Algoritmos criptográficos simétricos y asimétricos

Computación, Computo Paralelo

Nombres: Jhon Llivicota, Michael Alvarez

Fecha: 31/11/2023

Los algoritmos criptográficos se definen como la ciencia y practica de asegurar la comunicación y la información a través de técnicas de codificación y cifrado. Estos algoritmos son un conjunto de pasos o reglas matemáticas diseñadas para definir como se convierten los datos originales (texto sin formato o "plano") en un formato ilegible (cifrado) mediante un proceso de codificación, y cómo se convierten nuevamente a su forma original (descifrado), de una manera segura.

Los algoritmos criptográficos utilizan operaciones matemáticas como operaciones algebraicas, funciones de hash, operaciones modulares, sustituciones y permutaciones para transformar los datos originales en una forma enmascarada o cifrada. Pueden utilizar claves, que son valores específicos utilizados para cifrar y descifrar datos [1].

Estos algoritmos se dividen principalmente en dos categorías: simétricos y asimétricos. Cada uno tiene su propio conjunto de características y utiliza principios matemáticos diferentes para cifrar y descifrar datos.

Algoritmos Criptográficos Simétricos

Los algoritmos simétricos se basan en el uso de una clave única secreta, esto significa que la misma clave se utiliza tanto para cifrar como para descifrar la información, esto lleva a que todos los usuarios, como mínimo dos, que quieran cifrar o descifrar el mensaje, tengan esta clave secreta, de lo contrario, no podrán hacerlo.

Tenemos dos categorías principales dentro de los algoritmos simétricos que son:

- Algoritmos de Flujo: Generan una secuencia de bits llamada flujo, que se combina con el texto sin formato mediante una operación XOR (o exclusiva). Ejemplos de algoritmos de flujo incluyen RC4, HC-128 y A5/1.
- Algoritmos de Bloque: operan en bloques de datos fijos, cifrando cada bloque de manera independiente. Uno de los más importantes y ampliamente utilizados es el Advanced Encryption Standard (AES) [2].

Advanced Encryption Standard (AES)

También conocido como Rijndael es un algoritmo de cifrado simétrico que reemplazó al algoritmo Data Encryption Standard (DES) debido a su mayor seguridad y eficiencia, demostrando una gran resistencia a ataques criptoanalíticos y de fuerza bruta. Opera en bloques de 128 bits y admite claves de longitud de 128, 192 o 256 bits. Su estructura consiste en rondas de transformación que varían según el tamaño de la clave: 10 rondas para claves de 128 bits, 12 rondas para claves de 192 bits y 14 rondas para claves de 256 bits.

El algoritmo AES tiene varios modos de operación que determinan cómo se cifran bloques de datos más grandes:

- ECB: Electronic Codebook Mode (Modo de Libro de Códigos Electrónico): Este modo cifra cada bloque de datos de forma independiente utilizando la misma clave, lo que puede resultar en patrones repetitivos si los bloques de datos idénticos se cifran de la misma manera.

- CBC: Cipher Block Chaining Mode (Modo de Encadenamiento de Bloque de Cifrado): Antes de cifrar cada bloque, se realiza una operación XOR con el bloque anterior cifrado. Esto asegura que incluso bloques idénticos en el texto sin formato se cifren de manera diferente.
- CFB: Cipher Feedback Mode (Modo de Retroalimentación de Cifrado): Convierte el bloque de cifrado en un flujo de bits para cifrar el siguiente bloque de datos, permitiendo cifrar un flujo continuo de datos y no solo bloques.
- OFB: Output Feedback Mode (Modo de Retroalimentación de Salida): Convierte el bloque de cifrado en un flujo de bits, que luego se utiliza para cifrar el texto sin formato mediante una operación XOR. Similar a CFB, pero con la diferencia de que OFB no tiene efecto de error de propagación [3].

Las operaciones individuales que realiza el algoritmo son sustituciones no lineales SubBytes(), permutaciones ShiftRows(), mezclas de columnas MixColumns() y adición de claves AddRoundKey().

La operación de sustitución no lineal SubBytes(), opera de forma independiente en cada byte del Estado utilizando una tabla de sustitución (S-box). Esta S-box que es invertible, se construye a partir de dos transformaciones:

1. Cálculo del inverso multiplicativo en el campo finito GF(2^8), en este campo se debe calcular el inverso multiplicativo para cada elemento, excepto para el elemento {00}, que se mapea a sí mismo en este contexto. El inverso multiplicativo en un campo finito implica encontrar un número que, cuando se multiplica por un determinado número en ese campo, produce el elemento identidad (que es 1 en este caso). Cabe mencionar que en el campo GF(2^8), las operaciones se realizan en bytes (8 bits) y el inverso multiplicativo se calcula utilizando la aritmética modular.
2. Aplicación de la transformación afín sobre GF(2) que se aplicará a cada byte de datos. La transformación se describe mediante la fórmula:

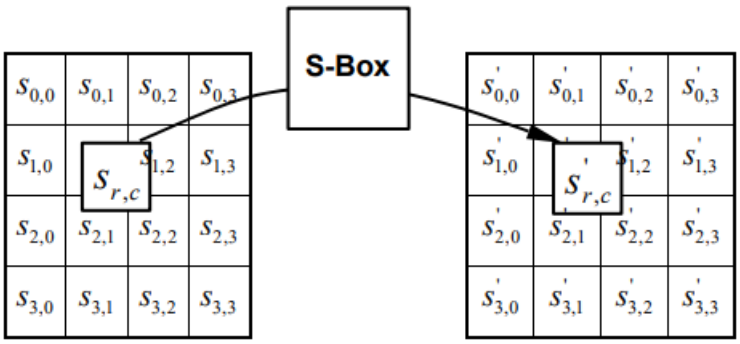
$$b_i' = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i$$

Donde cada bit i del byte b se actualiza según una operación afín que involucra bits específicos de b y de un byte c con valor {63} o {01100011}. Esta fórmula utiliza operaciones como desplazamientos circulares (mod 8) para actualizar los bits i del byte. El uso de un apóstrofe (prime) en una variable, por ejemplo, b′, significa que la variable se actualizará con el valor que se encuentra a la derecha de la expresión.

En forma matricial, el elemento de transformación afín de la S-box se puede expresar como:

$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}.$$

Esta matriz ilustra el efecto de la transformación SubBytes() en el Estado.

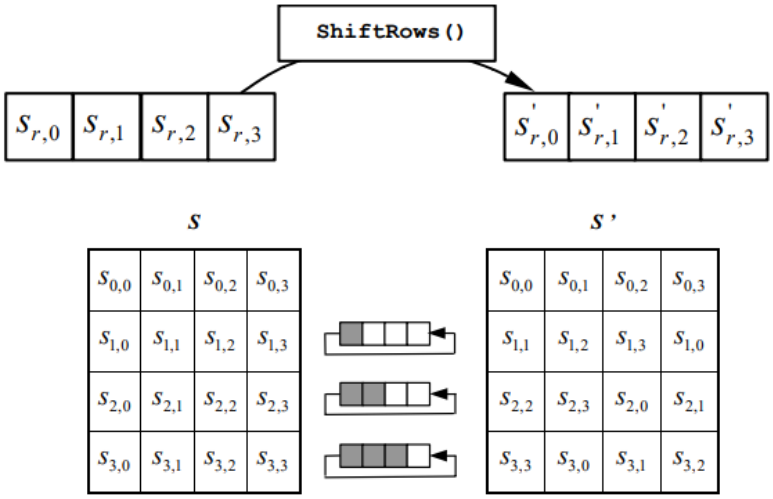


Aquí podemos observar cómo SubBytes() aplica el S-box a cada byte del Estado.

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Y por último se aprecia la S-box con valores de sustitución para el byte xy (en formato hexadecimal).

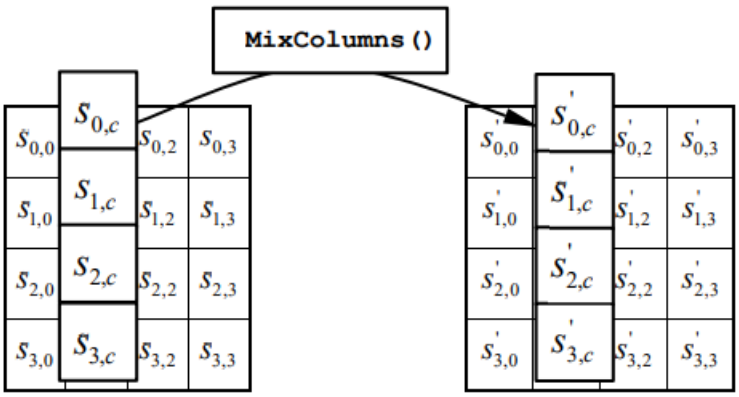
En la operación ShiftRows(), los bytes de las últimas tres filas del Estado se desplazan cíclicamente entre diferentes números de bytes (compensaciones). La primera fila, r = 0, no se desplaza, se puede ver en la siguiente imagen como se desplaza cíclicamente las últimas tres filas del Estado.



La transformación MixColumns() opera en el estado columna por columna, tratando cada columna como un polinomio de cuatro términos. Las columnas se consideran polinomios sobre GF(2^8) y se multiplican módulo x^4 + 1 con un polinomio fijo a(x), dado por

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} .$$

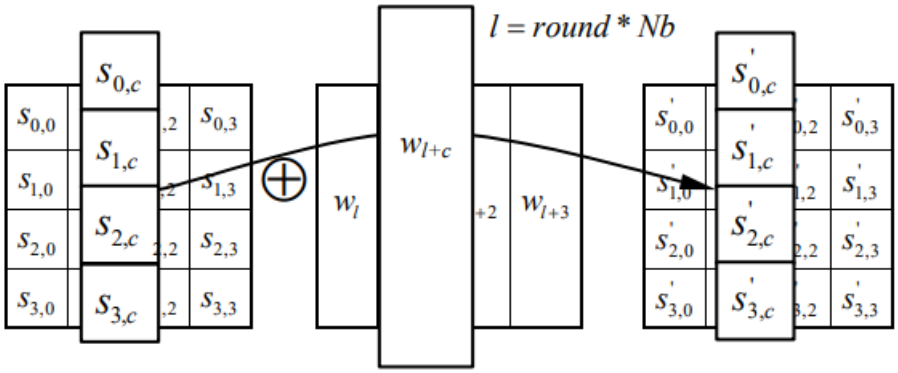
En la siguiente imagen se puede ver como MixColunms()opera en el Estado columna por columna



Y por último en la operación AddRoundKey(), se agrega una clave redonda al estado mediante una simple operación XOR bit a bit. Cada clave de ronda consta de Nb palabras. Cada una de esas palabras Nb se agrega a las columnas del Estado, de modo que

$$[s'_{0,c}, s'_{1,c}, s'_{2,c}, s'_{3,c}] = [s_{0,c}, s_{1,c}, s_{2,c}, s_{3,c}] \oplus [w_{round*Nb+c}] \quad \text{for } 0 \le c < \mathbf{Nb},$$

En la siguiente imagen se aprecia como AddRoundKey() realiza XOR en cada columna del estado con una palabra del programa clave [4].



Algoritmos Criptográficos Asimétricos

Los algoritmos criptográficos asimétricos, también conocidos como criptografía de clave pública, son fundamentales en la seguridad de la información al utilizar un par de claves matemáticamente relacionadas pero con funciones distintas: una clave pública y una clave privada. Esta configuración permite una serie de funciones de seguridad, autenticación y confidencialidad en la comunicación y transmisión de datos.

La clave pública se distribuye libremente y se utiliza para cifrar datos o verificar firmas digitales. Esta clave puede ser conocida por cualquier persona. La clave privada se mantiene en secreto por su propietario, esta clave se utiliza para descifrar datos cifrados con la clave pública o para firmar digitalmente información. Es esencial mantenerla segura y protegida.

Estos algoritmos permiten verificar la identidad de las partes comunicantes mediante firmas digitales, estas firmas digitales garantizan que los datos no han sido alterados por terceros los algoritmos asimétricos también facilitan el cifrado de datos para que solo el destinatario correcto pueda descifrarlos. La firma digital desempeña un papel crucial en la autenticación y la integridad de los mensajes, proporcionando una garantía de que el remitente es genuino y de que el contenido no ha sido alterado. Su seguridad radica en la dificultad prácticamente insuperable de falsificarla sin acceso a la clave privada del firmante, que debe ser resguardada con extremo cuidado. El proceso de firma digital consta de dos fases. En la primera etapa, el remitente cifra los datos utilizando su clave privada y los envía al destinatario. En la fase de verificación, el receptor descifra los datos con la clave pública del emisor y verifica sí coinciden con los datos originales; su congruencia indica que no han sido modificados. En este procedimiento, se emplean funciones hash, como SHA2-256 y SHA2-512, ya que el cifrado asimétrico tiende a ser más lento. El remitente aplica esta función al mensaje original para obtener una "huella digital" o resumen único. Posteriormente, cifra esta huella con su clave privada y la envía al destinatario a través de un canal inseguro. Al recibir la huella, el destinatario realiza la función hash en sus propios datos y compara los resultados con la huella recibida. La coherencia entre ambas huellas indica que el contenido no ha sido alterado; cualquier discrepancia sugiere una modificación en los datos. Esto garantiza tres aspectos fundamentales: la autenticidad del remitente, la integridad del mensaje y la imposibilidad de negar el envío del mensaje por parte del emisor. Para agregar confidencialidad a la comunicación, simplemente se cifra el mensaje original con la clave pública del receptor, asegurando que solo el destinatario legítimo pueda descifrarlo [2].

Algunos desafíos que podemos encontrar es que algunos algoritmos asimétricos pueden ser computacionalmente más costosos que los algoritmos simétricos y pueden requerir claves más largas para alcanzar un nivel deseado de seguridad. Además hay que considerar que la seguridad de la clave privada es fundamental; su gestión y protección son críticas. Uno de los algoritmos asimétricos más conocidos es RSA.

Algoritmo RSA (Rivest-Shamir-Adleman)

RSA es un algoritmo de cifrado asimétrico que utiliza operaciones matemáticas con números primos grandes. La seguridad de RSA se basa en la dificultad de factorizar el producto de dos números primos grandes para obtener los factores originales, lo que sería necesario para descifrar los mensajes sin conocer la clave privada correspondiente.

El algoritmo RSA implica la generación de claves pública y privada mediante la elección de dos números primos grandes y luego lleva a cabo operaciones matemáticas utilizando estos números para cifrar y descifrar datos. La clave pública es el producto de dos números primos, mientras que la clave privada se obtiene a través de operaciones matemáticas adicionales con los números primos. El proceso de cifrado se realiza mediante la exponenciación modular, donde el mensaje se eleva a una potencia y se toma el módulo del resultado. Este cálculo se realiza con la clave pública. El proceso de descifrado también utiliza la exponenciación modular, pero con la clave privada. Esta operación permite recuperar el mensaje original a partir del texto cifrado.

Los desafíos que encontramos con el algoritmo RSA es que con el tiempo, el aumento en la capacidad de cómputo hace necesario el uso de claves más largas para mantener un nivel adecuado de seguridad.

Los pasos que utiliza el algoritmo RSA son los siguientes:

1. Generación de claves RSA:
- Selección de números primos: Se eligen dos números primos grandes distintos, p y q.
- Cálculo de n: Se calcula n = p * q, que es parte de la clave pública y privada.

- Función Phi de Euler: Se calcula φ(n) = (p - 1) * (q - 1). Esta función es crucial para la generación de claves y el proceso de descifrado.
- Selección de e (clave pública): Se elige un número e que sea coprimo con φ(n), comúnmente un número primo pequeño (por ejemplo, 65537).
- Cálculo de d (clave privada): Se calcula d ≡ e^(-1) (mod φ(n)). Esto significa que d es el inverso multiplicativo modular de e modulo φ(n).

2. Proceso de Cifrado:

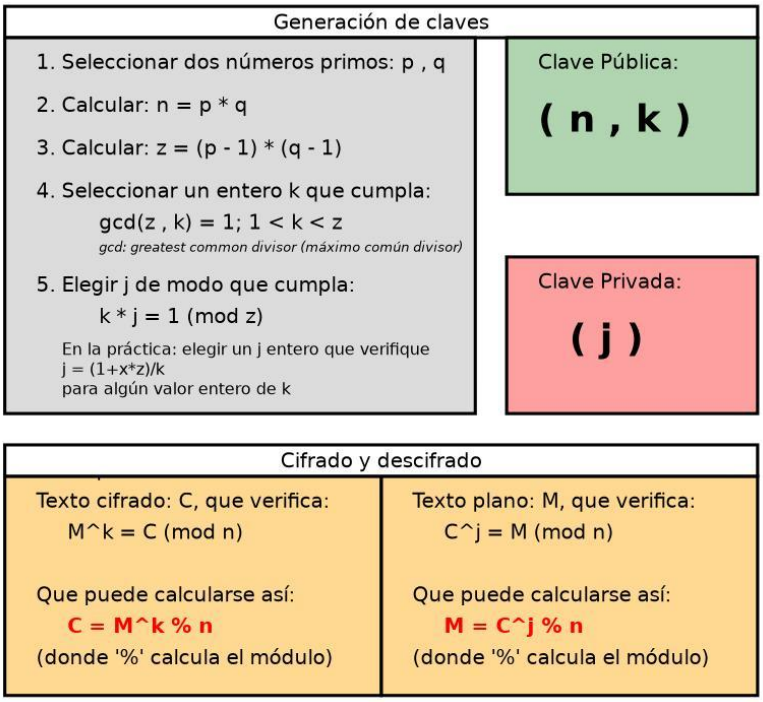
El mensaje M se convierte en un número entero y se eleva a la potencia de la clave pública e: C ≡ M^e (mod n).

3. Proceso de Descifrado:

El texto cifrado C se eleva a la potencia de la clave privada d: M ≡ C^d (mod n).

4. Verificación del Algoritmo:

El texto descifrado M es igual al texto original [5].



Referencias

[1] R. Palacios and V. Delgado, "Introducción a la Criptografía: tipos de algoritmos," in Anales de Mecánica y Electricidad, January 2006.

[2] G. J. Simmons, "Symmetric and asymmetric encryption," ACM Computing Surveys (CSUR), vol. 11, no. 4, pp. 305-330, 1979.

[3] Federal Information Processing Standards Publication. (2001). "Announcing the advanced encryption standard (AES)." Standard No. 197(1-51), pp. 3-3.

[4] V. Rijmen and J. Daemen, "Advanced encryption standard," in Proceedings of Federal Information Processing Standards Publications, National Institute of Standards and Technology, vol. 19, pp. 22, 2001.

[5] Cordoba, D. (2016), "RSA: ¿Cómo funciona este algoritmo de cifrado?"