Advanced Encryption Standard: Introducción y discusión

Samuel Gómez Sánchez

Teoría de la Información y Teoría de códigos

Abstract

En este documento presentamos el criptosistema AES. Dada su vocación introductoria, se presentan primeramente las estructuras criptográficas que se han considerado relevantes para la cabal comprensión del criptosistema y su potencia. A continuación se realiza una exposición del algoritmo que constituye el núcleo del sistema, y se presenta para terminar cierta información sobre los ataques tentativos contra el mismo.

Advanced Encryption Standard: introducción y discusión

Preliminares criptográficos

Definición. Un criptosistema es una quíntupla (M, C, K, E, D) tal que

M es un conjunto de “mensajes de texto plano”.

C es un conjunto de “mensajes de texto cifrado”.

K es un conjunto de “claves”.

E es un conjunto de aplicaciones $\varepsilon : M\,x\,K \rightarrow C$.

D es un conjunto de aplicaciones $\delta : C\,x\,K \rightarrow M$, de modo que $\delta(\varepsilon(m,k),k') = m'$, $m = m' \Leftrightarrow k = k'$.

Un criptosistema permite a dos partes compartir información de modo seguro sobre un canal público. Uno de los principales tipos de criptosistemas son los *criptosistemas de clave simétrica* (a cuya discusión nos limitamos en este documento). En estos casos, las dos partes que desean comunicarse comparten una clave *k* y un algoritmo *(ε, δ)*, de tal modo que pueden comunicarse sobre un canal público sin que nadie, en ausencia de conocimiento de la clave *k*, pueda comprender sus comunicaciones.

Diremos que un criptosistema es *perfectamente seguro* sobre un espacio de texto plano *M* si para cualquier distribución de probabilidad sobre *M*, se cumple la siguiente relación entre cada mensaje $m \in M$ y cada texto cifrado $c \in C$ con probabilidad no nula

$$P(m|c) = P(m)$$

lo que se prueba fácilmente como equivalente a, dados dos mensajes m_1 y m_2,

$$P(c|m_1) = P(c|m_2)$$

Es decir: las distribuciones de probabilidad de los mensajes de texto plano y los mensajes de texto cifrado son independientes, de modo que no es posible obtener información sobre *M* a partir de *C*.

Teorema (de Shannon). Sea *(M, C, K, E, D)* un criptosistema perfectamente seguro. Entonces $|K| \geq |M|$. Se puede probar que un criptosistema es perfectamente seguro si

1. Cada clave se selecciona con igual probabilidad *1/|K|*.
2. Para cada mensaje $m \in M$ y cada texto cifrado $c \in C$ existe una única clave $k \in K$ tal que para una función de encriptación $\varepsilon, \varepsilon(m,k) = c$.

El problema de utilizar criptosistemas completamente seguros radica por tanto en la necesidad de transmitir – como mínimo – tanta información redundante, correspondiente a la clave, como información valiosa, lo cual obliga a gastar una gran cantidad de recursos de modo evitable. Es por esto que la criptografía de clave simétrica se centra en el estudio de algoritmos donde el tamaño de *K* es mucho menor que el de *M*. Desde esta óptica, se redefine la noción de seguridad como *pseudoaleatoreidad*. Dada una distribución de elementos *D* de un conjunto *S*, decimos que es *pseudoaleatoria* si resulta «indistinguible» de la distribución uniforme de elementos de *S*. La cuestión aquí radica en que, para un observador con información y tiempo limitados, no hay manera de diferenciar el sistema de uno completamente seguro.

Definición 2. Un criptosistema cuyo espacio de texto plano y espacio de texto cifrado es el conjunto $\sum^n$ de palabras de longitud fija *n* sobre el alfabeto $\sum$ se denomina *cifrado en bloque*. Al valor entero positivo *n* lo denominamos *longitud del bloque*. Se prueba fácilmente que las funciones de encriptación de todo cifrado en bloque son permutaciones. El cifrado del César es un ejemplo clásico de criptosistema en bloque de longitud 1. En general, este tipo de cifras de longitud 1 se denominan *cifras de sustitución*. A lo largo de este documento trabajaremos sobre

el alfabeto {0, 1}, pero fácilmente se puede extender lo aquí expuesto a cualquier otro cuerpo finito.

Un cifrado en bloque encripta el mensaje original agrupando los símbolos en grupos (bloques) de dos o más elementos, de tal modo que cada símbolo depende de los adyacentes, y cada bloque se cifra siempre igual, independientemente de su lugar en el mensaje. En consecuencia, dos mensajes cifrados con la misma clave producen siempre mensajes iguales, y para descifrar parte de un mensaje basta descifrarlo a partir del bloque que interese.

Definición 3. Una red de sustitución-permutación (SPN, del inglés *Substitution-Permutation Network*) es un criptosistema en bloque cuya función de encriptación es una aplicación

$$C_k:\{0,1\}^n \to \{0,1\}^n \quad , \quad k=(k_0,\dots,k_r)\in (F^n)^{r+1}$$

Se supone que existe un proceso sistemático para dividir la clave de encriptación k en $r+1$ partes iguales. Esta aplicación se computa mediante $r\in\mathbb{N}$ iteraciones o rondas. La i-ésima ronda, $1\le i\le r$, se calcula del modo siguiente:

1. Para una entrada $x\in\{0,1\}^n$, la entrada de la primera iteración es $x + k_0$.

2. Se aplica la función S

$$S:GF(2^b)\to GF(2^b) \quad , \quad b\in\mathbb{N}$$

a $x + k_0$, dividiendo este valor en $m\in\mathbb{N}$ partes, de tal modo que $n=m\cdot b$.

3. Después, se aplica la permutación P

$$P:(GF(2^b))^m\to(GF(2^b))^m$$

al resultado del paso anterior.

4. Tras ello, se suma el resultado del paso anterior con la componente i-ésima de la clave de encriptación, que llamaremos *clave de la ronda i-ésima*.

5. Por último, la salida de la ronda *i* se convierte en la entrada de la ronda *i+1*. La salida de C_k es la salida de la iteración r-ésima.

El proceso de desencriptar el mensaje tiene lugar simplemente invirtiendo el proceso (para lo cual, las aplicaciones *S* y *P* deben ser inyectivas).

Como el lector podrá observar, no especificamos en ningún momento cuáles son las funciones *S* y *P* (ni si son las mismas en cada ronda, etc.), y tampoco cómo obtener la descomposición de la clave en subclaves para cada ronda; la selección de todos estos elementos determina si el cifrado es seguro o se puede romper de modo trivial, como veremos a continuación.

Nótese que las funciones *S* y *P* por separado son criptográficamente muy débiles: la aplicación de *S* puede pensarse como un cifrado por sustitución, mientras que *P* es una simple transposición. Ahora bien, conviene notar lo siguiente: la bondad de una SPN viene determinada por la medida en que la red consigue incorporar *difusión y confusión* en su funcionamiento. Por *confusión* entendemos el incremento de la complejidad de la relación estadística entre el texto plano y la clave. Esto se puede conseguir haciendo depender cada símbolo del texto cifrado de diferentes partes de la clave. Por *difusión* nos referimos a romper la estructura estadística del texto plano, distribuyéndola por todo el texto cifrado. Esto se consigue haciendo depender cada todos los símbolos de la salida de todos los símbolos de la entrada. Un criptosistema con estas propiedades tiene el aspecto de una función aleatoria a ojos de un observador con tiempo limitado; esto es, un atacante no puede obtener información sobre el sistema (su funcionamiento, sobre la clave, etc.) a partir de los datos disponibles, y por tanto, no puede romperlo.

Por otra parte, aunque los cifrados en bloque son construcciones complejas, resulta en la práctica muy complicado conseguir un cifrado seguro, y sorprendentemente sencillo diseñar

uno fácil de romper. Presentamos a continuación dos herramientas fundamentales que pueden emplearse en el criptoanálisis de cifrados en bloque.

Definición 3. Dadas dos entradas de texto plano x_1 y x_2, y sus textos cifrados correspondientes para la clave k, y_1 e y_2, podemos definir la diferencia entre x_1 y x_2 como $\Delta x = x_1 + x_2$., y la diferencia entre y_1 e y_2, $\Delta y = y_1 + y_2$. En ese caso llamamos *diferencial* al par ordenado $(\Delta x, \Delta y)$.

El *criptoanálisis diferencial* busca debilidades potenciales en un criptosistema en bloque por medio de la búsqueda de diferenciales cuya probabilidad de aparición es mayor de lo esperado para una transformación aleatoria. Para un cifrado de longitud de bloque n, la situación ideal sería que la probabilidad de cada diferencial se encontrara en un entorno de 2^{-n}. Es posible tratar de localizar porciones de la clave por medio de la aplicación de entradas aleatorias al sistema y llevar a cabo un análisis de los diferenciales resultantes.

Definición 4. De modo similar, el *criptoanálisis lineal* busca encontrar relaciones lineales entre la entrada y la salida del sistema. Así, para una entrada $x = x_1x_2\ldots x_n$ y una salida $y = y_1y_2\ldots y_n$, el criptoanálisis lineal considera ecuaciones de la forma $x_{i1} + \ldots + x_{il} + y_{i1} + \ldots y_{il} = 0$. Si la función de encriptación fuera aleatoria, estas ecuaciones lineales sólo podrían ocurrir con probabilidad ½. Por tanto buscando ecuaciones cuya probabilidad sea significativamente superior es posible encontrar la clave completa. En este caso cabe notar que es perfectamente legítimo afirmar que una función para la cual estas relaciones lineales se dan con probabilidad mayor a ½ no es pseudoaleatoria.

Tanto para el criptoanálisis diferencial como para el criptoanálisis lineal, la tarea se vuelve impracticable si es necesario un gran número de intentos para obtener información

significativa. Este es el enfoque que debe buscarse para la construcción de cifrados en bloque seguros.

Criptosistema AES

El criptosistema AES (*Advanced Encryption Standard*) fue desarrollado por V. Rijmen y J. Daemen en respuesta a la salida a concurso público del estándar de cifrado que relevaría al DES como estándar de cifrado del Instituto Nacional de Estándares y Tecnología de Estados Unidos (*National Institute of Standards and Technology*, NIST). El AES es una variante del algoritmo ganador, Rijndael, publicada en la FIPS 197 [1].

Algoritmo. Rjindael es una red de sustitución-permutación que opera con bloques de longitud igual a *Nb* palabras de 32 bits, con un mínimo de 128 y un máximo de 256 bits; la longitud de la clave puede tomar los mismos valores, con un número de subclaves igual a *Nk*. AES restringe el tamaño de los bloques a un valor fijo de 128 bits. A partir de aquí nos referiremos al algoritmo indiferentemente como Rijndael o AES, aunque presentaremos la estructura estandarizada publicada definitivamente como estándar, con un valor de *Nb* igual a 4.

AES estructura los bloques en una matriz de 4 filas y *Nb* columnas (4 en nuestro caso), agrupándolos en sub-bloques de 8 bits. A esta matriz se la denomina *estado* (*state*, en inglés).

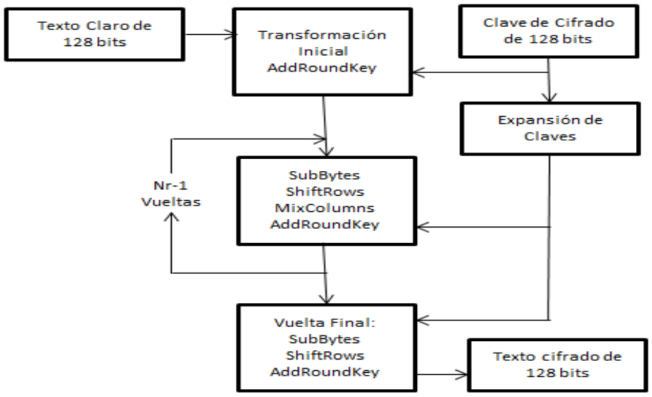
$$\begin{pmatrix} s_{0,0} & s_{1,0} & s_{2,0} & s_{3,0} \\ s_{0,1} & s_{1,1} & s_{2,1} & s_{3,1} \\ s_{0,2} & s_{1,2} & s_{2,2} & s_{3,2} \\ s_{0,3} & s_{1,3} & s_{2,3} & s_{3,3} \end{pmatrix}$$

Para comenzar, se copia el contenido del bloque en la matriz estado. A partir de aquí, a vista de pájaro, el procedimiento tiene tres etapas:

- Una transformación inicial.

- *Nr* – 1 rondas regulares.
- Una ronda final.

El procedimiento completo es como sigue:



AddRoundKey. Supondremos a partir de aquí que el algoritmo trabaja con elementos de GF(2). *AddRoundKey* es la transformación inicial. Consiste en una suma de los primeros 128 bits de la clave con el mensaje claro. Así, dado el estado inicial

$$\begin{pmatrix} en_{0,0} & en_{1,0} & en_{2,0} & en_{3,0} \\ en_{0,1} & en_{1,1} & en_{2,1} & en_{3,1} \\ en_{0,2} & en_{1,2} & en_{2,2} & en_{3,2} \\ en_{0,3} & en_{1,3} & en_{2,3} & en_{3,3} \end{pmatrix}$$

la operación *AddRoundKey* supone la suma de los vectores $\{en_i\}$ con la matriz construida de modo análogo al estado a partir de los 128 primeros bits de la clave:

$$S_0 = \begin{pmatrix} s_{0,0} & s_{1,0} & s_{2,0} & s_{3,0} \\ s_{0,1} & s_{1,1} & s_{2,1} & s_{3,1} \\ s_{0,2} & s_{1,2} & s_{2,2} & s_{3,2} \\ s_{0,3} & s_{1,3} & s_{2,3} & s_{3,3} \end{pmatrix} = \begin{pmatrix} en_{0,0} & en_{1,0} & en_{2,0} & en_{3,0} \\ en_{0,1} & en_{1,1} & en_{2,1} & en_{3,1} \\ en_{0,2} & en_{1,2} & en_{2,2} & en_{3,2} \\ en_{0,3} & en_{1,3} & en_{2,3} & en_{3,3} \end{pmatrix} + \begin{pmatrix} k_0 & k_4 & k_8 & k_{12} \\ k_1 & k_5 & k_9 & k_{13} \\ k_2 & k_6 & k_{10} & k_{14} \\ k_3 & k_7 & k_{11} & k_{15} \end{pmatrix}$$

El número de vueltas *Nr* - 1 depende de la longitud de la clave: *Nr* = 10 para *Nk* = 4 (claves de 128 bits), *Nr* = 12 para *Nk* = 6 (claves de 192 bits), y *Nr* = 14 para *Nk* = 8 (claves de 256 bits). En cada vuelta regular tienen lugar las siguientes operaciones:

- *SubBytes:* sustitución no lineal de bytes que denominaremos *caja S*.
- *ShiftColumns*: permutación cíclica de las últimas tres filas del estado.
- *MixColumns*: combinación lineal de las columnas del estado.
- *AddRoundKey*: suma del estado con la subclave de ronda correspondiente (siguientes 128 bits).

SubBytes. La caja S corresponde con la aplicación siguiente: se interpreta el byte a transformar como un polinomio en *GF(2)*. Después, se halla su inverso multiplicativo en $GF(2^8) = GF(2)[x]/(x^8 + x^4 + x^2 + x + 1)$ – el polinomio 0 se mantiene invariante –. Por último, se efectúa una transformación afín sobre GF(2) definida por, dado el inverso de un byte, $B^{-1} = (b_0, b_1, \ldots, b_7)$

$$\begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

Esto supone que la salida es la suma de múltiples rotaciones del byte como vector de $GF(2)^8$. En conjunto, la operación es, si llamamos *A* a la matriz de 8x8 elementos de la expresión anterior, y *v* al vector sumado

$$s_{i,j} \rightarrow A \cdot s_{i,j}^{-1} + v$$

Los objetivos de esta transformación son minimizar la correlación entre la salida y la entrada, y la probabilidad de propagación de diferencias en el texto plano.

Esta aplicación es 1-1, por lo que en la operación de descifrado simplemente se invierte la relación entre bytes de entrada y bytes de salida.

ShiftRows. Esta transformación supone la siguiente permutación cíclica de las tres últimas filas del estado

$$\sigma = (s_{00})(s_{10})(s_{20})(s_{30})(s_{01}, s_{31}, s_{21}, s_{11})(s_{02}, s_{22})(s_{12}, s_{32})(s_{03}, s_{13}, s_{23}, s_{33})$$

$$\begin{pmatrix} s_{0,0} & s_{1,0} & s_{2,0} & s_{3,0} \\ s_{0,1} & s_{1,1} & s_{2,1} & s_{3,1} \\ s_{0,2} & s_{1,2} & s_{2,2} & s_{3,2} \\ s_{0,3} & s_{1,3} & s_{2,3} & s_{3,3} \end{pmatrix} \rightarrow \begin{pmatrix} s_{0,0} & s_{1,0} & s_{2,0} & s_{3,0} \\ s_{1,1} & s_{2,1} & s_{3,1} & s_{0,1} \\ s_{2,2} & s_{3,2} & s_{0,2} & s_{1,2} \\ s_{3,3} & s_{0,3} & s_{1,3} & s_{2,3} \end{pmatrix}$$

Esta transformación también proporciona difusión, convirtiendo al algoritmo en resistente a ataques diferenciales.

De nuevo, es una aplicación inyectiva que se invierte en la fase de descifrado mediante la permutación inversa.

MixColumns. Esta es una transformación lineal que, junto con *ShiftRows*, constituye la principal fuente de difusión del AES. La aplicación utilizada opera sobre el estado, columna a columna. Dada una columna $S_i = (s_{i,0}, s_{i,1}, s_{i,2}, s_{i,3})$, la transformación es la siguiente

$$\begin{pmatrix} s'_{i,0} \\ s'_{i,1} \\ s'_{i,2} \\ s'_{i,3} \end{pmatrix} = \begin{pmatrix} 10_2 & 11_2 & 01_2 & 01_2 \\ 01_2 & 10_2 & 11_2 & 01_2 \\ 01_2 & 01_2 & 10_2 & 11_2 \\ 11_2 & 01_2 & 01_2 & 10_2 \end{pmatrix} \times \begin{pmatrix} s_{i,0} \\ s_{i,1} \\ s_{i,2} \\ s_{i,3} \end{pmatrix}$$

Esto supone tratar cada byte como polinomios sobre $GF(2^8)$, de tal modo que los bits son los coeficientes de los polinomios. Los bytes resultantes $s_{i,j}$ son combinaciones lineales de los

polinomios de entrada. En la operación de descifrado se invierte la transformación y se aplica sobre cada columna.

AddRoundKey. Por último se suman los bytes del estado con la subclave de la vuelta *Kr* correspondiente. Ahora bien, hasta aquí no hemos mencionado cómo se obtienen las subclaves de cada vuelta. La longitud de la clave en AES puede ser de 128, 192 o 256 bits, y es necesario generar muchas subclaves a partir de ella: la subclave inicial y *Nr* subclaves de vuelta, todas ellas de 32 bits.

Esto se consigue mediante el siguiente procedimiento:

Esquema de claves del AES. Sean

- *N*, la longitud de la clave en palabras de 32 bits: 4 para AES-128, 6 para AES-192 y 8 para AES-256.
- $K_0, K_1, \ldots, K_{N-1}$ las palabras de 32 bits de la clave original.
- $R = Nr$ - 1 el número de rondas necesarias: 11 para AES-128, 13 para AES-192 y 15 para AES-256.
- $W_0, W_1, \ldots, W_{4R-1}$ las palabras de 32 bits de la *clave expandida*.
- *RotWord* es una aplicación sobre una palabra de 32 bits *P = (p0, p1, p2, p3)* expresada en coordenadas de GF(8) tal que

$$RotWord : GF(8)^4 \rightarrow GF(8)^4$$
$$(p_0, p_1, p_2, p_3) \rightarrow (p_1, p_2, p_3, p_0)$$

- *SubWord* es la aplicación de la caja S, S, definida previamente, a cada uno de los cuatro elementos de una palabra de 32 bits:

$$SubWord : GF(8)^4 \rightarrow GF(8)^4$$
$$(p_0, p_1, p_2, p_3) \rightarrow (S(p_0), S(p_1), S(p_2), S(p_3))$$

- La constante de la ronda i-ésima, que es una función definida como

$$\text{rcon}_i = [rc_i \quad 0 \quad 0 \quad 0]$$

donde rc_i son ocho bits definidos por

$$\text{rc}_i = \begin{cases} 1, & \text{si } i = 1 \\ 2 \cdot rc_{\text{i-1}}, & \text{si } i > 1 \text{ y } rc_{\text{i-1}} < 1000000_2 \\ (2 \cdot rc_{\text{i-1}}) + 00011011_2, & \text{si } i > 1 \text{ y } rc_{\text{i-1}} \geq 1000000_2 \end{cases}$$

Dada una clave de cifrado K_0, K_1, …, K_{N-1}, la expansión de la clave de Rijndael se realiza construyendo una secuencia W_0, W_1, …, W_{4R-1} de palabras de 32 bits. En AES-128 se generarán 44 palabras de 32 bits, en AES-192, 52 palabras de 32 bits, y en AES-256, 60 palabras de 32 bits. La palabra W_i se genera según la siguiente expresión

$$W_i = \begin{cases} \text{K}_\text{i}, & \text{si } i < N \\ \text{W}_{\text{i-N}} + SubWord(RotWord(W_{\text{i-1}})) + rcon_{\text{i}/N} & \text{si } i \geq N \text{ y } i \equiv 0 \,(mod\, N) \\ \text{W}_{\text{i-N}} + SubWord(W_{\text{i-1}}) & \text{si } i \geq N,\ N > 6, \text{ y } i \equiv 4 \,(mod\, N) \\ \text{W}_{\text{i-N}} + W_{\text{i-1}} & \text{en otro caso} \end{cases}$$

La ronda final del algoritmo es idéntica a una ronda normal, sólo que sin aplicar la transformación *MixColumns*.

* * *

En resumen, AES es una red de sustitución-permutación clásica, que requiere 10, 12 o 14 rondas de encriptación en total – en función de la longitud de la clave. AES está orientado a byte, y depende notablemente de operaciones en el cuerpo GF(2^8) – o equivalentemente el anillo de polinomios con coeficientes en GF(2), o el espacio vectorial GF(2)^8–.

Criptoanálisis del AES

En este apartado damos una introducción al análisis de las vulnerabilidades del AES.

Ataques de canal lateral. Estos ataques son los más frecuentes al AES. Se basan en información conocida sobre la implementación física del sistema, más que en debilidades del algoritmo. Estos ataques pueden consistir, por ejemplo, en someter al equipo a perturbaciones físicas y realizar un examen similar al análisis diferencial (entre los cifrados con y sin perturbación); también existen ataques que dependen del análisis de la sincronización de procesos en el equipo, o en el análisis temporal, que permite obtener claves basándose en el tiempo necesario para la ejecución de un algoritmo complejo (|| este tipo de ataque se ha utilizado también contra algoritmos de exponenciación modular, como RSA o Diffie-Hellman).

Criptanálisis basado en claves relacionadas. Aunque basados en un modelo poco realista, y fácilmente evitable, aunque produce buenos resultados. Este tipo de ataque se basa en buscar información sobre el sistema a través del análisis de los resultados de cifrado con claves que presentan una relación conocida (e.g., un cierto patrón de bits). Son fácilmente evitables no permitiendo claves similares. En AES no son un peligro real, pues los cálculos más optimistas arrojan una complejidad impractiable para algoritmos bien implementados.

Ataques MQ sobre GF(2). El problema de resolver un sistema de *m* ecuaciones cuadráticas con *n* incógnitas sobre un cuerpo arbitrario *K* es NP-difícil, para cualquier K. Restringiéndonos a los cuerpos finitos que venimos manejando, su solución es de complejidad exponencial en general. De hecho, algunos criptosistemas se basan en la dificultad de este problema.

Sin embargo, si $m > n$, existen algoritmos que permiten resolver el problema, para sistemas «suficientemente» sobredeterminados, en tiempo polinómico. El Ataque de Linealización Dispersa Extendido (*eXtended Sparse Linearazation*, XSL) es un método de

criptanálisis de cifrados en bloque que, contando con un algoritmo eficiente de resolución del problema MQ, puede utilizarse para atacar ciertos cifrados.

En 2002 se observó que AES podía ser reducido a un sistema de ecuaciones cuadráticas, con variables que representan el texto plano, el cifrado, las claves, y valores intermedios. El punto más débil del algoritmo parecen ser las cajas S, que se basan en una simple inversión para eliminar la linealidad.

Ha habido largas disputas sobre si es posible utilizar este hecho como una ventaja para atacar AES. Los resultados más optimistas, sin embargo, aún siendo más rápidos que un ataque de fuerza bruta, son enormes, de modo que cualquier debilidad conocida hasta la fecha posee poca peligrosidad en el nivel práctico.

Conclusiones

AES es un algoritmo algebraicamente muy sencillo, y sin embargo, a pesar de la constante investigación en criptografía para intentar encontrar una debilidad que consiga romperlo, hasta el día de hoy ningún ataque conocido permite a un atacante sin conocimiento adicional atacar un sistema encriptado con él si el algoritmo ha sido correctamente implementado.

Referencias

[1] *Announcing the Advanced Encryption Standard (AES).* Federal Information Processing Standards Publication 197. United Stated States National Institute of Standards and Technology (NIST). November 26, 2001. Recuperado de https://csrc.nist.gov/csrc/media/publications/fips/197/final/documents/fips-197.pdf el 13 de abril de 2019.

[2] Buchmann, J. (2001). *Introduction to cryptography.* New York: Springer.

[3] Fúster Sabater, A., et al. (2004). *Técnicas criptográficas de protección de datos*. Tercera edición. Paracuellos de Jarama (Madrid): Ra-Ma. Impreso.

[4] Katz, J. (2007). *Introduction to modern cryptography: Principles and protocols*. Chapman & Hall.

[5] Miles, E., & Viola, E. (2015). Substitution-permutation networks, pseudorandom functions, and natural proofs. *Journal of the ACM (JACM)*, *62*(6), 46.

[6] Nover, H. (2005). *Algebraic cryptanalysis of AES: an overview*. University of Wisconsin, USA. Recuperado de http://www.math.wisc.edu/~boston/nover.pdf el 30 de mayo de 2019.