cla-

El Dr. Javier Areitio Bertolín,autor de este artículo, es Director de Redes y Sistemas. ESIDE. Facultad de Ingeniería. Universidad de Deusto

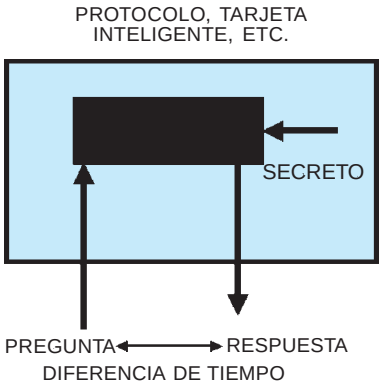

Figura 1. Esquema del Principio sobre el que se Fundamenta el Ataque de Tiempo.

ve adecuada y las cifrará con la otra clave para enviarla a la otra parte. Las partes creerán que se están comunicando de forma segura, pero de hecho el adversario está escuchando y entendiendo todo.

Una forma de prevenir los ataques de este tipo es que ambos extremos calculen una función criptográfica unidireccional hash (por ejemplo, MD5, SHA/SHA1, etc.) del intercambio de clave (o al menos de las claves de cifrado), la firmen utilizando un algoritmo de firma digital y envíen la firma al otro extremo. El receptor entonces verificará que la firma viene de la otra parte deseada y que el «hash» de la firma coincide con el calculado localmente. Este método se utiliza por ejemplo en Photuris.

Ataques de Tiempo

Este ataque es muy reciente y se basa en la medida repetida de los tiempos de ejecución exactos de las operacioes de exponenciación modular (figura 1). Es relevante al menos a los métodos criptográficos RSA, Diffie-Hellman y de Curvas Elipticas.

Las implementaciones de algoritmos criptográficos a menudo realizan cálculos en tiempo no constante, debido a optimizaciones del rendimiento. Si dichas operaciones implican parámetros secretos, estas variaciones de tiempo pueden fugar cierta información y si se proporciona suficiente conocimiento de las implementaciones, un cuidadoso análisis estadístico puede incluso conducir a la recuperación total de estos parámetros secretos. La idea inicial fue presentarda por Kocker en 1996 y demandaba que el atacante dispusiese de un conocimiento muy detallado de la implementación del sistema a atacar. Actualmente otros investigadores han mejorado el método de Kocker y no se requiere un conocimiento tan detallado del sistema.

Criptoanálisis Diferencial, Lineal y Lineal-Diferencial

El criptoanálisis diferencial es un tipo de ataque que puede aplicarse a cifradores de bloque iterativos (como por ejemplo, los algoritmos simétricos o de clave secreta DES, 3DES, IDEA, etc.). Estas técnicas fueron introducidas por Murphy en un ataque sobre FEAL-4, pero fueron mejoradas y perfeccionadas posteriormente por Biham y Shamir que las utilizaron para atacar el DES.

El criptoanálisis diferencial es básicamente un ataque sobre texto sin cifrar escogido y se basa en un análisis de la evolución de las diferencias entre dos textos sin cifrar relacionados cuando se cifran bajo la misma clave. Mediante un cuidadoso análisis de los datos disponibles, las probabilidades pueden asignarse a cada una de las posibles claves y eventualmente la clave más probable se identifica como la correcta.

El criptoanálisis diferencial se ha utilizado contra cifradores muy grandes con grados de éxito variables. En ataques contra el DES su efectividad es limitada, por lo que fue muy cuidadoso el diseño de las cajas S durante el diseño del DES a mediados del año 1970.

Nyberg, Knudsen Lai, Massey y Murphy han realizado estudios sobre protección de cifradores contra criptoanálisis diferencial. El criptoanálisis diferencial ha sido útil en ataques a otros algoritmos criptográficos como por ejemplo las funciones criptográficas unidireccionales «hash».

El criptoanálisis lineal fue ideado por Matsui y Yamagishi para un ataque sobre FEAL. Fue extendido por Matsui para atacar al DES. Es un ataque de texto sin cifrar conocido y utiliza una aproximación lineal para describir el comportamiento del cifrador de bloque. Dados suficientes pares de texto sin cifrar y los correspondientes textos cifrados, se pueden obtener los bits de información acerca de la clave y cantidades crecientes de datos darán normalmente una elevada probabilidad de éxito.

Existen una variedad de mejoras al ataque básico. Langford y Hellman introdujeron un ataque denominado criptoanálisis lineal-diferencial que combina los elementos del criptoanálisis diferencial con los del criptoanálisis lineal. Así mismo, Kaliski y Robshaw demostraron que un ataque criptoanalítico lineal utilizando múltiples aproximaciones, puede permitir una reducción de la cantidad de datos requerido para un ataque con éxito. Nyberg, Knudsen y O'Conner han estudiado la protección de cifradores contra ataques criptoanalíticos lineales.

Ataques de diccionario

Utilizados para romper palabras de paso de los sistemas operativos. En el «ataque de diccionario» no se pretende obtener la clave, si no directamente el texto en claro ya que el método de cifrado es público y aunque no se disponga de la clave se puede reproducir. Este es el caso de los sistemas de cifrado de claves de acceso en sistemas operativos.

Cuando un usuario quiere darse de alta en un sistema, introduce su código y su clave de acceso, ésta se cifra y posteriormente se compara el resultado con la clave cifrada que se almacena en el fichero de claves. Si son iguales, el sistema considera que el usuario es quien dice ser y le permite el acceso.

Los programas rompedores de "palabras de paso" parten del hecho de que se ha obtenido una copia del

fichero de claves (por ejemplo, el denominado /etc/passwd de Unix) y comprueban si existe alguna cuenta sin clave, en cuyo caso utilizan, y con el resto se realiza el siguiente ataque. Por una parte se dispone de un diccionario en claro con gran cantidad de palabras y combinaciones muy comunes y válidas como claves. Posteriormente se realiza su cifrado con la utilidad del sistema a atacar o una copia de la misma, se comprueba el resultado del cifrado con el contenido del fichero de claves y en el caso de que se produzca una coincidencia inferimos cuál es la palabra en claro.

Ataque de búsqueda exhaustiva

El ataque de búsqueda exhaustiva o fuerza bruta para encontrar la clave de un cifrador, es útil cuando el tamaño de la clave a atacar es reducido. Se realiza generando aleatoriamente todos los valores posibles de las claves de acceso y transformándolas. De esta forma se puede elegir la clave de acceso cuya transformada coincida con la interceptada. Para claves de gran longitud este ataque se agiliza con un hardware ASIC de ruptura de elevadísmo costo.

Generadores de números aleatorios criptográficos. Criptoanálisis

Generan números aleatorios utilizados en aplicaciones criptográficas como claves, passwords, etc. Los convencionales disponibles en la mayoría de lenguajes de programación o entornos de programación, no son adecuados para aplicaciones criptográficas, ya que estan diseñados para aleatoriedad estadística y no resisten la predicción por criptoanálisis. En el caso óptimo, los números aleatorios se basan en fuentes verdaderamente físicas de aleatoriedad que no pueden ser predecibles. Dichas fuentes son por ejemplo el ruido de los dispositivos semiconductores, el bit menos significativo de una entrada de audio, los intervalos entre interrupciones de dispositivos o las pulsaciones de teclado de usuario. El ruido obtenido de una fuente física se «destila» con una función hash criptográfica para hacer que cada bit dependa de los otros. A menudo una batería grande (varios miles de bits) se usa para contener aleatoriedad y cada bit de la batería se hace dependiente de cada bit del ruido de entrada y cada uno de los otros bits de la batería de una forma fuerte criptográfica.

Cuando la aleatoriedad física verdadera no está disponible, se deben usar números pseudoaleatorios. Esta situación no es deseable, pero a menudo se utiliza en ordenadores de propósito general. Siempre es deseable obtener algún ruido ambiental (latencias de dispositivos, estadísticas de utilización de recursos y de red, cadencia de pulsación de teclados, etc.). Los datos no deben ser predecibles por ningún observador externo; para conseguir esto, la batería aleatoria debe contener al menos 128 bits de entropía verdadera.

Los generadores pseudoaleatorios criptográficos poseen una gran batería (valor de semilla) que contiene aleatoriedad. Los bits son devueltos desde esta batería tomando datos de la batería, opcionalmente ejecutando los datos a través de una función hash criptográfica para evitar revelación de los contenidos de la batería. Cuando se necesitan más bits, la batería se agita cifrando sus contenidos con un cifrador adecuado con una clave aleatoria (que puede cogerse de una parte de la batería no devuelta) en un modo que hace que cada bit de la batería dependa de cada otro bit de la batería. Se debe mezclar nuevo ruido ambiental a la batería antes de agitar para hacer más imposible la predicción de valores previos o futuros.

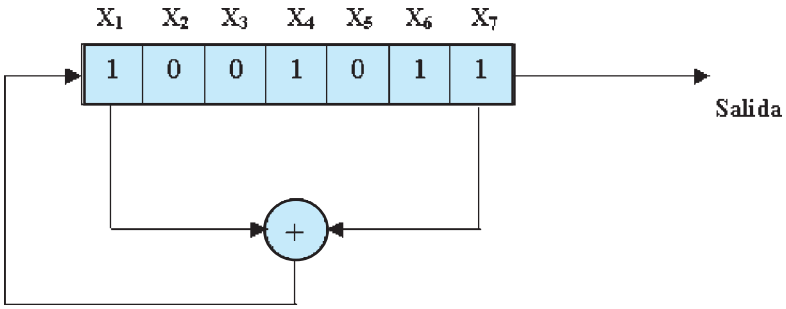

Figura 2. PRNG basado en Registro de Desplazamiento con Realimentación Lineal de longitud 7, de función de realimentación $f(x)=1+x+x^7$. Su estado inicial es 1001011 o expresado en forma polinómica $1+x^3+x^5+x^6$. Su salida de longitud 20 será 11010010110001101111.

Síntesis e implementación de un módulo software de criptoanálisis para la ruptura de un PRNG basado en registro de desplazamiento

La figura 2 muestra un PRNG basado en un registro de desplazamiento con realimentación de longitud 7, posee una función de realimentación lineal $f(x)=1+x+x^7$ y su estado inicial es 1001011 (MSB a la derecha) en binario o expresado en forma polinómica $1+x^3+x^5+x^6$, su salida en binario de longitud 20 es 11010010110001101111.

La figura 3 muestra la especificación en C de un mecanismo software de criptoanálisis que permite obtener la «complejidad lineal» (L) de una secuencia binaria producida por un PRNG basado en registro de desplazamiento con realimentación lineal, así como su polinomio de realimentación. La ayuda del programa se obtiene tecleando «criptoanálisis h».

Supongamos que se conoce una cadena de bits que proviene de un PRNG lineal desconocido. Esa cadena es la entrada del módulo de criptoanálisis. Supongamos que el PRNG

Figura 3. Listado del Módulo Software de Criptoanálisis para un Generador de Números Pseudoaleatorio (PRNG).

```
#define ALL_STEPS 0
#define AMAX 200
#include <stdio.h>
#include "inpvect.h"
#include "printpol.h"
#include "printv~1.h"
#include "ber_mass.h"
int main(int nargs, char * arg[]){
int pol[AMAX], *gen;
int n, complexity;
if (nargs==2)
    {
    if (strcmp(arg[2],"h"))
        {
        printf("Calculo de la complejidad lineal de una secuencia binaria.");
        printf("Entrada: Secuencia binaria, ej. 100000100010011\n");
        printf("Salida: Complejidad Lineal L Polinomio de Realimentacion f(x) \n");
        return 1;
        }
    }
printf("Secuencia?");
n=inputVector(pol);
gen=berlekamp(pol,n,&complexity);
printf("L= %d\nPolinomio de Realimentacion=",complexity);
printPoly(gen, complexity+1);
printf(" \n");
free(gen);
}
```

Programa principal CRIPTOANALISIS.C.

```
#ifndef PRINTPOL_H
#define PRINTPOL_H
void printPoly(int* C,int Csize)
    {
    int i;
    if (C[0])
        printf("%d",C[0]);
    for (i=1;i<Csize;i++)
        {
        if (C[i])
            {
            if(C[i]==1)
                printf("+x");
            else
                printf("+%dx",C[i]);
            if (i!=1)
                printf("^%d",i);
            }
        }
}
#endif
```

PRINTPOL.H

```
#ifndef INPVECT_H
#define INPVECT_H
int inputVector(int* pol){
    char poly[AMAX];
    int i;
    scanf("%s",poly);
    for(i=0;i<AMAX && poly[i]!='\0' ;i++)
        pol[i]=poly[i]-'0';
return i;
}
#endif
```

INPVECT.H

```
#ifndef PRINTVEC_H
#define PRINTVEC_H
void printVector(int* C,int Csize)
    {
    int i;
    for (i=0;i<Csize;i++)
        printf("%d",C[i]);
    printf("\n");
}
#endif
```

PRINTVECT.H

esta basado en un registro de desplazamiento de realimentación lineal (o LFSR) formado por el polinomio primitivo x^7+x+1 y dicho registro se inicializa con el valor (1001011) o en notación polinómica $1+x^3+x^5+x^6$.

Veinte bits de la cadena generada por este PRNG son:

11010010110001101111.

Si sometemos esta cadena de bits al ataque/criptoanálisis de Berlekamp-Massey dado por el módulo "criptoanálisis.exe" se obtine la siguiente salida:

Secuencia?

```
#ifndef BER_MASS_H
#define BER_MASS_H
/* if ALL_STEPS= 1 el programa mostrará el polinomio de conexión actual
calculado por el algoritmo */
#include "printpol.h"
#include <stdlib.h>
/*  Algoritmo de Berlekamp-Massey.
Devuelve el polinimio generador; S:  es la secuencia de entrada; n: es el
número de bits de S; L al final será la Complejidad Lineal de S;
 NOTA: La longitud del polinomio del generador será L+1 */
int* berlekamp(int S[], int n,int *l){
    int *C,* B,* Bp,* T, *gen;
    int  b=1, d;        /* variables del Algoritmo de Berlekamp */
    int xvar, shift;
    int N=0;        /*N: número actal de bits procesados por el algoritmo*/
    int x=1;        /*x: número de bits procesados desde el último cambio de
                    la complejidad lineal*/
    int L=0;            /* Complejidad Lineal */
    int i;
/*reserve memory*/
C =(int*)malloc(sizeof(int)*n);
B =(int*)malloc(sizeof(int)*n);
Bp=(int*)malloc(sizeof(int)*n);
T =(int*)malloc(sizeof(int)*n);
if ((C==NULL) + (B==NULL) + (Bp==NULL) + (T==NULL))
    {
    printf("No hay suficiente memoria\n");
    *l=0;
    free(C); free(B); free(Bp); free(T);
    return  C;
    }
/* Limpiar todo C y arrays B */
    for (i=1; i<n; i++)
        {
        C[i]=0;
        B[i]=0;
        }
    C[0]=B[0]=1; /*Condición Inicial*/
    while (N<n)
        {
        xvar=0;
        for(i=1;i<=L;i++)
            xvar=xvar+C[i]*S[N-i];
        d=(S[N] + xvar) % 2;
        if (d==0)
            {
            x=x+1;      /* No existe cambio de L*/
            }
        if (d && 2*L>N)   /* No existe cambio de L. C se ha modificado */
            {
            if (d*b)
                {
                for(i=n-1;i>=x;i--)
                    Bp[i]=B[i-x];
                for(i=0;i<x;i++)
                    Bp[i]=0;
                for (i=0; i<n; i++)
                    C[i]=C[i]^ Bp[i];
                }
            x=x+1;
            }
        if (d &&  2*L<=N)  /*L se ha
                    modificado de modo
                    que C tambien se ha
                    modificado */
            {
            for(i=0; i<n; i++)
                T[i]=C[i];
            if (d*b)
                {
                for(i=n-1;i>=x;i--)
                    Bp[i]=B[i-x];
                for(i=0;i<x;i++)
                    Bp[i]=0;
                for (i=0; i<n; i++)
                    C[i]=C[i] ^ Bp[i];
                }
            L=N+1-L;
            for(i=0;i<n;i++)
                B[i]=T[i];
            b=d;
            x=1;
            }
        N=N+1;
if (ALL_STEPS){
        printf("\nN= %d",N);
        printf("\tL= %d\t",L);
        printPoly(C,L+1);
        }
        }
        l[0]=L;
gen=(int*)malloc(sizeof(int)*(L+1));
for(i=0;i<=L;i++)
    gen[i]=C[i];
free(C); free(B); free(Bp); free(T);
return gen;
}
#endif
```

BER-MASS.H

11010010110001101111

L = 7

Polinomio de realimentación = $1+x+x^7$.

Esto significa que la "complejidad lineal" de dicha función ($1+x+x^7$) es 7 y el ataque criptoanalítico ha conseguido obtener el polinomio original del que partía la secuencia.

Si se introduce una secuencia de menos de veinte bits, por ejemplo doce:

110100101100

el módulo software diseñado

Grado n	Nº polinomios primitivos
1	1
2	1
3	2
4	2
5	6
6	6
7	18
8	16
9	48
10	60
11	176
12	144
13	630
14	756
15	1800
16	2048
17	7710
18	7776
19	27594
20	24000
21	84672
22	120032
23	356960
24	276480
25	1296000
26	1719900
27	4202496
28	4741632
29	18407808
30	17820000
31	69273666
32	67108864
.....	.....

Cuadro 1.

"criptoanálisis.exe" también podrá adivinar el polinomio de realimentación deseado del PRNG. Si en cambio sólo introducimos once o menos bits el criptoanálisis será incorrecto y la ruptura no tendrá éxito, debido a que poseemos poca información para que el criptoanálisis infiera lo deseado.

En este ejemplo de criptoanálisis se ha utilizado un PRNG basado en un LFSR o registro de desplazamiento realimentado linealmente por un polinomio primitivo basado en el operador suma módulo 2 (o O-exclusiva), es decir perteneciente a GF(2) (Grupo Finito de orden 2). El número de polinomios de realimentación primitivos sobre GF(2) para grado n=1,2,3,..., es el indicado en el cuadro 1, es decir para grado n=1 sólo hay un polinomio el (1+x), para n=2 sólo existe uno, el ($1+x+x^2$); para n=3 existen dos, el ($1+x+x^3$) y el ($1+x^2+x^3$), etc.

En general el número de polinomios primitivos sobre GF(2) para grado k es:

$$[FI(2^k - 1)]/k$$

donde la función FI(m) de Euler representa el número de valores enteros positivos menores que m y relativamente primos con m; p. ejemplo:

$$FI(2^6 - 1) = FI(63) = 36 =$$
$$= \text{cardinalidad}\{2,4,5,8,...,62\}$$

por tanto, el número de polinomios de grado k=6 es 36/6=6.

Una secuencia de salida se dice que es m si es generada por un LFSR de longitud m y si el polinomio de realimentación del LFSR es primitivo; se dice que es primitivo si es irreducible y divisible por $1+x^p$ donde el período p de la secuencia es (2^m-1). Las secuencias m poseen un número casi equilibrado de unos y ceros en un período; la diferencia entre unos y ceros en un período es sólo uno. Aparentemente una secuencia m posee un comportamiento aleatorio. La complejidad lineal de una secuencia es la longitud del LFSR más corto que puede generar dicha secuencia.

El algoritmo Berlekamp-Massey obtiene la complejidad lineal de forma iterativa y también obtiene el polinomio de realimentación asociado al LFSR. Puesto que el grado del polinomio es igual al del registro de desplazamiento LFSR, el grado de este polinomio es la complejidad lineal. El número de bits necesarios para roomper una secuencia generada por un LFSR de longitud n dado, utilizando el algoritmo de Berlekamp-Massey es en el peor de los casos 2n.

El orden de un polinomio de realimentación de un LFSR f(x), para el que f(0) es distinto de cero, es el menor entero "e" para el que f(x) divide a $1+x^e$; un polinomio sobre GF(2) es primitivo si posee orden (2^n-1), por ejemplo $1+x+x^2$ posee orden e = 3 = 2^2-1 ya que $(1+x+x^2)(1+x)=(1+x^3)$.

Otra ruptura con el módulo software diseñado sería para la señal de entrada procedente del PRNG a romper con, por lo menos, 12 bits: 101010100110 y la salida es L=7 y el polinomio es $1+x+x^7$; en este caso el estado de partida del LFSR es 1010101 ($1+x^2+x^4+x^6$).

Clasificación de clases de ataques criptoanalíticos a PRNGs

Ataque de Criptoanálisis Directo

Cuando un atacante puede distinguir directamente entre las salidas del PRNG (o Generador de Números Pseudoaleatorios) y las salidas aleatorias, se dice que es un ataque criptográfico directo. Este tipo de ataque es aplicable a la mayoría, pero no a todos los métodos de PRNGs.

Por ejemplo, un PRNG que únicamente se utilice para generar claves de un cifrador, como puede ser triple-DES, nunca puede ser vulnerable a este tipo de ataque, ya que las salidas del PRNG no se ven de forma directa.

Ataques Basados en la Entrada

Un ataque de entrada ocurre cuando un atacante puede utilizar el conocimiento o control de las entradas del PRNG para criptoanalizar el PRNG, es decir para distinguir entre la salida del PRNG y los valores aleatorios. Los ataques de entrada pueden ser a su vez clasificados en tres categorias:

a) Ataques de Entrada Conocida.- Pueden ser prácticos en cualquier situación en la que alguna de las entradas del PRNG ideadas por el diseñador del sistema difíciles de predecir, resultan ser fácilmente predichas en algunos casos especiales. Un ejemplo de esto es una aplicación que utiliza la latencia del «harddrive» para algunas de las entradas del PRNG, pero tienen que estar ejecutándose por medio de un "drive" de red cuyos «timings» (diagramas de tiempo) son observables por el atacante.

b*) Ataques de Entrada Elegida.-* Pueden ser prácticos contra tarjetas inteligentes y otros "token" (o testigos) resistentes a intentos de forzar bajo un ataque físico/criptoanalítico; pueden también ser prácticos para aplicaciones que suministren mensajes entrantes, palabras de paso seleccionadas por el usuario, estadísticas de red, etc. en su PRNG como muestras de entropía.

- c) Ataques de Entrada Repetida.- Es probable que sean prácticos en las mismas situaciones que los anteriormente mencionados ataques de entrada elegida, pero requieren ligeramente menos control/sofisticación por parte del atacante.

Ataques de Extensión que Comprometen el Estado

Los intentos de ataque de extensión que comprometen el estado ex-

tienden al máximo posible las ventajas de un esfuerzo previamente con éxito que ha recuperado el estado interno «S» (figura 4). Supóngase que por cualquier razón (una penetración temporal a la seguridad del ordenador, una fuga inadvertida, un suceso criptoanalítico, etc.) un adversario conoce el estado interno «S» en algún instante de tiempo «t». Un ataque de extensión que compromete al estado sucede cuando el atacante puede recuperar las salidas desconocidas del PRNG (o distinguir aquellas salidas del PRNG de los valores aleatorios) desde antes de que "S" se haya visto comprometido, o recuperar las salidas desde después de que el PRNG haya recogido una secuencia de entrada que el atacante no puede adivinar.

Los ataques de extensión que comprometen al estado son más probables que operen cuando un PRNG se arranca en un estado inseguro (adivinable) debido a la insuficiencia entropía de arranque. También pueden operar cuando "S" se ha visto comprometido por cualquiera de los ataques señalados o por cualquier otro método.

En la práctica es prudente asumir que puede suceder que se vea ocasionalmente comprometido el estado interno "S"; para preservar la robusted del sistema, los PRNGs deberían ser resistentes al máximo a los ataques de extensión que comprometen el estado.

Esta categoría de ataques puede a su vez clasificarse en las siguientes categorias:

a) Ataques de Vuelta Atrás (o «backtracking»).- Utilizan la situación de verse comprometido el estado interno «S» del PRNG en el instante de tiempo «t» para aprender las salidas del PRNG previas.

b) Ataques de verse comprometido en el estado de forma permanente.- Ocurren si una vez que un atacante compromete el estado interno «S» del PRNG en el instante de tiempo «t», todos los valores futuros y pasados de «S» son vulnerables al ataque.

c) Ataques de Adivinación Iterativa.- Utilizan el conocimiento del estado interno «S» en el instante de tiempo «t» y las salidas del PRNG intermedias para aprender el estado interno «S» en el instante de tiempo (t + e) cuando las entradas recogidas durante este intervalo de tiempo son adivinables (pero no conocidas) por el atacante.

d) Ataques «man/meet-in-the-middle».- Son en esencia una combinación de un ataque de adivinación iterativo con un ataque de vuelta atrás. El conocimiento del estado interno «S» en los instantes de tiempo «t» y «t + 2e» permite al atacante recuperar el estado interno "S" en el instante de tiempo (t + e).

Cifradores secretos. Requisitos de Kerckhoff

Aunque en algunos casos el criptoanálisis puede realizarse incluso aunque el proceso de cifrado no se conozca, parece que ésto puede en principio dificultar el trabajo del adversario. Por tanto, se puede argumentar que el proceso de cifrado debería permanecer en secreto, de hecho los sistemas de cifrado militares no se publican. En criptografía comercial se asume normalmente (los requisitos de Kerckhoff) que el adversario/oponente conoce los detalles del cifrador (aunque no la clave). Existen varias razones para esto:

- Es común para todo cifrador la existencia de ciertas debilidades (o «bugs») no esperadas que no detectan sus diseñadores. Pero si el diseño del cifrador se mantiene en secreto, no podrá ser examinado por otras partes competentes y, por tanto, la debilidad no se expondrá públicamente. Esto viene a significar que la debilidad puede explotarse en la práctica mientras el cifrador está utilizándose.
- Si un cifrador es secreto, ese secreto se ve comprometido de forma creciente por el hecho de hacerse disponible para su uso. Para que un cifrador sea utilizado debe estar presente en varios lugares y a más amplia utilización mayor es el riesgo de que el secreto se vea expuesto. Por tanto, cualquier ventaja que puede proporcionar un cifrador secreto no puede mantenerse y el oponente eventualmente tendrá las mismas ventajas que tendría al revelarse públicamente.

Los diseñadores de mecanismos criptográficos (cifradores simétricos/asimétricos, PRNGs, funciones hash, firmas digitales, etc.) se pueden autoengañar de forma peligrosa, suponiendo que su oponente no posee información que de hecho puede poseer. Los requisitos de Kerckoff formulados en 1883 relativos a los criptosistemas generales son:

– *El sistema debería ser irrompible*, si no teóricamente, al menos en la práctica. No existen sistemas reales que sean «irrompibles», y tiene poco sentido utilizar cifradores rompibles conocidos. El «One Time Pad» (OTP) que utiliza una clave tan larga como el mensaje a cifrar y absolutamente aleatoria, es el único método impracticable que cumple la condición de «secreto perfecto». En la práctica se buscan sistemas que sean «computacionalmente seguros», es decir el costo de obtener el mensaje original sin conocer la clave es extremadamente alto en tiempo y/o en espacio de computación.

– *Comprometer los detalles del sistema* criptográfico no debería preocupar. Hoy en día se asume que el oponente tiene suficientes detalles del cifrador, ya que para que el cifrador se pueda utilizar ampliamente debe estar presente en muchos lugares y por tanto es probable que pueda quedar expuesto. También se asume que el oponente tendrá cierta cantidad de texto sin cifrar (o «pla-

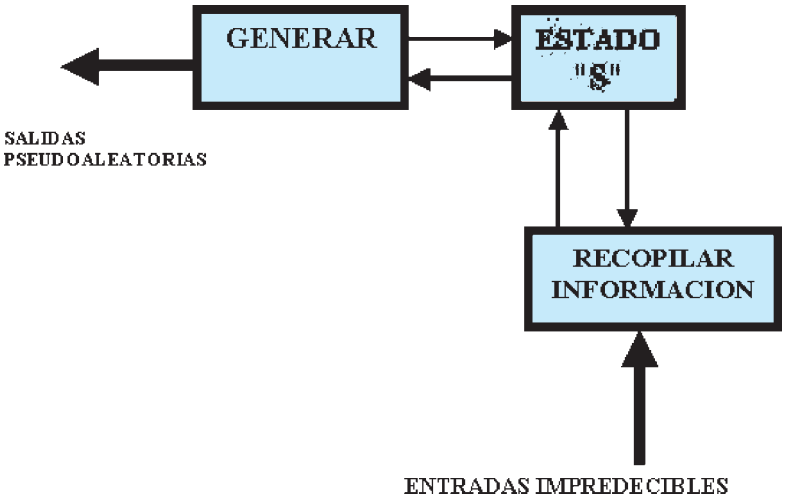

Figura 4. Representación de Bloques Simplificada de las Operaciones Internas de la mayoria de PRGNs.

intext») conocido para poder trabajar con él.

– *La clave debería ser recordable* sin necesidad de anotaciones y fácilmente cambiable. Esto aún en nuestros días es un problema. El «hash» nos permite recurrir al empleo de frases de lenguaje largas; por ello, el mejor enfoque consiste en utilizar simultáneamente una tarjeta hardware para la clave y una frase de paso para la clave.

– *El criptograma debería ser transmisible* por telégrafo. Esto no es muy importante hoy en día ya que incluso el texto cifrado binario puede convertirse a ASCII para la transmisión si es necesario.

- *El cifrador debería ser portable*, operable y fácil de utilizar por cualquier persona. ❑

Bibliografia

Areitio, J. «Análisis y Evaluación de la Seguridad en Redes». Revista Española de Electrónica. Nº 521. Abril 1998.

Gleason, AM. «Elementary Course In Probability for The Cryptananyst». Aegean Park Press. 1998.

Foster, C.C. «Cryptanalysis for Microcomputers». Hayden Books. Rochelle Park. 1990.