Ingeniería de Software y Sistemas Computacionales

Seguridad en la Información

Prof. Héctor Puga Zavala

Algoritmos de Encriptación

José de Jesús Reyes Delgado

Grupo 511

ÍNDICE

ÍNDICE....1
INTRODUCCIÓN....3
CIFRADO SIMÉTRICO....3
CIFRADO ASIMÉTRICO....3
ALGORITMOS....4
AVANCED ENCRYPTION STANDAR (AES)....4
CÓDIGO O PSEUDOCÓDIGO....5
VENTAJAS....5
DESVENTAJAS....6
ATÁQUES AL ALGORITMO....6
DATA ENCRYPTION STANDAR (DES)....7
CÓDIGO O PSEUDOCÓDIGO....7
VENTAJAS....39
DESVENTAJAS....39
ATÁQUES AL ALGORITMO....39
BLOWFISH....43
CÓDIGO O PSEUDOCÓDIGO....43
VENTAJAS....112
DESVENTAJAS....113
ATÁQUES AL ALGORITMO....113
RSA....113
CÓDIGO O PSEUDOCÓDIGO....114
VENTAJAS....116
DESVENTAJAS....116
ATÁQUES AL ALGORITMO....116
CONCLUSIÓN....117

BIBLIOGRAFÍA.....118

MÉDIOS IMPRESOS.....118

MÉDIOS ELECTRÓNICOS.....118

MÉDIOS ELECTRÓNICOS

Todos los s.o. que hay ahí fuera son más o menos iguales. Todos somos una porquería". Brian Valentine, vicepresidente de Microsoft describiendo el edo., del arte en seguridad de s.o. en 2003

INTRODUCCIÓN

Nos toca hablar sobre los diferentes, por no decir que solo hablaremos de algunos o de los más comerciales dado que hablar de todos ello sería imposible sin mencionar que algunas empresas han desarrollado su propio algoritmo de encriptación, algoritmos de encriptación, como recordaran existen dos tipos de encriptación o cifrado más propiamente dicho: simétrico y asimétrico, pero para que lo recordemos un poco mejor, volvamos a definir.

CIFRADO SIMÉTRICO

A este método es conocido también por el uso de una clave privada, es decir el cifrador pone una clave para cifrar la información que se requiere, el destinatario final deberá de tener la misma clave para poder descifrar el contenido del cifrado, este tipo de método es utilizado en condiciones que no exijan requisitos de seguridad muy estrictos, debido a que en él envió de la clave para su descifrado disminuye su robustez, por otra parte, una ventaja de este mecanismo es que al ser más simple su ejecución será mucho más rápida.

CIFRADO ASIMÉTRICO

Un cifrado asimétrico o de clave pública, es aquel cifrado que se basa en el uso de una pareja de clave, es decir, una privada y una pública, de las cuales una se usa para cifrar y otra para descifrar, ambas claves están relacionadas por una función “trampa”, que suele ser una función matemática, lo que ya comentábamos anteriormente, aplicando también la inversa de dicha fórmula, lo que la convierte en una función “trampa”, al ser difícil o imposible de calcular. Una de sus desventajas es que el mensaje cifrado ocupara mayor espacio que el original.

ALGORITMOS

Bien una vez, que ya hemos recordado los dos tipos de cifrado que existen, empecemos a describir algunos de los algoritmos que podemos encontrar, incluyo el código, de igual manera solo de algunos, así mismo trataremos sus ventajas y desventajas.

AVANCED ENCRYPTION STANDAR (AES)

Conocido también como Rijndael (pronunciado "Rain Doll" en inglés), es un esquema de cifrado por bloques adoptado como un estándar de cifrado por el gobierno de los Estados Unidos. El AES fue anunciado por el Instituto Nacional de Estándares y Tecnología (NIST) como FIPS PUB 197 de los Estados Unidos (FIPS 197) el 26 de noviembre de 2001 después de un proceso de estandarización que duró 5 años. Se transformó en un estándar efectivo el 26 de mayo de 2002.

El cifrador fue desarrollado por dos criptólogos belgas, Joan Daemen y Vincent Rijmen, ambos estudiantes de la Katholieke Universiteit Leuven, y enviado al proceso de selección AES bajo el nombre "Rijndael".

Estrictamente hablando, AES no es precisamente Rijndael (aunque en la práctica se los llama de manera indistinta) ya que Rijndael permite un mayor rango de tamaño de bloques y longitud de claves; AES tiene un tamaño de bloque fijo de 128 bits y tamaños de llave de 128, 192 ó 256 bits, mientras que Rijndael puede ser especificado por una clave que sea múltiplo de 32 bits, con un mínimo de 128 bits y un máximo de 256 bits. La mayoría de los cálculos del algoritmo AES se hacen en un campo finito determinado.

AES opera en una matriz de 4×4 bytes, llamada state (algunas versiones de Rijndael con un tamaño de bloque mayor tienen columnas adicionales en el state). Desde 2006, el AES es uno de los algoritmos más populares usados en criptografía simétrica.

CÓDIGO O PSEUDOCÓDIGO

Pseudocódigo AES

Todos los s.o. que hay ahí fuera son más o menos iguales. Todos somos una porquería". Brian Valentine, vicepresidente de Microsoft describiendo el edo., del arte en seguridad de s.o. en 2003

- Expansión de la clave usando el esquema de claves de Rijndael.
- Etapa inicial:
  - AddRoundKey
- Rondas:
  - SubBytes — en este paso se realiza una sustitución no lineal donde cada byte es reemplazado con otro de acuerdo a una tabla de búsqueda.
  - ShiftRows — en este paso se realiza una transposición donde cada fila del «state» es rotada de manera cíclica un número determinado de veces.
  - MixColumns — operación de mezclado que opera en las columnas del «state», combinando los cuatro bytes en cada columna usando una transformación lineal.
  - AddRoundKey — cada byte del «state» es combinado con la clave «round»; cada clave «round» se deriva de la clave de cifrado usando una iteración de la clave.
- Etapa final:
  - SubBytes
  - ShiftRows
  - AddRoundKey

VENTAJAS

- Bloque de entrada más grande: 128 bits
- Más operaciones son paralelizables a diferentes niveles

- Las operaciones se manejan a nivel de bits lo que lo hace altamente eficiente en la mayoría de las plataformas especialmente en software
- Alta flexibilidad en el nivel de seguridad: 128/192/256 bits

DESVENTAJAS

- En algunos casos la encriptación y desencriptación no son idénticas
- Esta descrito por una fórmula algebraica

ATÁQUES AL ALGORITMO

Conseguir un método viable para descifrar documentos cifrados con AES puede marcar un punto de inflexión en el uso de este estándar, dado que muchos documentos críticos y sistemas de protección usan este sistema de cifrado simétrico. Sin hablar de los que lo usan para evitar la piratería. Aquí pongo a su disposición algunos libros sobre ataques al algoritmo.

- "Cache-timing attacks on AES" - D. J. Bernstein
- "A Collision-Attack on AES Combining Side Channel- and Differential-Attack" - K. Schramm, G. Leander, P. Felke, C. Paar
- "Multiple-Differential Side-Channel Collision Attacks on AES" - A. Bogdanov
- "Related-Key Impossible Differential Attacks on 8-Round AES-192" - E. Biham, O. Dunkelman, N. Keller
- "Successfully Attacking Masked AES Hardware Implementations" - S. Mangard, N. Pramstaller, E. Oswald

Todos los s.o. que hay ahí fuera son más o menos iguales. Todos somos una porquería". Brian Valentine, vicepresidente de Microsoft describiendo el edo., del arte en seguridad de s.o. en 2003

DATA ENCRYPTION STANDAR (DES)

En 1972, tras terminar un estudio sobre las necesidades del gobierno de E.U.A. en materia de seguridad informática, la autoridad de estándares estadounidense NBS (National Bureau of Standards) — ahora rebautizado NIST (National Institute of Standards and Technology) — concluyó en la necesidad de un estándar a nivel gubernamental para cifrar información confidencial. En consecuencia, el 15 de mayo de 1973, tras consultar con la NSA, el NBS solicitó propuestas para un algoritmo que cumpliera rigurosos criterios de diseño. A pesar de todo, ninguna de ellas parecía ser adecuada. Una segunda petición fue realizada el 27 de agosto de 1974. En aquella ocasión, IBM presentó un candidato que fue considerado aceptable, un algoritmo desarrollado durante el periodo 1973–1974 basado en otro anterior, el algoritmo Lucifer de Horst Feistel. El equipo de IBM dedicado al diseño y análisis del algoritmo estaba formado por Feistel, Walter Tuchman, Don Coppersmith, Alan Conheim, Carl Meyer, Mike Matyas, Roy Adler, Edna Grossman, Bill Notz, Lynn Smith, y Bryant Tuckerman.

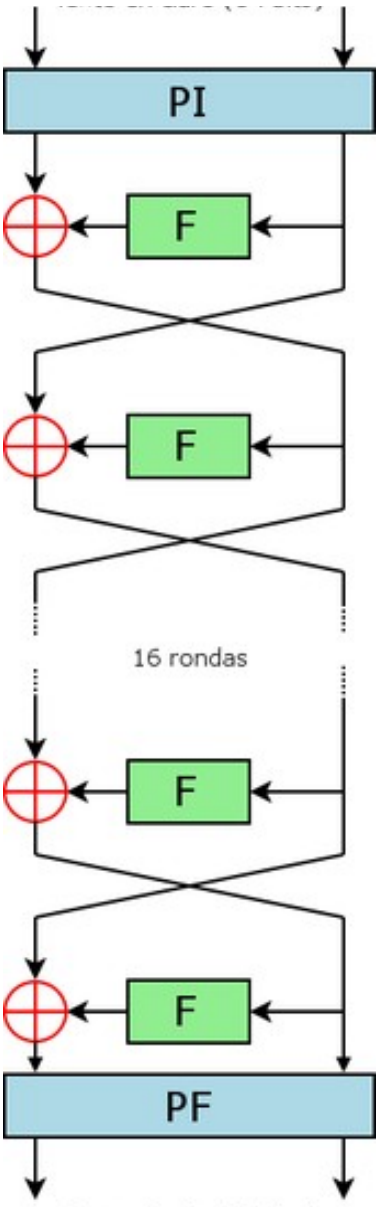
DES fue aprobado como estándar federal en noviembre de 1976, y publicado el 15 de enero de 1977 como FIPS PUB 46, autorizado para el uso no clasificado de datos. Fue posteriormente confirmado como estándar en 1983, 1988 (revisado como FIPS-46-1), 1993 (FIPS-46-2), y de nuevo en 1998 (FIPS-46-3), éste último definiendo "TripleDES". El 26 de mayo de 2002, DES fue finalmente reemplazado por AES (Advanced Encryption Standard), tras una competición pública. Hasta hoy día (2006), DES continúa siendo ampliamente utilizado. DES es también parte la familia de cifrado simétrico.

CÓDIGO O PSEUDOCÓDIGO

DES es el algoritmo prototipo del cifrado por bloques — un algoritmo que toma un texto en claro de una longitud fija de bits y lo transforma mediante una serie de operaciones básicas en otro texto cifrado de la misma longitud. En el caso de DES el tamaño del bloque es de 64 bits. DES utiliza también una clave criptográfica para modificar la transformación, de modo que el descifrado sólo puede ser realizado por aquellos que conozcan la clave concreta utilizada en el cifrado. La clave mide 64 bits, aunque en realidad, sólo 56 de ellos son empleados por el algoritmo. Los ocho bits restantes se utilizan únicamente para comprobar la paridad, y después son descartados. Por tanto, la longitud de clave efectiva en DES es de 56 bits, y así es como se suele especificar.

Al igual que otros cifrados de bloque, DES debe ser utilizado en el modo de operación de cifrado de bloque si se aplica a un mensaje mayor de 64 bits. FIPS-81 específica varios modos para el uso con DES, incluyendo uno para autenticación. Se pueden consultar más documentos sobre el uso de DES en FIPS-74.

La estructura básica del algoritmo aparece representada en la Figura: hay 16 fases idénticas de proceso, denominadas rondas. También hay una permutación inicial y final, denominadas PI y PF, que son funciones inversas entre sí (PI "deshace" la acción de PF, y viceversa). PI y PF no son criptográficamente significativas, pero se incluyeron presuntamente para facilitar la carga y descarga de bloques sobre el hardware de mediados de los 70. Antes de las rondas, el bloque es dividido en dos mitades de 32 bits y procesadas alternativamente. Este entrecruzamiento se conoce como esquema Feistel.



La estructura de Feistel asegura que el cifrado y el descifrado sean procesos muy similares — la única diferencia es que las subclaves se aplican en orden inverso cuando desciframos. El resto del algoritmo es idéntico. Esto simplifica enormemente la implementación, en especial sobre hardware, al no haber necesidad de algoritmos distintos para el cifrado y el descifrado.

El símbolo rojo "⊕" representa la operación OR exclusivo (XOR). La función-F mezcla la mitad del bloque con parte de la clave. La salida de la función-F se combina entonces con la otra mitad del bloque, y los bloques son intercambiados antes de la siguiente ronda. Tras la última ronda, las mitades no se intercambian; ésta es una característica de la estructura de Feistel que hace que el cifrado y el descifrado sean procesos parecidos.

Implementación en C++:

```
#include <iostream.h>
#include <string.h>
#include <stdlib.h>

#include "DES.h"
```

//

Todos los s.o. que hay ahí fuera son más o menos iguales. Todos somos una porquería". Brian Valentine, vicepresidente de Microsoft describiendo el edo., del arte en seguridad de s.o. en 2003

```
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

CDES::CDES(char* data, char* key)
{
	strcpy(this->data,data);
	this->data[strlen(this->data)] = '\0';
	strcpy(this->Compkey,key);
	this->Compkey[strlen(this->Compkey)] = '\0';

	cout << endl << "DATA:";
	print(this->data,DATA_LENGTH);
	cout << endl << "KEY:";
	print(this->Compkey,KEY_TOTAL_LENGTH);

	FillAllS();
//	SampleKeys();
}

CDES::~CDES()
{

}

char* CDES::Encrypt()
{
	// Calculate Keys
	GenKeys();

	// Begin Encryption
#ifdef SDES
	char *ip = DataIP(); // Initial Permutation on Input bits
#else
	char *ip = DataIP_DES();
#endif

	cout << endl << "Initial Permutation";
	print(ip,DATA_LENGTH);

	char *loopData = ip;
	char *FKr;

	for (int i = 1 ; i <= NO_OF_KEYS ; i++)
	{
		FKr = FKFunction(loopData,i);

		cout << endl << "FK" << i << " Result";
		print(FKr,DATA_LENGTH);

		if (i != NO_OF_KEYS)
		{
			loopData = Switch(FKr);
```

```
                cout << endl << "Switch Result";
                print(loopData,DATA_LENGTH);
            }
            else
            {
                loopData = FKr;
            }
        }

        char *IPinv = IPInverse(loopData);

        cout << endl << "IP Inverse";
        print(IPinv,DATA_LENGTH);

        return IPinv;
}

char* CDES::keyIPermutation()
{
#ifdef SDES
        char *keyP = new char[KEY_TOTAL_LENGTH];

        keyP[0] = Compkey[3-1];
        keyP[1] = Compkey[5-1];
        keyP[2] = Compkey[2-1];
        keyP[3] = Compkey[7-1];
        keyP[4] = Compkey[4-1];
        keyP[5] = Compkey[10-1];
        keyP[6] = Compkey[1-1];
        keyP[7] = Compkey[9-1];
        keyP[8] = Compkey[8-1];
        keyP[9] = Compkey[6-1];
#else
        char *keyP = new char[KEY_USED_BITS];

        keyP[0] = Compkey[57-1];
        keyP[1] = Compkey[49-1];
        keyP[2] = Compkey[41-1];
        keyP[3] = Compkey[33-1];
        keyP[4] = Compkey[25-1];
        keyP[5] = Compkey[17-1];
        keyP[6] = Compkey[9-1];
        keyP[7] = Compkey[1-1];
        keyP[8] = Compkey[58-1];
        keyP[9] = Compkey[50-1];
        keyP[10] = Compkey[42-1];
        keyP[11] = Compkey[34-1];
        keyP[12] = Compkey[26-1];
        keyP[13] = Compkey[18-1];
        keyP[14] = Compkey[10-1];
```

```
	keyP[15] = Compkey[2-1];
	keyP[16] = Compkey[59-1];
	keyP[17] = Compkey[51-1];
	keyP[18] = Compkey[43-1];
	keyP[19] = Compkey[35-1];
	keyP[20] = Compkey[27-1];
	keyP[21] = Compkey[19-1];
	keyP[22] = Compkey[11-1];
	keyP[23] = Compkey[3-1];
	keyP[24] = Compkey[60-1];
	keyP[25] = Compkey[52-1];
	keyP[26] = Compkey[44-1];
	keyP[27] = Compkey[63-1];
	keyP[28] = Compkey[63-1];
	keyP[29] = Compkey[55-1];
	keyP[30] = Compkey[47-1];
	keyP[31] = Compkey[39-1];
	keyP[32] = Compkey[31-1];
	keyP[33] = Compkey[23-1];
	keyP[34] = Compkey[15-1];
	keyP[35] = Compkey[7-1];
	keyP[36] = Compkey[62-1];
	keyP[37] = Compkey[54-1];
	keyP[38] = Compkey[46-1];
	keyP[39] = Compkey[38-1];
	keyP[40] = Compkey[30-1];
	keyP[41] = Compkey[22-1];
	keyP[42] = Compkey[14-1];
	keyP[43] = Compkey[6-1];
	keyP[44] = Compkey[61-1];
	keyP[45] = Compkey[53-1];
	keyP[46] = Compkey[45-1];
	keyP[47] = Compkey[37-1];
	keyP[48] = Compkey[29-1];
	keyP[49] = Compkey[21-1];
	keyP[50] = Compkey[13-1];
	keyP[51] = Compkey[5-1];
	keyP[52] = Compkey[28-1];
	keyP[53] = Compkey[20-1];
	keyP[54] = Compkey[12-1];
	keyP[55] = Compkey[4-1];
#endif

	return keyP;
}

char* CDES::LeftShift(char *keyPF, char LorR, int bits)
{
	char *sbits = new char[KEY_USED_BITS/2];
	int i,k;

	if (LorR == 'L')	// shift left bits
```

```
	{
		for (i = 0,k = 0 ; i < (KEY_USED_BITS/2) ; i++,k++)
		{
			sbits[i] = keyPF[k];
		}
	}
	else if (LorR == 'R')   // shift right bits
	{
		for (i = 0,k = (KEY_USED_BITS/2) ; i < (KEY_USED_BITS/2) ; i+
+,k++)
		{
			sbits[i] = keyPF[k];
		}
	}

	char temb;

	// Shift the Bits
	for (i = 1 ; i <= bits ; i++)
	{
		temb = sbits[(KEY_USED_BITS/2)-1];
		for (int k = 0 ; k < (KEY_USED_BITS/2) ; k++)
		{
			if ((KEY_USED_BITS/2)-1 == k)
				sbits[(k+((KEY_USED_BITS/2)-1))                    %
(KEY_USED_BITS/2)] = temb;
			else
				sbits[(k+((KEY_USED_BITS/2)-1))                    %
(KEY_USED_BITS/2)] = sbits[k];
		}
	}

	return sbits;
}

char* CDES::JoinShifted(char *Lbits, char *Rbits)
{
	char *key1 = new char[KEY_USED_BITS];

	int i,j;

	for (i = 0 ; i < (KEY_USED_BITS/2) ; i++)
	{
		key1[i] = Lbits[i];
	}

	for (i = (KEY_USED_BITS/2),j = 0 ; j < (KEY_USED_BITS/2) ; i++,j++)
	{
		key1[i] = Rbits[j];
	}
```

```
	return key1;
}

void CDES::keyPermutationA(char *keyJoined,int k)
{
#ifdef SDES
	key[k-1][0] = keyJoined[6-1];
	key[k-1][1] = keyJoined[3-1];
	key[k-1][2] = keyJoined[7-1];
	key[k-1][3] = keyJoined[4-1];
	key[k-1][4] = keyJoined[8-1];
	key[k-1][5] = keyJoined[5-1];
	key[k-1][6] = keyJoined[10-1];
	key[k-1][7] = keyJoined[9-1];
#else
	key[k-1][0] = keyJoined[14-1];
	key[k-1][1] = keyJoined[17-1];
	key[k-1][2] = keyJoined[11-1];
	key[k-1][3] = keyJoined[24-1];
	key[k-1][4] = keyJoined[1-1];
	key[k-1][5] = keyJoined[5-1];
	key[k-1][6] = keyJoined[3-1];
	key[k-1][7] = keyJoined[28-1];
	key[k-1][8] = keyJoined[15-1];
	key[k-1][9] = keyJoined[6-1];
	key[k-1][10] = keyJoined[21-1];
	key[k-1][11] = keyJoined[10-1];
	key[k-1][12] = keyJoined[23-1];
	key[k-1][13] = keyJoined[19-1];
	key[k-1][14] = keyJoined[12-1];
	key[k-1][15] = keyJoined[4-1];
	key[k-1][16] = keyJoined[26-1];
	key[k-1][17] = keyJoined[8-1];
	key[k-1][18] = keyJoined[16-1];
	key[k-1][19] = keyJoined[7-1];
	key[k-1][20] = keyJoined[27-1];
	key[k-1][21] = keyJoined[20-1];
	key[k-1][22] = keyJoined[13-1];
	key[k-1][23] = keyJoined[2-1];
	key[k-1][24] = keyJoined[41-1];
	key[k-1][25] = keyJoined[52-1];
	key[k-1][26] = keyJoined[31-1];
	key[k-1][27] = keyJoined[37-1];
	key[k-1][28] = keyJoined[47-1];
	key[k-1][29] = keyJoined[55-1];
	key[k-1][30] = keyJoined[30-1];
	key[k-1][31] = keyJoined[40-1];
	key[k-1][32] = keyJoined[51-1];
	key[k-1][33] = keyJoined[45-1];
	key[k-1][34] = keyJoined[33-1];
	key[k-1][35] = keyJoined[48-1];
	key[k-1][36] = keyJoined[44-1];
```

```
	key[k-1][37] = keyJoined[49-1];
	key[k-1][38] = keyJoined[39-1];
	key[k-1][39] = keyJoined[56-1];
	key[k-1][40] = keyJoined[34-1];
	key[k-1][41] = keyJoined[53-1];
	key[k-1][42] = keyJoined[46-1];
	key[k-1][43] = keyJoined[42-1];
	key[k-1][44] = keyJoined[50-1];
	key[k-1][45] = keyJoined[36-1];
	key[k-1][46] = keyJoined[29-1];
	key[k-1][47] = keyJoined[32-1];
#endif
}

char* CDES::DataIP()
{
	char *dataIP = new char[DATA_LENGTH];

	dataIP[0] = data[2-1];
	dataIP[1] = data[6-1];
	dataIP[2] = data[3-1];
	dataIP[3] = data[1-1];
	dataIP[4] = data[4-1];
	dataIP[5] = data[8-1];
	dataIP[6] = data[5-1];
	dataIP[7] = data[7-1];

	return dataIP;
}

char* CDES::DataIP_DES()
{
	char *dataIP = new char[DATA_LENGTH];

	dataIP[0] = data[58-1];
	dataIP[1] = data[50-1];
	dataIP[2] = data[42-1];
	dataIP[3] = data[34-1];
	dataIP[4] = data[26-1];
	dataIP[5] = data[18-1];
	dataIP[6] = data[10-1];
	dataIP[7] = data[2-1];
	dataIP[8] = data[60-1];
	dataIP[9] = data[52-1];
	dataIP[10] = data[44-1];
	dataIP[11] = data[36-1];
	dataIP[12] = data[28-1];
	dataIP[13] = data[20-1];
	dataIP[14] = data[12-1];
	dataIP[15] = data[4-1];
	dataIP[16] = data[62-1];
```

```
	dataIP[17] = data[54-1];
	dataIP[18] = data[46-1];
	dataIP[19] = data[38-1];
	dataIP[20] = data[30-1];
	dataIP[21] = data[22-1];
	dataIP[22] = data[14-1];
	dataIP[23] = data[6-1];
	dataIP[24] = data[64-1];
	dataIP[25] = data[56-1];
	dataIP[26] = data[48-1];
	dataIP[27] = data[40-1];
	dataIP[28] = data[32-1];
	dataIP[29] = data[24-1];
	dataIP[30] = data[16-1];
	dataIP[31] = data[8-1];
	dataIP[32] = data[57-1];
	dataIP[33] = data[49-1];
	dataIP[34] = data[41-1];
	dataIP[35] = data[33-1];
	dataIP[36] = data[25-1];
	dataIP[37] = data[17-1];
	dataIP[38] = data[9-1];
	dataIP[39] = data[1-1];
	dataIP[40] = data[59-1];
	dataIP[41] = data[51-1];
	dataIP[42] = data[43-1];
	dataIP[43] = data[35-1];
	dataIP[44] = data[27-1];
	dataIP[45] = data[19-1];
	dataIP[46] = data[11-1];
	dataIP[47] = data[3-1];
	dataIP[48] = data[61-1];
	dataIP[49] = data[53-1];
	dataIP[50] = data[45-1];
	dataIP[51] = data[37-1];
	dataIP[52] = data[29-1];
	dataIP[53] = data[21-1];
	dataIP[54] = data[13-1];
	dataIP[55] = data[5-1];
	dataIP[56] = data[63-1];
	dataIP[57] = data[55-1];
	dataIP[58] = data[47-1];
	dataIP[59] = data[39-1];
	dataIP[60] = data[31-1];
	dataIP[61] = data[23-1];
	dataIP[62] = data[15-1];
	dataIP[63] = data[7-1];

	return dataIP;
}

char* CDES::ExtendedPermutation(char *right)
```

```
{
	char* ep = new char[KEY_P_LENGTH];
	int i = DATA_LENGTH/2;

#ifdef SDES
	ep[0] = right[i+4-1];
	ep[1] = right[i+1-1];
	ep[2] = right[i+2-1];
	ep[3] = right[i+3-1];
	ep[4] = right[i+2-1];
	ep[5] = right[i+3-1];
	ep[6] = right[i+4-1];
	ep[7] = right[i+1-1];
#else

	ep[0] = right[i+32-1];
	ep[1] = right[i+1-1];
	ep[2] = right[i+2-1];
	ep[3] = right[i+3-1];
	ep[4] = right[i+4-1];
	ep[5] = right[i+5-1];
	ep[6] = right[i+4-1];
	ep[7] = right[i+5-1];
	ep[8] = right[i+6-1];
	ep[9] = right[i+7-1];
	ep[10] = right[i+8-1];
	ep[11] = right[i+9-1];
	ep[12] = right[i+8-1];
	ep[13] = right[i+9-1];
	ep[14] = right[i+10-1];
	ep[15] = right[i+11-1];
	ep[16] = right[i+12-1];
	ep[17] = right[i+13-1];
	ep[18] = right[i+12-1];
	ep[19] = right[i+13-1];
	ep[20] = right[i+14-1];
	ep[21] = right[i+15-1];
	ep[22] = right[i+16-1];
	ep[23] = right[i+17-1];
	ep[24] = right[i+16-1];
	ep[25] = right[i+17-1];
	ep[26] = right[i+18-1];
	ep[27] = right[i+19-1];
	ep[28] = right[i+20-1];
	ep[29] = right[i+21-1];
	ep[30] = right[i+20-1];
	ep[31] = right[i+21-1];
	ep[32] = right[i+22-1];
	ep[33] = right[i+23-1];
	ep[34] = right[i+24-1];
	ep[35] = right[i+25-1];
```


Todos los s.o. que hay ahí fuera son más o menos iguales. Todos somos una porquería". Brian Valentine, vicepresidente de Microsoft describiendo el edo., del arte en seguridad de s.o. en 2003

```
	ep[36] = right[i+24-1];
	ep[37] = right[i+25-1];
	ep[38] = right[i+26-1];
	ep[39] = right[i+27-1];
	ep[40] = right[i+28-1];
	ep[41] = right[i+29-1];
	ep[42] = right[i+28-1];
	ep[43] = right[i+29-1];
	ep[44] = right[i+30-1];
	ep[45] = right[i+31-1];
	ep[46] = right[i+32-1];
	ep[47] = right[i+1-1];

#endif

	return ep;
}

char* CDES::XORwithKey(char *b, int k)
{
	char *result = new char[DATA_LENGTH];
	int i;

	for (i = 0 ; i < DATA_LENGTH ; i++)
	{
		result[i] = XOR(b[i],key[k-1][i]);
	}

	return result;
}

char CDES::XOR(char a, char b)
{
	if (a == b)
		return '0';
	else
		return '1';
}

void CDES::FillS0_SDES()
{
	S[0][0][0] = 1;
	S[0][0][1] = 0;
	S[0][0][2] = 3;
	S[0][0][3] = 2;

	S[0][1][0] = 3;
	S[0][1][1] = 2;
	S[0][1][2] = 1;
	S[0][1][3] = 0;

	S[0][2][0] = 0;
```

```
	S[0][2][1] = 2;
	S[0][2][2] = 1;
	S[0][2][3] = 3;

	S[0][3][0] = 3;
	S[0][3][1] = 1;
	S[0][3][2] = 3;
	S[0][3][3] = 2;
}

void CDES::FillS1_SDES()
{
	S[1][0][0] = 0;
	S[1][0][1] = 1;
	S[1][0][2] = 2;
	S[1][0][3] = 3;

	S[1][1][0] = 2;
	S[1][1][1] = 0;
	S[1][1][2] = 1;
	S[1][1][3] = 3;

	S[1][2][0] = 3;
	S[1][2][1] = 0;
	S[1][2][2] = 1;
	S[1][2][3] = 0;

	S[1][3][0] = 2;
	S[1][3][1] = 1;
	S[1][3][2] = 0;
	S[1][3][3] = 3;
}

char* CDES::GetSValue(char *b, int s)
{
	char ra,rb,ca,cb;
	char* svalue;

#ifdef SDES
	if (s == 0)			// value from S0
	{
		ra = b[1-1];
		rb = b[4-1];

		ca = b[2-1];
		cb = b[3-1];
	}
	else if (s == 1)  // value from S1
	{
		ra = b[5-1];
		rb = b[8-1];
```


Todos los s.o. que hay ahí fuera son más o menos iguales. Todos somos una porquería". Brian Valentine, vicepresidente de Microsoft describiendo el edo., del arte en seguridad de s.o. en 2003

```
            ca = b[6-1];
            cb = b[7-1];
      }

      svalue = Bin2bit(GetValue(Dec2bit(ra,rb),Dec2bit(ca,cb),s));
#else
      char cc,cd;

      if (s == 0)
      {
            ra = b[1-1];
            rb = b[6-1];

            ca = b[2-1];
            cb = b[3-1];
            cc = b[4-1];
            cd = b[5-1];
      }
      else if (s == 1)
      {
            ra = b[7-1];
            rb = b[12-1];

            ca = b[8-1];
            cb = b[9-1];
            cc = b[10-1];
            cd = b[11-1];
      }
      else if (s == 2)
      {
            ra = b[13-1];
            rb = b[18-1];

            ca = b[14-1];
            cb = b[15-1];
            cc = b[16-1];
            cd = b[17-1];
      }
      else if (s == 3)
      {
            ra = b[19-1];
            rb = b[24-1];

            ca = b[20-1];
            cb = b[21-1];
            cc = b[22-1];
            cd = b[23-1];
      }
      else if (s == 4)
      {
            ra = b[25-1];
```

```
            rb = b[30-1];

            ca = b[26-1];
            cb = b[27-1];
            cc = b[28-1];
            cd = b[29-1];
      }
      else if (s == 5)
      {
            ra = b[31-1];
            rb = b[36-1];

            ca = b[32-1];
            cb = b[33-1];
            cc = b[34-1];
            cd = b[35-1];
      }
      else if (s == 6)
      {
            ra = b[37-1];
            rb = b[42-1];

            ca = b[38-1];
            cb = b[39-1];
            cc = b[40-1];
            cd = b[41-1];
      }
      else if (s == 7)
      {
            ra = b[43-1];
            rb = b[48-1];

            ca = b[44-1];
            cb = b[45-1];
            cc = b[46-1];
            cd = b[47-1];
      }

      svalue = Bin4bit(GetValue(Dec2bit(ra,rb),Dec4bit(ca,cb,cc,cd),s));
#endif
      return svalue;
}

int CDES::GetValue(int a, int b, int s)
{
      cout << endl << "Getting S" << s << " value of " << a << " " << b;

      return S[s][a][b];
}

int CDES::Dec2bit(char a, char b)
```

```
{
	if (a == '0' && b == '0')
		return 0;
	else if (a == '0' && b == '1')
		return 1;
	else if (a == '1' && b == '0')
		return 2;
	else
		return 3;
}

int CDES::Dec4bit(char a, char b, char c, char d)
{
	if		(a == '0' && b == '0' && c == '0' && d == '0')
		return 0;
	else if (a == '0' && b == '0' && c == '0' && d == '1')
		return 1;
	else if (a == '0' && b == '0' && c == '1' && d == '0')
		return 2;
	else if (a == '0' && b == '0' && c == '1' && d == '1')
		return 3;
	else if (a == '0' && b == '1' && c == '0' && d == '0')
		return 4;
	else if (a == '0' && b == '1' && c == '0' && d == '1')
		return 5;
	else if (a == '0' && b == '1' && c == '1' && d == '0')
		return 6;
	else if (a == '0' && b == '1' && c == '1' && d == '1')
		return 7;
	else if (a == '1' && b == '0' && c == '0' && d == '0')
		return 8;
	else if (a == '1' && b == '0' && c == '0' && d == '1')
		return 9;
	else if (a == '1' && b == '0' && c == '1' && d == '0')
		return 10;
	else if (a == '1' && b == '0' && c == '1' && d == '1')
		return 11;
	else if (a == '1' && b == '1' && c == '0' && d == '0')
		return 12;
	else if (a == '1' && b == '1' && c == '0' && d == '1')
		return 13;
	else if (a == '1' && b == '1' && c == '1' && d == '0')
		return 14;
	else
		return 15;
}

char* CDES::Bin2bit(int a)
{
	if (a == 0)
		return "00";
	else if (a == 1)
```

```
		return "01";
	else if (a == 2)
		return "10";
	else
		return "11";
}

char* CDES::Bin4bit(int a)
{
	if		(a == 0)
		return "0000";
	else if (a == 1)
		return "0001";
	else if (a == 2)
		return "0010";
	else if (a == 3)
		return "0011";
	else if (a == 4)
		return "0100";
	else if (a == 5)
		return "0101";
	else if (a == 6)
		return "0110";
	else if (a == 7)
		return "0111";
	else if (a == 8)
		return "1000";
	else if (a == 9)
		return "1001";
	else if (a == 10)
		return "1010";
	else if (a == 11)
		return "1011";
	else if (a == 12)
		return "1100";
	else if (a == 13)
		return "1101";
	else if (a == 14)
		return "1110";
	else
		return "1111";
}

char* CDES::JoinSres(char *r1, char *r2)
{
	char *res = new char[DATA_LENGTH/2];
	int i,j;

	for (i = 0 ; i < ((DATA_LENGTH/2)/2) ; i++)
	{
		res[i] = r1[i];
```

```
	}

	for (i = ((DATA_LENGTH/2)/2),j = 0 ; j < ((DATA_LENGTH/2)/2) ; i+
+,j++)
	{
		res[i] = r2[j];
	}

	return res;
}

char* CDES::PermuteSres(char *sr)
{
	char *r = new char[(DATA_LENGTH/2)];

	r[0] = sr[2-1];
	r[1] = sr[4-1];
	r[2] = sr[3-1];
	r[3] = sr[1-1];

	return r;
}

void CDES::SampleKeys()
{
	key[1-1][0] = '1';
	key[1-1][1] = '0';
	key[1-1][2] = '1';
	key[1-1][3] = '0';
	key[1-1][4] = '0';
	key[1-1][5] = '1';
	key[1-1][6] = '0';
	key[1-1][7] = '0';

	key[2-1][0] = '0';
	key[2-1][1] = '1';
	key[2-1][2] = '0';
	key[2-1][3] = '0';
	key[2-1][4] = '0';
	key[2-1][5] = '0';
	key[2-1][6] = '1';
	key[2-1][7] = '1';
}

void CDES::print(char *d, int length)
{
	int i;

	cout << endl;
	for (i = 0 ; i < length ; i++)
	{
		cout << d[i];
```

```
	}
	cout << endl;
}

char* CDES::XORLandP(char *left, char *p4r)
{
	int i;
	char *res = new char[DATA_LENGTH/2];

	for (i = 0 ; i < (DATA_LENGTH/2) ; i++)
	{
		res[i] = XOR(left[i],p4r[i]);
	}

	return res;
}

char* CDES::FKFinal(char *xr, char *right)
{
	char* res = new char[DATA_LENGTH];
	int i;

	for (i = 0 ; i < (DATA_LENGTH/2) ; i++)
	{
		res[i] = xr[i];
	}

	for (i = (DATA_LENGTH/2) ; i < DATA_LENGTH ; i++)
	{
		res[i] = right[i];
	}

	return res;
}

char* CDES::FKFunction(char *ip,int f)
{
	// begin the KF function
	char *ep = ExtendedPermutation(ip);

	cout << endl << "Extended Permutation";
	print(ep,KEY_P_LENGTH);

	cout << endl << "The k" << f;
	print(key[f-1],KEY_P_LENGTH);

	char *keyXor = XORwithKey(ep,f);

	cout << endl << "XOR with k" << f;
	print(keyXor,KEY_P_LENGTH);
```

```
	char *S0res = GetSValue(keyXor,0);

	cout << endl << "S0 Result";
#ifdef SDES
	print(S0res,DATA_LENGTH/4);
#else
	print(S0res,4);
#endif

	char *S1res = GetSValue(keyXor,1);

	cout << endl << "S1 Result";
#ifdef SDES
	print(S1res,DATA_LENGTH/4);
#else
	print(S1res,4);
#endif

#ifdef DES
	char *S2res = GetSValue(keyXor,2);
	cout << endl << "S2 Result";
	print(S2res,4);

	char *S3res = GetSValue(keyXor,3);
	cout << endl << "S3 Result";
	print(S3res,4);

	char *S4res = GetSValue(keyXor,4);
	cout << endl << "S4 Result";
	print(S4res,4);

	char *S5res = GetSValue(keyXor,5);
	cout << endl << "S5 Result";
	print(S5res,4);

	char *S6res = GetSValue(keyXor,6);
	cout << endl << "S6 Result";
	print(S6res,4);

	char *S7res = GetSValue(keyXor,7);
	cout << endl << "S7 Result";
	print(S7res,4);
#endif

#ifdef SDES
	char *Sresult = JoinSres(S0res,S1res);
	cout << endl << "Joined S0 and S1 Results";
#else
	char                              *Sresult                              =
JoinSres_DES(S0res,S1res,S2res,S3res,S4res,S5res,S6res,S7res);
	cout << endl << "Joined Printed S Results";
#endif
```

```
	print(Sresult,DATA_LENGTH/2);

#ifdef SDES
	char *pSres = PermuteSres(Sresult);
	cout << endl << "Permutation P4";
#else
	char *pSres = PermuteSres_DES(Sresult);
	cout << endl << "Permutation P32";
#endif

	print(pSres,DATA_LENGTH/2);

	char *XorLandPr = XORLandP(ip,pSres);
#ifdef SDES
	cout << endl << "XOR L and P4 Result";
#else
	cout << endl << "XOR L and P32 Result";
#endif

	print(XorLandPr,DATA_LENGTH/2);

	char *FKFinalRes = FKFinal(XorLandPr,ip);

	return FKFinalRes;
}

char* CDES::Switch(char *p)
{
	int i,j;
	char *res = new char[DATA_LENGTH];

	for (i = 0,j = (DATA_LENGTH/2) ; i < (DATA_LENGTH/2) ; i++,j++)
	{
		res[i] = p[j];
	}

	for (i = (DATA_LENGTH/2),j = 0 ; i < DATA_LENGTH ; i++,j++)
	{
		res[i] = p[j];
	}

	return res;
}

char* CDES::IPInverse(char *r)
{
	char *res = new char[DATA_LENGTH];

#ifdef SDES
	res[0] = r[4-1];
```

```
        res[1] = r[1-1];
        res[2] = r[3-1];
        res[3] = r[5-1];
        res[4] = r[7-1];
        res[5] = r[2-1];
        res[6] = r[8-1];
        res[7] = r[6-1];
#else
        res[0] = r[40-1];
        res[1] = r[8-1];
        res[2] = r[48-1];
        res[3] = r[16-1];
        res[4] = r[56-1];
        res[5] = r[24-1];
        res[6] = r[64-1];
        res[7] = r[32-1];
        res[8] = r[39-1];
        res[9] = r[7-1];
        res[10] = r[47-1];
        res[11] = r[15-1];
        res[12] = r[55-1];
        res[13] = r[23-1];
        res[14] = r[63-1];
        res[15] = r[31-1];
        res[16] = r[38-1];
        res[17] = r[6-1];
        res[18] = r[46-1];
        res[19] = r[14-1];
        res[20] = r[54-1];
        res[21] = r[22-1];
        res[22] = r[62-1];
        res[23] = r[30-1];
        res[24] = r[37-1];
        res[25] = r[5-1];
        res[26] = r[45-1];
        res[27] = r[13-1];
        res[28] = r[53-1];
        res[29] = r[21-1];
        res[30] = r[61-1];
        res[31] = r[29-1];
        res[32] = r[36-1];
        res[33] = r[4-1];
        res[34] = r[44-1];
        res[35] = r[12-1];
        res[36] = r[52-1];
        res[37] = r[20-1];
        res[38] = r[60-1];
        res[39] = r[28-1];
        res[40] = r[35-1];
        res[41] = r[3-1];
        res[42] = r[43-1];
        res[43] = r[11-1];
```

```
	res[44] = r[51-1];
	res[45] = r[19-1];
	res[46] = r[59-1];
	res[47] = r[27-1];
	res[48] = r[34-1];
	res[49] = r[2-1];
	res[50] = r[42-1];
	res[51] = r[10-1];
	res[52] = r[50-1];
	res[53] = r[18-1];
	res[54] = r[58-1];
	res[55] = r[26-1];
	res[56] = r[33-1];
	res[57] = r[1-1];
	res[58] = r[41-1];
	res[59] = r[9-1];
	res[60] = r[49-1];
	res[61] = r[17-1];
	res[62] = r[57-1];
	res[63] = r[25-1];
#endif

	return res;
}

char* CDES::Decrypt()
{
	// Calculate Keys
	GenKeys();

	// Begin Encryption
#ifdef SDES
	char *ip = DataIP(); // Initial Permutation on Input bits
#else
	char *ip = DataIP_DES();
#endif

	cout << endl << "Initial Permutation";
	print(ip,DATA_LENGTH);

	char *loopData = ip;
	char *FKr;

	for (int i = NO_OF_KEYS ; i >= 1 ; i--)
	{
		FKr = FKFunction(loopData,i);

		cout << endl << "FK" << i << " Result";
		print(FKr,DATA_LENGTH);

		if (i != 1)
```

```
		{
			loopData = Switch(FKr);

			cout << endl << "Switch Result";
			print(loopData,DATA_LENGTH);
		}
		else
		{
			loopData = FKr;
		}
	}

	char *IPinv = IPInverse(loopData);

	cout << endl << "IP Inverse";
	print(IPinv,DATA_LENGTH);

	return IPinv;
}

void CDES::FillAllS()
{
#ifdef SDES
	FillS0_SDES();
	FillS1_SDES();
#else DES
	FillS0_DES();
	FillS1_DES();
	FillS2_DES();
	FillS3_DES();
	FillS4_DES();
	FillS5_DES();
	FillS6_DES();
	FillS7_DES();
#endif
}

void CDES::FillS0_DES()
{
	S[0][0][0] = 14;  S[0][0][1] = 4;         S[0][0][2] = 13;  S[0][0]
[3] = 1;
	S[0][0][4] = 2;         S[0][0][5] = 15;  S[0][0][6] = 11;  S[0][0]
[7] = 8;
	S[0][0][8] = 3;         S[0][0][9] = 10;  S[0][0][10] = 6;  S[0][0]
[11] = 12;
	S[0][0][12] = 5;  S[0][0][13] = 9;  S[0][0][14] = 0;  S[0][0][15]   =
7;

	S[0][1][0] = 0;         S[0][1][1] = 15;  S[0][1][2] = 7;
	S[0][1][3] = 4;
	S[0][1][4] = 14;  S[0][1][5] = 2;         S[0][1][6] = 13;  S[0][1]
[7] = 1;
```

```
	S[0][1][8] = 10;  S[0][1][9] = 6;         S[0][1][10] = 12; S[0][1]
[11] = 11;
	S[0][1][12] = 9;  S[0][1][13] = 5;  S[0][1][14] = 3;  S[0][1][15]   =
8;

	S[0][2][0] = 4;         S[0][2][1] = 1;         S[0][2][2]    =    14;
	S[0][2][3] = 8;
	S[0][2][4] = 13;  S[0][2][5] = 6;         S[0][2][6] = 2;
	S[0][2][7] = 11;
	S[0][2][8] = 15;  S[0][2][9] = 12;  S[0][2][10] = 9;  S[0][2][11]   =
7;
	S[0][2][12] = 3;  S[0][2][13] = 10; S[0][2][14] = 5;  S[0][2][15]   =
0;

	S[0][3][0] = 15;  S[0][3][1] = 12;  S[0][3][2] = 8;         S[0][3]
[3] = 2;
	S[0][3][4] = 4;         S[0][3][5] = 9;         S[0][3][6] = 1;
	S[0][3][7] = 7;
	S[0][3][8] = 5;         S[0][3][9] = 11;  S[0][3][10] = 3;  S[0][3]
[11] = 14;
	S[0][3][12] = 10; S[0][3][13] = 0;  S[0][3][14] = 6;  S[0][3][15]   =
13;
}

void CDES::FillS1_DES()
{
	S[1][0][0] = 15;  S[1][0][1] = 1;         S[1][0][2] = 8;
	S[1][0][3] = 14;
	S[1][0][4] = 6;         S[1][0][5] = 11;  S[1][0][6] = 3;
	S[1][0][7] = 4;
	S[1][0][8] = 9;         S[1][0][9] = 7;         S[1][0][10]    =    2;
	S[1][0][11] = 13;
	S[1][0][12] = 12; S[1][0][13] = 0;  S[1][0][14] = 5;  S[1][0][15]   =
10;

	S[1][1][0] = 3;         S[1][1][1] = 13;  S[1][1][2] = 4;
	S[1][1][3] = 7;
	S[1][1][4] = 15;  S[1][1][5] = 2;         S[1][1][6] = 8;
	S[1][1][7] = 14;
	S[1][1][8] = 12;  S[1][1][9] = 0;         S[1][1][10] = 1;  S[1][1]
[11] = 10;
	S[1][1][12] = 6;  S[1][1][13] = 9;  S[1][1][14] = 11; S[1][1][15]   =
5;

	S[1][2][0] = 0;         S[1][2][1] = 14;  S[1][2][2] = 7;
	S[1][2][3] = 11;
	S[1][2][4] = 10;  S[1][2][5] = 4;         S[1][2][6] = 13;  S[1][2]
[7] = 1;
	S[1][2][8] = 5;         S[1][2][9] = 8;         S[1][2][10]    =    12;
	S[1][2][11] = 6;
```

```
	S[1][2][12] = 9;  S[1][2][13] = 3;  S[1][2][14] = 2;  S[1][2][15]   =
15;

	S[1][3][0] = 13;  S[1][3][1] = 8;         S[1][3][2] = 10;  S[1][3]
[3] = 1;
	S[1][3][4] = 3;         S[1][3][5] = 15;  S[1][3][6] = 4;
	S[1][3][7] = 2;
	S[1][3][8] = 11;  S[1][3][9] = 6;         S[1][3][10] = 7;  S[1][3]
[11] = 12;
	S[1][3][12] = 0;  S[1][3][13] = 5;  S[1][3][14] = 14; S[1][3][15]   =
9;
}

void CDES::FillS2_DES()
{
	S[2][0][0] = 10;  S[2][0][1] = 0;         S[2][0][2] = 9;
	S[2][0][3] = 14;
	S[2][0][4] = 6;         S[2][0][5] = 3;         S[2][0][6]    =    15;
	S[2][0][7] = 5;
	S[2][0][8] = 1;         S[2][0][9] = 13;  S[2][0][10] = 12; S[2][0]
[11] = 7;
	S[2][0][12] = 11; S[2][0][13] = 4;  S[2][0][14] = 2;  S[2][0][15]   =
8;

	S[2][1][0] = 13;  S[2][1][1] = 7;         S[2][1][2] = 0;
	S[2][1][3] = 9;
	S[2][1][4] = 3;         S[2][1][5] = 4;         S[2][1][6] = 6;
	S[2][1][7] = 10;
	S[2][1][8] = 2;         S[2][1][9] = 8;         S[2][1][10]    =    5;
	S[2][1][11] = 14;
	S[2][1][12] = 12; S[2][1][13] = 10; S[2][1][14] = 15; S[2][1][15]   =
1;

	S[2][2][0] = 13;  S[2][2][1] = 6;         S[2][2][2] = 4;
	S[2][2][3] = 9;
	S[2][2][4] = 8;         S[2][2][5] = 15;  S[2][2][6] = 3;
	S[2][2][7] = 0;
	S[2][2][8] = 11;  S[2][2][9] = 1;         S[2][2][10] = 2;  S[2][2]
[11] = 12;
	S[2][2][12] = 5;  S[2][2][13] = 10; S[2][2][14] = 14; S[2][2][15]   =
7;

	S[2][3][0] = 1;         S[2][3][1] = 10;  S[2][3][2] = 13;  S[2][3]
[3] = 0;
	S[2][3][4] = 6;         S[2][3][5] = 9;         S[2][3][6] = 8;
	S[2][3][7] = 7;
	S[2][3][8] = 4;         S[2][3][9] = 15;  S[2][3][10] = 14; S[2][3]
[11] = 3;
	S[2][3][12] = 11; S[2][3][13] = 5;  S[2][3][14] = 2;  S[2][3][15]   =
12;
}
```

```
void CDES::FillS3_DES()
{
      S[3][0][0] = 7;         S[3][0][1] = 13;  S[3][0][2] = 14;  S[3][0]
[3] = 3;
      S[3][0][4] = 0;         S[3][0][5] = 6;         S[3][0][6] = 9;
      S[3][0][7] = 10;
      S[3][0][8] = 1;         S[3][0][9] = 2;         S[3][0][10]    =    8;
      S[3][0][11] = 5;
      S[3][0][12] = 11; S[3][0][13] = 12; S[3][0][14] = 4;  S[3][0][15]   =
15;

      S[3][1][0] = 13;  S[3][1][1] = 8;         S[3][1][2] = 11;  S[3][1]
[3] = 5;
      S[3][1][4] = 6;         S[3][1][5] = 15;  S[3][1][6] = 0;
      S[3][1][7] = 3;
      S[3][1][8] = 4;         S[3][1][9] = 7;         S[3][1][10]    =    2;
      S[3][1][11] = 12;
      S[3][1][12] = 1;  S[3][1][13] = 10; S[3][1][14] = 14; S[3][1][15]   =
9;

      S[3][2][0] = 10;  S[3][2][1] = 6;         S[3][2][2] = 9;
      S[3][2][3] = 0;
      S[3][2][4] = 12;  S[3][2][5] = 11;  S[3][2][6] = 7;         S[3][2]
[7] = 13;
      S[3][2][8] = 15;  S[3][2][9] = 1;         S[3][2][10] = 3;  S[3][2]
[11] = 14;
      S[3][2][12] = 5;  S[3][2][13] = 2;  S[3][2][14] = 8;  S[3][2][15]   =
4;

      S[3][3][0] = 3;         S[3][3][1] = 15;  S[3][3][2] = 0;
      S[3][3][3] = 6;
      S[3][3][4] = 10;  S[3][3][5] = 1;         S[3][3][6] = 13;  S[3][3]
[7] = 8;
      S[3][3][8] = 9;         S[3][3][9] = 4;         S[3][3][10]    =    5;
      S[3][3][11] = 11;
      S[3][3][12] = 12; S[3][3][13] = 7;  S[3][3][14] = 2;  S[3][3][15]   =
14;
}

void CDES::FillS4_DES()
{
      S[4][0][0] = 2;         S[4][0][1] = 12;  S[4][0][2] = 4;
      S[4][0][3] = 1;
      S[4][0][4] = 7;         S[4][0][5] = 10;  S[4][0][6] = 11;  S[4][0]
[7] = 6;
      S[4][0][8] = 8;         S[4][0][9] = 5;         S[4][0][10]    =    3;
      S[4][0][11] = 15;
      S[4][0][12] = 13; S[4][0][13] = 0;  S[4][0][14] = 14; S[4][0][15]   =
9;
```

```
      S[4][1][0] = 14;  S[4][1][1] = 11;  S[4][1][2] = 2;         S[4][1]
[3] = 12;
      S[4][1][4] = 4;         S[4][1][5] = 7;         S[4][1][6]    =    13;
      S[4][1][7] = 1;
      S[4][1][8] = 5;         S[4][1][9] = 0;         S[4][1][10]   =   15;
      S[4][1][11] = 10;
      S[4][1][12] = 3;  S[4][1][13] = 9;  S[4][1][14] = 8;  S[4][1][15]   =
6;

      S[4][2][0] = 4;         S[4][2][1] = 2;         S[4][2][2] = 1;
      S[4][2][3] = 11;
      S[4][2][4] = 10;  S[4][2][5] = 13;  S[4][2][6] = 7;         S[4][2]
[7] = 8;
      S[4][2][8] = 15;  S[4][2][9] = 9;         S[4][2][10] = 12; S[4][2]
[11] = 5;
      S[4][2][12] = 6;  S[4][2][13] = 3;  S[4][2][14] = 0;  S[4][2][15]   =
14;

      S[4][3][0] = 11;  S[4][3][1] = 8;         S[4][3][2] = 12;  S[4][3]
[3] = 7;
      S[4][3][4] = 1;         S[4][3][5] = 14;  S[4][3][6] = 2;
      S[4][3][7] = 13;
      S[4][3][8] = 6;         S[4][3][9] = 15;  S[4][3][10] = 0;  S[4][3]
[11] = 9;
      S[4][3][12] = 10; S[4][3][13] = 4;  S[4][3][14] = 5;  S[4][3][15]   =
3;
}

void CDES::FillS5_DES()
{
      S[5][0][0] = 12;  S[5][0][1] = 1;         S[5][0][2] = 10;  S[5][0]
[3] = 15;
      S[5][0][4] = 9;         S[5][0][5] = 2;         S[5][0][6] = 6;
      S[5][0][7] = 8;
      S[5][0][8] = 0;         S[5][0][9] = 13;  S[5][0][10] = 3;  S[5][0]
[11] = 4;
      S[5][0][12] = 14; S[5][0][13] = 7;  S[5][0][14] = 5;  S[5][0][15]   =
11;

      S[5][1][0] = 10;  S[5][1][1] = 15;  S[5][1][2] = 4;         S[5][1]
[3] = 2;
      S[5][1][4] = 7;         S[5][1][5] = 12;  S[5][1][6] = 9;
      S[5][1][7] = 5;
      S[5][1][8] = 6;         S[5][1][9] = 1;         S[5][1][10]   =   13;
      S[5][1][11] = 14;
      S[5][1][12] = 0;  S[5][1][13] = 11; S[5][1][14] = 3;  S[5][1][15]   =
8;

      S[5][2][0] = 9;         S[5][2][1] = 14;  S[5][2][2] = 15;  S[5][2]
[3] = 5;
      S[5][2][4] = 2;         S[5][2][5] = 8;         S[5][2][6]    =    12;
      S[5][2][7] = 3;
```

```
	S[5][2][8] = 7;		S[5][2][9] = 0;		S[5][2][10]	=	4;
	S[5][2][11] = 10;
	S[5][2][12] = 1;  S[5][2][13] = 13; S[5][2][14] = 11; S[5][2][15]	=
6;

	S[5][3][0] = 4;		S[5][3][1] = 3;		S[5][3][2] = 2;
	S[5][3][3] = 12;
	S[5][3][4] = 9;		S[5][3][5] = 5;		S[5][3][6]	=	15;
	S[5][3][7] = 10;
	S[5][3][8] = 11;  S[5][3][9] = 14;  S[5][3][10] = 1;  S[5][3][11]	=
7;
	S[5][3][12] = 6;  S[5][3][13] = 0;  S[5][3][14] = 8;  S[5][3][15]	=
13;
}

void CDES::FillS6_DES()
{
	S[6][0][0] = 4;		S[6][0][1] = 11;  S[6][0][2] = 2;
	S[6][0][3] = 14;
	S[6][0][4] = 15;  S[6][0][5] = 0;		S[6][0][6] = 8;
	S[6][0][7] = 13;
	S[6][0][8] = 3;		S[6][0][9] = 12;  S[6][0][10] = 9;  S[6][0]
[11] = 7;
	S[6][0][12] = 5;  S[6][0][13] = 10; S[6][0][14] = 6;  S[6][0][15]	=
1;

	S[6][1][0] = 13;  S[6][1][1] = 0;		S[6][1][2] = 11;  S[6][1]
[3] = 7;
	S[6][1][4] = 4;		S[6][1][5] = 9;		S[6][1][6] = 1;
	S[6][1][7] = 10;
	S[6][1][8] = 14;  S[6][1][9] = 3;		S[6][1][10] = 5;  S[6][1]
[11] = 12;
	S[6][1][12] = 2;  S[6][1][13] = 15; S[6][1][14] = 8;  S[6][1][15]	=
6;

	S[6][2][0] = 1;		S[6][2][1] = 4;		S[6][2][2]	=	11;
	S[6][2][3] = 13;
	S[6][2][4] = 12;  S[6][2][5] = 3;		S[6][2][6] = 7;
	S[6][2][7] = 14;
	S[6][2][8] = 10;  S[6][2][9] = 15;  S[6][2][10] = 6;  S[6][2][11]	=
8;
	S[6][2][12] = 0;  S[6][2][13] = 5;  S[6][2][14] = 9;  S[6][2][15]	=
2;

	S[6][3][0] = 6;		S[6][3][1] = 11;  S[6][3][2] = 13;  S[6][3]
[3] = 8;
	S[6][3][4] = 1;		S[6][3][5] = 4;		S[6][3][6]	=	10;
	S[6][3][7] = 7;
	S[6][3][8] = 9;		S[6][3][9] = 5;		S[6][3][10]	=	0;
	S[6][3][11] = 15;
```

```
	S[6][3][12] = 14; S[6][3][13] = 2;  S[6][3][14] = 3;  S[6][3][15]   =
12;
}

void CDES::FillS7_DES()
{
	S[7][0][0] = 13;  S[7][0][1] = 2;         S[7][0][2] = 8;
	S[7][0][3] = 4;
	S[7][0][4] = 6;         S[7][0][5] = 15;  S[7][0][6] = 11;  S[7][0]
[7] = 1;
	S[7][0][8] = 10;  S[7][0][9] = 9;         S[7][0][10] = 3;  S[7][0]
[11] = 14;
	S[7][0][12] = 5;  S[7][0][13] = 0;  S[7][0][14] = 12; S[7][0][15]   =
7;

	S[7][1][0] = 1;         S[7][1][1] = 15;  S[7][1][2] = 13;  S[7][1]
[3] = 8;
	S[7][1][4] = 10;  S[7][1][5] = 3;         S[7][1][6] = 7;
	S[7][1][7] = 4;
	S[7][1][8] = 12;  S[7][1][9] = 5;         S[7][1][10] = 6;  S[7][1]
[11] = 11;
	S[7][1][12] = 0;  S[7][1][13] = 14; S[7][1][14] = 9;  S[7][1][15]   =
2;

	S[7][2][0] = 7;         S[7][2][1] = 11;  S[7][2][2] = 4;
	S[7][2][3] = 1;
	S[7][2][4] = 9;         S[7][2][5] = 12;  S[7][2][6] = 14;  S[7][2]
[7] = 2;
	S[7][2][8] = 0;         S[7][2][9] = 6;         S[7][2][10]   =   10;
	S[7][2][11] = 13;
	S[7][2][12] = 15; S[7][2][13] = 3;  S[7][2][14] = 5;  S[7][2][15]   =
8;

	S[7][3][0] = 2;         S[7][3][1] = 1;         S[7][3][2]    =    14;
	S[7][3][3] = 7;
	S[7][3][4] = 4;         S[7][3][5] = 10;  S[7][3][6] = 8;
	S[7][3][7] = 13;
	S[7][3][8] = 15;  S[7][3][9] = 12;  S[7][3][10] = 9;  S[7][3][11]   =
0;
	S[7][3][12] = 3;  S[7][3][13] = 5;  S[7][3][14] = 6;  S[7][3][15]   =
11;
}

char*  CDES::JoinSres_DES(char  *a,  char  *b,  char  *c,  char  *d,  char  *e,
char *f, char *g, char *h)
{
	char* JoinTres = new char[32];
	int i,j;

	for (i = 0 ; i < 4 ; i++)
	{
		JoinTres[i] = a[i];
```

```
	}

	for (i = 4,j = 0 ; i < 8 ; i++,j++)
	{
		JoinTres[i] = b[j];
	}

	for (i = 8,j = 0 ; i < 12 ; i++,j++)
	{
		JoinTres[i] = c[j];
	}

	for (i = 12,j = 0 ; i < 16 ; i++,j++)
	{
		JoinTres[i] = d[j];
	}

	for (i = 16,j = 0 ; i < 20 ; i++,j++)
	{
		JoinTres[i] = e[j];
	}

	for (i = 20,j = 0 ; i < 24 ; i++,j++)
	{
		JoinTres[i] = f[j];
	}

	for (i = 24,j = 0 ; i < 28 ; i++,j++)
	{
		JoinTres[i] = g[j];
	}

	for (i = 28,j = 0 ; i < 32 ; i++,j++)
	{
		JoinTres[i] = h[j];
	}

	return JoinTres;
}

char* CDES::PermuteSres_DES(char *r)
{
	char *res = new char[32];

	res[0] = r[16-1];
	res[1] = r[7-1];
	res[2] = r[20-1];
	res[3] = r[21-1];
	res[4] = r[29-1];
	res[5] = r[12-1];
	res[6] = r[28-1];
```

```
	res[7] = r[17-1];
	res[8] = r[1-1];
	res[9] = r[15-1];
	res[10] = r[23-1];
	res[11] = r[26-1];
	res[12] = r[5-1];
	res[13] = r[18-1];
	res[14] = r[31-1];
	res[15] = r[10-1];
	res[16] = r[2-1];
	res[17] = r[8-1];
	res[18] = r[24-1];
	res[19] = r[14-1];
	res[20] = r[32-1];
	res[21] = r[27-1];
	res[22] = r[3-1];
	res[23] = r[9-1];
	res[24] = r[19-1];
	res[25] = r[13-1];
	res[26] = r[30-1];
	res[27] = r[6-1];
	res[28] = r[22-1];
	res[29] = r[11-1];
	res[30] = r[4-1];
	res[31] = r[25-1];

	return res;
}

void CDES::GenKeys()
{
	char *keyPF = keyIPermutation();
	char *keyLshifted = LeftShift(keyPF,'L',1); // Left shift 1st Five
bits to 1 bit
	char *keyRshifted = LeftShift(keyPF,'R',1); // Left shift 2nd Five
bits to 1 bit

	keyPermutationA(JoinShifted(keyLshifted,keyRshifted),1);  //  create
Key1

#ifdef SDES
	keyLshifted = LeftShift(keyPF,'L',3);
	keyRshifted = LeftShift(keyPF,'R',3);
	keyPermutationA(JoinShifted(keyLshifted,keyRshifted),2);  //  create
Key2
#else
	keyLshifted = LeftShift(keyPF,'L',2);
	keyRshifted = LeftShift(keyPF,'R',2);
	keyPermutationA(JoinShifted(keyLshifted,keyRshifted),2);  //  create
Key2

	keyLshifted = LeftShift(keyPF,'L',4);
```

```
      keyRshifted = LeftShift(keyPF,'R',4);
      keyPermutationA(JoinShifted(keyLshifted,keyRshifted),3);  //  create
key3

      keyLshifted = LeftShift(keyPF,'L',6);
      keyRshifted = LeftShift(keyPF,'R',6);
      keyPermutationA(JoinShifted(keyLshifted,keyRshifted),4);  //  create
key4

      keyLshifted = LeftShift(keyPF,'L',8);
      keyRshifted = LeftShift(keyPF,'R',8);
      keyPermutationA(JoinShifted(keyLshifted,keyRshifted),5);  //  create
key5

      keyLshifted = LeftShift(keyPF,'L',10);
      keyRshifted = LeftShift(keyPF,'R',10);
      keyPermutationA(JoinShifted(keyLshifted,keyRshifted),6);  //  create
key6

      keyLshifted = LeftShift(keyPF,'L',12);
      keyRshifted = LeftShift(keyPF,'R',12);
      keyPermutationA(JoinShifted(keyLshifted,keyRshifted),7);  //  create
key7

      keyLshifted = LeftShift(keyPF,'L',14);
      keyRshifted = LeftShift(keyPF,'R',14);
      keyPermutationA(JoinShifted(keyLshifted,keyRshifted),8);  //  create
key8

      keyLshifted = LeftShift(keyPF,'L',15);
      keyRshifted = LeftShift(keyPF,'R',15);
      keyPermutationA(JoinShifted(keyLshifted,keyRshifted),9);  //  create
key9

      keyLshifted = LeftShift(keyPF,'L',17);
      keyRshifted = LeftShift(keyPF,'R',17);
      keyPermutationA(JoinShifted(keyLshifted,keyRshifted),10);  //  create
key10

      keyLshifted = LeftShift(keyPF,'L',19);
      keyRshifted = LeftShift(keyPF,'R',19);
      keyPermutationA(JoinShifted(keyLshifted,keyRshifted),11);  //  create
key11

      keyLshifted = LeftShift(keyPF,'L',21);
      keyRshifted = LeftShift(keyPF,'R',21);
      keyPermutationA(JoinShifted(keyLshifted,keyRshifted),12);  //  create
key12

      keyLshifted = LeftShift(keyPF,'L',23);
      keyRshifted = LeftShift(keyPF,'R',23);
```

```
      keyPermutationA(JoinShifted(keyLshifted,keyRshifted),13); // create
key13

      keyLshifted = LeftShift(keyPF,'L',25);
      keyRshifted = LeftShift(keyPF,'R',25);
      keyPermutationA(JoinShifted(keyLshifted,keyRshifted),14); // create
key14

      keyLshifted = LeftShift(keyPF,'L',27);
      keyRshifted = LeftShift(keyPF,'R',27);
      keyPermutationA(JoinShifted(keyLshifted,keyRshifted),15); // create
key15

      keyLshifted = LeftShift(keyPF,'L',28);
      keyRshifted = LeftShift(keyPF,'R',28);
      keyPermutationA(JoinShifted(keyLshifted,keyRshifted),16); // create
key16
#endif
}
```

VENTAJAS

- Encriptación y desencriptación idénticas: ejecutando subkeys en el orden reverso
- Fácil implementación
- Análisis simplificado
- Relativamente rápido y seguro bajo antiguos estándares

DESVENTAJAS

- Es lento bajo estándares actuales
- Operaciones a nivel de bits por lo que se expande, operaciones con semiregistros XOR con subkey, operaciones con semibytes, permutación inicial y final.
- Limitado a bloques de 64bits.

ATÁQUES AL ALGORITMO

Aunque se ha publicado más información sobre el criptoanálisis de DES que de ningún otro cifrado de bloque, el ataque más práctico hoy en día sigue siendo por fuerza bruta. Se conocen varias propiedades criptoanalíticas menores, y son posibles tres tipos de ataques teóricos que, aun requiriendo una complejidad teórica menor que

un ataque por fuerza bruta, requieren una cantidad irreal de textos planos conocidos o escogidos para llevarse a cabo, y no se tienen en cuenta en la práctica.

Para cualquier tipo de cifrado, el método de ataque más simple es el ataque por fuerza bruta — probando una por una cada posible clave. La longitud de clave determina el número posible de claves, y por tanto la factibilidad del ataque. En el caso de DES, ya en sus comienzos se plantearon cuestiones sobre su longitud de clave, incluso antes de ser adoptado como estándar, fue su reducido tamaño de clave, más que el criptoanálisis teórico, el que provocó la necesidad de reemplazarlo. Se sabe que la NSA animó, o incluso persuadió a IBM para que redujera el tamaño de clave de 128 bits a 64, y de ahí a 56 bits; con frecuencia esto se ha interpretado como una evidencia de que la NSA poseía suficiente capacidad de computación para romper claves de éste tamaño incluso a mediados de los 70.

Académicamente, se adelantaron varias propuestas de una máquina para romper DES. En 1977, Diffie y Hellman propusieron una máquina con un coste estimado de 20 millones de dólares que podría encontrar una clave DES en un sólo día. Hacia 1993, Wiener propuso una máquina de búsqueda de claves con un coste de un millón de dólares que encontraría una clave en 7 horas. La vulnerabilidad de DES fue demostrada en la práctica en 1998 cuando la Electronic Frontier Foundation (EFF), un grupo dedicado a los derechos civiles en el ciberespacio, construyó una máquina a medida para romper DES, con un coste aproximado de 250000 dólares (véase EFF DES cracker). Su motivación era demostrar que se podía romper DES tanto en la teoría como en la práctica: "Hay mucha gente que no creerá una verdad hasta que puedan verla con sus propios ojos. Mostrarles una máquina física que pueda romper DES en unos pocos días es la única manera de convencer a algunas personas de que realmente no pueden confiar su seguridad a DES." La máquina rompió una clave por fuerza bruta en una búsqueda que duró poco más de 2 días; Más o menos al mismo tiempo, un abogado del Departamento de Justicia de los Estados Unidos proclamaba que DES era irrompible.

Existen tres ataques conocidos que pueden romper las dieciséis rondas completas de DES con menos complejidad que un ataque por fuerza bruta: el criptoanálisis diferencial (CD), el criptoanálisis lineal (CL) y el ataque de Davies. De todas maneras, éstos ataques son sólo teóricos y no es posible llevarlos a la práctica; éste tipo de ataques se denominan a veces debilidades certificacionales.

El criptoanálisis diferencial fue descubierto a finales de los 80 por Eli Biham y Adi Shamir, aunque era conocido anteriormente tanto por la NSA como por IBM y mantenido en secreto. Para romper las 16 rondas completas, el criptoanálisis

Todos los s.o. que hay ahí fuera son más o menos iguales. Todos somos una porquería". Brian Valentine, vicepresidente de Microsoft describiendo el edo., del arte en seguridad de s.o. en 2003

diferencial requiere 247 textos planos escogidos. DES fue diseñado para ser resistente al CD.

El criptoanálisis lineal fue descubierto por Mitsuru Matsui, y necesita 243 textos planos conocidos (Matsui, 1993); el método fue implementado (Matsui, 1994), y fue el primer criptoanálisis experimental de DES que se dio a conocer. No hay evidencias de que DES fuese adaptado para ser resistente a este tipo de ataque. Una generalización del CL — el criptoanálisis lineal múltiple — se propuso en 1994 Kaliski and Robshaw), y fue mejorada por Biryukov y otros (2004); su análisis sugiere que se podrían utilizar múltiples aproximaciones lineales para reducir los requisitos de datos del ataque en al menos un factor de 4 (es decir, 241 en lugar de 243). Una reducción similar en la complejidad de datos puede obtenerse con una variante del criptoanálisis lineal de textos planos escogidos (Knudsen y Mathiassen, 2000). Junod (2001) realizó varios experimentos para determinar la complejidad real del criptoanálisis lineal, y descubrió que era algo más rápido de lo predicho, requiriendo un tiempo equivalente a 239–241 comprobaciones en DES.

El ataque mejorado de Davies: mientras que el análisis lineal y diferencial son técnicas generales y pueden aplicarse a multitud de esquemas diferentes, el ataque de Davies es una técnica especializada para DES. Propuesta por vez primera por Davies en los 80, y mejorada por Biham and Biryukov (1997). La forma más potente del ataque requiere 250 textos planos conocidos, tiene una complejidad computacional de 250, y tiene un 51% de probabilidad de éxito.

Existen también ataques pensados para versiones del algoritmo con menos rondas, es decir versiones de DES con menos de dieciséis rondas. Dichos análisis ofrecen una perspectiva sobre cuantas rondas son necesarias para conseguir seguridad, y cuánto «margen de seguridad» proporciona la versión completa. El criptoanálisis diferencial-lineal fue propuesto por Langford y Hellman en 1994, y combina criptoanálisis diferencial y lineal en un mismo ataque. Una versión mejorada del ataque puede romper un DES de 9 rondas con 215.8 textos planos conocidos y tiene una complejidad temporal de 229.2 (Biham y otros, 2002).

BLOWFISH

Codificador de bloques simétricos, diseñado por Bruce Schneier en 1993 e incluido en un gran número de conjuntos de codificadores y productos de cifrado. Mientras que ningún analizador de cifrados de Blowfish efectivo ha sido encontrado hoy en día, se ha dado más atención de la decodificación de bloques con bloques más grandes, como AES y Twofish.

Schneier diseñó Blowfish como un algoritmo de uso general, que intentaba reemplazar al antiguo DES y evitar los problemas asociados con otros algoritmos. Al mismo tiempo, muchos otros diseños eran propiedad privada, patentados o los guardaba el gobierno. Schneier declaró “Blowfish no tiene patente, y así se quedará en los demás continentes. El algoritmo está a disposición del público, y puede ser usado libremente por cualquiera”.

Blowfish usa bloques de 64 bits y claves que van desde los 32 bits hasta 448 bits. Es un codificador de 16 rondas Feistel y usa llaves que dependen de las Cajas-S. Tiene una estructura similar a CAST-128, el cual usa Cajas-S fijas.

Cada línea representa 32 bits. El algoritmo guarda 2 arrays de subclaves: El array P de 18 entradas y 4 cajas-S de 256 entradas. Una entrada del array P es usada cada ronda, después de la ronda final, a cada mitad del bloque de datos se le aplica un XOR con uno de las 2 entradas del array P que no han sido utilizadas. La función divide las entradas de 32 bits en 4 bloques de 8 bits, y usa los bloques como entradas para las cajas-S. Las salidas deben estar en módulo 232 y se les aplica un XOR para producir la salida final de 32 bits. Blowfish en un ejemplo más de un cifrado simétrico.

CÓDIGO O PSEUDOCÓDIGO

Código implementado en Visual Fox Pro

```
LPARAMETERS pcTexto as String

LOCAL lnTemporalA, lnTemporalB, lnTemporalC, cResultado, lnContador


lnTemporalA=0

lnTemporalB=0
```


Todos los s.o. que hay ahí fuera son más o menos iguales. Todos somos una porquería”. Brian Valentine, vicepresidente de Microsoft describiendo el edo., del arte en seguridad de s.o. en 2003

```
lnTemporalC=0
cResultado=""
FOR lnContador=1 TO LEN(pcTexto)
	lnTemporalA=ASC(SUBSTR(pcTexto,lnContador,1))
	lnTemporalB=FLOOR(lnTemporalA/16)
	lnTemporalC=lnTemporalA % 16
	IF	lnTemporalB<10
		lnTemporalB=lnTemporalB+48
	ELSE
		lnTemporalB=lnTemporalB+55
	ENDIF
	IF	lnTemporalC<10
		lnTemporalC=lnTemporalC+48
	ELSE
		lnTemporalC=lnTemporalC+55
	ENDIF
	cResultado=cResultado+CHR(lnTemporalB)+CHR(lnTemporalC)
ENDFOR
RETURN cResultado

ENDPROC
PROCEDURE decodificar
LPARAMETERS pcTexto as String
LOCAL lnTemporalA, lnTemporalB, lnTemporalC, cResultado, lnContador
lnTemporalA=0
lnTemporalB=0
```

```
lnTemporalC=0
cResultado=""
FOR lnContador=1 TO LEN(pcTexto)
	lnTemporalB=ASC(SUBSTR(pcTexto,lnContador,1))
	lnContador=lnContador+1
	lnTemporalC=ASC(SUBSTR(pcTexto,lnContador,1))
	lnTemporalB=IIF(lnTemporalB=0,48,lnTemporalB)
	IF lnTemporalB<58
		lnTemporalB=lnTemporalB-48
	ELSE
		IF lnTemporalB>96
			lnTemporalB=lnTemporalB-87
		ELSE
			lnTemporalB=lnTemporalB-55
		ENDIF
	ENDIF
	IF lnTemporalC<58
		lnTemporalC=lnTemporalC-48
	ELSE
		IF lnTemporalC>96
			lnTemporalC=lnTemporalC-87
		ELSE
			lnTemporalC=lnTemporalC-55
		ENDIF
	ENDIF
```

```
        lnTemporalA=(lnTemporalB*16)+lnTemporalC
        cResultado=cResultado+CHR(lnTemporalA)
ENDFOR
RETURN cResultado
ENDPROC
PROCEDURE codificarsimple
LPARAMETERS pcTexto as String, pcPalabraClave as String
LOCAL nDelta, lnTemporalA, cSalidaDato, i, j
nDelta=5
lnTemporalA=0
cSalidaDato=""
j=1
IF LEN(pcPalabraClave)=0
        cSalidaDato=this.Codificar(pcTexto)
ELSE
        FOR i=1 TO LEN(pcTexto)
                IF j=LEN(pcPalabraClave)
                        j=1
                ENDIF
                lnTemporalA = ASC(SUBSTR(pcTexto,i,1))
                lnTemporalA = lnTemporalA + ASC(SUBSTR(pcPalabraClave,j,1))
                IF lnTemporalA > 255
                        lnTemporalA=lnTemporalA-256
                ENDIF
                IF lnTemporalA < nDelta
                        cSalidaDato = cSalidaDato + CHR(nDelta)
```

```
                    lnTemporalA = lnTemporalA+ nDelta
                ENDIF
                cSalidaDato = cSalidaDato + CHR(lnTemporalA)
                j=j+1
        ENDFOR
        cSalidaDato=this.Codificar(cSalidaDato)
ENDIF
RETURN cSalidaDato
ENDPROC
PROCEDURE decodificarsimple
LPARAMETERS pcTexto as String, pcPalabraClave as String
LOCAL delta, lnTemporalA, cSalidaDato, i ,j
nDelta=5
lnTemporalA=0
cSalidaDato=""
j=1
WITH this
        IF LEN(pcPalabraClave)=0
                cSalidaDato= .Decodificar(pcTexto) &&myunescape(pcTexto)
        ELSE
                pcTexto=.Decodificar(pcTexto) &&myunescape(pcTexto)
                FOR i=1 TO LEN(pcTexto)
                        IF j=LEN(pcPalabraClave)
                                j=1
                        ENDIF
```

```
                    lnTemporalA = ASC(SUBSTR(pcTexto,i,1))
                    IF lnTemporalA=nDelta
                        i=i+1
                        lnTemporalA = ASC(SUBSTR(pcTexto,i,1))
                        slnTemporalA = lnTemporalA - nDelta
                    ENDIF
                    lnTemporalA = lnTemporalA - ASC(SUBSTR(pcPalabraClave,j,1))
                    IF lnTemporalA < 0
                        lnTemporalA=lnTemporalA+256
                    ENDIF
                    cSalidaDato = cSalidaDato + CHR(lnTemporalA)
                    j=j+1
                ENDFOR
        ENDIF
ENDWITH
RETURN cSalidaDato
ENDPROC
PROCEDURE blosfishconstantes
*!* ATENCION LAS LINEAS SIGUIENTES SON VALORES CONSTANTES.
*!* NO CAMBIE NINGUNO DE LOS ANTERIORES
*----
WITH this
        STORE 0x243f6a88 TO .aP(1)
        STORE 0x85a308d3 TO .aP(2)
        STORE 0x13198a2e TO .aP(3)
        STORE 0x03707344 TO .aP(4)
```

STORE 0xa4093822 TO .aP(5)

STORE 0x299f31d0 TO .aP(6)

STORE 0x082efa98 TO .aP(7)

STORE 0xec4e6c89 TO .aP(8)

STORE 0x452821e6 TO .aP(9)

STORE 0x38d01377 TO .aP(10)

STORE 0xbe5466cf TO .aP(11)

STORE 0x34e90c6c TO .aP(12)

STORE 0xc0ac29b7 TO .aP(13)

STORE 0xc97c50dd TO .aP(14)

STORE 0x3f84d5b5 TO .aP(15)

STORE 0xb5470917 TO .aP(16)

STORE 0x9216d5d9 TO .aP(17)

STORE 0x8979fb1b TO .aP(18)

*---

STORE 0xd1310ba6 TO .aS1(1)

STORE 0x98dfb5ac TO .aS1(2)

STORE 0x2ffd72db TO .aS1(3)

STORE 0xd01adfb7 TO .aS1(4)

STORE 0xb8e1afed TO .aS1(5)

STORE 0x6a267e96 TO .aS1(6)

STORE 0xba7c9045 TO .aS1(7)

STORE 0xf12c7f99 TO .aS1(8)

STORE 0x24a19947 TO .aS1(9)

STORE 0xb3916cf7 TO .aS1(10)

Todos los s.o. que hay ahí fuera son más o menos iguales. Todos somos una porquería". Brian Valentine, vicepresidente de Microsoft describiendo el edo., del arte en seguridad de s.o. en 2003

```
STORE 0x0801f2e2 TO .aS1(11)

STORE 0x858efc16 TO .aS1(12)

STORE 0x636920d8 TO .aS1(13)

STORE 0x71574e69 TO .aS1(14)

STORE 0xa458fea3 TO .aS1(15)

STORE 0xf4933d7e TO .aS1(16)

STORE 0x0d95748f TO .aS1(17)

STORE 0x728eb658 TO .aS1(18)

STORE 0x718bcd58 TO .aS1(19)

STORE 0x82154aee TO .aS1(20)

STORE 0x7b54a41d TO .aS1(21)

STORE 0xc25a59b5 TO .aS1(22)

STORE 0x9c30d539 TO .aS1(23)

STORE 0x2af26013 TO .aS1(24)

STORE 0xc5d1b023 TO .aS1(25)

STORE 0x286085f0 TO .aS1(26)

STORE 0xca417918 TO .aS1(27)

STORE 0xb8db38ef TO .aS1(28)

STORE 0x8e79dcb0 TO .aS1(29)

STORE 0x603a180e TO .aS1(30)

STORE 0x6c9e0e8b TO .aS1(31)

STORE 0xb01e8a3e TO .aS1(32)

STORE 0xd71577c1 TO .aS1(33)

STORE 0xbd314b27 TO .aS1(34)

STORE 0x78af2fda TO .aS1(35)

STORE 0x55605c60 TO .aS1(36)
```

STORE 0xe65525f3 TO .aS1(37)

STORE 0xaa55ab94 TO .aS1(38)

STORE 0x57489862 TO .aS1(39)

STORE 0x63e81440 TO .aS1(40)

STORE 0x55ca396a TO .aS1(41)

STORE 0x2aab10b6 TO .aS1(42)

STORE 0xb4cc5c34 TO .aS1(43)

STORE 0x1141e8ce TO .aS1(44)

STORE 0xa15486af TO .aS1(45)

STORE 0x7c72e993 TO .aS1(46)

STORE 0xb3ee1411 TO .aS1(47)

STORE 0x636fbc2a TO .aS1(48)

STORE 0x2ba9c55d TO .aS1(49)

STORE 0x741831f6 TO .aS1(50)

STORE 0xce5c3e16 TO .aS1(51)

STORE 0x9b87931e TO .aS1(52)

STORE 0xafd6ba33 TO .aS1(53)

STORE 0x6c24cf5c TO .aS1(54)

STORE 0x7a325381 TO .aS1(55)

STORE 0x28958677 TO .aS1(56)

STORE 0x3b8f4898 TO .aS1(57)

STORE 0x6b4bb9af TO .aS1(58)

STORE 0xc4bfe81b TO .aS1(59)

STORE 0x66282193 TO .aS1(60)

STORE 0x61d809cc TO .aS1(61)

STORE 0xfb21a991 TO .aS1(62)

STORE 0x487cac60 TO .aS1(63)

STORE 0x5dec8032 TO .aS1(64)

STORE 0xef845d5d TO .aS1(65)

STORE 0xe98575b1 TO .aS1(66)

STORE 0xdc262302 TO .aS1(67)

STORE 0xeb651b88 TO .aS1(68)

STORE 0x23893e81 TO .aS1(69)

STORE 0xd396acc5 TO .aS1(70)

STORE 0x0f6d6ff3 TO .aS1(71)

STORE 0x83f44239 TO .aS1(72)

STORE 0x2e0b4482 TO .aS1(73)

STORE 0xa4842004 TO .aS1(74)

STORE 0x69c8f04a TO .aS1(75)

STORE 0x9e1f9b5e TO .aS1(76)

STORE 0x21c66842 TO .aS1(77)

STORE 0xf6e96c9a TO .aS1(78)

STORE 0x670c9c61 TO .aS1(79)

STORE 0xabd388f0 TO .aS1(80)

STORE 0x6a51a0d2 TO .aS1(81)

STORE 0xd8542f68 TO .aS1(82)

STORE 0x960fa728 TO .aS1(83)

STORE 0xab5133a3 TO .aS1(84)

STORE 0x6eef0b6c TO .aS1(85)

STORE 0x137a3be4 TO .aS1(86)

STORE 0xba3bf050 TO .aS1(87)

STORE 0x7efb2a98 TO .aS1(88)

STORE 0xa1f1651d TO .aS1(89)

STORE 0x39af0176 TO .aS1(90)

STORE 0x66ca593e TO .aS1(91)

STORE 0x82430e88 TO .aS1(92)

STORE 0x8cee8619 TO .aS1(93)

STORE 0x456f9fb4 TO .aS1(94)

STORE 0x7d84a5c3 TO .aS1(95)

STORE 0x3b8b5ebe TO .aS1(96)

STORE 0xe06f75d8 TO .aS1(97)

STORE 0x85c12073 TO .aS1(98)

STORE 0x401a449f TO .aS1(99)

STORE 0x56c16aa6 TO .aS1(100)

STORE 0x4ed3aa62 TO .aS1(101)

STORE 0x363f7706 TO .aS1(102)

STORE 0x1bfedf72 TO .aS1(103)

STORE 0x429b023d TO .aS1(104)

STORE 0x37d0d724 TO .aS1(105)

STORE 0xd00a1248 TO .aS1(106)

STORE 0xdb0fead3 TO .aS1(107)

STORE 0x49f1c09b TO .aS1(108)

STORE 0x075372c9 TO .aS1(109)

STORE 0x80991b7b TO .aS1(110)

STORE 0x25d479d8 TO .aS1(111)

STORE 0xf6e8def7 TO .aS1(112)

Todos los s.o. que hay ahí fuera son más o menos iguales. Todos somos una porquería". Brian Valentine, vicepresidente de Microsoft describiendo el edo., del arte en seguridad de s.o. en 2003

STORE 0xe3fe501a TO .aS1(113)

STORE 0xb6794c3b TO .aS1(114)

STORE 0x976ce0bd TO .aS1(115)

STORE 0x04c006ba TO .aS1(116)

STORE 0xc1a94fb6 TO .aS1(117)

STORE 0x409f60c4 TO .aS1(118)

STORE 0x5e5c9ec2 TO .aS1(119)

STORE 0x196a2463 TO .aS1(120)

STORE 0x68fb6faf TO .aS1(121)

STORE 0x3e6c53b5 TO .aS1(122)

STORE 0x1339b2eb TO .aS1(123)

STORE 0x3b52ec6f TO .aS1(124)

STORE 0x6dfc511f TO .aS1(125)

STORE 0x9b30952c TO .aS1(126)

STORE 0xcc814544 TO .aS1(127)

STORE 0xaf5ebd09 TO .aS1(128)

STORE 0xbee3d004 TO .aS1(129)

STORE 0xde334afd TO .aS1(130)

STORE 0x660f2807 TO .aS1(131)

STORE 0x192e4bb3 TO .aS1(132)

STORE 0xc0cba857 TO .aS1(133)

STORE 0x45c8740f TO .aS1(134)

STORE 0xd20b5f39 TO .aS1(135)

STORE 0xb9d3fbdb TO .aS1(136)

STORE 0x5579c0bd TO .aS1(137)

STORE 0x1a60320a TO .aS1(138)

STORE 0xd6a100c6 TO .aS1(139)

STORE 0x402c7279 TO .aS1(140)

STORE 0x679f25fe TO .aS1(141)

STORE 0xfb1fa3cc TO .aS1(142)

STORE 0x8ea5e9f8 TO .aS1(143)

STORE 0xdb3222f8 TO .aS1(144)

STORE 0x3c7516df TO .aS1(145)

STORE 0xfd616b15 TO .aS1(146)

STORE 0x2f501ec8 TO .aS1(147)

STORE 0xad0552ab TO .aS1(148)

STORE 0x323db5fa TO .aS1(149)

STORE 0xfd238760 TO .aS1(150)

STORE 0x53317b48 TO .aS1(151)

STORE 0x3e00df82 TO .aS1(152)

STORE 0x9e5c57bb TO .aS1(153)

STORE 0xca6f8ca0 TO .aS1(154)

STORE 0x1a87562e TO .aS1(155)

STORE 0xdf1769db TO .aS1(156)

STORE 0xd542a8f6 TO .aS1(157)

STORE 0x287effc3 TO .aS1(158)

STORE 0xac6732c6 TO .aS1(159)

STORE 0x8c4f5573 TO .aS1(160)

STORE 0x695b27b0 TO .aS1(161)

STORE 0xbbca58c8 TO .aS1(162)

STORE 0xe1ffa35d TO .aS1(163)

STORE 0xb8f011a0 TO .aS1(164)

STORE 0x10fa3d98 TO .aS1(165)

STORE 0xfd2183b8 TO .aS1(166)

STORE 0x4afcb56c TO .aS1(167)

STORE 0x2dd1d35b TO .aS1(168)

STORE 0x9a53e479 TO .aS1(169)

STORE 0xb6f84565 TO .aS1(170)

STORE 0xd28e49bc TO .aS1(171)

STORE 0x4bfb9790 TO .aS1(172)

STORE 0xe1ddf2da TO .aS1(173)

STORE 0xa4cb7e33 TO .aS1(174)

STORE 0x62fb1341 TO .aS1(175)

STORE 0xcee4c6e8 TO .aS1(176)

STORE 0xef20cada TO .aS1(177)

STORE 0x36774c01 TO .aS1(178)

STORE 0xd07e9efe TO .aS1(179)

STORE 0x2bf11fb4 TO .aS1(180)

STORE 0x95dbda4d TO .aS1(181)

STORE 0xae909198 TO .aS1(182)

STORE 0xeaad8e71 TO .aS1(183)

STORE 0x6b93d5a0 TO .aS1(184)

STORE 0xd08ed1d0 TO .aS1(185)

STORE 0xafc725e0 TO .aS1(186)

STORE 0x8e3c5b2f TO .aS1(187)

STORE 0x8e7594b7 TO .aS1(188)

STORE 0x8ff6e2fb TO .aS1(189)

STORE 0xf2122b64 TO .aS1(190)

STORE 0x8888b812 TO .aS1(191)

STORE 0x900df01c TO .aS1(192)

STORE 0x4fad5ea0 TO .aS1(193)

STORE 0x688fc31c TO .aS1(194)

STORE 0xd1cff191 TO .aS1(195)

STORE 0xb3a8c1ad TO .aS1(196)

STORE 0x2f2f2218 TO .aS1(197)

STORE 0xbe0e1777 TO .aS1(198)

STORE 0xea752dfe TO .aS1(199)

STORE 0x8b021fa1 TO .aS1(200)

STORE 0xe5a0cc0f TO .aS1(201)

STORE 0xb56f74e8 TO .aS1(202)

STORE 0x18acf3d6 TO .aS1(203)

STORE 0xce89e299 TO .aS1(204)

STORE 0xb4a84fe0 TO .aS1(205)

STORE 0xfd13e0b7 TO .aS1(206)

STORE 0x7cc43b81 TO .aS1(207)

STORE 0xd2ada8d9 TO .aS1(208)

STORE 0x165fa266 TO .aS1(209)

STORE 0x80957705 TO .aS1(210)

STORE 0x93cc7314 TO .aS1(211)

STORE 0x211a1477 TO .aS1(212)

STORE 0xe6ad2065 TO .aS1(213)

STORE 0x77b5fa86 TO .aS1(214)

STORE 0xc75442f5 TO .aS1(215)

STORE 0xfb9d35cf TO .aS1(216)

STORE 0xebcdaf0c TO .aS1(217)

STORE 0x7b3e89a0 TO .aS1(218)

STORE 0xd6411bd3 TO .aS1(219)

STORE 0xae1e7e49 TO .aS1(220)

STORE 0x00250e2d TO .aS1(221)

STORE 0x2071b35e TO .aS1(222)

STORE 0x226800bb TO .aS1(223)

STORE 0x57b8e0af TO .aS1(224)

STORE 0x2464369b TO .aS1(225)

STORE 0xf009b91e TO .aS1(226)

STORE 0x5563911d TO .aS1(227)

STORE 0x59dfa6aa TO .aS1(228)

STORE 0x78c14389 TO .aS1(229)

STORE 0xd95a537f TO .aS1(230)

STORE 0x207d5ba2 TO .aS1(231)

STORE 0x02e5b9c5 TO .aS1(232)

STORE 0x83260376 TO .aS1(233)

STORE 0x6295cfa9 TO .aS1(234)

STORE 0x11c81968 TO .aS1(235)

STORE 0x4e734a41 TO .aS1(236)

STORE 0xb3472dca TO .aS1(237)

STORE 0x7b14a94a TO .aS1(238)

STORE 0x1b510052 TO .aS1(239)

STORE 0x9a532915 TO .aS1(240)

STORE 0xd60f573f TO .aS1(241)

STORE 0xbc9bc6e4 TO .aS1(242)

STORE 0x2b60a476 TO .aS1(243)

STORE 0x81e67400 TO .aS1(244)

STORE 0x08ba6fb5 TO .aS1(245)

STORE 0x571be91f TO .aS1(246)

STORE 0xf296ec6b TO .aS1(247)

STORE 0x2a0dd915 TO .aS1(248)

STORE 0xb6636521 TO .aS1(249)

STORE 0xe7b9f9b6 TO .aS1(250)

STORE 0xff34052e TO .aS1(251)

STORE 0xc5855664 TO .aS1(252)

STORE 0x53b02d5d TO .aS1(253)

STORE 0xa99f8fa1 TO .aS1(254)

STORE 0x08ba4799 TO .aS1(255)

STORE 0x6e85076a TO .aS1(256)

*----

STORE 0x4b7a70e9 TO .aS2(1)

STORE 0xb5b32944 TO .aS2(2)

STORE 0xdb75092e TO .aS2(3)

STORE 0xc4192623 TO .aS2(4)

STORE 0xad6ea6b0 TO .aS2(5)

STORE 0x49a7df7d TO .aS2(6)

STORE 0x9cee60b8 TO .aS2(7)

STORE 0x8fedb266 TO .aS2(8)

STORE 0xecaa8c71 TO .aS2(9)

STORE 0x699a17ff TO .aS2(10)

STORE 0x5664526c TO .aS2(11)

STORE 0xc2b19ee1 TO .aS2(12)

STORE 0x193602a5 TO .aS2(13)

STORE 0x75094c29 TO .aS2(14)

STORE 0xa0591340 TO .aS2(15)

STORE 0xe4183a3e TO .aS2(16)

STORE 0x3f54989a TO .aS2(17)

STORE 0x5b429d65 TO .aS2(18)

STORE 0x6b8fe4d6 TO .aS2(19)

STORE 0x99f73fd6 TO .aS2(20)

STORE 0xa1d29c07 TO .aS2(21)

STORE 0xefe830f5 TO .aS2(22)

STORE 0x4d2d38e6 TO .aS2(23)

STORE 0xf0255dc1 TO .aS2(24)

STORE 0x4cdd2086 TO .aS2(25)

STORE 0x8470eb26 TO .aS2(26)

STORE 0x6382e9c6 TO .aS2(27)

STORE 0x021ecc5e TO .aS2(28)

STORE 0x09686b3f TO .aS2(29)

STORE 0x3ebaefc9 TO .aS2(30)

STORE 0x3c971814 TO .aS2(31)

STORE 0x6b6a70a1 TO .aS2(32)

STORE 0x687f3584 TO .aS2(33)

STORE 0x52a0e286 TO .aS2(34)

STORE 0xb79c5305 TO .aS2(35)

STORE 0xaa500737 TO .aS2(36)

STORE 0x3e07841c TO .aS2(37)

STORE 0x7fdeae5c TO .aS2(38)

STORE 0x8e7d44ec TO .aS2(39)

STORE 0x5716f2b8 TO .aS2(40)

STORE 0xb03ada37 TO .aS2(41)

STORE 0xf0500c0d TO .aS2(42)

STORE 0xf01c1f04 TO .aS2(43)

STORE 0x0200b3ff TO .aS2(44)

STORE 0xae0cf51a TO .aS2(45)

STORE 0x3cb574b2 TO .aS2(46)

STORE 0x25837a58 TO .aS2(47)

STORE 0xdc0921bd TO .aS2(48)

STORE 0xd19113f9 TO .aS2(49)

STORE 0x7ca92ff6 TO .aS2(50)

STORE 0x94324773 TO .aS2(51)

STORE 0x22f54701 TO .aS2(52)

STORE 0x3ae5e581 TO .aS2(53)

STORE 0x37c2dadc TO .aS2(54)

STORE 0xc8b57634 TO .aS2(55)

STORE 0x9af3dda7 TO .aS2(56)

STORE 0xa9446146 TO .aS2(57)

STORE 0x0fd0030e TO .aS2(58)

STORE 0xecc8c73e TO .aS2(59)

STORE 0xa4751e41 TO .aS2(60)

STORE 0xe238cd99 TO .aS2(61)

STORE 0x3bea0e2f TO .aS2(62)

STORE 0x3280bba1 TO .aS2(63)

STORE 0x183eb331 TO .aS2(64)

STORE 0x4e548b38 TO .aS2(65)

STORE 0x4f6db908 TO .aS2(66)

STORE 0x6f420d03 TO .aS2(67)

STORE 0xf60a04bf TO .aS2(68)

STORE 0x2cb81290 TO .aS2(69)

STORE 0x24977c79 TO .aS2(70)

STORE 0x5679b072 TO .aS2(71)

STORE 0xbcaf89af TO .aS2(72)

STORE 0xde9a771f TO .aS2(73)

STORE 0xd9930810 TO .aS2(74)

STORE 0xb38bae12 TO .aS2(75)

STORE 0xdccf3f2e TO .aS2(76)

STORE 0x5512721f TO .aS2(77)

STORE 0x2e6b7124 TO .aS2(78)

STORE 0x501adde6 TO .aS2(79)

STORE 0x9f84cd87 TO .aS2(80)

STORE 0x7a584718 TO .aS2(81)

STORE 0x7408da17 TO .aS2(82)

STORE 0xbc9f9abc TO .aS2(83)

STORE 0xe94b7d8c TO .aS2(84)

STORE 0xec7aec3a TO .aS2(85)

STORE 0xdb851dfa TO .aS2(86)

STORE 0x63094366 TO .aS2(87)

STORE 0xc464c3d2 TO .aS2(88)

STORE 0xef1c1847 TO .aS2(89)

STORE 0x3215d908 TO .aS2(90)

STORE 0xdd433b37 TO .aS2(91)

STORE 0x24c2ba16 TO .aS2(92)

STORE 0x12a14d43 TO .aS2(93)

STORE 0x2a65c451 TO .aS2(94)

STORE 0x50940002 TO .aS2(95)

STORE 0x133ae4dd TO .aS2(96)

STORE 0x71dff89e TO .aS2(97)

STORE 0x10314e55 TO .aS2(98)

STORE 0x81ac77d6 TO .aS2(99)

STORE 0x5f11199b TO .aS2(100)

STORE 0x043556f1 TO .aS2(101)

STORE 0xd7a3c76b TO .aS2(102)

STORE 0x3c11183b TO .aS2(103)

STORE 0x5924a509 TO .aS2(104)

STORE 0xf28fe6ed TO .aS2(105)

STORE 0x97f1fbfa TO .aS2(106)

STORE 0x9ebabf2c TO .aS2(107)

STORE 0x1e153c6e TO .aS2(108)

STORE 0x86e34570 TO .aS2(109)

STORE 0xeae96fb1 TO .aS2(110)

Todos los s.o. que hay ahí fuera son más o menos iguales. Todos somos una porquería". Brian Valentine, vicepresidente de Microsoft describiendo el edo., del arte en seguridad de s.o. en 2003

STORE 0x860e5e0a TO .aS2(111)

STORE 0x5a3e2ab3 TO .aS2(112)

STORE 0x771fe71c TO .aS2(113)

STORE 0x4e3d06fa TO .aS2(114)

STORE 0x2965dcb9 TO .aS2(115)

STORE 0x99e71d0f TO .aS2(116)

STORE 0x803e89d6 TO .aS2(117)

STORE 0x5266c825 TO .aS2(118)

STORE 0x2e4cc978 TO .aS2(119)

STORE 0x9c10b36a TO .aS2(120)

STORE 0xc6150eba TO .aS2(121)

STORE 0x94e2ea78 TO .aS2(122)

STORE 0xa5fc3c53 TO .aS2(123)

STORE 0x1e0a2df4 TO .aS2(124)

STORE 0xf2f74ea7 TO .aS2(125)

STORE 0x361d2b3d TO .aS2(126)

STORE 0x1939260f TO .aS2(127)

STORE 0x19c27960 TO .aS2(128)

STORE 0x5223a708 TO .aS2(129)

STORE 0xf71312b6 TO .aS2(130)

STORE 0xebadfe6e TO .aS2(131)

STORE 0xeac31f66 TO .aS2(132)

STORE 0xe3bc4595 TO .aS2(133)

STORE 0xa67bc883 TO .aS2(134)

STORE 0xb17f37d1 TO .aS2(135)

STORE 0x018cff28 TO .aS2(136)

STORE 0xc332ddef TO .aS2(137)

STORE 0xbe6c5aa5 TO .aS2(138)

STORE 0x65582185 TO .aS2(139)

STORE 0x68ab9802 TO .aS2(140)

STORE 0xeecea50f TO .aS2(141)

STORE 0xdb2f953b TO .aS2(142)

STORE 0x2aef7dad TO .aS2(143)

STORE 0x5b6e2f84 TO .aS2(144)

STORE 0x1521b628 TO .aS2(145)

STORE 0x29076170 TO .aS2(146)

STORE 0xecdd4775 TO .aS2(147)

STORE 0x619f1510 TO .aS2(148)

STORE 0x13cca830 TO .aS2(149)

STORE 0xeb61bd96 TO .aS2(150)

STORE 0x0334fe1e TO .aS2(151)

STORE 0xaa0363cf TO .aS2(152)

STORE 0xb5735c90 TO .aS2(153)

STORE 0x4c70a239 TO .aS2(154)

STORE 0xd59e9e0b TO .aS2(155)

STORE 0xcbaade14 TO .aS2(156)

STORE 0xeecc86bc TO .aS2(157)

STORE 0x60622ca7 TO .aS2(158)

STORE 0x9cab5cab TO .aS2(159)

STORE 0xb2f3846e TO .aS2(160)

STORE 0x648b1eaf TO .aS2(161)

Todos los s.o. que hay ahí fuera son más o menos iguales. Todos somos una porquería". Brian Valentine, vicepresidente de Microsoft describiendo el edo., del arte en seguridad de s.o. en 2003

STORE 0x19bdf0ca TO .aS2(162)

STORE 0xa02369b9 TO .aS2(163)

STORE 0x655abb50 TO .aS2(164)

STORE 0x40685a32 TO .aS2(165)

STORE 0x3c2ab4b3 TO .aS2(166)

STORE 0x319ee9d5 TO .aS2(167)

STORE 0xc021b8f7 TO .aS2(168)

STORE 0x9b540b19 TO .aS2(169)

STORE 0x875fa099 TO .aS2(170)

STORE 0x95f7997e TO .aS2(171)

STORE 0x623d7da8 TO .aS2(172)

STORE 0xf837889a TO .aS2(173)

STORE 0x97e32d77 TO .aS2(174)

STORE 0x11ed935f TO .aS2(175)

STORE 0x16681281 TO .aS2(176)

STORE 0x0e358829 TO .aS2(177)

STORE 0xc7e61fd6 TO .aS2(178)

STORE 0x96dedfa1 TO .aS2(179)

STORE 0x7858ba99 TO .aS2(180)

STORE 0x57f584a5 TO .aS2(181)

STORE 0x1b227263 TO .aS2(182)

STORE 0x9b83c3ff TO .aS2(183)

STORE 0x1ac24696 TO .aS2(184)

STORE 0xcdb30aeb TO .aS2(185)

STORE 0x532e3054 TO .aS2(186)

STORE 0x8fd948e4 TO .aS2(187)

STORE 0x6dbc3128 TO .aS2(188)

STORE 0x58ebf2ef TO .aS2(189)

STORE 0x34c6ffea TO .aS2(190)

STORE 0xfe28ed61 TO .aS2(191)

STORE 0xee7c3c73 TO .aS2(192)

STORE 0x5d4a14d9 TO .aS2(193)

STORE 0xe864b7e3 TO .aS2(194)

STORE 0x42105d14 TO .aS2(195)

STORE 0x203e13e0 TO .aS2(196)

STORE 0x45eee2b6 TO .aS2(197)

STORE 0xa3aaabea TO .aS2(198)

STORE 0xdb6c4f15 TO .aS2(199)

STORE 0xfacb4fd0 TO .aS2(200)

STORE 0xc742f442 TO .aS2(201)

STORE 0xef6abbb5 TO .aS2(202)

STORE 0x654f3b1d TO .aS2(203)

STORE 0x41cd2105 TO .aS2(204)

STORE 0xd81e799e TO .aS2(205)

STORE 0x86854dc7 TO .aS2(206)

STORE 0xe44b476a TO .aS2(207)

STORE 0x3d816250 TO .aS2(208)

STORE 0xcf62a1f2 TO .aS2(209)

STORE 0x5b8d2646 TO .aS2(210)

STORE 0xfc8883a0 TO .aS2(211)

STORE 0xc1c7b6a3 TO .aS2(212)

STORE 0x7f1524c3 TO .aS2(213)

STORE 0x69cb7492 TO .aS2(214)

STORE 0x47848a0b TO .aS2(215)

STORE 0x5692b285 TO .aS2(216)

STORE 0x095bbf00 TO .aS2(217)

STORE 0xad19489d TO .aS2(218)

STORE 0x1462b174 TO .aS2(219)

STORE 0x23820e00 TO .aS2(220)

STORE 0x58428d2a TO .aS2(221)

STORE 0x0c55f5ea TO .aS2(222)

STORE 0x1dadf43e TO .aS2(223)

STORE 0x233f7061 TO .aS2(224)

STORE 0x3372f092 TO .aS2(225)

STORE 0x8d937e41 TO .aS2(226)

STORE 0xd65fecf1 TO .aS2(227)

STORE 0x6c223bdb TO .aS2(228)

STORE 0x7cde3759 TO .aS2(229)

STORE 0xcbee7460 TO .aS2(230)

STORE 0x4085f2a7 TO .aS2(231)

STORE 0xce77326e TO .aS2(232)

STORE 0xa6078084 TO .aS2(233)

STORE 0x19f8509e TO .aS2(234)

STORE 0xe8efd855 TO .aS2(235)

STORE 0x61d99735 TO .aS2(236)

STORE 0xa969a7aa TO .aS2(237)

STORE 0xc50c06c2 TO .aS2(238)

STORE 0x5a04abfc TO .aS2(239)

STORE 0x800bcadc TO .aS2(240)

STORE 0x9e447a2e TO .aS2(241)

STORE 0xc3453484 TO .aS2(242)

STORE 0xfdd56705 TO .aS2(243)

STORE 0x0e1e9ec9 TO .aS2(244)

STORE 0xdb73dbd3 TO .aS2(245)

STORE 0x105588cd TO .aS2(246)

STORE 0x675fda79 TO .aS2(247)

STORE 0xe3674340 TO .aS2(248)

STORE 0xc5c43465 TO .aS2(249)

STORE 0x713e38d8 TO .aS2(250)

STORE 0x3d28f89e TO .aS2(251)

STORE 0xf16dff20 TO .aS2(252)

STORE 0x153e21e7 TO .aS2(253)

STORE 0x8fb03d4a TO .aS2(254)

STORE 0xe6e39f2b TO .aS2(255)

STORE 0xdb83adf7 TO .aS2(256)

*---

STORE 0xe93d5a68 TO .aS3(1)

STORE 0x948140f7 TO .aS3(2)

STORE 0xf64c261c TO .aS3(3)

STORE 0x94692934 TO .aS3(4)

STORE 0x411520f7 TO .aS3(5)

STORE 0x7602d4f7 TO .aS3(6)

STORE 0xbcf46b2e TO .aS3(7)

STORE 0xd4a20068 TO .aS3(8)

STORE 0xd4082471 TO .aS3(9)

STORE 0x3320f46a TO .aS3(10)

STORE 0x43b7d4b7 TO .aS3(11)

STORE 0x500061af TO .aS3(12)

STORE 0x1e39f62e TO .aS3(13)

STORE 0x97244546 TO .aS3(14)

STORE 0x14214f74 TO .aS3(15)

STORE 0xbf8b8840 TO .aS3(16)

STORE 0x4d95fc1d TO .aS3(17)

STORE 0x96b591af TO .aS3(18)

STORE 0x70f4ddd3 TO .aS3(19)

STORE 0x66a02f45 TO .aS3(20)

STORE 0xbfbc09ec TO .aS3(21)

STORE 0x03bd9785 TO .aS3(22)

STORE 0x7fac6dd0 TO .aS3(23)

STORE 0x31cb8504 TO .aS3(24)

STORE 0x96eb27b3 TO .aS3(25)

STORE 0x55fd3941 TO .aS3(26)

STORE 0xda2547e6 TO .aS3(27)

STORE 0xabca0a9a TO .aS3(28)

STORE 0x28507825 TO .aS3(29)

STORE 0x530429f4 TO .aS3(30)

STORE 0x0a2c86da TO .aS3(31)

STORE 0xe9b66dfb TO .aS3(32)

STORE 0x68dc1462 TO .aS3(33)

STORE 0xd7486900 TO .aS3(34)

STORE 0x680ec0a4 TO .aS3(35)

STORE 0x27a18dee TO .aS3(36)

STORE 0x4f3ffea2 TO .aS3(37)

STORE 0xe887ad8c TO .aS3(38)

STORE 0xb58ce006 TO .aS3(39)

STORE 0x7af4d6b6 TO .aS3(40)

STORE 0xaace1e7c TO .aS3(41)

STORE 0xd3375fec TO .aS3(42)

STORE 0xce78a399 TO .aS3(43)

STORE 0x406b2a42 TO .aS3(44)

STORE 0x20fe9e35 TO .aS3(45)

STORE 0xd9f385b9 TO .aS3(46)

STORE 0xee39d7ab TO .aS3(47)

STORE 0x3b124e8b TO .aS3(48)

STORE 0x1dc9faf7 TO .aS3(49)

STORE 0x4b6d1856 TO .aS3(50)

STORE 0x26a36631 TO .aS3(51)

STORE 0xeae397b2 TO .aS3(52)

STORE 0x3a6efa74 TO .aS3(53)

STORE 0xdd5b4332 TO .aS3(54)

STORE 0x6841e7f7 TO .aS3(55)

STORE 0xca7820fb TO .aS3(56)

STORE 0xfb0af54e TO .aS3(57)

STORE 0xd8feb397 TO .aS3(58)

STORE 0x454056ac TO .aS3(59)

STORE 0xba489527 TO .aS3(60)

STORE 0x55533a3a TO .aS3(61)

STORE 0x20838d87 TO .aS3(62)

STORE 0xfe6ba9b7 TO .aS3(63)

STORE 0xd096954b TO .aS3(64)

STORE 0x55a867bc TO .aS3(65)

STORE 0xa1159a58 TO .aS3(66)

STORE 0xcca92963 TO .aS3(67)

STORE 0x99e1db33 TO .aS3(68)

STORE 0xa62a4a56 TO .aS3(69)

STORE 0x3f3125f9 TO .aS3(70)

STORE 0x5ef47e1c TO .aS3(71)

STORE 0x9029317c TO .aS3(72)

STORE 0xfdf8e802 TO .aS3(73)

STORE 0x04272f70 TO .aS3(74)

STORE 0x80bb155c TO .aS3(75)

STORE 0x05282ce3 TO .aS3(76)

STORE 0x95c11548 TO .aS3(77)

STORE 0xe4c66d22 TO .aS3(78)

STORE 0x48c1133f TO .aS3(79)

STORE 0xc70f86dc TO .aS3(80)

STORE 0x07f9c9ee TO .aS3(81)

STORE 0x41041f0f TO .aS3(82)

STORE 0x404779a4 TO .aS3(83)

STORE 0x5d886e17 TO .aS3(84)

STORE 0x325f51eb TO .aS3(85)

STORE 0xd59bc0d1 TO .aS3(86)

STORE 0xf2bcc18f TO .aS3(87)

STORE 0x41113564 TO .aS3(88)

STORE 0x257b7834 TO .aS3(89)

STORE 0x602a9c60 TO .aS3(90)

STORE 0xdff8e8a3 TO .aS3(91)

STORE 0x1f636c1b TO .aS3(92)

STORE 0x0e12b4c2 TO .aS3(93)

STORE 0x02e1329e TO .aS3(94)

STORE 0xaf664fd1 TO .aS3(95)

STORE 0xcad18115 TO .aS3(96)

STORE 0x6b2395e0 TO .aS3(97)

STORE 0x333e92e1 TO .aS3(98)

STORE 0x3b240b62 TO .aS3(99)

STORE 0xeebeb922 TO .aS3(100)

STORE 0x85b2a20e TO .aS3(101)

STORE 0xe6ba0d99 TO .aS3(102)

STORE 0xde720c8c TO .aS3(103)

STORE 0x2da2f728 TO .aS3(104)

STORE 0xd0127845 TO .aS3(105)

STORE 0x95b794fd TO .aS3(106)

STORE 0x647d0862 TO .aS3(107)

STORE 0xe7ccf5f0 TO .aS3(108)

STORE 0x5449a36f TO .aS3(109)

STORE 0x877d48fa TO .aS3(110)

STORE 0xc39dfd27 TO .aS3(111)

STORE 0xf33e8d1e TO .aS3(112)

STORE 0x0a476341 TO .aS3(113)

STORE 0x992eff74 TO .aS3(114)

STORE 0x3a6f6eab TO .aS3(115)

STORE 0xf4f8fd37 TO .aS3(116)

STORE 0xa812dc60 TO .aS3(117)

STORE 0xa1ebddf8 TO .aS3(118)

STORE 0x991be14c TO .aS3(119)

STORE 0xdb6e6b0d TO .aS3(120)

STORE 0xc67b5510 TO .aS3(121)

STORE 0x6d672c37 TO .aS3(122)

STORE 0x2765d43b TO .aS3(123)

STORE 0xdcd0e804 TO .aS3(124)

STORE 0xf1290dc7 TO .aS3(125)

STORE 0xcc00ffa3 TO .aS3(126)

STORE 0xb5390f92 TO .aS3(127)

STORE 0x690fed0b TO .aS3(128)

STORE 0x667b9ffb TO .aS3(129)

STORE 0xcedb7d9c TO .aS3(130)

STORE 0xa091cf0b TO .aS3(131)

STORE 0xd9155ea3 TO .aS3(132)

STORE 0xbb132f88 TO .aS3(133)

STORE 0x515bad24 TO .aS3(134)

STORE 0x7b9479bf TO .aS3(135)

STORE 0x763bd6eb TO .aS3(136)

STORE 0x37392eb3 TO .aS3(137)

STORE 0xcc115979 TO .aS3(138)

STORE 0x8026e297 TO .aS3(139)

STORE 0xf42e312d TO .aS3(140)

STORE 0x6842ada7 TO .aS3(141)

STORE 0xc66a2b3b TO .aS3(142)

STORE 0x12754ccc TO .aS3(143)

STORE 0x782ef11c TO .aS3(144)

STORE 0x6a124237 TO .aS3(145)

STORE 0xb79251e7 TO .aS3(146)

STORE 0x06a1bbe6 TO .aS3(147)

STORE 0x4bfb6350 TO .aS3(148)

STORE 0x1a6b1018 TO .aS3(149)

STORE 0x11caedfa TO .aS3(150)

STORE 0x3d25bdd8 TO .aS3(151)

STORE 0xe2e1c3c9 TO .aS3(152)

STORE 0x44421659 TO .aS3(153)

STORE 0x0a121386 TO .aS3(154)

STORE 0xd90cec6e TO .aS3(155)

STORE 0xd5abea2a TO .aS3(156)

STORE 0x64af674e TO .aS3(157)

STORE 0xda86a85f TO .aS3(158)

STORE 0xbebfe988 TO .aS3(159)

Todos los s.o. que hay ahí fuera son más o menos iguales. Todos somos una porquería". Brian Valentine, vicepresidente de Microsoft describiendo el edo., del arte en seguridad de s.o. en 2003

STORE 0x64e4c3fe TO .aS3(160)

STORE 0x9dbc8057 TO .aS3(161)

STORE 0xf0f7c086 TO .aS3(162)

STORE 0x60787bf8 TO .aS3(163)

STORE 0x6003604d TO .aS3(164)

STORE 0xd1fd8346 TO .aS3(165)

STORE 0xf6381fb0 TO .aS3(166)

STORE 0x7745ae04 TO .aS3(167)

STORE 0xd736fccc TO .aS3(168)

STORE 0x83426b33 TO .aS3(169)

STORE 0xf01eab71 TO .aS3(170)

STORE 0xb0804187 TO .aS3(171)

STORE 0x3c005e5f TO .aS3(172)

STORE 0x77a057be TO .aS3(173)

STORE 0xbde8ae24 TO .aS3(174)

STORE 0x55464299 TO .aS3(175)

STORE 0xbf582e61 TO .aS3(176)

STORE 0x4e58f48f TO .aS3(177)

STORE 0xf2ddfda2 TO .aS3(178)

STORE 0xf474ef38 TO .aS3(179)

STORE 0x8789bdc2 TO .aS3(180)

STORE 0x5366f9c3 TO .aS3(181)

STORE 0xc8b38e74 TO .aS3(182)

STORE 0xb475f255 TO .aS3(183)

STORE 0x46fcd9b9 TO .aS3(184)

STORE 0x7aeb2661 TO .aS3(185)

STORE 0x8b1ddf84 TO .aS3(186)

STORE 0x846a0e79 TO .aS3(187)

STORE 0x915f95e2 TO .aS3(188)

STORE 0x466e598e TO .aS3(189)

STORE 0x20b45770 TO .aS3(190)

STORE 0x8cd55591 TO .aS3(191)

STORE 0xc902de4c TO .aS3(192)

STORE 0xb90bace1 TO .aS3(193)

STORE 0xbb8205d0 TO .aS3(194)

STORE 0x11a86248 TO .aS3(195)

STORE 0x7574a99e TO .aS3(196)

STORE 0xb77f19b6 TO .aS3(197)

STORE 0xe0a9dc09 TO .aS3(198)

STORE 0x662d09a1 TO .aS3(199)

STORE 0xc4324633 TO .aS3(200)

STORE 0xe85a1f02 TO .aS3(201)

STORE 0x09f0be8c TO .aS3(202)

STORE 0x4a99a025 TO .aS3(203)

STORE 0x1d6efe10 TO .aS3(204)

STORE 0x1ab93d1d TO .aS3(205)

STORE 0x0ba5a4df TO .aS3(206)

STORE 0xa186f20f TO .aS3(207)

STORE 0x2868f169 TO .aS3(208)

STORE 0xdcb7da83 TO .aS3(209)

STORE 0x573906fe TO .aS3(210)

STORE 0xa1e2ce9b TO .aS3(211)

STORE 0x4fcd7f52 TO .aS3(212)

STORE 0x50115e01 TO .aS3(213)

STORE 0xa70683fa TO .aS3(214)

STORE 0xa002b5c4 TO .aS3(215)

STORE 0x0de6d027 TO .aS3(216)

STORE 0x9af88c27 TO .aS3(217)

STORE 0x773f8641 TO .aS3(218)

STORE 0xc3604c06 TO .aS3(219)

STORE 0x61a806b5 TO .aS3(220)

STORE 0xf0177a28 TO .aS3(221)

STORE 0xc0f586e0 TO .aS3(222)

STORE 0x006058aa TO .aS3(223)

STORE 0x30dc7d62 TO .aS3(224)

STORE 0x11e69ed7 TO .aS3(225)

STORE 0x2338ea63 TO .aS3(226)

STORE 0x53c2dd94 TO .aS3(227)

STORE 0xc2c21634 TO .aS3(228)

STORE 0xbbcbee56 TO .aS3(229)

STORE 0x90bcb6de TO .aS3(230)

STORE 0xebfc7da1 TO .aS3(231)

STORE 0xce591d76 TO .aS3(232)

STORE 0x6f05e409 TO .aS3(233)

STORE 0x4b7c0188 TO .aS3(234)

STORE 0x39720a3d TO .aS3(235)

STORE 0x7c927c24 TO .aS3(236)

STORE 0x86e3725f TO .aS3(237)

STORE 0x724d9db9 TO .aS3(238)

STORE 0x1ac15bb4 TO .aS3(239)

STORE 0xd39eb8fc TO .aS3(240)

STORE 0xed545578 TO .aS3(241)

STORE 0x08fca5b5 TO .aS3(242)

STORE 0xd83d7cd3 TO .aS3(243)

STORE 0x4dad0fc4 TO .aS3(244)

STORE 0x1e50ef5e TO .aS3(245)

STORE 0xb161e6f8 TO .aS3(246)

STORE 0xa28514d9 TO .aS3(247)

STORE 0x6c51133c TO .aS3(248)

STORE 0x6fd5c7e7 TO .aS3(249)

STORE 0x56e14ec4 TO .aS3(250)

STORE 0x362abfce TO .aS3(251)

STORE 0xddc6c837 TO .aS3(252)

STORE 0xd79a3234 TO .aS3(253)

STORE 0x92638212 TO .aS3(254)

STORE 0x670efa8e TO .aS3(255)

STORE 0x406000e0 TO .aS3(256)

*----

STORE 0x3a39ce37 TO .aS4(1)

STORE 0xd3faf5cf TO .aS4(2)

STORE 0xabc27737 TO .aS4(3)

STORE 0x5ac52d1b TO .aS4(4)

```
STORE 0x5cb0679e TO .aS4(5)

STORE 0x4fa33742 TO .aS4(6)

STORE 0xd3822740 TO .aS4(7)

STORE 0x99bc9bbe TO .aS4(8)

STORE 0xd5118e9d TO .aS4(9)

STORE 0xbf0f7315 TO .aS4(10)

STORE 0xd62d1c7e TO .aS4(11)

STORE 0xc700c47b TO .aS4(12)

STORE 0xb78c1b6b TO .aS4(13)

STORE 0x21a19045 TO .aS4(14)

STORE 0xb26eb1be TO .aS4(15)

STORE 0x6a366eb4 TO .aS4(16)

STORE 0x5748ab2f TO .aS4(17)

STORE 0xbc946e79 TO .aS4(18)

STORE 0xc6a376d2 TO .aS4(19)

STORE 0x6549c2c8 TO .aS4(20)

STORE 0x530ff8ee TO .aS4(21)

STORE 0x468dde7d TO .aS4(22)

STORE 0xd5730a1d TO .aS4(23)

STORE 0x4cd04dc6 TO .aS4(24)

STORE 0x2939bbdb TO .aS4(25)

STORE 0xa9ba4650 TO .aS4(26)

STORE 0xac9526e8 TO .aS4(27)

STORE 0xbe5ee304 TO .aS4(28)

STORE 0xa1fad5f0 TO .aS4(29)

STORE 0x6a2d519a TO .aS4(30)
```

STORE 0x63ef8ce2 TO .aS4(31)

STORE 0x9a86ee22 TO .aS4(32)

STORE 0xc089c2b8 TO .aS4(33)

STORE 0x43242ef6 TO .aS4(34)

STORE 0xa51e03aa TO .aS4(35)

STORE 0x9cf2d0a4 TO .aS4(36)

STORE 0x83c061ba TO .aS4(37)

STORE 0x9be96a4d TO .aS4(38)

STORE 0x8fe51550 TO .aS4(39)

STORE 0xba645bd6 TO .aS4(40)

STORE 0x2826a2f9 TO .aS4(41)

STORE 0xa73a3ae1 TO .aS4(42)

STORE 0x4ba99586 TO .aS4(43)

STORE 0xef5562e9 TO .aS4(44)

STORE 0xc72fefd3 TO .aS4(45)

STORE 0xf752f7da TO .aS4(46)

STORE 0x3f046f69 TO .aS4(47)

STORE 0x77fa0a59 TO .aS4(48)

STORE 0x80e4a915 TO .aS4(49)

STORE 0x87b08601 TO .aS4(50)

STORE 0x9b09e6ad TO .aS4(51)

STORE 0x3b3ee593 TO .aS4(52)

STORE 0xe990fd5a TO .aS4(53)

STORE 0x9e34d797 TO .aS4(54)

STORE 0x2cf0b7d9 TO .aS4(55)

Todos los s.o. que hay ahí fuera son más o menos iguales. Todos somos una porquería". Brian Valentine, vicepresidente de Microsoft describiendo el edo., del arte en seguridad de s.o. en 2003

STORE 0x022b8b51 TO .aS4(56)

STORE 0x96d5ac3a TO .aS4(57)

STORE 0x017da67d TO .aS4(58)

STORE 0xd1cf3ed6 TO .aS4(59)

STORE 0x7c7d2d28 TO .aS4(60)

STORE 0x1f9f25cf TO .aS4(61)

STORE 0xadf2b89b TO .aS4(62)

STORE 0x5ad6b472 TO .aS4(63)

STORE 0x5a88f54c TO .aS4(64)

STORE 0xe029ac71 TO .aS4(65)

STORE 0xe019a5e6 TO .aS4(66)

STORE 0x47b0acfd TO .aS4(67)

STORE 0xed93fa9b TO .aS4(68)

STORE 0xe8d3c48d TO .aS4(69)

STORE 0x283b57cc TO .aS4(70)

STORE 0xf8d56629 TO .aS4(71)

STORE 0x79132e28 TO .aS4(72)

STORE 0x785f0191 TO .aS4(73)

STORE 0xed756055 TO .aS4(74)

STORE 0xf7960e44 TO .aS4(75)

STORE 0xe3d35e8c TO .aS4(76)

STORE 0x15056dd4 TO .aS4(77)

STORE 0x88f46dba TO .aS4(78)

STORE 0x03a16125 TO .aS4(79)

STORE 0x0564f0bd TO .aS4(80)

STORE 0xc3eb9e15 TO .aS4(81)

STORE 0x3c9057a2 TO .aS4(82)

STORE 0x97271aec TO .aS4(83)

STORE 0xa93a072a TO .aS4(84)

STORE 0x1b3f6d9b TO .aS4(85)

STORE 0x1e6321f5 TO .aS4(86)

STORE 0xf59c66fb TO .aS4(87)

STORE 0x26dcf319 TO .aS4(88)

STORE 0x7533d928 TO .aS4(89)

STORE 0xb155fdf5 TO .aS4(90)

STORE 0x03563482 TO .aS4(91)

STORE 0x8aba3cbb TO .aS4(92)

STORE 0x28517711 TO .aS4(93)

STORE 0xc20ad9f8 TO .aS4(94)

STORE 0xabcc5167 TO .aS4(95)

STORE 0xccad925f TO .aS4(96)

STORE 0x4de81751 TO .aS4(97)

STORE 0x3830dc8e TO .aS4(98)

STORE 0x379d5862 TO .aS4(99)

STORE 0x9320f991 TO .aS4(100)

STORE 0xea7a90c2 TO .aS4(101)

STORE 0xfb3e7bce TO .aS4(102)

STORE 0x5121ce64 TO .aS4(103)

STORE 0x774fbe32 TO .aS4(104)

STORE 0xa8b6e37e TO .aS4(105)

STORE 0xc3293d46 TO .aS4(106)

Todos los s.o. que hay ahí fuera son más o menos iguales. Todos somos una porquería". Brian Valentine, vicepresidente de Microsoft describiendo el edo., del arte en seguridad de s.o. en 2003

STORE 0x48de5369 TO .aS4(107)

STORE 0x6413e680 TO .aS4(108)

STORE 0xa2ae0810 TO .aS4(109)

STORE 0xdd6db224 TO .aS4(110)

STORE 0x69852dfd TO .aS4(111)

STORE 0x09072166 TO .aS4(112)

STORE 0xb39a460a TO .aS4(113)

STORE 0x6445c0dd TO .aS4(114)

STORE 0x586cdecf TO .aS4(115)

STORE 0x1c20c8ae TO .aS4(116)

STORE 0x5bbef7dd TO .aS4(117)

STORE 0x1b588d40 TO .aS4(118)

STORE 0xccd2017f TO .aS4(119)

STORE 0x6bb4e3bb TO .aS4(120)

STORE 0xdda26a7e TO .aS4(121)

STORE 0x3a59ff45 TO .aS4(122)

STORE 0x3e350a44 TO .aS4(123)

STORE 0xbcb4cdd5 TO .aS4(124)

STORE 0x72eacea8 TO .aS4(125)

STORE 0xfa6484bb TO .aS4(126)

STORE 0x8d6612ae TO .aS4(127)

STORE 0xbf3c6f47 TO .aS4(128)

STORE 0xd29be463 TO .aS4(129)

STORE 0x542f5d9e TO .aS4(130)

STORE 0xaec2771b TO .aS4(131)

STORE 0xf64e6370 TO .aS4(132)

STORE 0x740e0d8d TO .aS4(133)

STORE 0xe75b1357 TO .aS4(134)

STORE 0xf8721671 TO .aS4(135)

STORE 0xaf537d5d TO .aS4(136)

STORE 0x4040cb08 TO .aS4(137)

STORE 0x4eb4e2cc TO .aS4(138)

STORE 0x34d2466a TO .aS4(139)

STORE 0x0115af84 TO .aS4(140)

STORE 0xe1b00428 TO .aS4(141)

STORE 0x95983a1d TO .aS4(142)

STORE 0x06b89fb4 TO .aS4(143)

STORE 0xce6ea048 TO .aS4(144)

STORE 0x6f3f3b82 TO .aS4(145)

STORE 0x3520ab82 TO .aS4(146)

STORE 0x011a1d4b TO .aS4(147)

STORE 0x277227f8 TO .aS4(148)

STORE 0x611560b1 TO .aS4(149)

STORE 0xe7933fdc TO .aS4(150)

STORE 0xbb3a792b TO .aS4(151)

STORE 0x344525bd TO .aS4(152)

STORE 0xa08839e1 TO .aS4(153)

STORE 0x51ce794b TO .aS4(154)

STORE 0x2f32c9b7 TO .aS4(155)

STORE 0xa01fbac9 TO .aS4(156)

STORE 0xe01cc87e TO .aS4(157)

Todos los s.o. que hay ahí fuera son más o menos iguales. Todos somos una porquería". Brian Valentine, vicepresidente de Microsoft describiendo el edo., del arte en seguridad de s.o. en 2003

STORE 0xbcc7d1f6 TO .aS4(158)

STORE 0xcf0111c3 TO .aS4(159)

STORE 0xa1e8aac7 TO .aS4(160)

STORE 0x1a908749 TO .aS4(161)

STORE 0xd44fbd9a TO .aS4(162)

STORE 0xd0dadecb TO .aS4(163)

STORE 0xd50ada38 TO .aS4(164)

STORE 0x0339c32a TO .aS4(165)

STORE 0xc6913667 TO .aS4(166)

STORE 0x8df9317c TO .aS4(167)

STORE 0xe0b12b4f TO .aS4(168)

STORE 0xf79e59b7 TO .aS4(169)

STORE 0x43f5bb3a TO .aS4(170)

STORE 0xf2d519ff TO .aS4(171)

STORE 0x27d9459c TO .aS4(172)

STORE 0xbf97222c TO .aS4(173)

STORE 0x15e6fc2a TO .aS4(174)

STORE 0x0f91fc71 TO .aS4(175)

STORE 0x9b941525 TO .aS4(176)

STORE 0xfae59361 TO .aS4(177)

STORE 0xceb69ceb TO .aS4(178)

STORE 0xc2a86459 TO .aS4(179)

STORE 0x12baa8d1 TO .aS4(180)

STORE 0xb6c1075e TO .aS4(181)

STORE 0xe3056a0c TO .aS4(182)

STORE 0x10d25065 TO .aS4(183)

STORE 0xcb03a442 TO .aS4(184)

STORE 0xe0ec6e0e TO .aS4(185)

STORE 0x1698db3b TO .aS4(186)

STORE 0x4c98a0be TO .aS4(187)

STORE 0x3278e964 TO .aS4(188)

STORE 0x9f1f9532 TO .aS4(189)

STORE 0xe0d392df TO .aS4(190)

STORE 0xd3a0342b TO .aS4(191)

STORE 0x8971f21e TO .aS4(192)

STORE 0x1b0a7441 TO .aS4(193)

STORE 0x4ba3348c TO .aS4(194)

STORE 0xc5be7120 TO .aS4(195)

STORE 0xc37632d8 TO .aS4(196)

STORE 0xdf359f8d TO .aS4(197)

STORE 0x9b992f2e TO .aS4(198)

STORE 0xe60b6f47 TO .aS4(199)

STORE 0x0fe3f11d TO .aS4(200)

STORE 0xe54cda54 TO .aS4(201)

STORE 0x1edad891 TO .aS4(202)

STORE 0xce6279cf TO .aS4(203)

STORE 0xcd3e7e6f TO .aS4(204)

STORE 0x1618b166 TO .aS4(205)

STORE 0xfd2c1d05 TO .aS4(206)

STORE 0x848fd2c5 TO .aS4(207)

STORE 0xf6fb2299 TO .aS4(208)

Todos los s.o. que hay ahí fuera son más o menos iguales. Todos somos una porquería". Brian Valentine, vicepresidente de Microsoft describiendo el edo., del arte en seguridad de s.o. en 2003

STORE 0xf523f357 TO .aS4(209)

STORE 0xa6327623 TO .aS4(210)

STORE 0x93a83531 TO .aS4(211)

STORE 0x56cccd02 TO .aS4(212)

STORE 0xacf08162 TO .aS4(213)

STORE 0x5a75ebb5 TO .aS4(214)

STORE 0x6e163697 TO .aS4(215)

STORE 0x88d273cc TO .aS4(216)

STORE 0xde966292 TO .aS4(217)

STORE 0x81b949d0 TO .aS4(218)

STORE 0x4c50901b TO .aS4(219)

STORE 0x71c65614 TO .aS4(220)

STORE 0xe6c6c7bd TO .aS4(221)

STORE 0x327a140a TO .aS4(222)

STORE 0x45e1d006 TO .aS4(223)

STORE 0xc3f27b9a TO .aS4(224)

STORE 0xc9aa53fd TO .aS4(225)

STORE 0x62a80f00 TO .aS4(226)

STORE 0xbb25bfe2 TO .aS4(227)

STORE 0x35bdd2f6 TO .aS4(228)

STORE 0x71126905 TO .aS4(229)

STORE 0xb2040222 TO .aS4(230)

STORE 0xb6cbcf7c TO .aS4(231)

STORE 0xcd769c2b TO .aS4(232)

STORE 0x53113ec0 TO .aS4(233)

STORE 0x1640e3d3 TO .aS4(234)

```
        STORE 0x38abbd60 TO .aS4(235)
        STORE 0x2547adf0 TO .aS4(236)
        STORE 0xba38209c TO .aS4(237)
        STORE 0xf746ce76 TO .aS4(238)
        STORE 0x77afa1c5 TO .aS4(239)
        STORE 0x20756060 TO .aS4(240)
        STORE 0x85cbfe4e TO .aS4(241)
        STORE 0x8ae88dd8 TO .aS4(242)
        STORE 0x7aaaf9b0 TO .aS4(243)
        STORE 0x4cf9aa7e TO .aS4(244)
        STORE 0x1948c25c TO .aS4(245)
        STORE 0x02fb8a8c TO .aS4(246)
        STORE 0x01c36ae4 TO .aS4(247)
        STORE 0xd6ebe1f9 TO .aS4(248)
        STORE 0x90d4f869 TO .aS4(249)
        STORE 0xa65cdea0 TO .aS4(250)
        STORE 0x3f09252d TO .aS4(251)
        STORE 0xc208e69f TO .aS4(252)
        STORE 0xb74e6132 TO .aS4(253)
        STORE 0xce77e25b TO .aS4(254)
        STORE 0x578fdfe3 TO .aS4(255)
        STORE 0x3ac372e6 TO .aS4(256)
ENDWITH
ENDPROC
PROCEDURE xor
```

```
LPARAMETERS pnValor1 as Number, pnValor2 as Number
LOCAL lnResultado, lnMaximo32
lnResultado=BITXOR(pnValor1,pnValor2)
lnMaximo32=0xffffffff
IF lnResultado< 0
	lnResultado= lnMaximo32 + 1 + lnResultado
ENDIF

RETURN lnResultado
ENDPROC
PROCEDURE blosfishinicializar
LPARAMETERS pcPalabraClave as String
LOCAL lnLlaveBytes, lcLlave, lnDato, i, j
lnLlaveBytes = LEN(pcPalabraClave)
lcLlave=''
lnDato =0x00000000
i=0
j=0
WITH this
	IF lnLlaveBytes = 0
		.BlowfishIniciado=0
	ELSE
		IF LEN(pcPalabraClave)>.maxLlaveBytes
			lcLlave=ALLTRIM(SUBSTR(pcPalabraClave,1,.maxLlaveBytes))
		ELSE
			lcLlave=ALLTRIM(pcPalabraClave)
```

```
ENDIF
*--- Iniciando las constantes de Blowfish en arreglo
.BlosfishConstantes()
j=0
FOR i=1 TO 18
        lnDato = ((ASC(SUBSTR(lcLlave,(j+0)%lnLlaveBytes+1,1))*256+;
                ASC(SUBSTR(lcLlave,(j+1)%lnLlaveBytes+1,1)))*256+;
                ASC(SUBSTR(lcLlave,(j+2)%lnLlaveBytes+1,1)))*256+;
                ASC(SUBSTR(lcLlave,(j+3)%lnLlaveBytes+1,1))
        .aP[i] = .XOR(.aP[i], lnDato)
        j = (j + 4) % lnLlaveBytes
ENDFOR
lcLlave=.Codificar(lcLlave)
.ParXL=0x00000000
.ParXR=0x00000000
FOR i = 1 TO 18 STEP 2
        .BlowfishEncriptar()
        .aP[i] = .ParXL
        .aP[i + 1] = .ParXR
ENDFOR
FOR j = 1  TO 256 STEP 2
        .BlowfishEncriptar()
        .aS1[j]   = .ParXL
        .aS1[j+1] = .ParXR
```

```
                    .BlowfishEncriptar()
                    .aS2[j]   = .ParXL
                    .aS2[j+1] = .ParXR
                    .BlowfishEncriptar()
                    .aS3[j]   = .ParXL
                    .aS3[j+1] = .ParXR
                    .BlowfishEncriptar()
                    .aS4[j]   = .ParXL
                    .aS4[j+1] = .ParXR
                ENDFOR
                .BlowfishIniciado=1
        ENDIF
ENDWITH
RETURN .BlowfishIniciado
ENDPROC
PROCEDURE blowfishdesencriptar
LOCAL xl, xr
WITH this
        Xl = .ParXL
        Xr = .ParXR
        Xl = .XOR(Xl, .aP[18])
        Xr = .BlowfishRedondear(Xr, Xl, 17)
        Xl = .BlowfishRedondear(Xl, Xr, 16)
        Xr = .BlowfishRedondear(Xr, Xl, 15)
        Xl = .BlowfishRedondear(Xl, Xr, 14)
        Xr = .BlowfishRedondear(Xr, Xl, 13)
```

```
        Xl = .BlowfishRedondear(Xl, Xr, 12)
        Xr = .BlowfishRedondear(Xr, Xl, 11)
        Xl = .BlowfishRedondear(Xl, Xr, 10)
        Xr = .BlowfishRedondear(Xr, Xl, 9)
        Xl = .BlowfishRedondear(Xl, Xr, 8)
        Xr = .BlowfishRedondear(Xr, Xl, 7)
        Xl = .BlowfishRedondear(Xl, Xr, 6)
        Xr = .BlowfishRedondear(Xr, Xl, 5)
        Xl = .BlowfishRedondear(Xl, Xr, 4)
        Xr = .BlowfishRedondear(Xr, Xl, 3)
        Xl = .BlowfishRedondear(Xl, Xr, 2)
        Xr = .XOR(Xr, .aP[1])
        .ParXL = Xr
        .ParXR = Xl
ENDWITH


ENDPROC
PROCEDURE blowfishencriptar
LOCAL xl, xr
WITH this
        Xl = .ParXL
        Xr = .ParXR
        Xl = .XOR(Xl, .aP[1])
        Xr = .BlowfishRedondear(Xr, Xl, 2)
        Xl = .BlowfishRedondear(Xl, Xr, 3)
```

```
        Xr = .BlowfishRedondear(Xr, Xl, 4)
        Xl = .BlowfishRedondear(Xl, Xr, 5)
        Xr = .BlowfishRedondear(Xr, Xl, 6)
        Xl = .BlowfishRedondear(Xl, Xr, 7)
        Xr = .BlowfishRedondear(Xr, Xl, 8)
        Xl = .BlowfishRedondear(Xl, Xr, 9)
        Xr = .BlowfishRedondear(Xr, Xl, 10)
        Xl = .BlowfishRedondear(Xl, Xr, 11)
        Xr = .BlowfishRedondear(Xr, Xl, 12)
        Xl = .BlowfishRedondear(Xl, Xr, 13)
        Xr = .BlowfishRedondear(Xr, Xl, 14)
        Xl = .BlowfishRedondear(Xl, Xr, 15)
        Xr = .BlowfishRedondear(Xr, Xl, 16)
        Xl = .BlowfishRedondear(Xl, Xr, 17)
        Xr = .XOR(Xr, .aP[18])
        .ParXL = Xr
        .ParXR = Xl
ENDWITH
ENDPROC
PROCEDURE blowfishredondear
LPARAMETERS pnValorA,pnValorB,pnValorN
LOCAL lnResultado as Number
WITH this
        lnResultado= (.XOR(pnValorA, .XOR( ((.XOR( (.aS1[.BytePalabra1(pnValorB)]
+.aS2[.BytePalabra2(pnValorB)]), .aS3[.BytePalabra3(pnValorB)]) ) +
.aS4[.BytePalabra4(pnValorB)] ), .aP[pnValorN]) ))
ENDWITH
```

```
RETURN lnResultado

ENDPROC

PROCEDURE decodificarpalabra

LPARAMETERS pnPalabra

LOCAL lcResultado, lnTemporalA, lnTemporalB, i

lcResultado=0

lnTemporalA=0

lnTemporalB=0

&& reverse byteorder for intel systems

FOR i=7 TO 1 STEP -2

	lnTemporalA=ASC(SUBSTR(pnPalabra,i,1))

	lnTemporalB=ASC(SUBSTR(pnPalabra,i+1,1))

	IF lnTemporalA<58

		lnTemporalA=lnTemporalA-48

	ELSE

		lnTemporalA=lnTemporalA-55

	ENDIF

	IF lnTemporalB<58

		lnTemporalB=lnTemporalB-48

	ELSE

		lnTemporalB=lnTemporalB-55

	ENDIF

	lcResultado=lcResultado*256+((lnTemporalA*16)+lnTemporalB)

ENDFOR

RETURN lcResultado
```

```
ENDPROC
PROCEDURE codificarpalabra
LPARAMETERS pnPalabra
LOCAL lcResultado, lnTemporalA, lnTemporalB, i
LOCAL ARRAY laBytes[4]
lcResultado=""
lnTemporalA=0
lnTemporalB=0
i=0
WITH this
	STORE .BytePalabra1(pnPalabra)-1 TO laBytes[1]
	STORE .BytePalabra2(pnPalabra)-1 TO laBytes[2]
	STORE .BytePalabra3(pnPalabra)-1 TO laBytes[3]
	STORE .BytePalabra4(pnPalabra)-1 TO laBytes[4]
	FOR i=4 TO 1 STEP -1
		lnTemporalA=FLOOR(laBytes[i]/16)
		lnTemporalB=laBytes[i] % 16
		IF lnTemporalA<10
			lnTemporalA=lnTemporalA+48
		ELSE
			lnTemporalA=lnTemporalA+55
		ENDIF
		IF lnTemporalB<10
			lnTemporalB=lnTemporalB+48
		ELSE
			lnTemporalB=lnTemporalB+55
```

```
            ENDIF

            lcResultado=lcResultado+CHR(lnTemporalA)+CHR(lnTemporalB)

        ENDFOR

ENDWITH

RETURN lcResultado

ENDPROC

PROCEDURE bytepalabra1

LPARAMETERS pnPalabra

        RETURN ((FLOOR(FLOOR(FLOOR(pnPalabra/256)/256)/256) % 256)+1)

ENDPROC

PROCEDURE bytepalabra2

LPARAMETERS pnPalabra

        RETURN ((FLOOR(FLOOR(pnPalabra/256)/256) % 256)+1)

ENDPROC

PROCEDURE bytepalabra3

LPARAMETERS pnPalabra

        RETURN ((FLOOR(pnPalabra/256) % 256)+1)

ENDPROC

PROCEDURE bytepalabra4

LPARAMETERS pnPalabra

        RETURN ((pnPalabra%256)+1)


ENDPROC

PROCEDURE decodificarblowfish

LPARAMETERS pcMensaje as String, pcPalabraClave as String
```

```
LOCAL lcPalabraClaveBackup, lnLongitudMensaje, lcSalidaDato, i
lcPalabraClaveBackup=""
lnLongitudMensaje=0
lcSalidaDato=""
i=0
pcMensaje=ALLTRIM(pcMensaje)
pcPalabraClave=ALLTRIM(pcPalabraClave)
WITH this
	IF lcPalabraClaveBackup="" OR pcPalabraClave!= lcPalabraClaveBackup
		.BlowfishIniciado=0
		lcPalabraClaveBackup = pcPalabraClave
	ENDIF
	IF .BlowfishIniciado=1 OR .BlosfishInicializar(pcPalabraClave) =1
		lnLongitudMensaje=LEN(pcMensaje)
		FOR i=1 TO ((lnLongitudMensaje-
INT(lnLongitudMensaje/16)*16)*MOD(lnLongitudMensaje,16)+
(INT(lnLongitudMensaje/16)*16))-lnLongitudMensaje
			pcMensaje=pcMensaje+"0"
		ENDFOR

		lnLongitudMensaje=LEN(pcMensaje)
		lcSalidaDato=""
		FOR i=1 TO lnLongitudMensaje STEP 16
			.ParXR=.DecodificarPalabra(SUBSTR(pcMensaje,i,8))
			.ParXL=.DecodificarPalabra(SUBSTR(pcMensaje,i+8,8))
			.BlowfishDesencriptar()
```

```
                                lcSalidaDato=lcSalidaDato+.CodificarPalabra(.ParXR)
+.CodificarPalabra(.ParXL)

                        ENDFOR

                        lcSalidaDato=.Decodificar(lcSalidaDato)

                ENDIF


lcSalidaDato=STRTRAN(lcSalidaDato,CHR(208)+CHR(204)+CHR(208)+PADR('',3,CHR(2
04)))

ENDWITH

RETURN STRTRAN(lcSalidaDato,CHR(0),'')

ENDPROC

PROCEDURE codificarblowfish

LPARAMETERS pcMensaje as String, pcPalabraClave as String

LOCAL lcPalabraClaveBackup, lnLongitudMensaje, lcSalidaDato, i

lcPalabraClaveBackup=""

lnLongitudMensaje=0

lcSalidaDato=""

i=0

pcMensaje=ALLTRIM(pcMensaje)

pcPalabraClave=ALLTRIM(pcPalabraClave)

WITH this

        IF lcPalabraClaveBackup="" OR pcPalabraClave!= lcPalabraClaveBackup

                .BlowfishIniciado=0

                lcPalabraClaveBackup = pcPalabraClave

        ENDIF

        IF .BlowfishIniciado=1 OR .BlosfishInicializar(pcPalabraClave)=1

                pcMensaje=.Codificar(pcMensaje)
```

```
		lnLongitudMensaje=LEN(pcMensaje)

		FOR i=1 TO ((lnLongitudMensaje-
INT(lnLongitudMensaje/16)*16)*MOD(lnLongitudMensaje,16)+
(INT(lnLongitudMensaje/16)*16))-lnLongitudMensaje

			pcMensaje=pcMensaje+"0"

		ENDFOR

		lcSalidaDato=""

		FOR i=1 TO LEN(pcMensaje) STEP 16

			.ParXR=.DecodificarPalabra(SUBSTR(pcMensaje,i,8))

			.ParXL=.DecodificarPalabra(SUBSTR(pcMensaje,i+8,8))

			.BlowfishEncriptar()

			lcSalidaDato=lcSalidaDato+.CodificarPalabra(.ParXR)
+.CodificarPalabra(.ParXL)

		ENDFOR

	ENDIF

ENDWITH

RETURN lcSalidaDato

ENDPROC

PROCEDURE Init

this.AddProperty('aP[18]')

this.AddProperty('aS1[256]')

this.AddProperty('aS2[256]')

this.AddProperty('aS3[256]')

this.AddProperty('aS4[256]')

this.BlosfishConstantes()

ENDPROC
```

ž_memberdata = 2077<VFPData><memberdata name="codificar" type="method" display="Codificar"/><memberdata name="decodificar" type="method" display="Decodificar"/><memberdata name="codificadorsimple" type="method" display="CodificadorSimple"/><memberdata name="decodificadorsimple" type="method" display="DecodificadorSimple"/><memberdata name="desencriptar" type="method" display="Desencriptar"/><memberdata name="encriptar" type="method" display="Encriptar"/><memberdata name="blosfishconstantes" type="method" display="BlosfishConstantes"/><memberdata name="xor" type="method" display="xOr"/><memberdata name="blosfishinicializar" type="method" display="BlosfishInicializar"/><memberdata name="blowfishdesencriptar" type="method" display="BlowfishDesencriptar"/><memberdata name="blowfishencriptar" type="method" display="BlowfishEncriptar"/><memberdata name="blowfishredondear" type="method" display="BlowfishRedondear"/><memberdata name="decodificarpalabra" type="method" display="DecodificarPalabra"/><memberdata name="codificarpalabra" type="method" display="CodificarPalabra"/><memberdata name="bytepalabra1" type="method" display="BytePalabra1"/><memberdata name="bytepalabra2" type="method" display="BytePalabra2"/><memberdata name="bytepalabra3" type="method" display="BytePalabra3"/><memberdata name="bytepalabra4" type="method" display="BytePalabra4"/><memberdata name="decodificarblowfish" type="method" display="DecodificarBlowfish"/><memberdata name="codificarblowfish" type="method" display="CodificarBlowfish"/><memberdata name="blowfishiniciado" type="property" display="BlowfishIniciado"/><memberdata name="maxllavebytes" type="property" display="MaxLlaveBytes"/><memberdata name="parxl" type="property" display="ParXL"/><memberdata name="parxr" type="property" display="ParXR"/><memberdata name="version" type="property" display="Version"/><memberdata name="_memberdata" type="property" display="_MemberData"/><memberdata name="codificarsimple" type="method" display="CodificarSimple"/><memberdata name="decodificarsimple" type="method" display="DecodificarSimple"/></VFPData>

parxl = 0

parxr = 0

blowfishiniciado = 0

version = '1.3'

maxllavebytes = 56

Name = "blowfish"

VENTAJAS

- Ejecuta muy rápido en cualquier maquina actual.
- Acepta una clave larga para hacer su cifrado

DESVENTAJAS

- Hay que hacer llegar esas claves de una persona a otra de forma segura, y esto es bastante difícil a no ser que nos encontremos en persona (ej. Ir al banco para que nos den una serie de claves para usar por internet).
- Si un grupo de N personas nos queremos comunicar entre nosotros, necesitamos claves para cada pareja. Por ejemplo para asegurarnos que nos podemos comunicar de forma secreta entre todos los miembros de un equipo de fútbol (11 personas), y que podemos escribirnos al menos 4 mensajes, cada miembro del equipo necesitaría manejar más de 40 claves.

ATÁQUES AL ALGORITMO

Aun no existe ningún ataque documentado que haya sido exitoso.

RSA

En criptografía, RSA (Rivest, Shamir y Adleman) es un sistema criptográfico de clave pública desarrollado en 1977. En la actualidad, RSA es el primer y más utilizado algoritmo de este tipo y es válido tanto para cifrar como para firmar digitalmente.

La seguridad de este algoritmo radica en el problema de la factorización de números enteros. Los mensajes enviados se representan mediante números, y el funcionamiento se basa en el producto, conocido, de dos números primos grandes elegidos al azar y mantenidos en secreto. Actualmente estos primos son del orden de 10200, y se prevé que su tamaño aumente con el aumento de la capacidad de cálculo de los ordenadores.

Como en todo sistema de clave pública, cada usuario posee dos claves de cifrado: una pública y otra privada. Cuando se quiere enviar un mensaje, el emisor busca la clave pública del receptor, cifra su mensaje con esa clave, y una vez que el mensaje cifrado llega al receptor, este se ocupa de descifrarlo usando su clave privada.

Se cree que RSA será seguro mientras no se conozcan formas rápidas de descomponer un número grande en producto de primos. La computación cuántica podría proveer de una solución a este problema de factorización.

El algoritmo fue patentado por el MIT en 1983 en Estados Unidos con el número 4.405.829. Esta patente expiró el 21 de septiembre de 2000. Como el algoritmo fue publicado antes de patentar la aplicación, esto impidió que se pudiera patentar en otros lugares del mundo. Dado que Cocks trabajó en un organismo gubernamental, una patente en Estados Unidos tampoco hubiera sido posible.

CÓDIGO O PSEUDOCÓDIGO

El algoritmo consta de tres pasos: generación de claves, cifrado y descifrado.

Idea del algoritmo

Supongamos que Bob quiere enviar a Alicia un mensaje secreto que solo ella pueda leer.

Alicia envía a Bob una caja con una cerradura abierta, de la que solo Alicia tiene la llave. Bob recibe la caja, escribe el mensaje, lo pone en la caja y la cierra con su cerradura (ahora Bob no puede leer el mensaje). Bob envía la caja a Alicia y ella la abre con su llave. En este ejemplo, la caja con la cerradura es la «clave pública» de Alicia, y la llave de la cerradura es su «clave privada».

Técnicamente, Bob envía a Alicia un «mensaje llano» M en forma de un número m menor que otro número n, mediante un protocolo reversible conocido como padding scheme («patrón de relleno»). A continuación genera el «mensaje cifrado» c mediante la siguiente operación:

$c = m^e \pmod{n}$, donde e es la clave pública de Alicia.

Ahora Alicia descifra el mensaje en clave c mediante la operación inversa dada por

$m = c^d \pmod{n}$, donde d es la clave privada que solo Alicia conoce.

Generación de claves

- Cada usuario elige dos números primos distintos p y q.

Todos los s.o. que hay ahí fuera son más o menos iguales. Todos somos una porquería". Brian Valentine, vicepresidente de Microsoft describiendo el edo., del arte en seguridad de s.o. en 2003

    - Por motivos de seguridad, estos números deben escogerse de forma aleatoria y deben tener una longitud en bits parecida. Se pueden hallar primos fácilmente mediante test de primalidad.
- Se calcula n = pq.
    - n se usa como el módulo para ambas claves, pública y privada.
- Se calcula $\varphi(n) = (p-1)(q-1)$, donde φ es la función φ de Euler.
- Se escoge un entero positivo e menor que $\varphi(n)$, que sea coprimo con $\varphi(n)$.
    - e se da a conocer como el exponente de la clave pública.
    - Si se escoge un e con una suma encadenada corta, el cifrado será más efectivo. Un exponente e muy pequeño (p. ej. e = 3) podría suponer un riesgo para la seguridad.1
- Se determina un d (mediante aritmética modular) que satisfaga la congruencia $de \equiv 1 \pmod{\varphi(n)}$.
    - Expresado de otra manera, de -1 divide a $\varphi(n) = (p-1)(q-1)$.
    - Esto suele calcularse mediante el algoritmo de Euclides extendido.
    - d se guarda como el exponente de la clave privada.

La clave pública es (n,e), esto es, el módulo y el exponente de cifrado. La clave privada es (n,d), esto es, el módulo y el exponente de descifrado, que debe mantenerse en secreto. Nota:

- PKCS#1 v2.0 y PKCS#1 v2.1 se especifican mediante la función de Carmichael λ(n) = mcm(p − 1,q − 1) en vez de la función φ de Euler, donde mcm es el mínimo común múltiplo.
- Para una mayor eficiencia los siguientes valores se calculan de antemano y se almacenan como parte de la clave privada:
    - p y q: los primos para la generación de las claves,
    - $d_p \equiv d \pmod{p-1}$ y $d_q \equiv d \pmod{q-1}$,
    - $q_{inv} \equiv q^{-1} \pmod{p}$.

Cifrado

Alicia comunica su clave pública (n,e) a Bob y guarda la clave privada en secreto. Ahora Bob desea enviar un mensaje M a Alicia.

Primero, Bob convierte M en un número entero m menor que n mediante un protocolo reversible acordado de antemano. Luego calcula el texto cifrado c mediante la operación

Esto puede hacerse rápido mediante el método de exponenciación binaria. Ahora Bob transmite c a Alicia.

VENTAJAS

- No requiere claves secretas.
- Permite encriptar y Firmar digitalmente.
- Utilizado conjuntamente con DES otorga una mayor velocidad de operación.
- Es un estándar internacional.

DESVENTAJAS

- Al ser combinado toma las debilidades del otro algoritmo

ATÁQUES AL ALGORITMO

Wing H. Wong. Timing Attacks on RSA: Revealing Your Secrets through the Fourth Dimension

CONCLUSIÓN

La criptografía podría ser considerada como métodos que sirven para ocultar cierta información o determinados datos que podrían ser de vital importancia para ciertos sectores, sin embargo, para otros podría representar el guardar información personal como lo son las firmas digitales, sin embargo para mi muy en lo personal me queda al aire la pregunta de ¿si realmente con el uso de la criptografía moderna podamos garantizar las propiedades de integridad y confidencialidad y con ello resolver si no al 100% el problema, casi en su totalidad la seguridad de la información?, a pesar de ello siempre habrá un punto débil en la criptografía, como ya lo veíamos en algunos casos de ataques, que por lo regular, podríamos decir, que es, por el hecho de que es desarrollado por personas, y en segundo por malos diseños de los sistemas, creo que por lo anterior, que no solamente hay que desarrollar algoritmos que nos permitan ocultar o mostrar información de algún modo ya sea matemático, por sustitución o incluso por bloques, y en este punto me surge la siguiente pregunta ¿la información algún día dejara de necesitar ser protegida?

BIBLIOGRAFÍA

MÉDIOS IMPRESOS

- Singh, Simon: The Code Book. Fourth State, London 2000.
- Neumann, John von: La Computadora y el cerebro. Bon Ton 1999.
- Singh, Simon: Fermat's last Theorem. Fourth Estate, 1998.pp.168-174.
- Gardner, Martin: Juegos matemáticos. Investigación y ciencia, Noviembre, 1977.
- Barrow, John D.: ¿Por qué el mundo es matemático? Grijalbo, Barcelona, 1997. Trad.: Javier Gracía Sanz.

MÉDIOS ELECTRÓNICOS

- http://www.dwheeler.com/secure-programs/Secure-Programs-HOWTO/crypto.html
- http://vexpert.mvps.org/articles/vbEncrypt.htm
- http://digisign.50megs.com/info040.html
- http://es.wikipedia.org/wiki/RSA
- http://es.wikipedia.org/wiki/Data_Encryption_Standard
- http://www.ekontsulta.net/ekontsulta/wiki/index.php/Cifrado_de_datos
- http://lattice.ft.uam.es/perpag/alberto/doc/tutorials/html/cripto/node4.html
- http://www.programatium.com/vfox/tutoriales/encriptar-con-blowfish.htm#clase
- http://www.codeproject.com/KB/library/DES.aspx

Todos los s.o. que hay ahí fuera son más o menos iguales. Todos somos una porquería". Brian Valentine, vicepresidente de Microsoft describiendo el edo., del arte en seguridad de s.o. en 2003