Escuela de Ingeniería en Informática y Telecomunicaciones

Universidad Diego Portales

Criptografía y seguridad en redes

Pauta Ayudantía Nº 1

Ayudantes: Marcelo Fuenzalida, Diego Pineda

1 Ejercicio 1

- Cifra la palabra **"Primera Ayudantia"** en ROT-16.
- Después, cifra el texto cifrado proveniente del rot-16 a vigenere. Con una llave creada por ustedes.

1.1 Opción de respuesta

Primero traducimos la palabra con ROT-16 moviendo cada carácter 16 veces (se utilizó el alfabeto en inglés), obtenemos: **"Fhycuhq Qoktqdjyq"**.

Luego, hay que escoger una llave en nuestro caso **"Pez"**, utilizamos la tabla de vigenere y obtenemos el siguiente resultado: **"Ulxrygf Unzxpsnxf"**.

2 Ejercicio 2

Transformar la codificación a base64 del siguiente texto plano **"Mucho Gusto"**.

2.1 Respuesta

Primero transformamos los caracteres de ASCII a binario.

Quedando de la siguiente manera:

M	u	c	h	o		G	u	s	t	o
01001101	01110101	01100011	01101000	01101111	00100000	01100111	01110101	01110011	01110100	01101111

Luego juntamos los bits para separar en 6 bits contenidos en grupos de 24 bits:

010011	010111	010101	100011	011010	000110	111100	100000	011001	110111	010101	110011	011101	000110	1111

Observamos que hay 4 grupos, por lo que rellenamos los espacios nulos:

010011	010111	010101	100011	011010	000110	111100	100000	011001	110111	010101	110011	011101	000110	111100	=

Buscamos el valor numérico en cada uno de los 6 bits:

19	23	21	35	26	6	60	32	25	55	21	51	29	6	60	=

Transformamos los caracteres restantes con la tabla de base 64 y nos queda de la siguiente manera:

T	X	V	j	a	G	8	g	Z	3	V	z	d	G	8	=

Que quedaría como: **TXVjaG8gZ3VzdG8=**.

Ejercicio 3

Cifrar el texto plano **"hola"** a vigenere, con la llave **"gato"**. Luego cifrar con ROT-6 y por último cambiar la codificación a base64.

3.1 Opción de respuesta

Utilizando la tabla de vigenere, llegamos al siguiente mensaje: **"noeo"**.
Luego utilizando ROT-6 llegamos al siguiente mensaje: **"tuku"**.
Transformamos la palabra con la tabla ASCII a binario.

01110100	01110101	01101011	01110101

Dividimos en 6 bits y agregamos el padding necesario:

011101	000111	010101	101011	011101	010000	=	=

Transformamos a números:

29	7	21	43	29	16

Transformamos usando la tabla de base64:

d	H	V	r	d	Q

Que quedaría como: **dHVrdQ==**.

4 Ejercicio 4

Si tengo un cifrado cuyo mensaje es **"Aeo leoxk kienkxdsk"** y solo se conoce los primeros 3 caracteres del texto plano **"Que"**. ¿Qué tipo de cifrado está utilizando y cuáles son sus parámetros?

4.1 Opción de respuesta

Sabiendo que **"Que" = "Aeo"**.
Podemos notar que está utilizando ROT-10, puesto que todas las palabras están desplazadas 10.
El texto original del mensaje cifrado es: **"Que buena ayudantía"**.

5 Ejercicio 5

Indique un mensaje que estando en base 64, los primeros 2 caracteres sean "NE" y los últimos 2 sean "s=".

5.1 Opción de respuesta

Se sabe que base64 siempre codificará con 4 caracteres base64, es decir, si se da un mensaje, siempre dará algún grupo de 4 caracteres. Por lo que hay que tener lo siguiente en claro:

x → xx==

xx → xxx=

xxx → xxxx

Dónde:
x: serían caracteres en ASCII.
x: serían caracteres en base64.

Con esto y sabiendo que solo se requiere que se inicie con **NE** y termine con **s=**, se podría intuir que por lo menos un grupo del mensaje debe tener 2 caracteres en ASCII.

También si se percatan, solo el decodificar **NEs=** serviría, esto debido a que cumple con los requisitos de tener 4 caracteres de base64.

NOTA: Hay que tomar su valor en base64, para poder decodificarlo y luego transformarlo a ASCII.

N	E	s	=
001101	000100	101100	Padding

Ahora se agrupan en grupos de 8 bits y se procede a descartar los bits sobrantes (Los que sean 0 más a la derecha que no estén en un grupo de 8 bits).

Luego de agrupar, esto quedaría de la siguiente forma:

Grupo 1	Grupo 2	Grupo 3
00110100	01001011	00
34	1B	Descartado
4	K	

Con esto se puede concluir que un posible mensaje que cumpla con estas condiciones es el valor **4K**.

Anexos

REGULAR ASCII CHART (character codes 0 - 127)

DEC	HEX	CHAR		DEC	HEX	CHAR		DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR
000	00		(nul)	016	10	►	(dle)	032	20		048	30	0	064	40	@	080	50	P	096	60	`	112	70	p
001	01	☺	(soh)	017	11	◄	(dc1)	033	21	!	049	31	1	065	41	A	081	51	Q	097	61	a	113	71	q
002	02	☻	(stx)	018	12	↕	(dc2)	034	22	"	050	32	2	066	42	B	082	52	R	098	62	b	114	72	r
003	03	♥	(etx)	019	13	‼	(dc3)	035	23	#	051	33	3	067	43	C	083	53	S	099	63	c	115	73	s
004	04	♦	(eot)	020	14	¶	(dc4)	036	24	$	052	34	4	068	44	D	084	54	T	100	64	d	116	74	t
005	05	♣	(enq)	021	15	§	(nak)	037	25	%	053	35	5	069	45	E	085	55	U	101	65	e	117	75	u
006	06	♠	(ack)	022	16	▬	(syn)	038	26	&	054	36	6	070	46	F	086	56	V	102	66	f	118	76	v
007	07	•	(bel)	023	17	↨	(etb)	039	27	'	055	37	7	071	47	G	087	57	W	103	67	g	119	77	w
008	08	◘	(bs)	024	18	↑	(can)	040	28	(	056	38	8	072	48	H	088	58	X	104	68	h	120	78	x
009	09		(tab)	025	19	↓	(em)	041	29	)	057	39	9	073	49	I	089	59	Y	105	69	i	121	79	y
010	0A	◙	(lf)	026	1A		(eof)	042	2A	*	058	3A	:	074	4A	J	090	5A	Z	106	6A	j	122	7A	z
011	0B	♂	(vt)	027	1B	←	(esc)	043	2B	+	059	3B	;	075	4B	K	091	5B	[	107	6B	k	123	7B	{
012	0C		(np)	028	1C	∟	(fs)	044	2C	,	060	3C	<	076	4C	L	092	5C	\	108	6C	l	124	7C	|
013	0D	♪	(cr)	029	1D	↔	(gs)	045	2D	-	061	3D	=	077	4D	M	093	5D	]	109	6D	m	125	7D	}
014	0E	♫	(so)	030	1E	▲	(rs)	046	2E	.	062	3E	>	078	4E	N	094	5E	^	110	6E	n	126	7E	~
015	0F	☼	(si)	031	1F	▼	(us)	047	2F	/	063	3F	?	079	4F	O	095	5F	_	111	6F	o	127	7F	⌂

EXTENDED ASCII CHART (character codes 128 - 255) LATIN1/CP1252

DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR
128	80	€	144	90		160	A0		176	B0	°	192	C0	À	208	D0	Ð	224	E0	à	240	F0	ð
129	81		145	91	‘	161	A1	¡	177	B1	±	193	C1	Á	209	D1	Ñ	225	E1	á	241	F1	ñ
130	82	‚	146	92	’	162	A2	¢	178	B2	²	194	C2	Â	210	D2	Ò	226	E2	â	242	F2	ò
131	83	ƒ	147	93	“	163	A3	£	179	B3	³	195	C3	Ã	211	D3	Ó	227	E3	ã	243	F3	ó
132	84	„	148	94	”	164	A4	¤	180	B4	´	196	C4	Ä	212	D4	Ô	228	E4	ä	244	F4	ô
133	85	…	149	95	•	165	A5	¥	181	B5	µ	197	C5	Å	213	D5	Õ	229	E5	å	245	F5	õ
134	86	†	150	96	–	166	A6	¦	182	B6	¶	198	C6	Æ	214	D6	Ö	230	E6	æ	246	F6	ö
135	87	‡	151	97	—	167	A7	§	183	B7	·	199	C7	Ç	215	D7	×	231	E7	ç	247	F7	÷
136	88	ˆ	152	98	˜	168	A8	¨	184	B8	¸	200	C8	È	216	D8	Ø	232	E8	è	248	F8	ø
137	89	‰	153	99	™	169	A9	©	185	B9	¹	201	C9	É	217	D9	Ù	233	E9	é	249	F9	ù
138	8A	Š	154	9A	š	170	AA	ª	186	BA	º	202	CA	Ê	218	DA	Ú	234	EA	ê	250	FA	ú
139	8B	‹	155	9B	›	171	AB	«	187	BB	»	203	CB	Ë	219	DB	Û	235	EB	ë	251	FB	û
140	8C	Œ	156	9C	œ	172	AC	¬	188	BC	¼	204	CC	Ì	220	DC	Ü	236	EC	ì	252	FC	ü
141	8D		157	9D		173	AD		189	BD	½	205	CD	Í	221	DD	Ý	237	ED	í	253	FD	ý
142	8E	Ž	158	9E	ž	174	AE	®	190	BE	¾	206	CE	Î	222	DE	Þ	238	EE	î	254	FE	þ
143	8F		159	9F	Ÿ	175	AF	¯	191	BF	¿	207	CF	Ï	223	DF	ß	239	EF	ï	255	FF	ÿ

Hexadecimal to Binary

0	0000	4	0100	8	1000	C	1100
1	0001	5	0101	9	1001	D	1101
2	0010	6	0110	A	1010	E	1110
3	0011	7	0111	B	1011	F	1111

Groups of ASCII-Code in Binary

Bit 6	Bit 5	Group
0	0	Control Characters
0	1	Digits and Punctuation
1	0	Upper Case and Special
1	1	Lower Case and Special

Table 1: Tabla ASCII

Valor	Encoding	Valor	Encoding	Valor	Encoding	Valor	Encoding
0	A	17	R	34	i	51	z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	+
12	M	29	d	46	u	63	/
13	N	30	e	47	v		
14	O	31	f	48	w	(pad)	=
15	P	32	g	49	x		
16	Q	33	h	50	y		

Table 2: Tabla Base64

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Table 3: Tabla Vigenere

Escuela de Ingeniería en Informática y Telecomunicaciones

Universidad Diego Portales

Criptografía y seguridad en redes

Pauta Ayudantía Nº 2

Ayudantes: Marcelo Fuenzalida, Diego Pineda

1 Ejercicio 1

Como ingeniero informático, le piden hacer una contraseña usando el alfabeto en inglés (base 26), cuya entropía debe cumplir con un valor aproximado de 47 bits. Mencione una contraseña válida que cumpla con todos los parámetros mencionados anteriormente.

1.1 Opción de respuesta

Sabiendo que nuestros parámetros son:

$$H = 47$$
$$W = 26$$
$$L = ?$$

Con estos datos se procede a usar la forma de entropía, reemplazando los datos:

$$\begin{aligned} 47 &= L \cdot \log_2(25) \\ &\approx L \cdot 5 \\ L &\approx \frac{47}{5} \\ L &\approx 10 \end{aligned}$$

Dejando que las contraseñas deben ser mínimo de 10 carácteres, como por ejemplo: "**AAAAAAAAAA**" (Bastante insegura la contraseña, pero sirve para este caso).

2 Ejercicio 2

Un hacker está llevando a cabo un ataque de fuerza bruta contra una página web cuyo sistema de contraseñas tiene un tamaño máximo de 7 carácteres. Además, sabe que la página solo utiliza el alfabeto en inglés, incluyendo mayúsculas y minúsculas. ¿Cuál es la entropía de la contraseña?

2.1 Opción de respuesta

Sabiendo que nuestros parámetros son:

$$H = ?$$
$$W = 52$$
$$L = 7$$

Con estos datos se procede a usar la forma de entropía, reemplazando los datos:

$$\begin{aligned} H &= 7 \cdot \log_2(52) \\ &\approx 7 \cdot 6 \\ H &\approx 42 \end{aligned}$$

Por lo que la entropía de la contraseña es de aproximadamente 42 bits.

3 Ejercicio 3

Supongamos un alfabeto de caracteres cuyos valores pertenecen a la tabla ASCII de la siguiente manera: [! - ;] y [a - z].
¿Cuál es la cantidad de caracteres distintos que se utilizan? Luego, si se utiliza un passcode de 10 carácteres de dicho alfabeto, indique la entropía de bits utilizando el alfabeto mencionado anteriormente.
¿Es posible alcanzar una entropía mayor a 12 bits con 10 carácteres de ese alfabeto?

3.1 Opción de respuesta

Sabiendo que los valores de los carácteres son:

$! \Rightarrow 33$
$; \Rightarrow 59$
$a \Rightarrow 97$
$z \Rightarrow 122$

Y sabiendo que se debe considerar el primer carácter queda que la primera parte del regex queda como:

$$59 - 33 = 26$$

Sumandole el primer carácter que hay que usar queda que se tienen 27 carácteres de uso.
Y para el segundo queda como:

$$122 - 97 = 25$$

Sumandole el primer carácter que hay que usar queda que se tienen 26 carácteres de uso.
Teniendo en total:

$$27 + 26 = 53$$

Quedando con 53 carácteres de uso.

Luego si L=10 la fórmula de la entropía queda como:

$$\begin{aligned} H &= 10 \cdot \log_2(53) \\ &\approx 10 \cdot 6 \\ H &\approx 60 \end{aligned}$$

Cómo se alcanza una entropía de 60 bits, es posible alcanzar la entropía mayor a 12 bits con 10 carácteres en este alfabeto.

4 Ejercicio 4

Proporcione dos mensajes con una longitud mayor a 17 bits. Luego, codifique ambos mensajes a base64, asegurando que los últimos dos caracteres sean "w=". También, explique los parámetros y condiciones que deben cumplirse para que el resultado termine en "w=".

4.1 Opción de respuesta

Primero conseguimos que bits necesitamos obtener que valores provocan que se termine en "w=".
Sabiendo que:

x → xx== 4 bits
xx → xxx= 2 bits
xxx → xxxx

Dónde:
x: serían caracteres en ASCII.
x: serían caracteres en base64.

Cómo necesitamos que termine en "w=", se puede hacer de la siguiente forma:

Grupo 1	Grupo 2	w	=
[6 bits]	[6 bits]	110000	Padding

Con esto y sabiendo que cada grupo de base64 tiene 24 bits con 4 subgrupos de 6 bits cada uno. En ASCII cada carácter tiene 8 bits, por lo que debe hacer un grupo de 2 carácteres, que el último carácter tenga un valor de sus últimos 4 bits sea 1100 en binario o C en hex.

Por lo que las condiciones quedan cómo:

1. Cantidad de carácteres

Dónde $n \in \mathbb{Z}$

2. Carácter final tiene un valor hexadecimal que sea de la forma XC.

Por lo que algunos mensajes podrían ser:

- LLLLL
- AAAAL

Anexos

REGULAR ASCII CHART (character codes 0 - 127)

DEC	HEX	CHAR		DEC	HEX	CHAR		DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR
000	00		(nul)	016	10	►	(dle)	032	20		048	30	0	064	40	@	080	50	P	096	60	`	112	70	p
001	01	☺	(soh)	017	11	◄	(dc1)	033	21	!	049	31	1	065	41	A	081	51	Q	097	61	a	113	71	q
002	02	☻	(stx)	018	12	↕	(dc2)	034	22	"	050	32	2	066	42	B	082	52	R	098	62	b	114	72	r
003	03	♥	(etx)	019	13	‼	(dc3)	035	23	#	051	33	3	067	43	C	083	53	S	099	63	c	115	73	s
004	04	♦	(eot)	020	14	¶	(dc4)	036	24	$	052	34	4	068	44	D	084	54	T	100	64	d	116	74	t
005	05	♣	(enq)	021	15	§	(nak)	037	25	%	053	35	5	069	45	E	085	55	U	101	65	e	117	75	u
006	06	♠	(ack)	022	16	▬	(syn)	038	26	&	054	36	6	070	46	F	086	56	V	102	66	f	118	76	v
007	07	•	(bel)	023	17	↨	(etb)	039	27	'	055	37	7	071	47	G	087	57	W	103	67	g	119	77	w
008	08	◘	(bs)	024	18	↑	(can)	040	28	(	056	38	8	072	48	H	088	58	X	104	68	h	120	78	x
009	09		(tab)	025	19	↓	(em)	041	29	)	057	39	9	073	49	I	089	59	Y	105	69	i	121	79	y
010	0A	◙	(lf)	026	1A		(eof)	042	2A	*	058	3A	:	074	4A	J	090	5A	Z	106	6A	j	122	7A	z
011	0B	♂	(vt)	027	1B	←	(esc)	043	2B	+	059	3B	;	075	4B	K	091	5B	[	107	6B	k	123	7B	{
012	0C		(np)	028	1C	∟	(fs)	044	2C	,	060	3C	<	076	4C	L	092	5C	\	108	6C	l	124	7C	|
013	0D	♪	(cr)	029	1D	↔	(gs)	045	2D	-	061	3D	=	077	4D	M	093	5D	]	109	6D	m	125	7D	}
014	0E	♫	(so)	030	1E	▲	(rs)	046	2E	.	062	3E	>	078	4E	N	094	5E	^	110	6E	n	126	7E	~
015	0F	☼	(si)	031	1F	▼	(us)	047	2F	/	063	3F	?	079	4F	O	095	5F	_	111	6F	o	127	7F	⌂

EXTENDED ASCII CHART (character codes 128 - 255) LATIN1/CP1252

DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR
128	80	€	144	90		160	A0		176	B0	°	192	C0	À	208	D0	Ð	224	E0	à	240	F0	ð
129	81		145	91	‘	161	A1	¡	177	B1	±	193	C1	Á	209	D1	Ñ	225	E1	á	241	F1	ñ
130	82	‚	146	92	’	162	A2	¢	178	B2	²	194	C2	Â	210	D2	Ò	226	E2	â	242	F2	ò
131	83	ƒ	147	93	“	163	A3	£	179	B3	³	195	C3	Ã	211	D3	Ó	227	E3	ã	243	F3	ó
132	84	„	148	94	”	164	A4	¤	180	B4	´	196	C4	Ä	212	D4	Ô	228	E4	ä	244	F4	ô
133	85	…	149	95	•	165	A5	¥	181	B5	µ	197	C5	Å	213	D5	Õ	229	E5	å	245	F5	õ
134	86	†	150	96	-	166	A6	¦	182	B6	¶	198	C6	Æ	214	D6	Ö	230	E6	æ	246	F6	ö
135	87	‡	151	97	—	167	A7	§	183	B7	·	199	C7	Ç	215	D7	×	231	E7	ç	247	F7	÷
136	88	ˆ	152	98	˜	168	A8	¨	184	B8	¸	200	C8	È	216	D8	Ø	232	E8	è	248	F8	ø
137	89	‰	153	99	™	169	A9	©	185	B9	¹	201	C9	É	217	D9	Ù	233	E9	é	249	F9	ù
138	8A	Š	154	9A	š	170	AA	ª	186	BA	º	202	CA	Ê	218	DA	Ú	234	EA	ê	250	FA	ú
139	8B	‹	155	9B	›	171	AB	«	187	BB	»	203	CB	Ë	219	DB	Û	235	EB	ë	251	FB	û
140	8C	Œ	156	9C	œ	172	AC	¬	188	BC	¼	204	CC	Ì	220	DC	Ü	236	EC	ì	252	FC	ü
141	8D		157	9D		173	AD		189	BD	½	205	CD	Í	221	DD	Ý	237	ED	í	253	FD	ý
142	8E	Ž	158	9E	ž	174	AE	®	190	BE	¾	206	CE	Î	222	DE	Þ	238	EE	î	254	FE	þ
143	8F		159	9F	Ÿ	175	AF	¯	191	BF	¿	207	CF	Ï	223	DF	ß	239	EF	ï	255	FF	ÿ

Hexadecimal to Binary

0	0000	4	0100	8	1000	C	1100
1	0001	5	0101	9	1001	D	1101
2	0010	6	0110	A	1010	E	1110
3	0011	7	0111	B	1011	F	1111

Groups of ASCII-Code in Binary

Bit 6	Bit 5	Group
0	0	Control Characters
0	1	Digits and Punctuation
1	0	Upper Case and Special
1	1	Lower Case and Special

Table 1: Tabla ASCII

Value	Encoding	Value	Encoding	Value	Encoding	Value	Encoding
0	A	17	R	34	i	51	z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	+
12	M	29	d	46	u	63	/
13	N	30	e	47	v		
14	O	31	f	48	w	(pad)	=
15	P	32	g	49	x		
16	Q	33	h	50	y		

Table 2: Tabla Base64

Escuela de Ingeniería en Informática y Telecomunicaciones
Universidad Diego Portales

Criptografía y seguridad en redes

Pauta Ayudantía Nº 3

Ayudantes: Marcelo Fuenzalida, Diego Pineda

1 Ejercicios

Responda las siguientes preguntas:

1.1 Si en la función Hash, solo se conoce el valor $H(M)$, ¿es posible obtener el mensaje M?

En general, no es posible obtener el mensaje original M a partir de su valor hash H(M) solamente. Esto se debe a la propiedad de unidireccionalidad de las funciones hash, la cual indica que es computacionalmente imposible encontrar un mensaje M' que genere el mismo valor hash H(M') que el mensaje original M.

La unidireccionalidad es una característica fundamental de las funciones hash y es crucial para su uso en aplicaciones de seguridad como:

- **Verificación de integridad:** Se utiliza para asegurar que un mensaje no ha sido alterado durante su transmisión o almacenamiento. Comparando el hash del mensaje recibido con el hash generado originalmente, se puede detectar cualquier manipulación.
- **Almacenamiento de contraseñas:** Se utilizan para almacenar contraseñas de forma segura. En lugar de almacenar la contraseña en texto plano, se almacena su hash. Si un atacante accede a la base de datos, solo obtendrá los valores hash, no las contraseñas originales.

1.2 Si en la función Hash, solo se conoce el valor M, ¿es posible obtener el mensaje $H(M)$?

Si solo se conoce el valor del mensaje M, **siempre es posible obtener el valor hash H(M)** utilizando la función hash correspondiente. La función hash está diseñada para tomar un mensaje de entrada M y generar un valor hash H(M) de salida de longitud fija. El proceso de generar el hash es determinista, lo que significa que para un mensaje M dado, siempre se producirá el mismo valor hash H(M).

1.3 ¿Qué significa que un hash debe ser resistente a colisiones?

En el contexto de las funciones hash, la resistencia a colisiones se refiere a la propiedad que dificulta, o hace computacionalmente inviable, encontrar dos entradas diferentes (mensajes) que generen el mismo valor hash (salida).

En otras palabras, una función hash resistente a colisiones debe ser capaz de producir valores hash únicos para cada mensaje de entrada con una alta probabilidad, incluso si se realizan muchos intentos de encontrar colisiones.

1.4 Si se presentan colisiones, ¿cómo podrían evitarse?

Para lograr una alta resistencia a colisiones, las funciones hash utilizan algoritmos matemáticos complejos y se diseñan con un gran espacio de salida (cantidad posible de valores hash). Algunas de las características que contribuyen a la resistencia a colisiones incluyen:

- **Función de mezcla:** La función de mezcla combina y transforma los bits del mensaje de entrada de manera no lineal, lo que dificulta que dos entradas diferentes produzcan el mismo resultado.
- **Tamaño del hash:** Un tamaño de hash más grande aumenta el número posible de valores hash, lo que reduce la probabilidad de colisiones.
- **Algoritmos probados:** Las funciones hash se basan en algoritmos matemáticos bien estudiados y que han demostrado ser resistentes a ataques conocidos.

1.5 Indique 5 parámetros, con su nombre y definición que podría pasarle a una función HASH

1. **Mensaje de entrada (input_message):** Es la información que se desea transformar mediante la función hash. Puede ser cualquier tipo de datos, como texto, números, o incluso archivos binarios.
2. **Longitud del hash (hash_length):** Indica la longitud deseada para el resultado de la función hash. Determina la cantidad de bits o caracteres que conformarán el hash generado.
3. **Algoritmo de hash (hash_algorithm):** Especifica el método o algoritmo que se utilizará para realizar la función hash. Ejemplos comunes incluyen SHA-256, MD5, y SHA-1, entre otros.
4. **Semilla (seed):** Es un valor inicial que se utiliza en algunos algoritmos de hash para influir en la generación del hash resultante. La inclusión de una semilla puede afectar la aleatoriedad y la seguridad del hash.
5. **Iteraciones (iterations):** Indica la cantidad de veces que se aplicará el algoritmo de hash sobre el mensaje de entrada. Aumentar el número de iteraciones puede mejorar la seguridad del hash, pero también aumenta el tiempo de procesamiento.

Escuela de Ingeniería en Informática y Telecomunicaciones
Universidad Diego Portales

Criptografía y seguridad en redes

Pauta Ayudantía Nº 4

Ayudantes: Marcelo Fuenzalida, Diego Pineda

1 Ejercicio 1

HMAC permite autenticar un mensaje sin necesidad de transmitir la clave secreta, utilizando un relleno interno (`0x363636...3636`) de 64 bytes y un relleno externo (`0x5c5c5c...5c5c`) de 64 bytes. Consideremos HMAC-TRUNKn, donde el algoritmo hash TRUNK simplemente trunca el mensaje a los n primeros bytes (retorna los n primeros bytes del flujo de bytes).

Indique, en formato hexadecimal, una clave de 12 bytes y el HMAC-TRUNK20 resultante para el mensaje "**se vienen las solemnes**". La clave no puede contener el carácter nulo.

1.1 Opción de respuesta

Sabiendo que la fórmula de HMAC es la siguiente (según RFC2104):

$$HMAC(K, m) = H[(K \oplus opad)||H((K \oplus ipad)||m)]$$

Dónde:

- H: Función de Hash Criptográfica.
- K: Clave Criptográfica.
- m: Mensaje.
- $opad$: Outer Padding.
- $ipad$: Inner Padding.
- $\oplus$: XOR.
- $||$: Concatenación.

En este caso, se dice que el Hash es TRUNK20, es decir, se utilizan los primeros 20 bytes del mensaje, con esa consideración al momento de hacer el hash solo se tendrán los primeros 20 bytes, los cuales serán el $\oplus$ entre la llave criptográfica con el outer padding, y sabiendo que el outer padding tiene 64 bytes de información (de los cuales se tomarán los primeros 20 bytes de izquierda a derecha) y la clave que tendrá 12 bytes únicamente. Y estás deben aplicar el $\oplus$.

En otras palabras el mensaje se original es decir "**se vienen las solemnes**"no es necesario hacerle ningún procedimiento, y tampoco se debe hacer el proceso de hash con el inner padding.

Gracias a esto solo se debe obtener una clave que no sea el carácter nulo para terminar el ejercicio. Una posible clave podría ser alguna que en formato hexadecimal sea la que provoque que la primera parte del mensaje hasheado sea con `0xff`.

Para este caso, y sabiendo que el outer padding es `0x5c` 20 veces para este caso, se procede aplicar un $\oplus$ que de como resultado un mensaje `0xff`, se procede a buscar un valor que cumpla con la siguiente ecuación:

$$\begin{array}{rr} & \texttt{0x5c} \\ \oplus & \texttt{0xff} \\ \hline & \texttt{0xXX} \end{array}$$

```
   0x5c
⊕  0xff
-------
   0xa3
```

HINT: Una forma de utilizar el ⊕ con hexadecimal es hacer la suma hasta el número 15, cuando es mayor que 15 el valor vuelve a ser 0, que a la larga es lo mismo que sumar en binario, igualmente se mostrará el proceso con los binarios.

```
   01011100
⊕  11111111
-----------
   XXXXXXXX

   01011100
⊕  11111111
-----------
   10100011
```

Que igualmente si se vuelve a transformar a hexadecimal vuelve a ser `0xa3`, por lo tanto la clave criptográfica quedaría como:

Clave: `0xa3a3a3a3a3a3a3a3a3a3a3a3`
Mensaje: `0xffffffffffffffffffffffff5c5c5c5c5c5c5c5c`

Nota: Para escribir un mensaje hexadecimal **siempre** debe tener un `0x` al inicio, para decir que es hexadecimal.

2 Ejercicio 2

Sea el algoritmo de hash Left-N, que retorna los N bits más significativos del mensaje. Calcule en base64 el resultado de HMAC-Left-8 para el mensaje en ASCII **"Pregunta bonus"**. Considere una clave en base64 igual a **"SmE="**. Considere los siguientes valores de padding:

- Outer Padding: `0x010203...3F40`
- Inner Padding: `0x8A8B8C...C8C9`

2.1 Opción de Respuesta

Al igual que el ejercicio anterior, el mensaje en este caso no importa, por lo que solo importa la clave y el outer padding.

Paso 1

Se pasa la clave a digitos binarios:

S	m	E	=
010010	100110	000100	=

NOTA: Como cada mensaje que está en base64 tiene un "=", significa que el mensaje tiene solo 16 bits, por lo tanto, los últimos 2 ceros se obviarán al momento de usar el ⊕.

Paso 2

Se procederá a pasar los primeros 2 bytes de la clave hexadecimal a binario:

0x0	0x1	0x0	0x2
0000	0001	0000	0010

Paso 3

Se aplica XOR con el arreglo de bits y se transforma a base64, agrupando en grupos de 6 los bits de la clave:

```
   010010 100110 000100
⊕  000000 010000 001000
-----------------------
   010010 110110 001100
```

La parte roja es relleno obligatorio por eso es 0, luego transformando a base64 el mensaje queda como **"S2M="**

3 Ejercicio 3

Se le pide encontrar dos strings en ASCII que al pasar a base64, cumpla con las siguientes restricciones:
Que posea un largo mayor a 19 bits y finalizan en "g==".
Así mismo las contraseñas vienen dados por la siguiente expresión regular [/-:A-Za-z]
Mencione la cantidad de caracteres que pueden utilizar nuestros mensajes
Por último, se le pide calcular la entropía de algún mensaje que se rige por la siguiente norma de n = 2 (Hint: Se calculo para saber cuantos caracteres ASCII se usan al momento de transformar a base64). Además, se le agregan a los últimos 3 bytes menos significativos, los caracteres '$', '%' y '&'.

3.1 Opción de Respuesta

Lo primero que hay que hacer en este tipo de ejercicios, es ver las restricciones que son dadas, es decir, que se puede y que no se puede hacer, por lo tanto se partirá viendo cuales son estas restricciones.
Por ende se consiguen que bits se necesitan obtener que valores provocan que se termine en "**g==**".

Sabiendo que:

x → xx==

xx → xxx=

xxx → xxxx

Dónde:
x: serían caracteres en ASCII.
x: serían caracteres en base64.

Se puede saber que se debe cumplir la siguiente ecuación para obtener la cantidad de carácteres ASCII a utilizar

$$3n + 1 \tag{1}$$

Dónde $n \in \mathbb{Z}$

Después se busca que carácteres terminan en "**g==**", haciendo el siguiente procedimiento:

o	g	=	=
o	100000	=	=

Y con esta consideración se puede intuir que solo se necesitan los primeros 2 bits más significativos de el cuadro **g**, que en este caso sería **10**, es decir, todos los mensajes disponibles deben terminar en **10**, lo que deja solamente 1/4 de los carácteres del Regex disponibles para utilizar.

Luego se procede a ver la cantidad de carácteres que se tienen disponible (osea la base).

Sabiendo que los valores de los carácteres son:

$$/ \Rightarrow 47$$
$$: \Rightarrow 58$$
$$A \Rightarrow 65$$
$$Z \Rightarrow 90$$
$$a \Rightarrow 97$$
$$z \Rightarrow 122$$

Y sabiendo que se debe considerar el primer carácter queda que la primera parte del regex queda como:

$$58 - 47 = 11$$

Sumandole el primer carácter que hay que usar queda que se tienen 27 carácteres de uso.
Para el segundo queda como:

$$90 - 65 = 25$$

Sumandole el primer carácter que hay que usar queda que se tienen 26 carácteres de uso.
Y para el tercero queda como:

$$122 - 97 = 25$$

Sumandole el primer carácter que hay que usar queda que se tienen 26 carácteres de uso.
Teniendo en total:

$$12 + 26 + 26 = 64$$

Quedando con 64 carácteres de uso en toda la regex.

Finalmente se divide por 4 esta cantidad de carácteres para ver cuantos caráctes hay disponibles para el carácter final del mensaje, el cual sería 16 carácteres para el carácter final del mensaje.

Por lo que 2 mensajes posibles podrían ser sin problemas:

"AAAB"y "BBBB"

Para la segunda parte, hay que considerar que el $n = 2$, para ver la entropía, que reemplazando el la fórmula dada en la ecuación (1), dejando que la cantidad de carácteres sea de 7.
Utilizando la fórmula de entropía

$$H = L \cdot \log_2(W) \tag{2}$$

Dónde:

H: Entropía en bits
L: Cantidad de carácteres
W: Base del mensaje
Con las consideraciones dadas, 6 de 7 carácteres tienen una base de 64, el séptimo tiene una base de 16 y los últimos 3 carácteres, que deben se carácteres obligatorios tienen una base de 1. Quedando de la siguiente forma:

$$\begin{aligned} H &= 6 \cdot \log_2(64) + 1 \cdot \log_2(16) + 3 \cdot \log_2(1) \\ H &= 6 \cdot 6 + 1 \cdot 4 + 3 \cdot 0 \\ H &= 36 + 4 + 0 \\ H &= 40 \end{aligned}$$

Gracias a esto se puede ver que si se agregan carácteres extras obligatorios y fijos no afectan a la entropía y cualquier mensaje con 7 carácteres en este caso tendrá siempre una entropía de 40 bits.

4 Pregunta 4

Les dan el siguiente programa:

```bash
#!/bin/bash

find_char_position() {
    local str="$1"
    local char="$2"
    local pos=-1

    for (( i=0; i<${#str}; i++ )); do
        if [[ "${str:$i:1}" == "$char" ]]; then
            pos=$i
            break
        fi
    done

    echo $pos
}

encrypt_char() {
    local char="$1"
    local keychar="$2"
    local alphabet="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789"
    local shift=$(find_char_position "$alphabet" "$keychar")
    if [ $shift -gt 0 ]; then
        echo "$char" | tr "${alphabet}" "$(echo ${alphabet} | cut -b$((shift+1))-$(#alphabet))$(echo ${alphabet} | cut -b1-$shift)"
    else
        echo "$char"
    fi
}

encrypt() {
    local plaintext="$1"
    local key="$2"
    local encrypted=""
    for (( i=0; i<${#plaintext}; i++ )); do
        local char="${plaintext:$i:1}"
        local keychar="${key:$((i % ${#key})):1}"
        encrypted+=$(encrypt_char "$char" "$keychar")
    done
    echo "$encrypted"
}

# Verificación de argumentos
if [ $# -ne 2 ]; then
    echo "Uso: $0 '<texto_plano>' '<clave>'"
    exit 1
fi

plaintext="$1"
key="$2"
encrypted=$(encrypt "$plaintext" "$key")
echo "Texto plano: $plaintext"
echo "Clave: $key"
echo "Texto cifrado: $encrypted"
```

Este programa solo lo pueden ver, no pueden modificarlo de ninguna forma.

Con este programa deben identificar, que algoritmo de cifrado clásico utiliza y la base utilizada. Utilizando este algoritmo, se cifra un mensaje (que usa la misma base del algoritmo) y se concatena el caracter "$" en el byte menos significativo del mensaje. Indique la entropía en bits de la password.

Indique alguna forma para que este programa, al momento de ejecutarlo se transforme en otro que se haya visto en clases y ¿cual sería?.

4.1 Opción de respuesta

Se puede ver que el programa que fue dado, consta de 3 funciones, lo importante en este caso, es darse cuenta que lo que hace la función `find_char_position()`, es encontrar la posición que tiene cada carácter del string `alphabet`. Que la función `encrypt_char()` hace un corrimiento en base a la posición del carácter entregado, dejando ya 2 opciones posibles para saber cuales son los posibles cifrados clásicos disponibles, dejando solo las opciones de Vigenère y Cesar. Finalmente con la funcion `encrypt()` se puede ver que la clave es un string, lo que solo deja la opción de que sea un cifrado Vigenère. Con la base se puede ver que es alfanúmerica case sensitive, es decir, toma tanto mayúsculas como mínusculas, teniendo en sí 62 carácteres de base.

Si se cifra un mensaje con este algoritmo se podría decir, que su entropía en bits sería en base a la ecuación (2), quedaría como:

$$H = L \cdot \log_2(62) + 1 \cdot \log_2(1)$$
$$H \approx L \cdot 6$$

En otras palabras su entropía en bits sería de 6L, dónde se depende del valor que tenga L.

Como se ha visto en ayudantías anteriores, otra forma de hacer un cifrado ROT-N sería aplicar Vigenère, considerando que el carácter a utilizar es el valor que tiene en base a su posición del alfabeto, por lo que este programa podría cifrar de forma ROT-N si la clave es un solo carácter.

Escuela de Ingeniería en Informática y Telecomunicaciones

Universidad Diego Portales

Criptografía y seguridad en redes

Pauta Ayudantía Nº 5

Ayudantes: Marcelo Fuenzalida, Diego Pineda

¡¡ SIMETRICO !!

① Ejercicio 1

→ cifrar, descifrar y cifrar, con llaves distintas o iguales (3).

Considerando el cifrado 3DES, en modo de operacion ECB, con bloques de 4 bytes. Con el añadido de al ingresar un mensaje se le concatena el simbolo '#' y caracter '0'. Indique 2 mensajes para cada caso: → 32 bits.

→ no dependo del bloque anterior.

- Primer bloque de cifrado igual, el resto distinto
- Primer y tercer bloque igual, el resto distinto
- Segundo bloque igual, el resto distinto
- Tercer bloque igual, el resto distinto

1.1 Opción de respuesta

Lo que hay considerar en este ejercicio es solamente el tipo de cifrado que se utiliza además de los bloques que se piden, además de la cantidad de bytes por bloque. Ya que lo que piden especificamente que al momento de cifrar con 3DES-ECB debe quedar los bloques correspondientes iguales.
Con eso se deben poner 2 mensajes, que pueden ser:

1. Para dos mensajes que el primer bloque de cifrado y los otros distintos podrían ser:

	Bloque 1	Bloque 2	Bloque 3
Mensaje 1	Hola	como	esta
Mensaje 2	Hola	diga	cosa

→ 8 bits. = cada caracter.

Siendo en si los mensajes "Holacomoesta"y "Holadigacosa".

2. Para que el primer y tercer es igual y el resto distinto.

	Bloque 1	Bloque 2	Bloque 3
Mensaje 1	Hola	como	esta
Mensaje 2	Hola	diga	esta

Siendo en si los mensajes "Holacomoesta"y "Holadigaesta".

3. Para que el segundo bloque sea igual y el resto distinto.

	Bloque 1	Bloque 2	Bloque 3
Mensaje 1	Hola	[illegible]diga	esta
Mensaje 2	Chao	diga	cosa

Siendo en si los mensajes "Holacomoesta"y "Chaodigacosa".

4. Para que el tercer bloque sea igual y el resto distinto.

	Bloque 1	Bloque 2	Bloque 3
Mensaje 1	Hola	como	esta
Mensaje 2	Chao	diga	esta

Siendo en si los mensajes "Holacomoesta"y "Chaodigaesta".

2 Ejercicio 2

Utilice el mensaje "HOLACHAO", escoja una llave y cifrelo con tres modos de cifrados distintos. Considere bloques de 4 bytes. Considere nonce y counter de tamaño 16 bits y un IV de 32 bits en caso de requerirlos. Para los modos de cifrado escogidos, considere que la encriptacion por bloques de cifrado, utiliza solo XOR. Sea la llave 0x00000000.

2.1 Opción de respuesta

- **ECB:**
  1. El primer bloque es HOLA → 0x484F4C41.
  2. El cifrado del primer bloque es: 0x484F4C41 (Lo mismo).
  3. El segundo bloque es CHAO → 0x4348414F.
  4. El cifrado del segundo bloque es: 0x4348414F (Lo mismo).

  Dejando que el mensaje cifrado, usando ECB es 0x484F4C414348414F (Lo mismo).

- **CBC:**
  1. Sea un IV igual a 0x00000000.
  2. El primer bloque es HOLA → 0x484F4C41.
  3. El cifrado del primer bloque es: 0x484F4C41 (Lo mismo).
  4. El segundo bloque es CHAO → 0x4348414F.
  5. Se aplica un XOR con el cifrado del primer bloque, dejando 0x0B070D0E.
  6. El cifrado del segundo bloque es: 0x0B070D0E.

  Dejando que el mensaje cifrado, usando ECB es 0x0x484F4C410B070D0E

- **CTR:**
  1. Sea un nonce igual a 0x0000.
  2. Sea un counter igual a 0x0000.
  3. Se aplica un XOR entre Nonce⊕COUNTER, dejando la clave como 0x00000000.
  4. El primer bloque es HOLA → 0x484F4C41.
  5. El cifrado del primer bloque es: 0x484F4C41 (Lo mismo).
  6. El counter aumenta quedando ahora como 0x0001.
  7. Se aplica un XOR entre Nonce⊕COUNTER, dejando la clave como 0x00000001.
  8. El segundo bloque es CHAO → 0x4348414F.
  9. El cifrado del segundo bloque es: 0x4348414E.

  Dejando que el mensaje cifrado, usando ECB es 0x484F4C414348414E.

0x18 → 0001 1000 → 24.
hexa → binario → decimal.

Ejercicio 3

Escoja un stream (en ASCII) de largo 0x18 bytes (no es necesario que el mensaje tenga sentido). Considere que se opera sobre bloques de 16 bytes. En caso de requerir relleno, deberá utilizar PKCS#7. Aplique cifrado de flujo intermitente cada 32 bits a sus bloques, comenzando desde el bit más significativo del stream utilizando como llave la secuencia "10" (para cuatro bits, la llave será 1010). Utilice la operación XOR entre el mensaje y la llave para generar su mensaje cifrado. Indique 2 mensajes distintos, donde sus mensajes cifrados sean idénticos y que el bit de paridad generado a partir de ambos mensajes cifrados sea impar.

32 → :8 = 4 bytes.

3.1 Opción de respuesta

Lo primero que se debe hacer es traducir 0x18 que está en hexadecimal, por lo que se debe pasar a decimal.

[cifraren una sí, en otra no]

0x1	0x8
0001	1000
$2^7 \cdot 0 + 2^6 \cdot 0 + 2^5 \cdot 0 + 2^4 \cdot 1$	$2^3 \cdot 1 + 2^2 \cdot 0 + 2^1 \cdot 0 + 2^0 \cdot 0$
16	8

Dando como resultado 24, por lo tanto se piden 24 bytes que sería lo mismo que 24 carácteres. 24-16=8
Si se opera sobre bloques de 16 bytes, se tendrá el primer bloque de 16 bytes y un segundo bloque 8 bytes y se deberá rellenar con 8 bytes de valor 0x08, esto debido estándar PKCS#7.
La clave a utilizar para cada byte será 10101010.
El segundo, cuarto y sexto grupo de mensajes de 4 bytes deben ser idénticos, ya que deben mantener su forma.
Para que los cifrados sean idénticos, toca trabajar el texto en una base que no tenga carácteres que se puedan imprimir, en dónde los carácteres cifrados se representen de la misma forma, ya que de otra manera no será factible obtener dos mensajes idénticos. misma llave.
El carácter correspondiente a 10101010 es 170, el cual es "¬", por lo que dicho valor retornará un carácter que no se puede imprimir, ya que la interfaz está límitada y estos carácteres se muestran como un punto.
El carácter ® tiene diferencia de dos bits (valor 169), por lo que representará a otro carácter no imprimible y se mantenderá con paridad par.
Dado que se requiere que el cifrado tenga paridad impar, se sabe que: "da lo mismo que los cifrados retornen par o impar, dado que la cantidad de bloques es par".
Por lo que solo es necesario que los bloque sin cifrar, únicamente uno de ellos sea impar.
Dando el siguiente resultado:

Mensaje 1: ¬¬¬¬ BBBB ¬¬¬¬ BBBB ¬¬¬¬ BBBA

Mensaje 2: ® ® ® ® BBBB ® ® ® ® BBBB ® ® ® ® BBBA

(1) padding hasta 256 bytes / puro 0xff
PKCS#7: 01 → 1 byte → 8 bits → 4 bits clave.
02 02 → 2 byte → 16 bits
03 03 03 → 3 byte → 24 bits.
↓ mismo procedimiento

(2) PKCS#5 / puros 8
8 bytes / 0808080808080808 max

Escuela de Ingeniería en Informática y Telecomunicaciones

Universidad Diego Portales

Criptografía y seguridad en redes

Pauta Ayudantía Nº 6

Ayudantes: Marcelo Fuenzalida, Diego Pineda

1 Ejercicio 1 Diffie-Hellman.

Alice desea enviar un mensaje secreto a Bob, pero no le ha compartido una llave de forma previa y se encuentra en un canal inseguro. Alice elige un número secreto a=4 y le envía a Bob el número primo p = 11, la base g = 2, y el resultado del logaritmo discreto. Bob, a partir de su número secreto b=5, calcula la operación y se la transmite a Alice. Determine la llave que generan ambos usuarios al utilizar la operatoria de logaritmo discreto. Considerando que el valor numérico resultante está representado como caracteres de la tabla ASCII, conviértalo a su respectiva base64.

1.1 Opción de respuesta

Cabe aclarar que la operatoria del logaritmo discreto es lo que se utiliza en el método de intercambio de claves de Diffie-Hellman, esta función se puede resumir en el siguiente procedimiento:

1.1.1 Algoritmo

- Se establece un número primo p y un generador $g \in \mathbb{Z}_p^*$ (Este dato es público).
- El emisor escoge $a \in \mathbb{Z}_{p-1}^*$ al azar, calcula $A = g^a \bmod p$ y se envía A al receptor.
- El receptor se escoge $b \in \mathbb{Z}_{p-1}^*$ al azar, calcula $B = g^b \bmod p$ y envía B al emisor

Nótese que A y B pueden calcular el valor de $K = A^b \bmod p$ que el valor del logaritmo discreto.

También se puede llegar de estas formas:

$$\begin{aligned} K &= A^b \bmod p \\ &= (g^a \bmod p)^b \\ &= g^{a \cdot b} \bmod p \\ &= (g^b \bmod p)^a \\ &= B^a \bmod p \end{aligned} \quad (*)$$

IMPORTANTE: mod es aplicar el módulo, es decir dar el resto de la función al dividirla.
Sabiendo esto se hace el siguiente procedimiento para obtener la clave:
Considerando que:

- p = 11
- g = 2
- a = 4
- b = 5

Se aplica el logaritmo discreto:

- Alice calcula A: $2^4 \bmod 11 = 5$
- Bob calcula B: $2^5 \bmod 11 = 10$
- Bob calcula el número secreto: $5^5 \bmod 11 = 1$
- Alice calcula el número secreto: $10^4 \bmod 11 = 1$

El valor de la clave es 1 es ASCII, el cual corresponde el valor 49 en decimal, que en bits sería igual a 00110001. Que expresandolo a base64 se obtiene que:

001100	010000	=	=
M	Q	=	=

El cual es el valor MQ==.

2 Ejercicio 2 RSA.

Sean los valores p=5, q=11, e=7, d=23. Verifique que los valores cumplen con las condiciones necesarias para generar una llave privada y una publica, y luego cifre el siguiente mensaje: "bote ". Considere rellenar con PKCS#7 en caso de ser necesario. Represente el resultado en base64.

2.1 Opción de respuesta

Viendo los parámetros que son dados, se puede deducir que se debe usar RSA, el cual es un algoritmo de clave pública, el cual es el más útilizado. Resuelve el problema de como enviar un mensaje codificado a alguien sin tener que enviar previamente el código de ello.

2.1.1 Algoritmo

1. Se escogen dos números diferentes p y q, que se deben ser aleatorios, grandes y de una longitud similar. y primos.
2. Se halla el producto de p y q para hallar n.
$$n = p \cdot q$$
3. Con la función ϕ de Euler se calcula:
$$\phi(n) = (p-1) \cdot (q-1)$$ } verificaciones del e y el d.
4. Se escoge un número natural e que sea menor y primo de $\phi(n)$
$$1 < e < \phi(n)$$
5. Se encuentra una clave privada d por media de una operación modular en dónde d es el inverso de e mod ϕ
$$1 = e \cdot d \bmod \phi(n)$$

Sus formas de cifrar y descifrar son las siguientes:

- Cifrado: clave privada.
$$C = M^e \bmod n \rightarrow p \cdot q$$
mensaje en binario en bloques.
- Descifrado: → de la publica.
$$M = C^d \bmod n$$
mensaje cifrado en binario en bloques.

Cabe aclarar que m se divide en bloques tal que el tamaño sea:

$$TRUNK(log_2 n)$$
↳ Solo el entero!

Para que la clave pública e sea utilizable en RSA, se debe corrobar que no tenga múltiplos comunes con $(p-1)\cdot(q-1)$, además con los datos del ejercicio:

- $e = 7$ } ? No se puede hacer RSD, p, q, e y d me lo tienen que dar.
- $\phi(n) = (5-1)\cdot(11-1) = 40$
- $1 < 7 < 40$

Por otro lado, para determinar la clave privada d se debe cumplir que:

$$1 = e \cdot d \bmod \phi(n)$$

Y reemplazando se tiene que:

$$\begin{aligned} 1 &= 7 \cdot 23 \bmod 40 \\ &= 161 \bmod 40 \end{aligned}$$ → cualquier otro valor esta malo, siempre 1.

Y para cifrar el mensaje "bote "

Y considerando que $p \cdot q = 5 \cdot 11 = 55$, y que:

$2^x = 55$.

$$TRUNK(log_2 55) = 5$$ → [numero exato } parte entera.] → decimal cercano.

Al pasar el mensaje "bote " a bits queda como 01100010 01101111 01110100 01100101 00100000 Que al dividir los bloques en grupos de 5 queda como: 01100 01001 10111 10111 01000 11001 01001 00000] trunk para hacer los bloques.

Que daría que cada carácter del mensaje fuera:

$M^e \bmod n$

$01100 = 12$	$C_1 = 12^7 \bmod 55 = 23$
$01001 = 9$	$C_2 = 9^7 \bmod 55 = 4$
$10111 = 23$	$C_3 = 23^7 \bmod 55 = 12$
$10111 = 23$	$C_4 = 23^7 \bmod 55 = 12$
$01000 = 8$	$C_5 = 8^7 \bmod 55 = 2$
$11001 = 25$	$C_6 = 25^7 \bmod 55 = 20$
$01001 = 9$	$C_7 = 9^7 \bmod 55 = 4$
$00000 = 0$	$C_8 = 0^7 \bmod 55 = 0$

"bote_" → ASCII / HEXA → binario [8 bits]

- Sino caba tenes que usar PSKA.
- sino 0.

5 bits

Que pasando a binario queda como 10111 00100 01100 01100 00010 10100 00100 00000

101110	010001	100011	000001	010100	001000	000000	=
u	R	j	B	U	I	A	=

agrupas en 6 para base 64.

Siendo el mensaje **uRjBUIA=**.

$N = C^d \bmod n$

Ejercicio 3

Suponiendo que Alice se comunica con Bob a través de un canal inseguro utilizando Diffie-Hellman. Al crear una comunicación, un atacante utilizando MitM intercepto el mensaje. Indique 3 medidas que pudiera tomar Alice o Bob para detectar o prevenir este tipo de ataque.

3.1 Opción de respuesta

1. Autenticación mediante contraseñas compartidas
   - **Método:** Alice y Bob pueden usar una contraseña secreta previamente compartida, conocida solo por ellos. Esta contraseña se utiliza para autenticar los valores intercambiados durante el establecimiento de la clave.
   - **Detección/Prevención:** Alice y Bob pueden utilizar la contraseña secreta para derivar un valor adicional que sea intercambiado y verificado junto con los valores de Diffie-Hellman. Un atacante que no conoce esta contraseña secreta no podrá generar el valor correcto, y así Alice y Bob podrán detectar la presencia del atacante.
2. Hashing del intercambio de claves con un secreto compartido (HMAC):
   - **Método:** Alice y Bob pueden aplicar una función hash con clave (HMAC) utilizando una clave secreta compartida previamente conocida solo por ellos para los valores intercambiados en el proceso Diffie-Hellman.
   - **Detección/Prevención:** Una vez completado el intercambio, ambos pueden comparar los resultados del HMAC. Si los resultados coinciden, pueden estar seguros de que no hubo intervención de un MitM. Un atacante que no conoce la clave secreta no podrá generar un HMAC válido, y su intervención será detectada.
3. Utilización de valores públicos pre-acordados:
   - **Método:** Alice y Bob pueden acordar previamente valores públicos (claves públicas estáticas) que no cambian con cada sesión. Estos valores se intercambian una sola vez en un entorno seguro.
   - **Detección/Prevención:** Durante el intercambio de claves, ambos pueden utilizar estos valores preacordados para realizar el cálculo del secreto compartido. Un atacante no podrá reemplazar estos valores con los suyos propios sin ser detectado, ya que Alice y Bob conocen de antemano cuáles son los valores correctos.

4. Que el mensaje venga firmado.

Escuela de Ingeniería en Informática y Telecomunicaciones

Universidad Diego Portales

Criptografía y seguridad en redes

Pauta Ayudantía Nº 7

Ayudantes: Marcelo Fuenzalida, Diego Pineda

IMPORTANTE

Estas respuestas no son una solución definitiva, sino una sugerencia entre varias opciones posibles. En el ámbito de las preguntas abiertas, la creatividad y la diversidad de perspectivas son clave. Por lo tanto, no se sientan presionados a memorizar esta respuesta, sino que utilícenla como punto de partida para desarrollar sus propias ideas.

1 Ejercicio 1

Sean 2 clientes (Alice y Bob con sus llaves privadas y publicas respectivas). ¿Que llaves y en que orden debería aplicar Alice sobre un mensaje para cifrarlo (y que solo Bob lo pueda leer) y luego firmarlo (para que Bob sepa que su emisor es Alice)?

1.1 Opción de respuesta

Para cifrar y firmar un mensaje de forma segura entre Alice y Bob, se deben seguir estos pasos.

Cifrado [simetrico / asimetrico].

- **Alice obtiene la clave pública de Bob:** Es importante que Alice verifique la autenticidad de la clave pública de Bob para evitar suplantaciones. Esto se puede hacer mediante certificados digitales o canales de comunicación confiables.
- **Alice cifra el mensaje con la clave pública de Bob:** Alice utiliza la clave pública de Bob para cifrar el mensaje original (M). El resultado del cifrado se denomina texto cifrado (C). El método de cifrado puede ser por ejemplo RSA, que debe utilizar la siguiente fórmula:

$$C = M^e mod \text{ n}$$

Dónde:

  - **C:** Texto cifrado
  - **M:** Mensaje original
  - **e:** Exponente público de Bob
  - **n:** Módulo de RSA compartido entre Alice y Bob

Firma Asimetrico:

- **Alice calcula el hash del mensaje:** Alice crea un resumen del mensaje original (M) utilizando una función hash criptográfica. El hash (H) es una cadena de caracteres única que representa el mensaje.
- **Alice firma el hash con su clave privada:** Alice utiliza su clave privada (d) para firmar el hash del mensaje. La firma digital (S) se calcula como:

$$S = H^d mod \text{ n}$$

Dónde:

- **S:** Firma digital
- **H:** Hash del mensaje
- **d:** Exponente privado de Alice
- **n:** Módulo de RSA compartido entre Alice y Bob

- **Alice envía el mensaje cifrado (C) y la firma digital (S) a Bob.** Bob es el único que puede descifrar y verificar el mensaje.

Descifrado y verificación de la firma

- **Bob descifra el mensaje con su clave privada:** Bob utiliza su clave privada (d) para descifrar el texto cifrado (C). El mensaje original (M) se obtiene como:

$$M = C^d mod n$$

- **Bob verifica la firma digital con la clave pública de Alice:** Bob utiliza la clave pública de Alice (e) para verificar la firma digital (S). El hash del mensaje original (H') se calcula como:

$$H' = S^e mod n$$

Bob compara el hash calculado (H') con el hash recibido (H). Si coinciden, significa que el mensaje no ha sido alterado y que efectivamente proviene de Alice.

2 Ejercicio 2

Usted está trabajando en una empresa, cuyo sistema de autenticación que utiliza al momento de firmar documentos sensibles, correos electrónicos, etc., es la firma electrónica simple. Explique si es aconsejable utilizar ese tipo de firma. Justifique, por qué sería bueno o por qué no sería bueno, la utilización de ese tipo de firma.

2.1 Opción de respuesta

La firma electrónica simple (FES) ofrece ventajas significativas para la autenticación de documentos y correos electrónicos en el ámbito empresarial, pero también presenta algunas limitaciones que deben considerarse:

Ventajas

- **Comodidad y facilidad de uso:** La FES es un método accesible y sencillo de implementar, no requiere hardware especializado ni conocimientos técnicos complejos. Los usuarios pueden firmar documentos y correos electrónicos desde cualquier dispositivo con acceso a internet.
- **Agilización de procesos:** La firma electrónica elimina la necesidad de imprimir, firmar manualmente y escanear documentos, lo que agiliza considerablemente los flujos de trabajo y reduce el tiempo dedicado a tareas administrativas.
- **Mejora la seguridad:** La FES agrega una capa de seguridad a las comunicaciones electrónicas, ya que permite verificar la identidad del firmante y la integridad del documento. Esto ayuda a prevenir fraudes, suplantaciones de identidad y manipulaciones de documentos.

Limitaciones

- **Nivel de seguridad moderado:** La FES no ofrece el mismo nivel de seguridad que la firma electrónica avanzada (FEA), ya que no garantiza la identidad del firmante de manera fehaciente. Esto la hace menos adecuada para documentos altamente confidenciales o de alto riesgo.
- **Posibilidad de repudio:** En algunos casos, un firmante podría negar haber firmado un documento electrónico con FES, lo que podría generar disputas legales.

Escuela de Ingeniería en Informática y Telecomunicaciones
Universidad Diego Portales
Criptografía y seguridad en redes
Pauta Ayudantía Nº 8

Firma
Simetrico
Asimetrico./Tipo Control

Ayudantes: Marcelo Fuenzalida, Diego Pineda

1 Ejercicio 1

LEER.

Explique el uso y que resulta de aplicar las siguientes implementaciones de 3DES, donde K1, K2, y K3 son las tres llaves utilizadas por el algoritmo (K3 es la que opera sobre el mensaje):

- Todas las claves son distintas
- K1=K2, K3 distinta
- K1=K3, K2 distinta
- K2=K3, K1 distinta
- K1=K2=K3

1.1 Opción de respuesta

Para esto se debe tener en cuenta de como funciona el algoritmo de cifrado 3DES, el cual es de la siguiente manera:

$$C_{3DES} = E_{K1}(D_{K2}(E_{K3}(M)))$$

$$C_{2DES} = E_{K1}(D_{K2}(M))$$

Dónde

- E_{K3}: Cifrar con la clave 3
- D_{K2}: Descifrar con la clave 2
- E_{K1}: Cifrar con la clave 1

Con esta consideración, se puede ver que cada clave tiene su nivel de importancia en este tipo de cifrado. Por lo tanto viendo cada caso, se podría concluir que:

1.1.1 Todas las claves ~~iguales~~ distintas.

Lo que provocaría esto es que se estaría en el modo más seguro de la aplicación de este algoritmo, ofreciendo la máxima protección posible contra ataques de fuerza bruta y análisis diferencial. Esto debido a que se tendrá una fortaleza porque se utilizarán 168 bits para aplicar.

1.1.2 K1=K2, K3 distinta

Esto provocaría que la seguridad baje, además de hacer que la seguridad sea reducida a un 2DES, esta es más vulnerable a ataques de fuerza bruta, ya que se reducirá el tiempo de buscar una tercera clave, además de ser más vulnerable a los ataques de análisis, debido a que se podrían identidicar patrones del texto cifrado más fácilmente.

1.1.3 K1=K3, K2 distinta

Similar al caso anterior, la clave K2 actúa como un factor diferenciador, pero la repetición de K1 y K3 la hace susceptible a ataques específicos. Estos son el ataque de fuerza bruta y a los ataques de meet-in-the-middle ya que la repetición de K1 y K3 puede ser explotada para realizar ataques de meet-in-the-middle, donde se cifran bloques de texto con K1 y se descifran con K3 buscando coincidencias.

1.1.4 K2=K3, K1 distinta

Análoga a la configuración anterior, esta implementación ofrece seguridad media. La clave K1 aporta diferenciación, pero la repetición de K2 y K3 la vuelve vulnerable a ataques similares.

1.1.5 K1=K2=K3

Esta configuración es la menos segura de las cinco, ya que utiliza una sola clave para las tres rondas de cifrado. Esto la hace equivalente a un cifrado DES simple, y por lo tanto, susceptible a ataques de fuerza bruta y análisis diferencial con mayor facilidad.

2 Ejercicio 2

Considere que la red de Feistel en vez de utilizar 16 rondas, ahora utiliza solo 2 rondas. Utilice padding ISO/IEC 7816-4 como función de expansión y relleno para bloques iniciales de 64 bits. Considere la llave 0xFFFFFFFFFFFFFFFF. Para la salida de las S-Box, considere solo los bits mas significativos de la entrada de las S-Box, eliminando los menos significativos. No considere funciones adicionales en la red. Cifre el mensaje "AAA".

2.1 Opción de respuesta

Hay que recordar que la red de Feistel es una estructura simétrica usada en la construcción de cifrado en bloque. Usa una función de ronda que tiene 2 inputs. El bloque de datos y la subclave. En cada ronda se corre en parte del dato a ser encriptado y la salida es la aplicación XOR $\oplus$ con la otra parte del dato.

Se puede resumir en las siguientes fórmulas:

- Encriptación:

$$L_{i+1} = R_i$$
$$R_{i+1} = L_i \oplus F(R_i, K_i)$$

- Desencriptación:

$$R_i = L_{i+1}$$
$$L_i = R_{i+1} \oplus F(L_{i+1}, K_i)$$

Dónde:

- L: Bloque en la posición izquierda.
- R: Bloque en la posición derecha.
- i: Posición en la parte del proceso dónde se está.
- F: Función utilizada para el cifrado.

Gráficamente se ve como en la Figura 1, más a detalle.

También es importante aclarar que el padding que es pedido, es decir, ISO/IEC 7816-4 es un método para rellenar datos con bytes adicionales para que tengan una longitud fija. Se utiliza comúnmente en criptografía para garantizar que los datos se puedan cifrar y descifrar correctamente.

Este funciona de la siguiente manera:

1. **Añadir un byte de marcador:** Se añade un único byte con el valor 0x80 al final de los datos. Este byte indica el inicio del padding.
2. **Rellenar con bytes de cero:** Se añaden bytes con el valor 0x00 al final de los datos después del byte de marcador. Se añaden tantos bytes de cero como sean necesarios para que la longitud total de los datos sea un múltiplo del tamaño del bloque del algoritmo de cifrado que se va a utilizar.

Lo mejor para esto es hacer toda la lógica en hexadecimal, por lo que se procederá a hacer de esta forma, (también considerar como esto está en hexadecimal se pueden hacer sumas que no superen el número 15, es decir, 0 – 15), y también hay que considerar que un dígito hexadecimal consiste en 4 bits.

Por eso mismo se puede ver que el bloque en 64 bits o 16 dígitos hexadecimal de la siguiente forma: "AAA" como `0x414141`.

También se puede notar que el mensaje no cumple los requisitos de un bloque de la red de Feistel, por eso mismo se debe rellenar con un byte `0x80` y con 8 bytes de valor `0x00`, que es el tipo de relleno utilizado para este caso.

Quedando el mensaje con relleno como: `0x4141418000000000`. La clave dada ya tiene el tamaño necesario para utilizarse en esta red, no se le debe aplicar ningún tipo de relleno, dejandola como: `0xFFFFFFFFFFFFFFFF`.

Ronda 1

Parte Derecha

Para la función F se toman los 32 bits menos significativos del mensaje, se expanden a 48 bits y se aplicar XOR con la subclave, quedando de la siguiente forma:

M		0x000000008000
K	⊕	0xffffffffffff
C		0xffffffff7fff

Cabe aclarar que el texto en rojo es el padding aplicado al bloque para que se cumpla la función de expansión a 48 bits, aplicando el padding **ISO/IEC 7816-4**.

Con todo esto se aplica la función de las S-Boxes, dónde se debe recordar que son 8 S-boxes, de las cuales tienen un input de 6 bits y un output de 4 bits, y como se pide que cada elemento de las S-boxes solo tomen los bits más significativos, quedaría de la siguiente manera.

Primero hay que transformar el mensaje a binario:

0xff	0xff	0xff	0xff	0x7f	0xff
11111111	11111111	11111111	11111111	01111111	11111111

Luego agrupar en sextetos, y como se piden los bits menos significativos se deben tomar los bits que estarán en azul en la siguiente tabla:

S1	S2	S3	S4	S5	S6	S7	S8
111111	111111	111111	111111	111111	110111	111111	111111

Quedando el output de las S-boxes como:

S1	S2	S3	S4	S5	S6	S7	S8
1111	1111	1111	1111	1111	1101	1111	1111

Dando como resultado en hexadecimal `0xFFFFFDFF`.

Con el resultado dado de la función F, se procede a hacer un ⊕ para tener el bloque derecho.

	0xFFFFFDFF
⊕	0x41414180
	0xBEBEBC7F

Parte izquierda

Para esto solo se toma la parte derecha antes de hacer algún proceso, quedando como resultado para esta parte como: `0x00000000`.

Dejando como resultado en la primera ronda el mensaje: `0x00000000BEBEBC7F`.

Ronda 2

Parte Derecha

Para la función F se toman los 32 bits menos significativos del mensaje, se expanden a 48 bits y se aplicar XOR con la subclave, quedando de la siguiente forma:

M		0xBEBEBC7F8000
K	⊕	0xffffffffffff
C		0x414143807fff

} agarras la parte derecha y lo mismo hacia abajo.

Con todo esto se aplica la función de las S-Boxes, dónde se debe recordar que son 8 S-boxes, de las cuales tienen un input de 6 bits y un output de 4 bits, y como se pide que cada elemento de las S-boxes solo tomen los bits más significativos, quedaría de la siguiente manera.

Primero hay que transformar el mensaje a binario:

0x41	0x41	0x43	0x80	0x7f	0xff
01000001	01000001	01000011	10000000	01111111	11111111

Luego agrupar en sextetos, y como se piden los bits menos significativos se deben tomar los bits que estarán en azul en la siguiente tabla:

S1	S2	S3	S4	S5	S6	S7	S8
010000	010100	000101	000011	100000	000111	111111	111111

Quedando el output de las S-boxes como:

S1	S2	S3	S4	S5	S6	S7	S8
0100	0101	0001	0000	1000	0001	1111	1111

Dando como resultado en hexadecimal `0x451081FF`.

Con el resultado dado de la función F, se procede a hacer un ⊕ para tener el bloque derecho.

		0x451081FF
	⊕	0x00000000
		0x451081FF

Parte izquierda

Para esto solo se toma la parte derecha antes de hacer algún proceso, quedando como resultado para esta parte como: `0xBEBEBC7F`.

Dejando como resultado en la primera ronda el mensaje: `0x451081FFBEBEBC7F`.

3 Ejercicio 3

firmas → cifrado asimetrico.

Compare el algoritmo DSA con RSA en términos de tiempo de computo y seguridad. Asimismo, describa como se utilizan las funciones de hash en DSS para asegurar la integridad de los mensajes.

3.1 Opción de respuesta

Comparación entre DSA y RSA

- **Tiempo de cómputo**
  - **Firma:** DSA es generalmente más lento que RSA para la firma de mensajes. } hacer la firma.
  - **Verificación:** RSA es generalmente más lento que DSA para la verificación de firmas. } verificar la firma.
  - **Cifrado:** RSA es más rápido que DSA tanto para el cifrado como para el descifrado.
- **Seguridad** ↓ es algoritmo de cifrado y DSA no.
  - La seguridad de RSA se basa en la dificultad de factorizar números grandes. La seguridad de DSA se basa en la dificultad del problema del logaritmo discreto.
  - Los expertos creen que RSA podría ser vulnerable a ataques cuánticos en el futuro. No se sabe si DSA es vulnerable a este tipo de ataques.

Funciones hash en DSS

Las funciones hash se utilizan en DSS para asegurar la integridad de los mensajes de la siguiente manera:

1. **Se calcula un hash del mensaje:** Un hash es una cadena de bits corta que se genera a partir del mensaje. La función hash debe ser resistente a colisiones, lo que significa que es difícil encontrar dos mensajes diferentes que produzcan el mismo hash.
2. **El hash se firma digitalmente con la clave privada del firmante:** La firma digital es una prueba matemática de que el mensaje fue enviado por el firmante y que no ha sido alterado.
3. **El mensaje, el hash y la firma digital se envían juntos al destinatario.**
4. **El destinatario verifica la firma digital utilizando la clave pública del firmante:** Si la firma es válida, el destinatario puede estar seguro de que el mensaje fue enviado por el firmante y que no ha sido alterado.
5. **El destinatario calcula el hash del mensaje recibido y lo compara con el hash incluido en el mensaje:** Si los hashes coinciden, el destinatario puede estar seguro de que el mensaje no ha sido alterado.

4 Ejercicio 4

Explique cada paso del proceso de cifrado y descifrado en PGP y la importancia de la clave de sesión.

4.1 Opción de respuesta

4.1.1 Cifrado

1. **Generación de clave de sesión:** El remitente genera una clave de sesión aleatoria y simétrica, la cual se utiliza para cifrar el mensaje en sí. Esta clave es única para cada mensaje y se desecha después de su uso, lo que proporciona un alto nivel de seguridad.
2. **Cifrado del mensaje:** El mensaje original se cifra utilizando la clave de sesión aleatoria generada en el paso 1. Esto transforma el mensaje en un formato indescifrable para cualquier persona que no tenga la clave correcta.
3. **Cifrado de la clave de sesión:** La clave de sesión utilizada en el paso 2 se cifra a continuación utilizando la clave pública del destinatario. La clave pública es de conocimiento público y se puede compartir de forma segura con cualquier persona con la que se desee comunicar.
4. **Firma digital (opcional):** El remitente puede firmar digitalmente el mensaje usando su clave privada. Esto crea una firma que verifica la identidad del remitente y garantiza la integridad del mensaje, previniendo su alteración o falsificación.
5. **Empaquetado del mensaje:** El mensaje cifrado, la clave de sesión cifrada y la firma digital (si se ha incluido) se empaquetan juntos en un único mensaje encriptado.
6. **Envío del mensaje:** El mensaje cifrado completo se envía al destinatario a través de un canal de comunicación seguro, como el correo electrónico o un sistema de mensajería instantánea.

4.1.2 Descifrado

1. **Obtención de la clave pública del destinatario:** El remitente debe obtener la clave pública del destinatario de una fuente confiable, como un servidor de claves o un intercambio directo.
2. **Descifrado de la clave de sesión:** El destinatario utiliza su clave privada para descifrar la clave de sesión cifrada recibida en el mensaje.
3. **Descifrado del mensaje:** La clave de sesión descifrada del paso 2 se utiliza para descifrar el mensaje original, convirtiéndolo de nuevo en un formato legible para el destinatario.
4. **Verificación de la firma digital (opcional):** Si el mensaje incluye una firma digital, el destinatario puede utilizar la clave pública del remitente para verificar la firma y garantizar la autenticidad e integridad del mensaje.

4.1.3 Importancia de la clave de sesión

La clave de sesión juega un papel crucial en el proceso de cifrado PGP, ya que es la clave que se utiliza para cifrar y descifrar el mensaje real. Las características clave de la clave de sesión y su importancia son las siguientes:

- **Aleatoriedad:** La clave de sesión debe ser aleatoria e impredecible para evitar que sea descifrada por fuerza bruta o mediante otros métodos criptoanalíticos.
- **Secreto:** La clave de sesión debe mantenerse en secreto tanto para el remitente como para el destinatario. Si la clave de sesión se filtra, cualquier persona con acceso a ella podría descifrar el mensaje correspondiente.
- **Uso único:** La clave de sesión debe ser única para cada mensaje y desecharse después de su uso. Esto evita que la misma clave se utilice para cifrar varios mensajes, lo que reduciría la seguridad general.

5 Ejercicio 5 (POSIBLE CONTROL).

A usted lo mandan a implementar un cifrado RSA, el problema es que el cliente que le está requiriendo esto, solo le pasa los siguientes datos, $p = 5$, $q = 11$, $e = 7$. En ese instante usted se dio cuenta de que por los datos que le proporcionó el cliente además de faltarle la clave privada (d), hay un problema que no permitirá que el algoritmo sea RSA realmente, usted al avisarle al cliente de este problema, le responde: **"Bueno no te puedo cambiar esos datos, así que hazlo igualmente, pero en vez de llamar este algoritmo como RSA, llamalo RSAn't y omite ese requerimiento, y usted encuentre la clave privada, sabiendo que es un número primo y es el mínimo valor que hace que se cumpla la operación modular del inverso"**

Con la consideración que le dió el cliente, debe especificar que error vió que no le permite que el algoritmo sea RSA. Y con el factor de que falta d, como podría obtenerlo y que valor podría ser para aplicar el algoritmo **RSAn't**.

NOTA: El valor de d es un número menor de 30.

$1 - e \cdot d \cdot \text{mod}\,\varphi$

5.1 Opción de respuesta

El problema que se puede ver por lo que no se puede aplicar **RSA** es que los números p y q, no son números primos grandes, por esta razón no se puede aplicar un verdadero **RSA**.

Para solucionar el problema de **RSAn't**,

Se procede a obtener el valor de la función Euler, que sería:

$N = p \cdot q \rightarrow 5 \cdot 11 = 55.$

$$\begin{aligned} \varphi(N) &= (p-1) \cdot (q-1) \\ \phi(55) &= (5-1) \cdot (11-1) \\ \phi(55) &= 4 \cdot 10 \end{aligned}$$

Con esto listo, se procede a obtener un número primero que cumpla con

$$1 = 7 \cdot d \bmod 40$$

Considerando que los números primeros que son descartables automaticamente son 2, 3, 5 y 7, ya que no cumplen con los requisitos que se necesitan para RSA, se debe encontrar el valor d correspondiente probando con los números primos que le siguen son: 11, 13, 19, 23 y 29. Tocará probar con estos para ver cual cumple con los requisitos.

Buscar 1

$$\begin{cases} 37 = 7 \cdot 11 \bmod 40 \\ 11 = 7 \cdot 13 \bmod 40 \\ 13 = 7 \cdot 19 \bmod 40 \\ 1 = 7 \cdot 23 \bmod 40 \\ 3 = 7 \cdot 29 \bmod 40 \end{cases} \rightarrow \text{el que me de 1 mesine.}$$

Se puede notar que el mínimo valor que cumple con estos requerimientos es el número $d = 23$, por lo tanto esa es la clave del cliente.

- Calcular con todos los numeros < 30 primos.] 2, 3, 5, 7, 11, 13, 17, 19, 23, 29.
- e y d siempre diferentes
- todo lo que sea menor al • del valor de ø.
- primero que se encuentra.

$= 7 \cdot 2 \bmod 40$

$14 \% 40 = 0$ ✗

14

$7 \cdot 3$

$21 \bmod 40$:

$21 \% 40 = 0$

21 → mod

Tipos de firma:

vulnerable.

1- firma digitalizada → foto de una firma.
2- " electronica → escribir en el elemento electronico
3: " digital → proceso criptografico para firmar

eficiente

Ventajas Firmas:

* son unicas
* infalsificables
* verificables
* irrenegables
* viables

Estandares:

DES: usa SHA-1 y DSA

PGP:

[redacted]. **Proceso para firmar:**

1: generar la llave de sesión, luego con esa clave cifras la key publica del destinatario, el remitente transmite su clave de sesion para que el destinatario pueda utilizar su key privada para decifrar.

decifrar y cifrar en la ayudantia 8.

Algoritmo: / SIMETRICOS.

DES: cifrado simetricos, es como funcona la red de Fesel con las S-Boxes. muy vulnerable.

→ feistel → formula (2).

3DES: Lo mismo pero tres veces; descifra, cifra, descifra.

AES: cifrado simetrico/funcion de expansion, añade clave de ronda en base KDF, en las rondas 9, 11 y 13 cambia el procedimiento, agrego sus bytes, mueve filas, reordena las columnas y añade una nueva llave de ronda. y hace lo mismo pero sin contar el orden de las columnas.

Blowfish: cifra en bloque pero la clave cambia / es como 3DES.
* es mas seguro, pero la clave es variable.

ASIMETRICO.

RSA: parametros: p, q, e, d. / numeros grandes, primos y longitud similar.

p, q → numeros primos / e → llave publica / d → llave privada. / $e \neq d$ /

formulas: $n = p \cdot q$

$\emptyset = (p-1)(q-1)$. → $1 < e < \emptyset$] e es co-primo de $\emptyset$.

$1 = e \cdot d \cdot \text{mod}\,\emptyset$ → se tiene que cumplir siempre

$\text{TRUNK}(\log_2(n)) = \text{int resultado}$ → cantidad de bloques de cada mensaje.

cifrar $\Rightarrow C = M^{e \to 2} \text{ mod } n$

descifrar $\Rightarrow M = C^{d} \cdot \text{mod } n$.

} me lo deben decir.

Diffie-Hellman: parametros, a, b, g, P, A, B., funcion de logaritmo discreto.

→ $K = A^b \cdot \text{mod } p$ / $K = B^a \cdot \text{mod } p$.

→ $A = g^a \cdot \text{mod } p$ / $B = g^b \cdot \text{mod } p$.]

} a, b numeros secreto.
A, B mensaje para sincronizar.

Firmas.

DSA: algoritmo de firmas: autenticar la firma / no se recomienda para cifrar.

Curva eliptica: se usa para crear firmas, funcion matematica en base a una curva eliptica / valor aleatorio de la curva y con ese numero, es el valor para hacer el numero.

Escuela de Ingeniería en Informática y Telecomunicaciones

Universidad Diego Portales

Criptografía y seguridad en redes

Pauta Ayudantía Nº 9

Ayudantes: Marcelo Fuenzalida, Diego Pineda

IMPORTANTE

Estas respuestas no son una solución definitiva, sino una sugerencia entre varias opciones posibles. En el ámbito de las preguntas abiertas, la creatividad y la diversidad de perspectivas son clave. Por lo tanto, no se sientan presionados a memorizar esta respuesta, sino que utilícenla como punto de partida para desarrollar sus propias ideas.

1 Ejercicio 1

Usted realiza una consultoría en donde su cliente le expone la necesidad de disponer confidencialidad y autenticidad, al mismo tiempo, de los datos contenidos en su servidor de backup. Indique 2 alternativas que podrían dar solución a sus requerimientos e indique dos características distintas para cada alternativa bajo las cuales usted recomienda su uso.

1.1 Opción de respuesta

Como alternativas que provean confidencialidad y autenticidad tenemos los algoritmos **AEAD**, tales como: **AES-GCM** y **Chacha20-poly1305**.
AES-GCM está implementado a nivel de hardware en algunas CPUs, por lo que se recomienda en caso que se disponga de aceleración por hardware. Es un algoritmo más antiguo que **chacha20**, por lo que asegura tener compatibilidad con equipos legacy.
Chacha20-poly1305 al ser un cifrador de flujo es más rápido a nivel de software que AES. Por otro lado, al utilizar llaves de 256 bits y utilizar menos rondas, contribuye a lograr un mejor desempeño en CPUs modernas.

2 Ejercicio 2

¿Que vulnerabilidades ve en las redes de Feistel? Mencione al menos 3.

2.1 Opción de respuesta

- **Ataques Diferenciales:** Este tipo de ataque analiza cómo las diferencias en la entrada pueden afectar las diferencias en la salida. Si el cifrado no está bien diseñado, un atacante puede deducir información sobre la clave utilizando un número limitado de textos cifrados y sus correspondientes textos en claro.
- **Ataques Lineales:** Similar al ataque diferencial, el ataque lineal utiliza aproximaciones lineales de las funciones de cifrado. Este método intenta encontrar relaciones lineales aproximadas entre los textos en claro, los textos cifrados y la clave.
- **Longitud de la Clave:** La seguridad de una red de Feistel también depende de la longitud de la clave. Por ejemplo, DES utiliza una clave de 56 bits, lo cual es susceptible a ataques de fuerza bruta en la actualidad debido al aumento de la capacidad computacional.
- **Número de Rondas:** La seguridad de un cifrado basado en una red de Feistel depende significativamente del número de rondas. Un número insuficiente de rondas puede hacer que el cifrado sea vulnerable a varios ataques criptográficos. Por ejemplo, DES tiene 16 rondas, pero con menos rondas, sería mucho menos seguro.

- **Función de Cifrado Interna:** La elección de la función de cifrado interna (también conocida como función de Feistel) es crucial. Si esta función es débil, toda la seguridad del cifrado puede verse comprometida, independientemente del número de rondas.
- **Ataques de Texto Plano Elegido:** Algunos algoritmos de cifrado basados en redes de Feistel pueden ser vulnerables a ataques de texto plano elegido, donde el atacante puede elegir textos en claro y analizar los textos cifrados resultantes para deducir información sobre la clave.

3 Ejercicio 3

Alice se quiere comunicar con Bob a través de un canal inseguro utilizando el protocolo Diffie-Hellman para establecer una clave compartida. Alice elige un número primo grande p y una base g. Alice elige un número secreto a, pero antes de poder enviar su clave pública a Bob, Eve interviene.

Eve decide ejecutar un ataque de Man-in-the-Middle. Cuando alice envía su clave pública $A = g^a \bmod p$ a Bob, Eve la intercepta y en su lugar envía a Bob una clave pública forjada $E_1 = g^e \bmod p$, donde e es el número secreto de Eve. Bob, sin saber de la interferencia, recibe E_1, calcula la clave compartida y envía su clave pública $B = g^b \bmod p$ a Alice. Eve intercepta esta clave también y envía a Alice una clave pública forjada $E_2 = g^e \bmod p$

Ahora, Alice y Bob creen que están comunicándose entre sí, pero en realidad, ambos están comunicándose con Eve.

A partir del escenario anterior responda a las siguientes preguntas con justificación:

- Indique 3 medidas que podrían tomar Alice y Bob para detectar o prevenir este tipo de ataque.
- Si Eve se aburre y deja de hacer el ataque MiTM. ¿Qué ocurrirá cuando Alice vuelva a conectarse con Bob?

3.1 Opción de respuesta

- Detección o prevención
  1. Alice podría cifrar los valores a compartir con la llave pública de Bob, con el fin que solo Bob los pueda ver, en caso que Alice tuviese la llave pública de Bob de forma previa.
  2. Utilizar una VPN hacia un servidor confiable, con el fin de que un MiTM no pueda interceptar y modificar el tráfico de red.
  3. Alice podría firmar los valores para que Bob verifique que efectivamente fue Alice quien le envió la información, en caso que Bob tuviese la llave pública de Alice de forma previa.
- Para el caso de que Eve deje de hacer MiTM

  Diffie-Hellman al volver a ejecutarse, generará un número secreto entre ambos equipos para crear un enlace seguro. (Se cae la conexión unos instantes).

4 Ejercicio 4

Indique 4 ventajas distintas de utilizar X.509 en vez de PGP. Para cada una, indique un escenario donde se evidencie la ventaja.

4.1 Opción de respuesta

1. En X.509, los certificados son emitidos por CA. En PGP alguien podría suplantar la identidad de un individuo subiendo una llave falsa sin previa autenticación.
2. X.509 actualmente es un protocolo ampliamente soportado por los navegadores. PGP no tiene el mismo nivel de adopción.
3. X.509 permite trabajar con distintas extensiones, donde unas están optimizadas a nivel máquina (lenguaje binario). PGP solo trabaja con extensiones en texto plano, ralentizando el proceso.
4. Los certificados X.509 se validan comprobando la firma, el período de validez, el estado de revocación y la ruta del certificado del certificado. En caso de PGP, la revocación no está explícita en los servidores de llaves públicas.

Ejercicio 5

A pesar de que en el año 2022 se haya lanzado una estrategia de transformación digital: Chile 2035, siguen existiendo brechas con respecto a los procesos de verificación de autenticidad e integridad de documentos y correos electrónicos. Explique cuatro falencias distintas de los actuales métodos de validación de autenticidad e integridad y de qué forma podrían evitarse.

5.1 Opción de respuesta

- A pesar de integrar una FEA, los sw recomendados para visualizar los documentos, no proveen de sistemas capaces de verificar las firmas. Una solución sería incorporar las firmas públicas en los sw recomendados para validar la autenticidad de los documentos.
- A pesar de validar la autenticidad de un documento a través de un ID, algunas plataformas solo indican que el documento es legítimo o no, sin hacer referencia al contenido del documento. Una solución sería subir el documento a verificar y que el sistema lo compare el hash del documento.
- Los correos electrónicos pueden ser suplantados a través de spoofing, facilitando ataques como el phishing. Habilitar interfaces que permitan que el usuario visualizar si los mecanismos de seguridad han sido validados o no permitiría que el usuario sepa si el correo ha cumplido con el mínimo de seguridad requerido a nivel de protocolo.
- Las cuentas de correo pueden llegar a ser víctimas de ataques, donde la identidad digital queda en manos del atacante, permitíendole utilizar la identidad y legitimidad de los servicios del atacado. Utilizar firmas, que solo el usuario legítimo posee, permitiría mitigar este tipo de escenarios.

6 Ejercicio 6

Dado el siguiente output `0xE4B7C3F9A58D`, obtenga el resultado de las S-Boxes en base a que debe ignorar el bit más significativo y el bit menos significativo.

6.1 Opción de respuesta

Primero hay que transformar el mensaje a binario:

0xE4	0xB7	0xC3	0xF9	0xA5	0x8D
11100100	10110111	11000011	11111001	10100101	10001101

Luego agrupar en sextetos, y como se piden los bits del medio, por lo que se deben tomar los bits que estarán en azul en la siguiente tabla:

S1	S2	S3	S4	S5	S6	S7	S8
111001	001011	011111	000011	111110	011010	010110	001101

Quedando el output de las S-boxes como:

S1	S2	S3	S4	S5	S6	S7	S8
1100	0101	1111	0001	1111	1101	1011	0110

Que en hexadecimal queda: `0xC5F1FDB5`

7 Ejercicio 7

Usted hace uso de CRL para verificar certificados. Esta configurado para descargar la lista cada 1 semana. Suponga que la última vez que se actualizó la lista fue el domingo pasado. El día miércoles se detecta un nuevo certificado inseguro y se agrega a la CRL. ¿Cuanto es el periodo de vulnerabilidad?

7.1 Opción de respuesta

Desde el miércoles hasta el próximo domingo (cuando se actualice la CRL nuevamente), hay un período de 4 días en el que el nuevo certificado inseguro no está incluido en la lista. Durante este tiempo, los sistemas que confían en la CRL podrían estar en riesgo. (En pocas palabras 4 días).

8 Ejercicio 8

Escoja un string de largo de 4 bytes. Aplique flujo intermitente cada 8 bits utilizando como llave el mismo mensaje. Utilice la operacion XOR entre la llave y el mensaje para generar su mensaje cifrado. Exprese el mensaje cifrado en base64. Realice este proceso para 2 mensajes distintos, donde sus mensajes cifrados sean identicos y que el bit de paridad de los mensajes cifrados sea par.

8.1 Opción de respuesta

Para este caso se escogerán los strings "AAAA"y "BBBB"en ASCII y sabiendo que la llave es el mismo mensaje. Para cifrar con flujo intermitente, se procede a aplicar $\oplus$ en ambos mensajes con su misma clave, considerando que solo se cifrarán el primer y tercer byte. Primero se transforman ambos mensajes a binario (o hexadecimal), quedando como:

A	A	A	A	B	B	B	B
0x41	0x41	0x41	0x41	0x42	0x42	0x42	0x42
01000001	01000001	01000001	01000001	01000010	01000010	01000010	01000010

Luego se procede a cifrar con flujo intermitente quedando de la siguiente forma:

$$\begin{array}{rr} & \texttt{0x41414141} \\ \oplus & \texttt{0x00410041} \\ \hline & \texttt{0x41004100} \end{array} \qquad \begin{array}{rr} & \texttt{0x42424242} \\ \oplus & \texttt{0x00420042} \\ \hline & \texttt{0x42004200} \end{array}$$

NOTA: Se considera en este $\oplus$ como `0x00` la parte que no se cifra por temas de interpretabilidad.
Finalmente los mensajes quedan como: `0x41004100` y `0x42004200`.

9 Ejercicio 9

Usted obtiene un archivo PDF desde un pendrive. Indique 3 mecanismos distintos con los cuales se podría verificar la autenticidad del documento, sin interactuar con personas. Indique para ello, con qué tendría que cumplir el PDF para poder llevar a cabo los mecanismos antes planteados. Indique para cada uno de los casos, de qué forma un atacante podría saltarse el mecanismo planteado.

9.1 Opción de respuesta

1. **Verificar su firma digital:** El documento debería estar firmado y su llave pública disponible. Un atacante podría suplantar la identidad de un usuario si utiliza PGP, al crear una llave asociada a un correo alternativo.
2. **Verificar su número de folio:** Debiese tener un número de folio asociado a un sistema que permita validar su validez. Se puede cambiar el contenido del pdf, en caso que lo único que haga el comprobador de folio, sea indicar si el documento es legítimo.
3. **Verificar si la metadata corresponde a documentos reales:** El documento debe poseer metadata. El atacante debe confeccionar un documento o editarlo con los mismos valores de metadata que un documento real.
4. **Verificar con un certificado de marca de tiempo:** este certificado asegura la existencia del documento sin modificaciones en una fecha y tiempo especifico. El atacante podria modificar/falsificar la marca de tiempo antes de que se verifique la validez del documento, alterando su contenido.
5. **Verificar con el valor de un HASH que actue como huella del documento:** El atacante puede cambiar el contenido y alterar el hash del documento y actualizar el valor almacenado

10 Ejercicio 10

Usted tiene la llave pública RSA de Alice, que tiene los valores p=5, q=11, e=2, d=23. Para el string "ALO", indique cuál sería el mensaje cifrado en base64 aplicando la llave pública de Alice. En caso de requerir relleno en RSA, utilice el estándar ISO/IEC 7816-4.

10.1 Opción de respuesta

Lo primero que se debe hacer en este momento es verificar si cumple los requerimentos de un algoritmo RSA, para eso se cálcula n y la función ϕ de Euler.

$$\begin{aligned} n = & \quad 13 \cdot 11 & = 55 \\ \phi(55) = & \quad (13-1) \cdot (11-1) & = 40 \end{aligned}$$

Luego se verifica si e cumple con:

$$1 < e < \phi(n)$$

Dónde e debe ser coprimo de $\phi(n)$. Y aunque cumple con que se menor que $\phi(n)$, e NO es coprimo de $\phi(n)$, por lo tanto en este ejercicio no se puede aplicar RSA con los valores dados.

11 Ejercicio 11

Indique 3 mecanismos distintos que le permitan saber a una ISP, a partir del tráfico generado por el cliente, el dominio asociado al servidor https al cual el cliente se intenta conectar. Recuerde que una IP puede hospedar varios dominios.

11.1 Opción de respuesta

- Obtener el dominio por el DNS request que hace el cliente a su server DNS.
- Obtener el dominio por el SNI que envía el cliente en el handshake TLS al server HTTPS.
- Obtener el dominio por el certificado que envía el server en el handshake TLS al server HTTPS.
- Obtener el dominio a traves de la inspeccion de paquetes en la red buscando el SNI (Extension del protocolo TLS durante el handshake).
- Obtener el dominio analizando patrones de flujo de datos que pueden pertenecer a servers HTTPS y analizando los paquetes de este.
- Obtener el dominio cuando un cliente intenta conectarse a un servidor HTTPS, primero realiza una consulta DNS para obtener la dirección IP del servidor asociada al dominio.

12 Ejercicio 12

Utilice el mensaje "#HASHTAG", use la llave 0x00000000 y cífrelo con tres modos de cifrados distintos. Considere bloques de 4 bytes. Considere nonce y counter de tamaño 16 bits y un IV de 32 bits en caso de requerirlos. Para los modos de cifrado escogidos, considere que la encriptación por bloques de cifrado, utiliza solo XOR. Indique el cifrado en base hexadecimal.

12.1 Opción de respuesta

- **ECB:**
  1. El primer bloque es #HAS → `0x23484153`.
  2. El cifrado del primer bloque es: `0x23484153` (Lo mismo).
  3. El segundo bloque es HTAG → `0x48544147`.
  4. El cifrado del segundo bloque es: `0x48544147` (Lo mismo).

Dejando que el mensaje cifrado, usando ECB es `0x2348415348544147` (Lo mismo).

- **CBC:**
  1. Sea un IV igual a `0x00000000`.
  2. El primer bloque es #HAS → `0x23484153`.
  3. El cifrado del primer bloque es: `0x23484153` (Lo mismo).
  4. El segundo bloque es HTAG → `0x48544147`.
  5. Se aplica un XOR con el cifrado del primer bloque, dejando `0x6B1C0014`.
  6. El cifrado del segundo bloque es: `0x6B1C0014`.

  Dejando que el mensaje cifrado, usando CBC es `0x234841536B1C0014`

- **CTR:**
  1. Sea un nonce igual a `0x0000`.
  2. Sea un counter igual a `0x0000`.
  3. Se aplica una concatenación con el Nonce y el COUNTER, dejando la clave como `0x00000000`.
  4. El primer bloque es #HAS → `0x23484153`.
  5. El cifrado del primer bloque es: `0x23484153` (Lo mismo).
  6. El counter aumenta quedando ahora como `0x0001`.
  7. Se aplica una concatenación con el Nonce y el COUNTER, dejando la clave como `0x00000001`.
  8. El segundo bloque es HTAG → `0x48544147`.
  9. El cifrado del segundo bloque es: `0x48544146`.

  Dejando que el mensaje cifrado, usando CTR es `0x2348415348544146`.

13 Ejercicio 13

Usted está viendo el cifrado Vigenère por mera curiosidad, cuando se da cuenta de un detalle bastante interesante de este, que tiene que ver con el cifrado simétrico, al darse cuenta de este gran detalle decide investigar si esto es cierto, y se da cuenta que para el caso de este cifrado se llama cifrado Vernam (o cifrado XOR pa los compas). ¿Qué tipo de cifrado sería y por qué sería este? De un ejemplo.

13.1 Opción de respuesta

Es importante aclarar que el cifrado Vernam puede verse como un caso especial del cifrado Vigenère en el que la clave secreta tiene la misma longitud que el mensaje original y se compone de valores aleatorios. También uno se puede percatar que el tipo de cifrado que hace es el cifrado simétrico de flujo. Se puede considerar cifrado de flujo porque:

- Procesa de datos bit a bit.
- Depende del estado interno.
- Es determinista.
- Es inmutable.
- Y tiene propagación de errores

Un ejemplo podría ser:

- Texto plano: HELLO
- Clave: OCVHW (La llave es random).
- Texto Cifrado: iRTZP

14 Ejercicio 14

¿Qué similitudes y diferencias tienen los algoritmos Rabbit y Salsa20?

14.1 Opción de respuesta

- **Similitudes:**
  - **Cifrado de flujo:** Ambos son cifrados de flujo, lo que significa que generan una secuencia de bits pseudorrandom (keystream) que se combina con el texto claro para producir el texto cifrado.
  - **Velocidad y eficiencia:** Tanto Rabbit como Salsa20 están diseñados para ser rápidos y eficientes, adecuados para aplicaciones que requieren cifrado en tiempo real.
  - **Uso en criptografía moderna:** Ambos algoritmos han sido considerados en competiciones de cifrado y tienen implementaciones en varias bibliotecas criptográficas.
  - **Tamaño de clave:** Los dos algoritmos soportan claves de 128 bits, lo que es suficiente para un nivel alto de seguridad.
- **Diferencias:**
  - **Diseño y estructura:**
    - **Rabbit:** Utiliza un sistema basado en registros y un proceso iterativo que incluye la actualización de estos registros y una función no lineal basada en cuadrados y sumas modulares.
    - **Salsa20:** Usa una estructura de matriz de 4x4 (16 palabras) y una serie de operaciones de mezcla que incluyen sumas modulares, rotaciones y operaciones XOR.
  - **Número de rondas:**
    - **Rabbit:** Realiza 8 iteraciones de su función interna en cada ciclo de generación de keystream.
    - **Salsa20:** Tiene variantes que operan con 8, 12 o 20 rondas, siendo Salsa20/20 (con 20 rondas) la versión estándar más conocida.
  - **Resistencia a ataques:**
    - **Rabbit:** Inicialmente fue cuestionado en cuanto a su seguridad, pero ha sido sometido a varios análisis criptográficos que no han encontrado vulnerabilidades críticas en su diseño original.
    - **Salsa20:** Ha sido ampliamente analizado y se considera muy seguro, sin vulnerabilidades prácticas conocidas hasta la fecha en su versión estándar.

15 Ejercicio 15

¿Que similitudes y diferencias se pueden ver entre el algoritmo DES y Blowfish en cifrado simétrico?

15.1 Opción de respuesta

- Similitudes:
  - Ambos algoritmos son cifrados simétricos, lo que implica el uso de la misma clave para cifrar y descifrar los datos.
  - Ambos pueden ser utilizados en varios modos de operación de cifrado como ECB (Electronic Codebook), CBC (Cipher Block Chaining), CFB (Cipher Feedback), y otros.
  - Ambos algoritmos operan sobre bloques de datos.
- Diferencias:
  - **Longitud de la Clave:**
    - **DES:** Utiliza una clave de 56 bits, lo que limita su seguridad y lo hace vulnerable a ataques de fuerza bruta.
    - **Blowfish:** Soporta claves de longitud variable, desde 32 bits hasta 448 bits, lo que permite un nivel de seguridad mucho más alto.

- **Seguridad:**
  - **DES:** Considerado inseguro hoy en día debido a su corta longitud de clave y la evolución de la capacidad computacional para realizar ataques de fuerza bruta.
  - **Blowfish:** Considerado seguro, siempre y cuando se utilice una longitud de clave adecuada (generalmente 128 bits o más). No se conocen vulnerabilidades significativas en el algoritmo cuando se usa correctamente.
- **Estructura del Algoritmo:**
  - **DES:** Basado en una estructura de Feistel con 16 rondas de procesamiento.
  - **Blowfish:** También utiliza una estructura de Feistel, pero es más flexible en el número de rondas (usualmente 16) y tiene un diseño más complejo con una fase inicial de expansión de la clave y generación de cajas S y P específicas para cada clave.
- **Desempeño:**
  - **DES:** Más rápido en hardware debido a su diseño simple y optimización para hardware.
  - **Blowfish:** Generalmente más rápido en software debido a su eficiencia y diseño, aunque su inicialización (expansión de la clave) puede ser más lenta.

16 Ejercicio 16

¿Es más seguro el algoritmo DES con más de 20 rondas en la red de Feistel y que pasa si se usan 14 rondas? Justifique.

16.1 Opción de respuesta

No, debido a que con más rondas aunque **teóricamente** es más seguro, baja el rendimiento del sistema, se hace más complejo y deja de ser compatible con los estándares.
Debido a que se tienen menos rondas es más suceptible a ataques de fuerza bruta con los recursos computacionales modernos.

17 Ejercicio 17

¿Cuales son los requisitos que cumple el cifrado simétrico AES, la cantidad de combinaciones posibles, que tamaño soportan los bloques y las llaves y cuantas rondas pueden tener (mencione todas las rondas)? Además de eso, mencione en 3 programas que usen este tipo de cifrado.

17.1 Opción de respuesta

Debe cumplir con la siguiente estructura:

- Tiene que tener bloques de 128, 192 y 256 bits.
- Llaves de 128, 192 y 256 bits.
- Tiene 9 combinaciones posibles.
- Soporta bloques y llaves de tamaño $n \cdot 32$ bits.
- Tiene que tener 10(AES-128), 12(AES-192) y 14(AES-256) rondas.

Algunos programas que usen AES son:

- OpenVPN.
- WinRAR.
- 7-Zip.
- Excel
- Portable Document Format desde la versión 1.6

18 Ejercicio 18

Matemáticamente, ¿como sería la diferencia entre 2DES y 3DES? Muestre ambas fórmulas.

18.1 Opción de respuesta

- **2DES:** $E_{K2}(E_{K1}(M))$
- **3DES:** $E_{K3}(D_{K2}(E_{K1}(M)))$

Como se puede ver la diferencia entre 2DES y 3DES es principalmente que 2DES en ningún momento desencripta mientras que 3DES lo hace a la mitad de su proceso.

19 Ejercicio 19

¿Qué diferencias se pueden ver entre RSA, ECDSA y EdDSA? y ¿Alguno de ellos podría sobrevivir a la computación cuántica? Justifique.

19.1 Opción de respuesta

- **RSA (Rivest-Shamir-Adleman)**
  - **Tipo:** Algoritmo asimétrico basado en la factorización de enteros grandes.
  - **Clave:** Utiliza claves grandes (2048 bits o más) para garantizar la seguridad.
  - **Rendimiento:** Más lento en comparación con ECDSA y EdDSA debido al tamaño de las claves.
  - **Seguridad:** La seguridad se basa en la dificultad de factorizar grandes números enteros.
- **ECDSA (Elliptic Curve Digital Signature Algorithm)**
  - **Tipo:** Algoritmo asimétrico basado en curvas elípticas.
  - **Clave:** Utiliza claves más pequeñas para un nivel de seguridad comparable al de RSA con claves mucho más grandes.
  - **Rendimiento:** Más rápido y eficiente que RSA, especialmente con claves más pequeñas.
  - **Seguridad:** La seguridad se basa en la dificultad del problema del logaritmo discreto en curvas elípticas.
- **EdDSA (Edwards-curve Digital Signature Algorithm)**
  - **Tipo:** Variante de ECDSA basada en curvas elípticas Edwards, como Curve25519.
  - **Clave:** Utiliza claves pequeñas y ofrece alta seguridad.
  - **Rendimiento:** Muy eficiente en términos de velocidad y tamaño de las firmas. Está diseñado para ser más rápido y seguro que ECDSA.
  - **Seguridad:** La seguridad se basa en la dificultad del problema del logaritmo discreto en curvas elípticas.

Ninguno de los tres algoritmos mencionados (RSA, ECDSA y EdDSA) es considerado seguro frente a la computación cuántica. El algoritmo de Shor, ejecutado en una computadora cuántica suficientemente potente, podría romper la seguridad de estos algoritmos resolviendo la factorización de enteros grandes (en el caso de RSA) y el problema del logaritmo discreto en curvas elípticas (en el caso de ECDSA y EdDSA) en tiempo polinómico.

Un algoritmo que podría sobrevivir a la computación cuántica es **Lattice-based Cryptography**.

20 Ejercicio 20

Mencione y especifique en qué situaciones es necesario utilizar el cifrado homomórfico. Proporcione tres ejemplos específicos donde su uso sea indispensable.

20.1 Opción de respuesta

Algunas situaciones pueden ser:

- El cifrado homomórfico es una técnica de cifrado que permite realizar operaciones sobre datos cifrados sin necesidad de descifrarlos. Esto es extremadamente útil en varias situaciones donde se necesita mantener la privacidad y seguridad de los datos mientras se procesan.
- El cifrado homomórfico asegura que los datos permanezcan confidenciales incluso mientras se procesan, lo que es crucial en entornos donde la privacidad y la seguridad de los datos son de máxima importancia.
- Para transacciones bancarias
- Transacciones con fichas medicas
- Procesamiento de datos en la nube
- Transacciones de informacion clasificada militar o del gobierno
- Toda aquella transaccion que utilice datos sensibles y necesiten operar con ellos sin necesidad de que otra personas los vea.

21 Ejercicio 21

¿Existe el verdadero RNG? ¿Eso compromete a la ciberseguridad?

21.1 Opción de respuesta

Sí, los verdaderos RNGs existen y se basan en fenómenos físicos impredecibles. Su implementación y uso adecuados son cruciales para mantener la seguridad en sistemas criptográficos. La ciberseguridad puede verse comprometida si se utilizan PRNGs inseguros o mal implementados, debido a que son fáciles de reproducir.

Los tipos de RNG que existen son:

- **Generadores de Números Aleatorios Pseudoaleatorios (PRNG):** Son algoritmos deterministas que utilizan un valor inicial (semilla) para producir una secuencia de números que parece aleatoria. Son rápidos y reproducibles, lo cual es útil para simulaciones y pruebas donde se necesita replicar resultados. No son verdaderamente aleatorios ya que, dado el estado inicial (semilla), se puede predecir la secuencia completa.
- **Generadores de Números Aleatorios Verdaderos (TRNG):** Utilizan fenómenos físicos impredecibles como el ruido térmico, la radiación de fondo, o el ruido electrónico para generar números aleatorios. Son no deterministas y, por lo tanto, proporcionan aleatoriedad genuina. Son más lentos y más difíciles de implementar correctamente en comparación con los PRNG.

22 Ejercicio 22

¿Cuál es la importancia de las subclaves en los algoritmos de cifrado simétrico?

22.1 Opción de respuesta

Las subclaves son componentes fundamentales en los algoritmos de cifrado simétrico porque permiten aumentar la seguridad del cifrado. En estos algoritmos, como el AES (Advanced Encryption Standard), las subclaves se generan a partir de una clave maestra inicial mediante un proceso conocido como expansión de clave. Cada subclave se utiliza para realizar una ronda de cifrado sobre los datos, y la combinación de múltiples rondas con diferentes subclaves aumenta la resistencia del cifrado frente a ataques criptoanalíticos.

Las subclaves son derivadas de la clave principal del cifrado y se utilizan para realizar operaciones de cifrado y descifrado. Al dividir la clave en subclaves, se mejora la seguridad del algoritmo ya que dificulta el análisis criptográfico. Si un atacante intenta romper el cifrado, no solo tendría que descifrar la clave principal, sino también las subclaves que se generan a partir de ella.

Ejercicio extra

Identifique de que materia es cada ejercicio :D

22.2 Respuesta

1. AEAD
2. Red de Feistel
3. Cifrado Asimétrico (Principalmente)
4. Firmas y Certificados
5. Firmas
6. S-Box
7. Certificados
8. Cifrado intermitente
9. Firmas
10. RSA
11. TLS
12. Modos de operación cifrado simétrico
13. Cifrado de flujo
14. eSTREAM
15. Blowfish y DES
16. DES
17. AES
18. 3DES
19. RSA, ECDSA y EdDSA
20. Cifrado homomórfico
21. RNG
22. Subclaves cifrado simétrico.