Algoritmos de encriptación o cifrado

Publicado el El Informatico - 20 de noviembre de 2020 -

Cuando nos venden algún tipo de servicio normalmente nos hablan de "encriptación" sin entrar en detalles. Muchos productos te los ofrecen con lo que en las oficinas de Marketing se denomina "algoritmo criptográfico de grado militar", lo cuál de cara a alguien que no sabe suena como algo muy imponente, pero que en realidad se usa para denominar algoritmos como RSA o AES, desarrollados por los servicios de inteligencia de varios países y usados también en el ámbito militar.

Pero un algoritmo de encriptación es mucho más que un nombre, así qué es necesario detallar un poco más sobre lo que se usa para conocer si es seguro o no, así como conocer en qué consisten, cómo funcionan, y para qué se aplican.

¿Qué es un algoritmo de cifrado?

Es básicamente un proceso (o algoritmo en el caso de que el proceso sea matemático) en el en el que se toma un mensaje como entrada, y se procesa de modo que su contenido parezca aleatorio o irreconocible para una tercera parte que intente acceder al contenido original. Es algo que ya existía en los tiempos de la antigua Roma, donde los militares usaban lo que se denomina el cifrado de César para que, en caso de que el enemigo interceptara las comunicaciones entre sus batallones, no pudiera revelar su contenido.

El cifrado de César es el algoritmo de cifrado más antiguo y más básico, y consiste en desplazar cada letra del mensaje un número determinado de espacios en el alfabeto. De éste modo, si el valor de desplazamiento fuese 1, cada letra A cambia por una B, y cada B por una C, etc. Por ejemplo, el texto:

Hola Mundo

Pasaría a ser (siguiendo el alfabeto Español):

Ipmb Nvñep

El texto de salida, a ojos de un tercero, parecería un texto aleatorio sin significado y nuestra comunicación quedaría segura, asumiendo que el enemigo no conozca la clave para descifrar el mensaje.

¿Cómo se aplica la criptografía en la informática?

La información de un equipo informático se compone de dígitos numéricos. Por lo general, cuando hablamos de **software**, dividimos éstos dígitos en **bytes**. Cada byte contiene un valor numérico entre el 0 y el 255, con un total de 256 posibles valores por byte (2 elevado a 8, ya que son 8 bits). El ordenador trabaja constantemente manipulando éstos valores, y la salida que nos muestra es una interpretación de todos éstos valores en conjunto.

Cuando nosotros almacenamos un archivo, lo que hacemos es almacenar una cadena de bytes en el medio de almacenamiento que representan el archivo. Y cuando nos comunicamos por internet, lo hacemos mandando paquetes de información cuyo contenido e información también se representan mediante bytes.

Por ejemplo, a la hora de representar un texto en un ordenador, por norma general lo hacemos mediante código ASCII. Éste código toma el valor de cada byte y lo asigna a un carácter específico dentro de una tabla de símbolos.

Dec	Hex	Oct	Binary	Char	Dec	Hex	Oct	Binary	Char	Dec	Hex	Oct	Binary	Char	Dec	Hex	Oct	Binary	Char
0	00	000	0000000	NUL (null character)	32	20	040	0100000	space	64	40	100	1000000	@	96	60	140	1100000	`
1	01	001	0000001	SOH (start of header)	33	21	041	0100001	!	65	41	101	1000001	A	97	61	141	1100001	a
2	02	002	0000010	STX (start of text)	34	22	042	0100010	"	66	42	102	1000010	B	98	62	142	1100010	b
3	03	003	0000011	ETX (end of text)	35	23	043	0100011	#	67	43	103	1000011	C	99	63	143	1100011	c
4	04	004	0000100	EOT (end of transmission)	36	24	044	0100100	$	68	44	104	1000100	D	100	64	144	1100100	d
5	05	005	0000101	ENQ (enquiry)	37	25	045	0100101	%	69	45	105	1000101	E	101	65	145	1100101	e
6	06	006	0000110	ACK (acknowledge)	38	26	046	0100110	&	70	46	106	1000110	F	102	66	146	1100110	f
7	07	007	0000111	BEL (bell (ring))	39	27	047	0100111	'	71	47	107	1000111	G	103	67	147	1100111	g
8	08	010	0001000	BS (backspace)	40	28	050	0101000	(	72	48	110	1001000	H	104	68	150	1101000	h
9	09	011	0001001	HT (horizontal tab)	41	29	051	0101001	)	73	49	111	1001001	I	105	69	151	1101001	i
10	0A	012	0001010	LF (line feed)	42	2A	052	0101010	*	74	4A	112	1001010	J	106	6A	152	1101010	j
11	0B	013	0001011	VT (vertical tab)	43	2B	053	0101011	+	75	4B	113	1001011	K	107	6B	153	1101011	k
12	0C	014	0001100	FF (form feed)	44	2C	054	0101100	,	76	4C	114	1001100	L	108	6C	154	1101100	l
13	0D	015	0001101	CR (carriage return)	45	2D	055	0101101	-	77	4D	115	1001101	M	109	6D	155	1101101	m
14	0E	016	0001110	SO (shift out)	46	2E	056	0101110	.	78	4E	116	1001110	N	110	6E	156	1101110	n
15	0F	017	0001111	SI (shift in)	47	2F	057	0101111	/	79	4F	117	1001111	O	111	6F	157	1101111	o
16	10	020	0010000	DLE (data link escape)	48	30	060	0110000	0	80	50	120	1010000	P	112	70	160	1110000	p
17	11	021	0010001	DC1 (device control 1)	49	31	061	0110001	1	81	51	121	1010001	Q	113	71	161	1110001	q
18	12	022	0010010	DC2 (device control 2)	50	32	062	0110010	2	82	52	122	1010010	R	114	72	162	1110010	r
19	13	023	0010011	DC3 (device control 3)	51	33	063	0110011	3	83	53	123	1010011	S	115	73	163	1110011	s
20	14	024	0010100	DC4 (device control 4)	52	34	064	0110100	4	84	54	124	1010100	T	116	74	164	1110100	t
21	15	025	0010101	NAK (negative acknowledge)	53	35	065	0110101	5	85	55	125	1010101	U	117	75	165	1110101	u
22	16	026	0010110	SYN (synchronize)	54	36	066	0110110	6	86	56	126	1010110	V	118	76	166	1110110	v
23	17	027	0010111	ETB (end transmission block)	55	37	067	0110111	7	87	57	127	1010111	W	119	77	167	1110111	w
24	18	030	0011000	CAN (cancel)	56	38	070	0111000	8	88	58	130	1011000	X	120	78	170	1111000	x
25	19	031	0011001	EM (end of medium)	57	39	071	0111001	9	89	59	131	1011001	Y	121	79	171	1111001	y
26	1A	032	0011010	SUB (substitute)	58	3A	072	0111010	:	90	5A	132	1011010	Z	122	7A	172	1111010	z
27	1B	033	0011011	ESC (escape)	59	3B	073	0111011	;	91	5B	133	1011011	[	123	7B	173	1111011	{
28	1C	034	0011100	FS (file separator)	60	3C	074	0111100	<	92	5C	134	1011100	\	124	7C	174	1111100	|
29	1D	035	0011101	GS (group separator)	61	3D	075	0111101	=	93	5D	135	1011101	]	125	7D	175	1111101	}
30	1E	036	0011110	RS (record separator)	62	3E	076	0111110	>	94	5E	136	1011110	^	126	7E	176	1111110	~
31	1F	037	0011111	US (unit separator)	63	3F	077	0111111	?	95	5F	137	1011111	_	127	7F	177	1111111	DEL

Tabla de caracteres ASCII. Fuente: asciitable.xyz

Cuando queremos digitalizar un texto usando ASCII, se toma el valor de cada carácter de ésta tabla. Así pues, si escribimos una 'A', su valor decimal sería 41. Y si escribimos una 'c', su valor decimal sería 99. Un texto como 'Hola' tendría una longitud de 4 bytes, con valores 72, 111, 108, 97.

Para encriptar ésta información, basta con aplicar un algoritmo matemático, tomando éstos valores como entrada, además de una clave de cifrado, para que nos dé a la salida **una cadena de valores de apariencia aleatoria** y **con una distribución de valores lo más cercana posible a ser uniforme**.

Si bien muchos algoritmos son de **dos pasos** (podemos encriptar y desencriptar la información para recuperarla), algunos lo son de **sólo uno** (Una vez encriptada la información, no será posible desencriptarla). Por ejemplo, algoritmos como SHA3 se usan como control de la integridad de información, otros se pueden usar para almacenamiento de contraseñas, o incluso para generación de claves privadas en comunicaciones que usan algoritmos de clave pública. **Si bien no todos están debidamente habilitados para ello**.

Por ejemplo, si encriptáramos la palabra "Hola" usando un algoritmo SHA3 con un hash de 224 bytes y un tamaño de palabra de 64, se produce la siguiente cadena (En hexadecimal) llamada HASH o DIGEST:

1F42ADD25C0A80A4C82AAE3A0E302ABF9261DCA7E7884FD869D96ED4CE88AAAA25304D2D79E1FA5CC1FA2C9
5899229BC87431AD06DA524F2140E70BD0536E9685EE7808F598D8A9FE15D40A72AEFF431239292C5F64BDB7F
620E5D160B329DEB58CF6D5C0665A3DED61AE4ADBCA94DC2B7B02CDF3992FDF79B3D93E546D5823C3A6309
23064ED24C3D974C4602A49DF75E49CF7BD51EDC7382214CBA850C4D3D11B40A70B1D926E3755EC79693620
C242AB0F23EA206BA337A7EDC5421D63126CB6C7094F6BC1CF9943796BE2A0D9EB74FC726AA0C0D3B3D3903
9DEAD39A7169F8C3E2365DD349E358BF08C717D2E436D65172A76ED5E1F1E694A75C19280B2A

Buena suerte descifrando "eso".

Un ejemplo de un algoritmo que no es criptográficamente seguro para contraseñas o claves y que sólo se usa para verificar la integridad de los datos es **MD5**. Es un algoritmo bastante antiguo que se usaba en tiempos de antaño para almacenar contraseñas en las bases de datos, pero con el tiempo se demostró que no era seguro para ello.

Componentes de un algoritmo de encriptación

Cada algoritmo es diferente, y normalmente en una comunicación se usa una cadena de algoritmos para hacer más seguras las comunicaciones. Pero por lo general, suelen componerse de una clave, un generador de números aleatorios (PRNG), un valor "semilla" para el generador PRNG, y un algoritmo que genera la

salida con todos los valores anteriores (cipher).

El generador de números aleatorios es el componente más crítico, ya que genera los valores con los que se va a cifrar o descifrar el mensaje. Para que éste sea válido, debe generar una serie de valores que sean aparentemente aleatorios, sin repeticiones, cuyos valores sean impredecibles, y cuya distribución sea prácticamente uniforme.

Un algoritmo que use un generador de números aleatorios débil puede generar repeticiones en los valores y distribuciones no uniformes que, por ejemplo, pueden conseguir que varias claves den un mismo resultado, consiguiendo que nuestro algoritmo sea **predecible e inseguro**, ya que podríamos obtener la clave mediante los valores que devuelve si los podemos predecir.

Es por ello que algunos algoritmos podrían incluir puertas traseras que permitieran a un atacante descifrar el contenido del mensaje mediante algoritmos débiles de generación de números aleatorios.

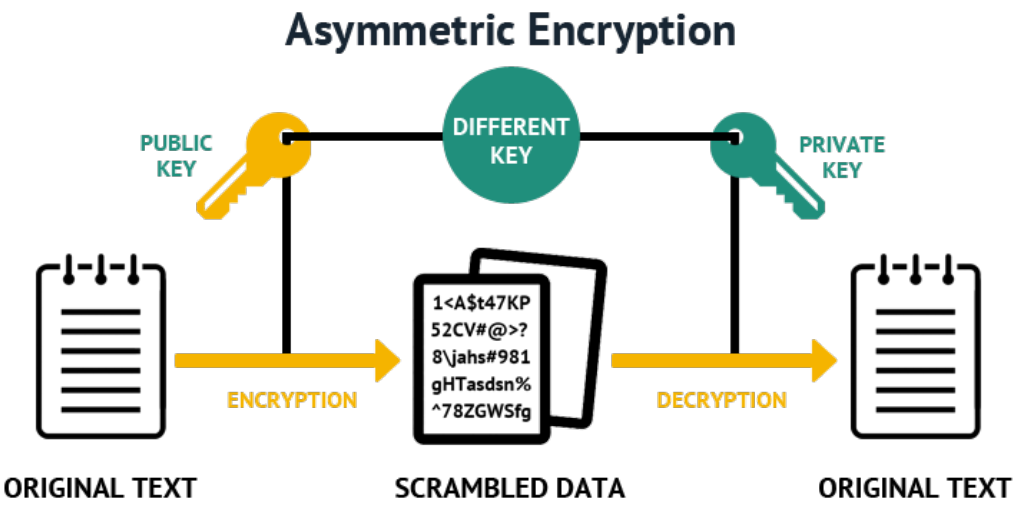
Por éste motivo, que desde marketing nos vendan que un algoritmo de encriptación sea "de grado militar" no significa automáticamente que éste sea seguro, si no nos garantizan que los componentes de dicho algoritmo sean **criptográficamente seguros**.

Algoritmos de clave pública (Encriptación asimétrica)

Los algoritmos comunes suelen utilizar una clave, pero algunos algoritmos pueden usar más. En los algoritmos de clave pública, se usan dos, una de las cuales es **pública** y la otra, **privada**. Dicha clave pública se genera a partir de la privada.

Al realizar una comunicación, las dos partes implicadas en la comunicación **intercambian sus claves públicas**, siendo las privadas un **secreto**. Una vez intercambiadas, cifran sus mensajes usando la clave pública que han intercambiado con la otra parte.

Una vez finalizada la encriptación, ambas partes intercambian sus mensajes ya cifrados, y una vez recibido usan sus claves **privadas** para descifrar el contenido del mensaje. Ésto es posible ya que utilizan un algoritmo matemático que permite cifrar un mensaje mediante una clave pública, pero no descifrarlo mediante dicha clave, ya que la clave pública se genera mediante la privada, y es la privada la que se usa para descifrarlo, la cuál se mantiene en secreto.



Esquema del funcionamiento de un algoritmo de encriptación asimétrico. Fuente: hackernoon.com

Este tipo de encriptación se usa muy comúnmente en internet, ya que permite intercambiar las claves entre las partes implicadas sin posibilitar que un posible atacante sea capaz de usarlas para descifrar el mensaje.

También se usa en algunos equipos electrónicos para protegerlos de manipulación. Algunas videoconsolas usan un algoritmo de éste tipo, cuyas claves van dentro del propio hardware de la consola, en los microchips, para dificultar su recuperación.

Certificados TLS (antíguamente SSL) en la web

Al conectarte a internet a través de tu navegador, notarás que en ocasiones al conectarte a una página web aparece **un candado con una franja roja** en la barra de direcciones. Ese candado significa que nos estamos conectando a una página web que no utiliza un **certificado TLS o SSL**.

Esto es muy importante, y **no debemos ignorarlo**. Un certificado nos ofrece dos capas de protección (aunque no sean infalibles): por un lado, nos garantiza que la comunicación está encriptada mediante clave pública. Y por otro, mediante el certificado, y acreditado por una autoridad, que la procedencia de la comunicación es de quien dice ser. Que, por ejemplo, alguien no nos esté atacando haciéndose pasar por un servicio conocido.

Si vamos a hacer compras en internet o si vamos a solicitar documentos o dar números de cuentas a un ayuntamiento, **debemos asegurarnos de que la página que usemos utilice un certificado de seguridad y que éste sea válido**. En caso contrario, nuestros datos podrían estar comprometidos, ya que no hay ninguna capa de encriptación en ellos ni tenemos la garantía de que el destinatario sea realmente quien dice ser.

Si necesitas un ejemplo de lo que podría pasar si lo ignoras, en éste enlace tienes un artículo donde se demuestra como se puede capturar tu

información simplemente capturando los paquetes durante una conexión no segura.

¿Por qué éste método no es completamente infalible? Como ya he dicho, todo depende de los algoritmos de encriptación que se usen, así como nuestra confianza en las autoridades que afirman que la fuente de procedencia del certificado es de fiar. No es 100% infalible (nada lo es), pero como de costumbre, le ponen las cosas mucho más complicadas a un posible atacante. Los certificados **TLS** solucionan algunos de los problemas de su predecesor, SSL, haciendo las conexiones algo más seguras.

Almacenamiento de contraseñas

Algo que debería de preocupar y mucho a los usuarios es el método usado para el **almacenamiento de las contraseñas** por parte de los administradores de los diferentes servicios. En la mayoría de casos, y como ya he mencionado más arriba, por lo general las contraseñas se encriptan al almacenarse en las bases de datos mediante algún algoritmo de un solo paso como, por ejemplo, SHA.

El funcionamiento de ésta técnica es muy sencillo. El sistema toma la contraseña introducida por el usuario, la encripta usando dicho método, y almacena la salida en la base de datos en el campo de la contraseña.

Al iniciar sesión o autentificarse, el sistema realiza ésta misma operación. Toma la contraseña de manos del usuario, y la vuelve a encriptar. Entonces compara la salida generada con el valor del campo de la contraseña previamente almacenada. Si ambas coinciden, entonces la contraseña es válida.

De este modo, si un atacante se hiciese con una copia de la base de datos, las contraseñas permanecerían encriptadas. En el caso de que un usuario usase la misma contraseña para todo (una práctica muy poco recomendable pero muy extendida), a los atacantes les resultaría mucho más complicado hacerse con dicha contraseña y, por tanto, el usuario tendría tiempo para cambiarla.

A pesar de éste riesgo, aún exísten muchas empresas y muchas plataformas que **todavía almacenan la contraseña en texto plano o incluso usando algoritmos como MD5**. Lo cuál quiere decir no sólo que un atacante pudiera llegar a recuperarla, sino que **los propios administradores del sistema tienen acceso a ella** desde la base de datos. Por eso es imprescindible encriptar las contraseñas adecuadamente, así como añadir otras capas de seguridad a los sistemas (2FA, encriptación de los medios de almacenamiento y las bases de datos, etc).

Encriptación en aplicaciones como Whatsapp e

Instagram

Que aplicaciones como Whatsapp ofrezcan encriptación punto a punto usando algoritmos de clave pública es algo un tanto “engañoso”.

Si bien es posible que nos ofrezcan dicha encriptación e incluso que sea segura frente a un ataque MITM, la información se encuentra **sin encriptar** dentro de la aplicación en nuestros terminales, e incluso un tercero podría seguir interceptando nuestros mensajes leyendo las notificaciones que recibimos al recibir un mensaje.

Por éste motivo, que nos ofrezcan métodos para encriptar las comunicaciones o incluso destruirlas, **no es suficiente**. Ya que para empezar, no tenemos garantías de que incluso dichas aplicaciones intercepten esas comunicaciones ya que, al fin y al cabo, se encuentran disponibles sin encriptar en nuestro dispositivo.

¿Qué algoritmos son los más seguros?

Por regla general se usan los algoritmos más antiguos y que aún no se han probado vulnerables. Por ejemplo: RSA para clave pública, AES-256 para claves y SHA-3 para claves y contraseñas. RSA es un algoritmo creado en nada más ni nada menos que 1977. Es decir, lleva 44 años sin haber sido vulnerado. SHA lleva algo menos, desde 1993 (23 años). Pero 23 años también son bastantes años, si bien lleva algunas revisiones a sus espaldas (Actualmente se usa SHA-3, las primera sí es vulnerable).

Usar éstos algoritmos con un generador de números que sea criptográficamente seguro (CSPRNG). Si usara por ejemplo Dual_EC_DRBG PRNG, entonces sería inseguro ya que éste generador tiene una puerta trasera, algo que ninguna empresa te va a especificar.

En resúmen...

Los puntos a tener en cuenta a la hora de usar internet son:

1. Asegúrate de que tus comunicaciones usen **algún tipo de encriptación de clave pública que sea criptográficamente seguro**.
2. Asegúrate de que las páginas web por las que navegas utilicen **un certificado TLS válido**. ***Sobre todo si vas de compras o a realizar algún tipo de trámite con datos personales***. Tu navegador te debe de informar de cuándo ésto no es así.
3. **NO confíes NUNCA en lo que te digan las empresas si no te dan pruebas de que algo es como dicen que es**. Verifica que los algoritmos son realmente seguros y desconfía de la encriptación punto a punto en clientes de mensajería instantánea como Whatsapp o Telegram.

4. **Tus comunicaciones pueden estar comprometidas desde el propio terminal que usas**, haciendo que la encriptación E2E sea inútil (motivo por el cuál debes desconfiar).
5. Si te enteras de que tu plataforma usa contraseñas en texto plano (sin encriptar), **DENUNCIALO**.
6. Usa TOR. Te ayudará a mantener tus comunicaciones encriptadas y a mantenerte anónimo por internet.

ANTERIOR

Redes VPN

SIGUIENTE

Usando youtube-dl para descargar audio y videos de Youtube (Y otras plataformas)

Buscar ...

Entradas Recientes

- Encriptación LUKS con CRYPTSETUP
- Se acabaron las bromas. A partir de ahora vas a estar constantemente vigilado en todas partes.
- Microsoft anuncia su nueva versión de su sistema operativo: Windows 11
- La historia de Internet en España
- Terminología moderna usada en tecnología digital
- Desactiva la ejecución de JavaScript de los archivos PDF, en Firefox y TOR browser.

Categorías

Actualidad | Android | Básicos | Ciberseguridad | Criptografía | Emulación / Virtualización | FOSS | Hacking | Informática | Internet | Juegos | Opinion | Otros | Personal | Privacidad | Programación | Tecnología

Time Machine | Tutoriales

RSS

Subscribirse al feed RSS