Responde detallada y razonadamente a las siguientes preguntas. Las respuestas no razonadas no serán consideradas como válidas, aunque sean correctas. Recuadra CLARAMENTE tu respuesta a cada apartado. Consigna tu NIA, nombre y apellidos completos en todas las hojas que entregues.

1. **PROBLEMA 1** (3,5 ptos.)

   Imagina que has podido interceptar el siguiente mensaje M firmado digitalmente, proveniente de Alicia, y dirigido a Bob, $M = HOLA|4$, donde la cadena 'HOLA' es el mensaje, el símbolo | indica concatenación y 4 es la firma del mensaje.

   Contesta razonada y detalladamente a las siguientes cuestiones:

   1. **(1 pto)** Comprueba si la firma del mensaje es válida o no. En el caso de que no lo sea, ¿qué conclusiones podría sacar Bob?
   2. **(2,5 ptos)** A continuación, deberás intentar falsificar el mensaje original, y conseguir que Bob acepte el mensaje $M' = ADIOS$ como auténtico e íntegro, y crea que proviene de Alicia.

   La función hash utilizada opera sobre todo el mensaje x, concatenando los códigos ASCII de cada caracter, y se define como: $h(x) = 3x + 5 \mod 7$. Por ejemplo, $h('UAM') = h(85|65|77) = h(856577)$.

   Las claves públicas de Alicia y Bob son {29,91} y {11,77}, respectivamente.

   NOTA: Explica con el máximo detalle posible cada paso que des. NO se aceptarán como válidas respuestas cuyos datos, parámetros o soluciones no estén justificados.

Figura 1: Tabla ASCII

Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex
(sp)	32	0040	0x20	@	64	0100	0x40	`	96	0140	0x60
!	33	0041	0x21	A	65	0101	0x41	a	97	0141	0x61
"	34	0042	0x22	B	66	0102	0x42	b	98	0142	0x62
#	35	0043	0x23	C	67	0103	0x43	c	99	0143	0x63
$	36	0044	0x24	D	68	0104	0x44	d	100	0144	0x64
%	37	0045	0x25	E	69	0105	0x45	e	101	0145	0x65
&	38	0046	0x26	F	70	0106	0x46	f	102	0146	0x66
'	39	0047	0x27	G	71	0107	0x47	g	103	0147	0x67
(	40	0050	0x28	H	72	0110	0x48	h	104	0150	0x68
)	41	0051	0x29	I	73	0111	0x49	i	105	0151	0x69
*	42	0052	0x2a	J	74	0112	0x4a	j	106	0152	0x6a
+	43	0053	0x2b	K	75	0113	0x4b	k	107	0153	0x6b
,	44	0054	0x2c	L	76	0114	0x4c	l	108	0154	0x6c
-	45	0055	0x2d	M	77	0115	0x4d	m	109	0155	0x6d
.	46	0056	0x2e	N	78	0116	0x4e	n	110	0156	0x6e
/	47	0057	0x2f	O	79	0117	0x4f	o	111	0157	0x6f
0	48	0060	0x30	P	80	0120	0x50	p	112	0160	0x70
1	49	0061	0x31	Q	81	0121	0x51	q	113	0161	0x71
2	50	0062	0x32	R	82	0122	0x52	r	114	0162	0x72
3	51	0063	0x33	S	83	0123	0x53	s	115	0163	0x73
4	52	0064	0x34	T	84	0124	0x54	t	116	0164	0x74
5	53	0065	0x35	U	85	0125	0x55	u	117	0165	0x75
6	54	0066	0x36	V	86	0126	0x56	v	118	0166	0x76
7	55	0067	0x37	W	87	0127	0x57	w	119	0167	0x77
8	56	0070	0x38	X	88	0130	0x58	x	120	0170	0x78
9	57	0071	0x39	Y	89	0131	0x59	y	121	0171	0x79
:	58	0072	0x3a	Z	90	0132	0x5a	z	122	0172	0x7a
;	59	0073	0x3b	[	91	0133	0x5b	{	123	0173	0x7b
<	60	0074	0x3c	\	92	0134	0x5c	|	124	0174	0x7c
=	61	0075	0x3d	]	93	0135	0x5d	}	125	0175	0x7d
>	62	0076	0x3e	^	94	0136	0x5e	~	126	0176	0x7e
?	63	0077	0x3f	_	95	0137	0x5f				

Solución:

1. Comenzaremos comprobando si la firma inicial del mensaje es válida o no. Para ello, recalculamos primero el hash:

$$h(HOLA) = h(72|79|76|65) = h(72797665) = 3 \cdot 72797665 + 5 \mod 7 = 0$$

   Comprobamos ahora si la firma de Alicia es válida, utilizando para ello su clave pública sobre el valor de la firma:

$$S = 4^{29} \mod 91 = 23$$

Como no coinciden ($23 \neq 4$), Bob sabe que el mensaje ha sido manipulado de alguna forma, bien intencionadamente o no. En cualquier caso, no debe confiar en el mensaje.

2. En general, para falsificar la firma digital de un mensaje, y hacerlo pasar como auténtico para Bob, haría falta la clave privada del emisor. En eso se basa toda la seguridad de estos esquemas. Puesto que los módulos implicados son pequeños, podemos directamente factorizar dicho módulo, para obtener los primos p y q originales.
Rápidamente vemos que $N = 91 = 13 \cdot 7$, luego $p = 13$ y $q = 7$ (o al revés, el orden es indiferente). Puesto que conocemos el exponente público, $e = 29$, podemos directamente plantear la ecuación:

$$e \cdot d = 1 \mod \phi(n) = 29 \cdot d = 1 \mod 72$$

Podemos resolver esta ecuación de varias formas, pero un simple prueba y error nos devuelve enseguida que para $d = 5$ obtenemos un valor válido. Solo resta recalcular el hash del mensaje que queremos falsificar:

$$h(ADIOS) = h(65|68|73|79|83) = h(6568737983) = 3 \cdot 6568737983 + 5 \mod 7 = 0$$

Vemos que tiene el mismo valor hash que la cadena 'HOLA', lo que no resulta difícil dado que el valor se reduce módulo 7. Hemos encontrado una **colisión**. Solo resta volver a firmarlo, esta vez con la clave privada:

$$S = 0^5 \mod 91 = 0$$

Finalmente, el mensaje que debemos enviar a Bob es $ADIOS|0$.

Resolución alternativa

En este ejercicio, se introdujo un pequeño *atajo* que hacía la resolución del ejercicio trivial. Al advertir que el hash del segundo mensaje ('ADIOS') es 0, ya no es siquiera necesario calcular la clave privada de Alicia, puesto que, sea cual sea ésta, suponga que x, se verifica siempre que $0^x \mod 91 = 0$ para cualquier x. Por tanto, podríamos concluir directamente que la firma sería $ADIOS|0$, como en el caso anterior.

NOTA: para que esta resolución se considere válida, debe razonarse correcta y adecuadamente, y explicar *explícitamente* que esta situación solo ocurre para h(x) = 0.

2. **PROBLEMA 2** (3,5 ptos.)

Imagina que se te encarga diseñar una aplicación Web, cuyo mecanismo principal de autenticación será un esquema de usuario/contraseña. Para almacenarlas de forma segura, se decide utilizar el método del hash con *salt*.

A lo largo de todo el ejercicio, si debes hacer alguna suposición, justifícala y razónala. Utilizando estrictamente la siguiente notación, responde razonada y detalladamente a las siguientes cuestiones:

Notación		**Tiempo de cálculo**	
h	Función hash	t_h	Tiempo de cálculo de una función hash
		t_c	Tiempo de comparación de dos hashes (por byte)
salt	Salt	t_s	Tiempo de generación de un byte para el salt
HMAC	Función HMAC	t_m	Tiempo de cálculo de una función HMAC

1. **(0,5 ptos)** Describe cómo se almacena una contraseña en la BD, y cómo se verifica cuando es introducida por un usuario. Utiliza esquemas u otros elementos que consideres necesarios para explicar los conceptos.
2. **(0,5 ptos)** Se considera que la contraseñas pueden estar formadas por letras mayúsculas (incluyendo la 'ñ', hay 27 en el alfabeto), minúsculas y números, y tener una longitud de entre 1 y 8 caracteres. ¿Cuántas contraseñas diferentes pueden elegir los usuarios?

3. **(0,5 ptos)** El responsable de desarrollo, decide ahora restringir la elección aplicando una política de contraseñas, que consiste en que éstas deben tener, al menos un número o una letra mayúscula. ¿Cuántas contraseñas diferentes pueden elegir ahora los usuarios?
4. **(0,5 ptos)** Considera que se decide utilizar un *salt* de 8 bytes. ¿Cuánto tiempo tardaría en ejecutarse el proceso de almacenamiento de una contraseña en la BD? ¿Y el de verificación?. Los tiempos no especificados en la tabla anterior pueden considerarse despreciables.
5. **(0,5 ptos)** Imagina ahora que la BD de contraseñas almacena en un momento dado n usuarios. El campo de usuario ocupa un tamaño fijo de u bytes y se utiliza el carácter ':' (que ocupa 1 byte), para separar los distintos campos de cada entrada. ¿Cuál es el tamaño ocupado por la BD?
6. **(0,5 ptos)** Con el salt especificado en nuestro sistema (8 caracteres), ¿en qué factor hace más difícil el ataque con tablas *rainbow* o precomputadas respecto a un sistema sin *salt*? Considera que los caracteres que pueden formar parte del salt son los últimos 128 caracteres de la tabla ASCII.

Solución:

1. En teoría
2. El alfabeto final está compuesto de $27 + 27 + 10 = 64$ símbolos, que pueden combinarse de 64^k formas diferentes, siendo k la longitud de la contraseña. Puesto que la longitud de esta puede variar entre 1 y 8, el número total de contraseñas, n_c, será:

$$n_c = \sum_{k=1}^{8} 64^k$$

3. En este caso, puede ser más sencillo afrontar el problema calculando el número de contraseñas que NO cumplen la política y restarlo del total de posibles contraseñas, obteniendo así el número que sí la cumplen. Por tanto, el número de contraseñas que NO tienen un número Y NO tienen una letra mayúscula (es decir, está compuesto sólo por letras minúsculas) es $\sum_{k=1}^{8} 27^k$ (recuerda que si $\overline{A \vee B} = \overline{A} \wedge \overline{B}$). Finalmente queda que el total de contraseñas en este caso es:

$$n_c = \sum_{k=1}^{8} 64^k - \sum_{k=1}^{8} 27^k$$

4. Comencemos por el proceso de generación de una contraseña almacenada. Para ello deberemos generar 8 bytes de *salt*, lo que tardaría un tiempo $8t_s$. Ahora, se calcula la función hash de ambas, quedando $t_h + 8t_s$. El tiempo de verificación implica recalcular el hash (t_h) y un t_c de comparación para cada byte del hash. Supondremos que utilizamos SHA256 (que es el hash de longitud mínima que deberíamos utilizar), que genera una salida de 32 bytes. Por tanto, el tiempo final sería $t_h + 32t_c$
5. Cada entrada de contraseña tendría el formato [salt:user:contraseña] (el orden es indiferente). El salt ocupa 8 bytes, el usuario u, la contraseña 32 bytes (puesto que es un hash 256), y 2 bytes de los caracteres separadores. Total: $n(42 + u)$ bytes.
6. Cada caracter diferente del salt hace que sea necesario calcular una tabla diferente. Nos dicen que los caracteres posibles del salt son 128, luego hay $128^8 = (2^7)^8 = 2^{56}$ posibles combinaciones, que es el factor que estamos buscando.