Las Redes Substitución-Permutación y el AES(Advanced Encryption Standard)

Jaime Gutiérrez

1. Introducción

Los algoritmos de cifrado simétrico más modernos son una combinación de los cifradores de substitución y de transposición o permutación (ver [10], [15]), que se corresponden con los conceptos de *confusión* y *difusión* introducidos por Claude Shannon. El principal objetivo de la confusión es esconder la relación entre el texto claro, el texto cifrado y la clave y, el de la difusión trata de propagar la información real en el mensaje cifrado. La aplicación sucesiva de un cierto número de tales criptosistemas simples puede dar origen a un sistema criptográfico iterado (cifrador producto) suficientemente seguro. La mayoría de los criptosistemas se basan en diferentes capas de substituciones y permutaciones, estructura denominada SPN (Substitution-Permutation Networks), redes de substitución-permutación y, en particular el AES (Advanced Encryption Standard).

2. Producto de Criptosistemas

Otra de las innovaciones de Shannon presentada en su famoso artículo [14] de 1949 es la de combinar criptosistemas formando su *producto*. Esta idea ha sido de importancia fundamental en el diseño de los cripsotistemas de hoy en día, como el DES(Data Encryption Standard) y el AES que serán analizados posteriormente. En esta sección definimos formalmente este concepto.

Por simplicidad en la presentación, supondremos que el espacio del texto claro $\mathcal{M}$ y del texto cifrado $\mathcal{C}$ coinciden, es decir, $\mathcal{M} = \mathcal{C}$ que son denominados criptosistemas endomórficos.

Sean $\mathbf{S_1} = (\mathcal{C}, \mathcal{C}, \mathcal{K}_1, \mathcal{E}_1, \mathcal{D}_1)$ y $\mathbf{S_2} = (\mathcal{C}, \mathcal{C}, \mathcal{K}_2, \mathcal{E}_2, \mathcal{D}_2)$ dos criptosistemas endomórficos. El criptosistema producto $\mathbf{S_1}$ y $\mathbf{S_2}$, denotado por $\mathbf{S_1} \times \mathbf{S_2}$ es:

$$\mathbf{S_1} \times \mathbf{S_2} = (\mathcal{C}, \mathcal{C}, \mathcal{K}_1 \times \mathcal{K}_2, \mathcal{E}, \mathcal{D}).$$

Una clave del criptosistema producto tiene la forma $K = (K_1, K_2)$ donde $K_1 \in \mathcal{K}_1$ y $K_2 \in \mathcal{K}_2$. Las transformaciones de cifrado y descifrado están definidas como sigue:

$$E_K(\mathbf{x}) = E_{K_2}(E_{K_1}(\mathbf{x})), \qquad D_K(\mathbf{y}) = D_{K_1}(D_{K_2}(\mathbf{y})).$$

Se tiene que $D_K(E_K(\mathbf{x})) = \mathbf{x}$, para cada $\mathbf{x} \in \mathcal{C}$ y para cada $K \in \mathcal{K}_1 \times \mathcal{K}_2$.

A continuación se muestra un ejemplo muy simple de un criptosistema producto. Definimos el cifrador $\mathbf{S_1}$

Sean r y N enteros mayores que 2. Sea $\mathcal{M} = \mathcal{C} = Z_N^r$ y sea
$\mathcal{K} = \{r \times r$ matrices inversibles sobre $Z_N\}$.
Para cada $K \in \mathcal{K}$, definimos
$$E_K(\mathbf{x}) = \mathbf{x}K,$$
$$D_K(y) = \mathbf{y}K^{-1}.$$

Ahora, definimos el cifrador $\mathbf{S_2}$:

Sean r y N enteros mayores que 2. Sea $\mathcal{M} = \mathcal{C} = Z_N^r$ y sea
$\mathcal{K} = \{b = (b_1, \ldots, b_r) \in Z_N^r\}$.
Para cada $b \in \mathcal{K}$, definimos
$$E_K(\mathbf{x}) = \mathbf{x} + b,$$
$$D_K(\mathbf{y}) = \mathbf{y} - b.$$

Precisamente el producto de estos dos criptosistemas es el código matricial ([10], [15]).

3. Redes de Substitución-Permutación

Una red de substitución-permutación SPN es un cifrador *iterado*. La idea general de estos cifradores consiste en dividir el mensaje en bloques de bits, generalmente del mismo tamaño fijo, y aplicar un número N_r (rondas o vueltas) de substituciones y permutaciones a cada bloque. Cada cifrador requiere la función ronda; la clave K, que generalmente es una cadena aleatoria de bits y una función de expansión EK de la clave K, proporcionando una lista de claves $EK(K) = (K^{(1)}, \ldots, K^{(N_r+1)})$ y que será construida con un algoritmo público.

En una SPN intervienen dos permutaciones: π_S, denominada S-caja, (la letra S proviene de la palabra substitución) y, π_P la cual permuta los bits de cada bloque. Sean l y m enteros positivos. Consideramos las permutaciones

$$\pi_S : \{0,1\}^l \to \{0,1\}^l$$

y

$$\pi_P : \{1, \ldots, lm\} \to \{1, \ldots, lm\}.$$

Dado el texto claro $\mathbf{x} = (x_1, \ldots, x_{lm})$, como un bloque de longitud lm, podemos interpretar $\mathbf{x}$ como una concatenación de m cadenas de bits, cada una de ellas con l bits, $\mathbf{x} = (x_{(1)}, \ldots, x_{(m)})$ donde cada

$$x_{(i)} = (x_{(i-1)l+1}, \ldots, x_{il}).$$

Finalmente, dada la salida $(K^1, \ldots, K^{N_r+1})$ de la función expansión EK de la clave $K \in \{0,1\}^{lm}$. Ciframos $\mathbf{x}$ mediante el siguiente algoritmo $SPN(\mathbf{x}, \pi_S, \pi_P, (K^1, \ldots, K^{N_r+1}) = \mathbf{y}$:

$$\begin{array}{l}
\mathbf{z}^0 \leftarrow \mathbf{x} \\
\textbf{for } r \leftarrow 1 \quad \textbf{to} \quad N_r - 1 \\
\textbf{do} \left\{ \begin{array}{l} \mathbf{x}^r \leftarrow \mathbf{z}^{r-1} \oplus K^r \\ \textbf{for} \quad i \leftarrow 1 \quad \textbf{to} \quad m \\ \qquad \textbf{do} \quad \mathbf{y}^r_{(i)} \leftarrow \pi(\mathbf{x}^r_{(i)}) \\ \mathbf{z}^r \leftarrow (\mathbf{y}^r_{\pi_P(1)}, \ldots, \mathbf{y}^r_{\pi_P(lm)}) \end{array} \right. \\
\mathbf{x}^{N_r} \leftarrow \mathbf{z}^{N_r-1} \oplus K^{N_r} \\
\textbf{for} \quad i \leftarrow 1 \quad \textbf{to} \quad m \\
\qquad \textbf{do } \mathbf{y}^{N_r}_{(i)} \leftarrow \pi_S(\mathbf{x}^{N_r}_{(i)}) \\
\mathbf{y} \leftarrow \mathbf{y}^{N_r} \oplus K^{N_r+1}
\end{array}$$

Donde $\oplus$ representa la función or-exclusivo y $\mathbf{z}$ es un vector variable local. La entrada de la S-caja en la vuelta r es $\mathbf{x}^r$ y la salida es $\mathbf{y}^r$. El hecho que la última vuelta no involucra la permutación π_P permite diseñar un algoritmo involutivo, es decir, las transformaciones cifrado y descifrado son idénticas.

Algunas variantes de la red SPN utilizan más de una S-caja [17].

Fundamentalmente, hay dos ataques consistentes a este tipo de redes: el criptoanálisis lineal ([12]) y el diferencial ([6]). Obviamente, el tamaño de la clave y el número de vueltas juegan un papel transcendental en la seguridad de estas redes.

4. Redes Feistel y DES

Muchos de los cifradores producto tienen en común que dividen cada bloque S_i en la ronda i-ésima en dos mitades: $S_i = L_i R_i$ y aplican una red tipo SPN, en el que la salida de cada ronda es utilizada como entrada para la siguiente:

$$\begin{aligned} L_i &= R_{i-1}, \\ R_i &= L_{i-1} \oplus f(R_{i-1}, K_i). \end{aligned}$$

La función f no requiere ninguna propiedad de inyectividad para el proceso de descifrado:

$$L_{i-1} = R_i \oplus f(L_i, K_i),$$
$$R_{i-1} = L_i.$$

Esta clase de estructuras fue introducida por H. Feistel ([8]) en 1973 y es utilizada por varios algoritmos de cifrado en bloque, como DES, CAST, FEAL, Lucifer, etcetera.

Precisamente, el DES es el criptosistema de clave privada más utilizado en el mundo, y muy especialmente en el ámbito financiero y bancario. En mayo de 1973 la National Bureau of Standards (ahora National Institute of Standards and Technology, NIST) del gobierno de los Estados Unidos de America solicitó a la comunidad científica criptosistemas simétricos, con vistas a adoptar un estándar que pudiese ser construido en masa y con máximas garantías de seguridad. Oficialmente fue publicado en Marzo de 1975 (ver [7]). Después de notorias discusiones públicas, se adoptó como estándar por el gobierno de EEUU, para comunicaciones no clasificadas.

El DES cifra y descrifra bloques de 64 bits y lo somete a 16 rondas, con una clave 64 bits (56 bits reales y 8 bits de control de paridad). Básicamente, la función f consiste en una permutación de expansión, convirtiendo un bloque de 32 bits en uno de 48 bits. A continuación realiza un or-exclusiva con el valor de la función clave K^i y aplica una substitución, utilizando las famosas ocho S-cajas del DES. Estas son matrices de 4 filas por 16 columnas, cuyos coeficientes están comprendidos entre 0 y 15.

Se esperaba que el DES pudiera tener una validez como uso estandard entre 10 y 15 años, sin embargo la realidad ha sido bien distinta.

Desde su nacimiento hubo una gran critica. Una objeción era la no linealidad de la computación de las S-caja, a diferencia del resto de las computaciones del DES. Se especulaba que tenían propiedades algebraicas, mantenidas en secreto por el gobierno de EEUU, lo cual permitiría descifrar facilmente los mensajes. Todo esto, en el cajón de las hipótesis, puesto que no se ha podido demostrar. Cuando E. Biham y A. Shamir inventaron en 1990 el criptoanálisis diferencial, descubrierón que las S-cajas fueron diseñadas para prevenir este tipo de ataques. Parece ser que investigadores de IBM conocían hace más de 20 años este tipo de ataque y lo mantuvieron en secreto hasta que, independientemente, Biham y Shamir lo descubrieron.

Sin embargo, la mayor crítica se refiere al tamaño real de la clave de 56 bits, el cual es demasiado pequeño para ser realmente segura. Por ejemplo, su predecesor el Lucifer, también, de IBM tenia 128 bits. En 1977 Diffie y Hellman sugirieron que se podría construir un chip que comprobase 10^6 claves por segundo. Una máquina con 10^6 chips encontraría todo el espacio de claves en un día de trabajo. Se estimó que el coste económico de construir esa máquina sería de unos 20,000,000 de euros. Precisamente en 1993, Michael Wiener describió explícitamente tal máquina. En julio de 1998, la empresa Electronic Frontier Foundation construyó el ordenador "DES Crackerçonteniendo 1536 chips, con capacidad para buscar 88 billones de claves por segundo. En enero de 1999, se unieron DES Challenge III construida

por la compañia RSA y DES-Cracker junto con otros 100.000 ordenadores a traves de internet, permitiendo romper DES en 22 horas y 15 minutos, comprobando ás de 245 billones de claves por segundo, (ver [16], [15], [17]).

Los dos más destacables ataques criptoanáliticos son el diferencial y el lineal. Este último fue inventado por Matsui, permite romper el DES con ataques a texto claro conocido, usando 2^{43} pares de texto claro conocido, en 40 días de trabajo. Sin embargo, estos métodos de criptoanálisis apenas tienen impacto en la robustez del DES, pues es muy improbable que se pueda disponer de tanto pares de texto claro conocido, [5], [6], [12].

A pesar de esto, en la actualidad DES sigue siendo utilizado en el mundo entero. Muchas compañías prefieren mantener este criptosistema, precisamente porque ha sido capaz de permanecer durante más de 20 años, y optan por utilizar variantes que eviten el riesgo de tener que confiar en nuevos criptosistemas. Aprovechando, también, la implementación en hardware existente de DES o algunas de sus variantes. Una de estas variantes, es conocida como el triple DES. El funcionamiento del TDES consiste, básicamente, en actuar tres veces el DES, con tres claves distintas aumentando, de este modo, el tamaño del espacio de claves.

5. AES

Parece claro que DES (ni ninguna de sus variantes) no va seguir siendo el criptosistema de uso estándar en las organizaciones estadounidenses y por lo tanto sera obligatorio sustituirlo por otro más robusto.

5.1. Un poco de historia

En enero de 1997, el Instituto Nacional de Estándares y Tecnología (National Institute of Standards and Technology, NIST) comenzó el proceso de elegir un estándar de cifrado avanzado AES, con una convocatoria pública a la comunidad científica. Se espera que el algoritmo seleccionado sea utilizado por el gobierno de Estados Unidos y voluntariamente por el sector privado, en substitución de DES. Por extensión, presumiblemente sería adoptado por el resto de países del mundo. Las bases de la convocatoria ([11]) se especificaron los siguientes criterios de evaluación y requisitos mínimos de aceptación:

- El algoritmo debe ser público. Disponible gratuitamente, y ajustándose a los requisitos de la política de patentes del Instituto Nacional Americano de Estándares.
- Debe ser un algoritmo de cifrado en bloque símétrico.
- Debe estar diseñado de forma que se permita aumentar la longitud de clave según las necesidades.
- Debe poderse implementar tanto en hardware como en software.

- Estos requisitos serán evaluados de acuerdo con los siguientes criterios: seguridad, eficiencia computacional y requisitos de memoria, adecuación hardware y software, simplicidad de diseño y flexibilidad, y requisitos de licencia.

- Obligatoriamente los candidatos deben soportar cifrados con una longitud de bloque de 128 bits y una longitud de clave de 128, 192 y 256 bits.

Esta última exigencia responde a los previsibles avances en la apasionante teoría de la computacion cuántica [13]. Con los recursos actuales, una ataque a fuerza bruta a una clave de 128 bits esta condenado al fracaso, pero no puede asegurarse que en un futuro próximo el estado del arte de la computación no vaya a permitir reventar claves de esa longitud. Dado que el estándar se desea que pueda utilizarse hasta bien entrado el siglo XXI, se debe tener en cuenta estos avances teóricos en computación cuántica. Groseramente, podemos decir que las velocidades previstas para una máquina cuántica tardaría el mismo tiempo en realizar una búsqueda exhaustiva en un espacio de claves de 256 bits que un ordenador digital actual en un espacio de claves de 128 bits.

Despues, se convocaron tres conferencias, en distintos lugares del mundo, en las que los algoritmos candidatos pudieron ser probados, comentados y revisados. Durante el desarrollo del proceso AES, todos los algoritmos y criterios de diseño estuvieron disponibles de forma pública y abierta, evitando así la controversia que hubo en su predecesor DES. Todos los participantes han contribuido al proceso, analizando las posibles vulnerabilidades de sus competidores.

La fecha límite para la presentación fue el 15 de junio de 1998. De las 21 propuestas, sólo 15 cumplían las exigencias de la convocatoria. En el congreso "First AES Candidate Conference "en agosto de 1998, ([1]), el NIST presentó los 15 candidatos:

Primera criba

- CAST-256, Entust Technologies, Inc. (C. Adams).

- CRYPTON, Future Systems, Inc. (Chae Hoon Lim).

- DEAL, L. Knudsen, R. Outerbridge.

- DFC, CNRS-Ecole Norrnale Superiere (S. Vaudenay).

- E2, NTT Nippon Telegraph and Telephone Corporation (M. Kanda).

- FROG, TecApro Intemational S.A. (D. Georgoudis, Leroux, Chaves).

- HPC, R. Schoeppel.

- LOKI97, L. Brown, J. Pieprzyk, J.Seberry.

- MAGENTA, Deutshe Telekom AG (K. Huber).

- MARS, IBM (N. Zunic).
- RC6, RSA Laboratories (Rivest, M. Robshaw, Sidney, Yin).
- RIJNDAEL, J. Daemen, V. Rijmen.
- SAFER+, Cy1ink Corporation (L. Chen).
- SERPENT, R. Anderson, E. Biham, L. Knudsen.
- TWOFISH, B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, N. Ferguson.

En el congreso "Second AES Candidate Conference "([2]), celebrado en marzo de 1999, se discutió los resultados de las numerosas pruebas y ejercicios de criptoanálisis realizados por la comunidad criptográfica mundial sobre los quince candidatos.
En esta tabla, que aparece en el trabajo [9] y presentada en el AES2, se muestran datos comparativos de los 15 finalistas (CRYPTON v1, es una variante de CRYPTON) de la eficiencia en lenguaje C++ con procesadores Pentium Pro y Pentium II. Se exhiben datos, tanto de la función expansión EK de la clave K (dependiendo de su tamaño), como de los algoritmos cifrado y descifrado. Los datos están en ciclos de reloj para Pentium Pro/II. Los dos valores para el Rijndael y CRYPTON son para el cifrado y descifrado. Las velocidades en las tres útimas filas son en megabits por segundo para una plataforma tipo Pentium Pro a 200MHz.

Basándose en numerosos análisis NIST seleccionó cinco candidatos finalistas: MARS, RC6, Rijndael, Serpent y Twofish. En abril de 2000 se celebró el "Third AES Candidate Conference"en Nueva York, ([3]) donde se presentaron nuevos estudios de evaluación y criptoanálisis de los últimos cinco candidatos.

El aspecto fundamental en este proceso de decisión fue la garantía de seguridad de los algoritmos. Todos tuvieron que pasar distintos ataques criptoanáliticos efectuados, tanto por los autores de algoritmos rivales, como por cualquier otra persona acreditada por el NIST. Los resultados obtenidos para cada algoritmo fueron:

- MARS: seguridad elevado. MARS recibió muchas críticas basadas en su complejidad, lo cual impido su análisis de seguridad durante el proceso de desarrollo del AES.
- RC6: seguridad adecuado. Sin embargo, RC6 recibió críticas debido a su bajo margen de seguridad respecto al que ofrecieron otros finalistas. Por otro lado, RC6 ha sido elogiado por su simplicidad, ayudando a su análisis de seguridad durante el tiempo especificado en el proceso de desarrollo del AES.
- RIJNDAEL: seguridad adecuado. El margen de seguridad es un poco dificil de medir, debido a que el número de rondas cambia con el tamaño de la clave. El margen de seguridad, es de los más bajos, entre los finalistas, pero su elegante estructura matemática ha facilitado su análisis.

- SERPENT: seguridad alta. Serpent también tiene una estructura simple, que ha facilitado su análisis de seguridad durante el tiempo especificado de desarrollo del AES.
- TWOFISH: seguridad alta. El concepto de margen de seguridad tiene menos significado para este algoritmo que para los demás finalistas. La dependencia de las S-cajas de Twofish en solo K/2 bits de entropía en el caso de clave K-bits ha producido algunas especulaciones acerca de posibles ataques, aunque no ha sido probado. También, se ha criticado su complejidad, haciendo dificil su análisis durante el tiempo establecido en el proceso del AES.

El 2 de octubre de 2000, el NIST anunció el algoritmo ganador: Rijndael [6], propuesto por los belgas Vincent Rijmen y Joan Daemen. Los motivos para seleccionar a RIJNDAEL", fue su buena combinación de seguridad-velocidad-eficiencia, sencillez y flexibilidad. Los resultados de la votación del concurso fue:

- 1. RIJNDAEL, 86 votos.
- 2. SERPERNT, 59 votos.
- 3. TWOFISH, 31 votos.
- 4. RC6, 23 votos.
- 5. MARS, 13 votos.

5.2. Las matemáticas

Una de las virtudes del AES, comparado con el DES, es la elegancia en la descripción, desde el punto de vista matemático. Varias operaciones en Rijndael estan definidas al nivel de bytes, entidades de 8 bits, tratados como elementos del cuerpo finito $\mathbb{F}_{2^8}$, es decir un byte $(b_7b_6b_5b_4b_3b_2b_1b_0)$ es considerado como un polinomio $f(x)$ con coeficientes en $\mathbb{F}_2 = Z_2 = \{0, 1\}$:

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4b_3x^3 + b_2x^2 + b_1x + b_0.$$

$$\Updownarrow$$

$$(b_7b_6b_5b_4b_3b_2b_1b_0).$$

A veces utilizaremos la notación hexadecimal para expresar un byte, por ejemplo,

$$(01011110) = x^6 + x^4 + x^3 + x^2 + x = \{5E\}.$$

La suma de dos polinomios es la suma de sus coeficientes módulo 2, y se corresponde con la or-exclusiva ($\oplus$) de dos bytes:

$$(x^6 + x^4 + x^3 + x^2 + x) + (x^7 + x^4 + x^3 + 1) = x^7 + x^6 + x^2 + x + 1$$

$$\Updownarrow$$

$$(01011110) \oplus (10011001) = (11000111).$$

La multiplicación en el cuerpo $\mathbb{F}_{2^8}$ se corresponde con la multiplicación de polinomios módulo un polinomio irreducible de grado 8. El polinomio propuesto por Rijndael es :

$$m(x) = x^8 + x^4 + x^3 + x + 1.$$

Multiplicar dos polinomios $f(x)$ y $g(x)$ módulo el polinomio $m(x)$, consiste en tomar el resto de la división de $f(x)g(x)$ entre $m(x)$.

$$(x^6 + x^4 + x^3 + x^2 + x) \times (x^7 + x^4 + x^3 + 1) \equiv x^{13} + x^{11} + x^6 + x^3 + x^2 + x \equiv$$

$$x^7 + x^6 + x^4 + x^3 + x + 1 \text{ mód } m(x).$$

Esta operación es asociativa, conmutativa, tiene un elemento neutro $\{01\}$, y cada elemento $f(x)$ tiene inverso $f(x)^{-1}$ en el cuerpo $\mathbb{F}_{2^8} = \mathbb{F}_2[x]/m(x)$. Para determinar este el inverso, se utiliza el famoso algoritmo extendido de Euclides a los polinomios $f(x)$ y $m(x)$ obteniendo

$$f(x)g(x) + m(x)n(x) = 1,$$

de lo que deducimos que

$$f(x)g(x) = 1 \text{ mód } m(x), \quad g(x) = f(x)^{-1}.$$

La multiplicación por x es muy simple y útil: todas las multiplicaciones entre polinomios pueden ser expresadas por éstas y por or-exclusivas. Por ejemplo,

$$f(x)(x^2 + 1) = x(x(f(x))) \oplus f(x).$$

Si $b_7 = 0$, la multiplicación es un desplazamiento a la izquierda : $x(01101110) = (11011100)$.
En cambio si $b_7 = 1$, la multiplicación es un desplazamiento a la izquierda seguido de una or-exclusiva con $(00011011) = \{1B\}$,

$$x(10101100) = (01011001) \oplus (00011011) = (01000001).$$

Rijndael opera, también, con palabras. Cada palabra es un fila de 4 bytes que es tratado como un polinomio de grado 3 con coeficientes en $\mathbb{F}_{2^8}$:

$$[a_3a_2a_1a_0] \mapsto a_3y^3 + a_2y^2 + a_1y + a_0,$$

donde los a_i son bytes, es decir, elementos del cuerpo $\mathbb{F}_{2^8}$. Para sumar dos palabras se suman los coeficientes (or-exclusivo, ya que nos movemos sobre el cuerpo $\mathbb{F}_{2^8}$). La multiplicación de dos palabras es un polinomio de grado menor o igual a 6, para que sea otra palabra necesitamos reducirlo módulo un polinomio de grado 4. En el algoritmo Rijndael el polinomio es

$$y^4 + 1.$$

Por lo tanto, trabajaremos en el anillo $\mathbb{F}_{2^8}[y]/(y^4+1)$. Este polinomio tiene la siguiente propiedad interesante:

$$y^i \text{ mód } (y^4 + 1) = y^{i \text{ mód } 4},$$

permitiendo realizar la multiplicación, sólo con or-exclusiva y desplazamientos. Denotamos $a(y) \bigotimes b(y)$ a la operacion $a(y)b(y)$ mód $y^4 + 1$,

$$a(y) \otimes b(y) = a(y)b(y) \text{ mód } y^4 + 1.$$

Los coeficientes $a(y), b(y)$ y $y^4+1 = \{01\}y^4+\{01\}$ son elementos de $\mathbb{F}_{2^8}$ y que $y^4+1 = (y^2 + 1)^2$ no es irreducible, con lo que no todos los elementos tienen inversos modulares. Rijndael elige un polinomio fijo $c(y)$, que es primo con $y^4 + 1$ y por lo tanto es inversible módulo $y^4 + 1$:

$$c(y) = \{03\}y^3 + \{01\}y^2 + \{01\}y + \{02\}.$$

El inverso es $d(y) = \{0E\} + \{09\}y + \{0D\}y^2 + \{0B\}y^3$, es decir:

$$d(y) \otimes c(y) = 1.$$

5.3. Algoritmo

Rijndael cifra bloques de 128, 192 ó 256 bits (4,6 ó 8 palabras) con claves, también, de 128, 192 ó 256 bits. El estándar AES solo permite cifrar y descifrar bloques de 128 bits. Al número de palabras del bloque lo denotamos por N_b y por N_k al número de palabras de la clave. Para cifrar un bloque Rijndael realiza N_r iteraciones o vueltas que depende de N_b y N_k siguiendo la siguiente tabla:

N_r	$N_b = 4$	$N_b = 6$	$N_b = 8$
$N_k = 4$	10	12	14
$N_k = 6$	12	12	14
$N_k = 8$	14	14	14

Internamente Rijndael utiliza estados S, que son matrices $4 \times N_b$ (de cuatro filas por N_b columnas), donde cada elemento de la matriz es un byte. Rijndael tiene la *función input* al comienzo que transforma un bloque en un estado y la *función output* al final que transforma un estado en un bloque. Cada byte de un bloque esta numerado desde 0 a $4(N_b - 1)$ la función input coloca al elemento de índice n en la fila $i = n$ mód $4, i = 0, 1, 2, 3$. Y en la columna $j = \lfloor n/4j \rfloor, j = 0, \ldots, N_b - 1$ de un estado. A su vez la transformación output

inserta al elemento de la fila i y la columna j de un estado en la posición $n = 4i + j$ de un bloque.

Un bloque al pasar a un estado, cada palabra es una columna, y el índice i denota un byte dentro de una palabra y j indica una palabra dentro de un bloque.

Para cifrar un bloque con una clave se hacen N_r rondas, en cada una de ellas intervienen 4 transformaciones o funciones invertibles:

- **SB** SubBytes (substitución de bytes),
- **SR** ShiftRows (desplazamiento de filas),
- **MC** MixColumns (mezcla de columnas) y
- **ARK** AddRoundKey (añadir elementos a la clave).

Tienen como objetivo dotar al algoritmo resistencia frente ataques criptoanalíticos, en particular, al criptoanálisis lineal y diferencial. En este contexto, podemos decir que las transformaciones lineales MC y SR proporcionan la *difusión*, la transformación SB es no lineal, (lo parejo a las S-cajas del DES) que junto a la ARK (un or-exclusiva entre el estado intermedio y la subclave intermedia $EK(i)$ en cada ronda i) proporcionan dosis de *confusión* al criptosistema. Podemos expresar este esquema en forma de pseudocodigo como sigue:

- $S := input(\mathbf{x}); EK(K);$
- $S_0 := ARK(S, EK(0));$
- $S_i := ARK(MC(SR(SB(S_{i-1}))), EK(i)), i = 1, \ldots, N_r - 1;$
- $S_{N_r} := ARK(SR(SB(S_{N_r-1})));$
- $\mathbf{y} := output(S_{N_r}).$

Donde $\mathbf{x}$ es el bloque en texto claro e $\mathbf{y}$, es el bloque en texto cifrado. Pasamos, a detallar cada una de estas transformaciones:

Substitución de Bytes SB

SB tiene como entrada y como salida un estado. Esta operación actúa independientemente sobre cada byte del estado mediante otra transformación F de modo que

$$F(\alpha) = \alpha', \quad \alpha' = (b_7'b_6'b_5'b_4'b_3'b_2'b_1'b_0').$$

Si $\alpha \neq \{00\}$, pongamos $\alpha^{-1} = (x_7x_6x_5x_4x_3x_2x_1x_0)$, el inverso de α en el cuerpo $\mathbb{F}_{2^8}$; si $\alpha = \{00\}$, pongamos $\alpha^{-1} = \{00\}$. Consideramos la matriz circulante $\mathbf{A}$ definida por el vector-byte $(10001111) = \{8F\}$, es decir,

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$F(\alpha) = \mathbf{A}(x_0x_1x_2x_3x_4x_5x_6x_7)^t + \{C6\})^t = (b'_0b'_1b'_2b'_3b'_4b'_5b'_6b'_7)^t,$$

donde, en general, β^t es el traspuesto del vector-byte β.

Por ejemplo $SB(\{53\} = \{ED\}$. F no es una aplicación lineal, puesto que involucra el inverso del byte α.

Por otro lado, la matriz $\mathbf{A}$ es una matriz inversible en el $\mathbb{F}_2$, por lo tanto la inversa de F es muy sencilla de calcular, se tiene:

$$F(\alpha') = (\quad \mathbf{A^{-1}} \quad [\quad (b'_0b'_1b'_2b'_3b'_4b'_5b'_6b'_7)^t + \{C6\}^t \quad] \quad)^{-1} = \alpha.$$

Desplazamiento de Filas SR

La transformación SR tiene como entrada y como salida un estado. Esta operación actúa sobre las filas mediante desplazamientos cíclicos a la izquierda, la primera fila no es desplazada, a la segunda fila se le aplican c_1 desplazamientos, c_2 a la tercera, y c_3 la cuarta. Donde c_1, c_2 y c_3 dependen de N_b:

N_b	c_1	c_2	c_3
4	1	2	3
6	1	2	3
8	1	3	4

La inversa de SR aplica a la fila i un total de $N_b - c_i$ desplazamientos cíclicos a la izquierda con $c_0 = 0$.

Mezcla de columnas MC

Esta operación, también, tiene como entrada y salida un estado. MC actua sobre cada columna, manipulandola como un polinomio con coeficientes el cuerpo $\mathbb{F}_{2^8}$. Multiplicando por el polinomio fijo $c(y) = \{03\}y^3 + \{01\}y^2 + \{01\}y + \{02\}$ y reduciendo módulo el polinomio

y^4+1, es decir la operación $\otimes$. Cada columna j tiene 4 bytes, la palabra $(a_0^j, a_1^j, a_2^j, a_3^j)$, lo interpretamos como el polinomio $a_0^j + a_1^j y + a_2^j y^2 + a_3^j y^3$, entonces

$$(a_0^j + a_1^j y + a_2^j y^2 + a_3^j y^3) \otimes c(y) = b_0^j + b_1^j y + b_2^j y^2 + b_3^j y^3.$$

Así, $MC(a_0^j, a_1^j, a_2^j, a_3^j) = (b_0^j, b_1^j, b_2^j, b_3^j)$.

Como los polinomios $c(y)$ y y^4+1 son primos entre sí, podemos determinar el inverso modular y por tanto la transformación inversa de MC, que será analoga a esta, cambiando $c(y)$ por su inverso $d(y)$, es decir,

$$(b_0^j + b_1^j y + b_2^j y^2 + b_3^j y^3) \otimes d(y) = a_0^j + a_1^j y + a_2^j y^2 + a_3^j y^3.$$

Añadir elementos a la clave ARK y la expansion de la clave EK

La función ARK tiene como entrada un estado S y una fila de N_b palabras y como salida otro estado S'. La fila de palabras de entrada lo denotamos por $EK(i) = (w(4i), w(4i+1), \ldots, w(4i+N_b-1))$ donde i representa la iteración en que nos encontramos y cada w es una palabra. La transformación ARK es una or-exclusiva de la palabra correspondiente a la columna j del estado S la palabra $w(4i+j), j = 0, \ldots, N_b$ transformándolo en el nuevo estado S'.

La función inversa de ARK es ella misma, puesto que es una or-exclusivo.

$EK(i)$ son palabras que se obtienen mediante la función expansión de clave EK. Esta nueva función EK, tiene como entrada la clave K y como salida $N_b(N_r+1)$ palabras:

$$EK(K) = (w(0), w(1), \ldots, w(N_b(N_r+1)).$$

Las primeras N_k palabras se corresponden con las de la clave K. Las siguientes palabras se obtienen mediante el siguiente simple algoritmo:

- Si i mód $N_k = 0$, entonces $w(i) = SW(RW(w(i-1))) \oplus Rcon(i/N_k) \oplus w(i-N_k)$.
- Si i mód $N_k = 4$ y $N_k > 6$, entonces $w(i) = SW(w(i-1)) \oplus w(i-N_k)$.
- El resto de las posibilidades, $w(i) = w(i-1) \oplus w(i-N_k)$

donde SW actúa sobre cada byte de una palabra igual que la función F descrita en la transformación de sustitución de bytes. RW actua del siguiente modo:

$$RW(a_0a_1a_2a_3) = (a_1a_2a_3a_0)),$$

donde cada a_i es un byte. Finalmente, la función $Rcon$ tiene como entrada un entero n y como salida una palabra,

$$Rcon(n) = (a_0a_1a_2a_3), \quad a_1 = a_2 = a_3 = \{00\}, \quad a_0 = \{02\}^{-1}.$$

EL proceso de descifrado

El proceso de descifrado es muy similar al del cifrado, sólo es necesario realizar todas las operaciones en orden inverso y usar la generación de la clave, también, en orden inverso. Además, se deben aplicar las transformaciones inversas, ya descritas en el los pasos anteriores.

- $S := input(\mathbf{y}); EK(K);$
- $S_{N_r} := SB^{-1}(SR^{-1}(ARK(S, EK(N_r))));$
- $S_{N_r-i} := SB^{-1}(SR^{-1}(MC^{-1}(ARK(S_{N_r+1-i}, EK(N_{r+1-i})), i = 1, \ldots, N_r - 1;$
- $S_0 := ARK(S_1, EK(0));$
- $\mathbf{x} := output(S_0).$

Donde SB^{-1}, SR^{-1} y MC^{-1} son las transformaciones inversas de SB, SR y MC, respectivamente. Con algunas minimas modificaciones, también, Rijndael puede ser adaptado a un criptosistema cifrado-descifrado, es decir, ambos procesos se realizan de forma similar. La importancia de este hecho reside en la eficiencia en la implementación.

5.4. Análisis de seguridad

Obviamente, de momento, no existen ningúna técnica para descifrar el algoritmo Rijndael más eficiente que la búsqueda exhaustiva de claves mediante un ataque de fuerza bruta. Esto implica que si utilizamos claves de 256 bits, el atacante debería generar 2^{256} combinaciones para asegurarse que ha descifrado un texto. El diseño de la S-caja obtenida mediante la transformación SB está basada en la protección frente ataques de tipo criptoanálisis diferencial y lineal. Los criterios seguidos para construirlas fueron :

- Invertibles.
- Minimizar la correlación entre las combinaciones lineales de bits a la entrada con las combinaciones de bits a la salida.
- Dificultar manipulaciones algebraicas, para prevenir a ataques de interpolación.

En principio, se podrían sustituir por otra S-caja verificando los criterios de arriba. Por otro lado, la estructura del cifrador y el número de vueltas está definido incluso para usar una S-caja que no optimice las propiedades contra ataques. El número a desplazar en la transformación SR, entre otras cosas, fueron elegidos para ofrecer resistencia contra ataques, como el basado en "truncated differentials". Finalmente, el número de rondas fue determinado por los autores, buscando el mínimo número de vueltas necesarias para ofrecer una margen de seguridad considerable. Por ejemplo, para un tamaño de bloque y de clave de 128 bits se utilizan 10 vueltas, esto es así porque, no se han encontrado atajos en ataques para versiones reducidas con mas de 6 vueltas. A estas 6, los autores, le añaden otras 4 vueltas como margen de seguridad.

5.5. Modos de operación

Los cuatro modos que fueron diseñados por DES para aumentar la seguridad, pueden ser adaptados por cuaquier cifrador en bloque y, en particular para el Rijndael.

- ECB (Electronic Codebook Mode), para mensajes cortos.
- CBC (Cipher Block Chaining Mode) para mensajes largos.
- CFB (Cipher Feedback Mode) para cifrar bit por bit ó byte por byte.
- OFB (Output Feedback Mode) el mismo uso pero evitando propagación.

En el modo ECB el texto claro es dividido en bloques de 128 bits que se cifran uno a uno y por separado usando el AES. La concatenación de los bloques cifrados da lugar al texto cifrado. Este modo de funcionamiento tiene la ventaja de que funciona bien en canales con ruido. Un fallo en la transmisión tan solo afecta a un bloque de 128 bits, no al mensaje completo. Por otro lado, es susceptible a ataques estadísticos y/o ataques sobre la clave con un texto en claro conocido.

Una alternativa es el CBC, se divide el texto en bloques y se hace depender el bloque i-ésimo del texto cifrado/descifrado del (i-1)-ésimo:

$$y_i = E_K(x_i \oplus y_{i-1}), \quad x_i = D_K(y_i) \oplus y_{i-1}.$$

El primer bloque de entrada al proceso AES esta formado por la or-exclusiva entre el primer bloque del mensaje y los 128 bits de un vector inicial fijo y_0. De este modo, se produce un encadenamiento (chaining) entre los distintos bloques, y el resultado de cifrar cada uno de ellos depende de todos los anteriores. Debido a esta última característica, el último bloque del texto cifrado puede actuar como firma digital o checksum del resto, permitiendo certificar que no ha sido alterado. En este modo de funcionamiento un error en el texto cifrado tan sólo afecta al descifrado de dos bloques. Este modo suele utilizarse para ciframeinto y autentificación.

En modo CFB, también, se parte de un vector inicial fijo convenido y_0 de 128 bits,

$$z_i = E_K(y_{i-1}), \quad y_i = x_i \oplus z_i.$$

Este método permite descifrar cualquier parte del texto cifrado sin conocer el resto. Además, tanto este modo como el siguiente realizan el cifrado a nivel de carácter, de modo que el texto cifrado va surgiendo de modo continuo (stream mode) y no por bloques. Es comun que se utilize para cifrar caracteres individuales.

Finalmente, OFB funciona de forma analoga al CFB, pero el byte realimentado en la cola es directamente el más significativo del cifrado de la misma.

$$z_i = E_K(z_{i-1}), \quad y_i = x_i \oplus z_i.$$

Un error en un bit del texto cifrado tan sólo afecta a un bit en el texto descifrado, por lo que este modo suele utilizarse para comunicaciones vía satélite.

Hay varias variantes de los modos OFB y CFB denominados k-bit feedback modos, $1 \leq k \leq 128$.

Referencias

[1] Proceedings of the Second Advanced Encryption Standard Candidate Conference AES1, (Ventura, California, USA). National Institute of Standards and Technology (NIST), August 1998.

[2] Proceedings of the Second Advanced Encryption Standard Candidate Conference AES2, (Rome, Italy). National Institute of Standards and Technology (NIST), March 1999.

[3] Proceedings of the Second Advanced Encryption Standard Candidate Conference AES3, (New York, NY USA). National Institute of Standards and Technology (NIST), April, 2000.

[4] E. Biham and A. Shamir, *Differential cryptanalysis of the DES*. Springer-Verlag, 1993.

[5] D. Coppersmith, 'The data encryption standard DES and its strengh against attacks', *IBM Journal of Research and Development*, **38** (1994), 243–250.

[6] J. Daem and V. Rijmen, *The Rijndael algorithm*. Springer-Verlag, 2002.

[7] Data Encryption Standard (DES). Federal Information Processing Standard Publication, 46, 1977.

[8] H. Feistel, 'Cryptography and Computer privacy', *Scientific American*, **228**(5) (1973), 23–23.

[9] B. Gladman. 'Implementtaion Experience with AES Candidates Algorithms'. Proceedings of the Second Advanced Encryption Standard Candidate Conference, (Rome, Italy). National Institute of Standards and Technology (NIST), March 1999.

[10] J. Gutierrez y A. Ibeas, *Criptografía*. Protocolos Criptograficos y Seguridad en Redes (Eds. J. Gutiérrez, T. Ayuso), Servicio de Publicaciones Universidad de Cantabria, 2003.

[11] NIST, 'Request for candidates nominations for the Advanves Encryption Satandard', *Federal Register* , **62**(177), September, 1997.

[12] M. Matsui, 'Linear cryptanalysis method for DES cipher', *Lectures Notes in Computer Science* , **765**, (1994), 386-397.

[13] J.Rifà *Quantum Computing and Quantum Cryptography*. Protocolos Criptograficos y Seguridad en Redes (Eds. J. Gutiérrez, T. Ayuso), Servicio de Publicaciones Universidad de Cantabria, 2003.

[14] C. Shannon, 'Communication Theory of Secret Systems', *Bell System Technical Journal*, **28** (1949), 657–715.

[15] B. Schneider, *Applied Criptography*. J. Wiley, 1994.

[16] M. Smid and D. Branstad, 'DES: past and future', *Contemporary Cryptology. The science of information Integrity* , 43–64, IEEE press 1992.

[17] D.R. Stinson, *Cryptography. Theory and Practice*, Second edition, C.R.C., 2002.