CRYPTOBENCH: UN BANCO DE PRUEBAS PARA COMPONENTES CRIPTOGRÁFICOS

Luis Cornide Arce y Gonzalo Álvarez Marañón

Instituto de Física Aplicada
Consejo Superior de Investigaciones Científicas
{luisc, gonzalo}@iec.csic.es

RESUMEN

En este artículo se describe el funcionamiento de CryptoBench cuya finalidad es proporcionar un entorno integrado, completo y flexible para la implementación y evaluación de componentes criptográficos. Para ello dispone de una importante batería de tests y de un interfaz de programación que permite el rápido desarrollo de componentes para su posterior inclusión en la aplicación.

1. INTRODUCCIÓN

La finalidad de CryptoBench es proporcionar un entorno integrado, completo y flexible para la implementación y evaluación de componentes criptográficos a través de una importante batería de test. Si bien ya existen otras herramientas que realizan estas pruebas, como CRYPT-X'98 [1], la principal aportación de este trabajo consiste en definir una interfaz de programación para el desarrollo rápido de cifradores y generadores pseudoaleatorios que más tarde se incluirán en la aplicación. De esta forma, se dispone de un entorno en el que el desarrollo y las pruebas de evaluación de los componentes se encuentran completamente integrados.
En este trabajo se describe el funcionamiento de CryptoBench. En la sección 2 se habla brevemente de las características de la aplicación. En la sección 3 se comentarán los distintos test implementados y, por último, se describirá el interfaz de programación.

2. LA APLICACIÓN

CryptoBench pretende ser un entorno de desarrollo y evaluación de componentes criptográficos, tales como cifradores de bloque y generadores de números pseudoaleatorios. Dispone de las características básicas de un editor de texto, con el que crear o editar los ficheros sobre los que se desee realizar las pruebas. Tras elegir el cifrador de entre todos los que disponga la aplicación[1], se deben seleccionar las opciones de cifrado. La aplicación permite utilizar los cuatro modos de operación definidos para los cifradores de bloque: ECB, CBC, CFB y OFB. También soporta la utilización de cifradores que permitan distintas longitudes de bloque o de clave. Con la idea de proporcionar un entorno integrado, CryptoBench dispone de herramientas para crear claves y vectores de inicialización aleatorios usando algunos de los generadores de la aplicación. La aplicación calcula los tiempos y velocidades de cifrado y descifrado y muestra la salida de cada proceso. Una vez terminado el cifrado y descifrado, se pueden realizar sobre la secuencia cifrada los distintos test, cuyos resultados se mostrarán a través de distintas gráficas.
Una de las características sobresalientes de CryptoBench reside en su extensibilidad, que con la ayuda de un asistente permite al usuario crear sus propios algoritmos de cifrado y de generación de números aleatorios, los cuales pueden añadirse a la aplicación en forma de bibliotecas de enlace dinámico (dll) para su posterior evaluación.

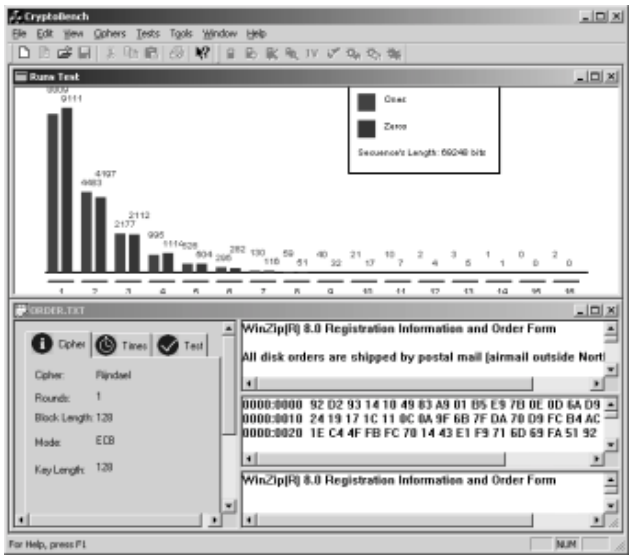

Figura 1. *Interfaz de la aplicación.*

3. TESTS

Los test que se aplicarán a las secuencias cifradas evalúan su grado de aleatoriedad, valiéndose de distintas técnicas: tests estadísticos basados en el contraste de hipótesis, complejidad lineal, rachas, autocorrelación, etc.

[1] En la primera versión de distribución están implementados el Rijndael, el IDEA y el RC5.

3.1 Independencia

Este test comprueba si los distintos bloques de una secuencia se distribuyen independientemente los unos de los otros. La secuencia resultante del cifrado se divide en bloques de 16 ó 32 bits y cada uno en dos subbloques. Este test toma como hipótesis nula, H_0, que la función de distribución conjunta es igual al producto de las funciones de distribución de cada una de las partes[2].

3.2 Autocorrelación

Este test comprueba las diferencias que existen entre una secuencia y versiones desplazadas de la misma. Se calcula la autocorrelación de la secuencia con la Ec. 1 para cada desplazamiento *k*.

$$AC(k) = \frac{2 \cdot A(k) - n}{n} \quad (1)$$

Donde $A(k)$ cuenta los bits diferentes entre una secuencia y ella misma desplazada *k* posiciones.

$$A(k) = \sum_{i=0}^{n-k-1} s_i \oplus s_{i+k} \quad (2)$$

Los resultados obtenidos se representan en una gráfica con los valores de la autocorrelación para cada desplazamiento desde 0 a *n*. Secuencias pseudolaeatorias buenas presentan una gráfica muy regular en la que no existen picos (valor de autocorrelación alto) salvo para $k = 0$ y $k = n$.

3.3 Correlación cruzada

Este test realiza la misma comprobación que el anterior, con la diferencia de que éste compara dos secuencias distintas y de igual longitud. Típicamente, se usa para comprobar las diferencias entre la secuencia formada por un mensaje en claro y versiones desplazadas de la que forma el mensaje cifrado.

3.4 Ortogonalidad

Este sencillo test cuenta los bits diferentes entre dos secuencias.

3.5 Homogeneidad

Este test comprueba que una serie de muestras pertenecen a la misma población. Para ello, se divide la secuencia en 32 bloques y éstos a su vez en 16 subbloques y se comprueba si la secuencia se comporta uniformemente o existen partes que no mantienen sus características[2].

3.6 Complejidad lineal

Este test calcula la complejidad lineal de la secuencia cifrada. La complejidad lineal de una secuencia *S* se define como la longitud del menor LFSR (*Linear Feedback Shift Register*) capaz de generar *S*. Para calcularla se utiliza el algoritmo de Berlekamp-Massey. Idealmente, una secuencia cifrada debe tener una complejidad lineal igual a la longitud de la misma.

3.7 Rachas

Una racha de longitud *k* es una secuencia de *k* dígitos iguales seguidos delimitada por dos dígitos distintos. En este test se comprueban dos de los tres postulados que según Golomb [3] debe cumplir una secuencia binaria para poder ser considerada pseudoaleatoria: i) el número de unos y de ceros de la secuencia debe ser aproximadamente igual, y ii) del total de rachas observado, la mitad de ellas debe tener longitud 1, la cuarta parte tendrá longitud 2, la octava parte tendrá longitud 3 y así sucesivamente. CryptoBench presenta en un gráfico de barras las relaciones entre las rachas de distintas longitudes (v. Fig. 1).

4. INTERFAZ DE PROGRAMACIÓN

El API de CryptoBench permite a los usuarios con ciertos conocimientos de programación en C++ implementar sus propios componentes (cifradores de bloque o generadores de números pseudoaleatorios) e integrarlos en la aplicación. Entre las herramientas de la aplicación se incluyen dos asistentes encargados de generar gran parte del código necesario para implementar nuevos componentes. El objetivo de esta aplicación reside en ofrecer un marco de desarrollo sencillo, orientado a objetos, que permita a programadores sin experiencia desarrollar rápidamente implementaciones de cifradores de bloque y generadores de números pseudoaleatorios, sin que para ello sea necesario tener unos grandes conocimientos acerca de programación orientada a objetos.

A la hora de diseñar este API se han tenido muy en cuenta las directrices del NIST para el reciente concurso del AES (*Advanced Encryption Standard*). En consecuencia, está dotado de la mayor flexibilidad posible, pudiéndose especificar libremente distintos parámetros de operación, como tamaños de bloque y de clave o número de vueltas.

5. CONCLUSIONES

CryptoBench proporciona un entorno integrado para la fácil creación y evaluación de componentes criptográficos. Por este motivo, se trata de una aplicación muy adecuada en ámbitos académicos, tanto para el docente como para el criptoanalista.

En la actualidad, se está trabajando en su extensión para incluir en futuras versiones otros componentes criptográficos, como cifradores de flujo, generadores de números primos, tests de primalidad, etc.

REFERENCIAS

[1] Queensland University of Technology. CRYPT-X'98. http://www.isrc.qut.edu.au/cryptx/. 1998

[2] Lillo Rodríguez, R. E., Fúster Sabater A. Visión Probabilística de las Secuencias Binarias de Aplicación Criptográfica. 2ª Reunión de Criptografía Española. 1992.

[3] Golomb, S. W., Shift Register Secuences, Holden-Day, San Francisco, 1967.