Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
Segundo Semestre de 2013

IIC 1103 ⋆ Introducción a la Programación

Proyecto 2

Objetivo General

- Aplicar el uso de funciones.
- Estudiar y aplicar el uso de strings, listas y listas multidimensionales.
- Implementar funciones de ordenación y búsqueda.

Introducción

Una de las principales funciones de la criptografía es lograr una comunicación segura entre dos puntos. Esto significa establecer un canal para enviar mensajes encriptados de tal forma que si un tercero intercepta un mensaje, no sea capaz de decriptarlo para obtener el contenido original.

El protocolo de encriptación One Time Pad (OTP) ofrece una forma de comunicación segura, basada en que los dos puntos que se comunican conocen una *clave* que no conoce nadie más. A continuación se explica cómo funciona este protocolo:

1. Los puntos que se van a comunicar se ponen de acuerdo en una clave. Esta clave debe ser una cadena de ceros y unos, por ejemplo

   $$01100001011000100110001111100001001100010011001000110011.$$

2. El mensaje que se desea enviar se transforma a una cadena de ceros y unos. Para esto, cada caracter se pasa a su número correspondiente de acuerdo a la codificación ASCII (ver Tabla1). Luego, estos números son transformados a números binarios de ocho dígitos, los cuales serán concatenados. Por ejemplo, si el texto es "Mensaje", entonces al pasar cada caracter a ASCII obtenemos 77, 101, 110, 115, 97, 106, 101, lo que corresponde a los números binarios de ocho dígitos que se muestran a continuación:

   $$01001101, 01100101, 01101110, 01110011, 01100001, 01101010, 01100101.$$

   El mensaje a utilizar es la concatenación de los números anteriores, es decir,

   $$01001101011001010110111001110011011000010110101001100101.$$

0	<NUL>	32	<SPC>	64	@	96	`	128	Ä	160	†	192	¿	224	‡
1	<SOH>	33	!	65	A	97	a	129	Å	161	°	193	¡	225	·
2	<STX>	34	"	66	B	98	b	130	Ç	162	¢	194	¬	226	‚
3	<ETX>	35	#	67	C	99	c	131	É	163	£	195	√	227	„
4	<EOT>	36	$	68	D	100	d	132	Ñ	164	§	196	ƒ	228	‰
5	<ENQ>	37	%	69	E	101	e	133	Ö	165	•	197	≈	229	Â
6	<ACK>	38	&	70	F	102	f	134	Ü	166	¶	198	Δ	230	Ê
7	<BEL>	39	'	71	G	103	g	135	á	167	ß	199	«	231	Á
8	<BS>	40	(	72	H	104	h	136	à	168	®	200	»	232	Ë
9	<TAB>	41	)	73	I	105	i	137	â	169	©	201	…	233	È
10	<LF>	42	*	74	J	106	j	138	ä	170	™	202		234	Í
11	<VT>	43	+	75	K	107	k	139	ã	171	´	203	À	235	Î
12	<FF>	44	,	76	L	108	l	140	å	172	¨	204	Ã	236	Ï
13	<CR>	45	-	77	M	109	m	141	ç	173	≠	205	Õ	237	Ì
14	<SO>	46	.	78	N	110	n	142	é	174	Æ	206	Œ	238	Ó
15	<SI>	47	/	79	O	111	o	143	è	175	Ø	207	œ	239	Ô
16	<DLE>	48	0	80	P	112	p	144	ê	176	∞	208	–	240	
17	<DC1>	49	1	81	Q	113	q	145	ë	177	±	209	—	241	Ò
18	<DC2>	50	2	82	R	114	r	146	í	178	≤	210	“	242	Ú
19	<DC3>	51	3	83	S	115	s	147	ì	179	≥	211	”	243	Û
20	<DC4>	52	4	84	T	116	t	148	î	180	¥	212	‘	244	Ù
21	<NAK>	53	5	85	U	117	u	149	ï	181	µ	213	’	245	ı
22	<SYN	54	6	86	V	118	v	150	ñ	182	∂	214	÷	246	ˆ
23	<ETB>	55	7	87	W	119	w	151	ó	183	∑	215	◊	247	˜
24	<CAN>	56	8	88	X	120	x	152	ò	184	∏	216	ÿ	248	¯
25	<EM>	57	9	89	Y	121	y	153	ô	185	π	217	Ÿ	249	˘
26	<SUB>	58	:	90	Z	122	z	154	ö	186	∫	218	⁄	250	˙
27	<ESC>	59	;	91	[	123	{	155	õ	187	ª	219	€	251	°
28	<FS>	60	<	92	\	124	|	156	ú	188	º	220	‹	252	¸
29	<GS>	61	=	93	]	125	}	157	ù	189	Ω	221	›	253	˝
30	<RS>	62	>	94	^	126	~	158	û	190	æ	222	fi	254	˛
31	<US>	63	?	95	_	127	<DEL>	159	ü	191	ø	223	fl	255	ˇ

Tabla 1: Codificación ASCII

3. Teniendo lo anterior se genera el mensaje codificado, que corresponde a la cadena de ceros y unos que será enviada. El n-ésimo caracter del mensaje codificado es un 0 si el n-ésimo caracter del mensaje es igual al n-ésimo caracter de la clave. De lo contrario, el mensaje codificado tendrá un 1 en dicha posición.

   A modo de ejemplo, veamos el mensaje codificado que se generaría con el mensaje del punto 2 y la clave del punto 1:

mensaje	$= 01001101011001010110111001110011011000010110101001100101$
clave	$= 01100001011000100110001111100001001100010011001000110011$
codificación	$= 00101100000001110000110110010010010100000101100001010110$

   La primera posición de ambos strings contiene un 0, por lo que el primer caracter del mensaje codificado es un 0. El segundo caracter del mensaje codificado será también 0, pues en la segunda posición ambos strings contienen un 1. En la tercera posición el primer string contiene un 1 y el segundo un 0, por lo que el tercer caracter del mensaje codificado será un 1. Así se procede con todo el mensaje

4. El mensaje codificado es enviado al receptor.

5. El receptor toma el mensaje que recibe, y aplica la misma operación utilizada

en el punto 3, pero esta vez con el mensaje codificado que recibío y la clave:

codificación	= 00101100000001110000110110010010010100000101100001010110
clave	= 01100001011000100110001111100001001100010011001000110011
mensaje	= 01001101011001010110111001110011011000010110101001100101

Puedes comprobar que de esta forma el receptor obtendrá el mensaje original.

Habrás notado que el mensaje codificado no tiene ningún significado para alguien que no conoce la clave, por lo que si alguien intercepta lo que se envió, no puede obtener mucha información sobre el mensaje original.

Una restricción de este protocolo es que el largo de un mensaje es a lo más el largo de la clave. Para encriptar un mensaje de largo menor al largo de la clave, simplemente se considera el sufijo de la clave con el mismo largo del mensaje y se hace la encriptación de la misma forma.

Enunciado

En este proyecto deberás realizar un programa que te permita participar en un *servicio seguro de mensajería* junto a tus compañeros. En particular, tu programa debe permitir:

- Enviar un mensaje encriptado a un compañero.
- Recibir y desencriptar los mensajes que te han enviado tus compañeros.
- Mostrar los mensajes en pantalla ordenados por fecha.
- Filtrar los mensajes que se muestran en pantalla. El filtro debe considerar también las direcciones de correo involucradas.

Todos los mensajes deberán tener a lo más 45 caracteres debido al tamaño de las claves. Se te entregará una clave personal con la que deberás encriptar los mensajes que envíes y desencriptar los mensajes que recibas. Para encontrar esta clave deberás ir al sitio web del curso y entrar a la página correspondiente a este proyecto.

Librerías

En esta sección se muestran las dos librerías con las que cuentas para llevar a cabo tu proyecto.

mensajeria

Esta librería servirá para interactuar con el servidor que manejará el servicio de mensajería. En particular, provee las siguientes funciones:

- `mensajeria.conectar(clave,mail)`: Esta función debe ser llamada antes que cualquier otra función de la librería `msg` para que la librería y el servidor sepan quién eres. Recibe como parámetros tu clave y tu correo (@puc.cl).

Retorna `True` si se puede conectar correctamente con el servidor, y `False` de lo contrario. Tu clave la encontrarás en el sitio web del curso, en la página de este proyecto.

- `mensajeria.mensajes_recibidos()`: Retorna una lista con todos los mensajes que te han enviado tus compañeros. Cada mensaje es a su vez una lista que contiene tres elementos. El primero es el correo de quien lo envía, el segundo es la fecha y hora en que fue enviado, y el tercero es el mensaje (que viene encriptado con tu clave).
- `mensajeria.mensajes_enviados()`: Retorna una lista con todos los mensajes que has enviado. Cada mensaje es a su vez una lista que contiene tres elementos. El primero es el correo del destinatario, el segundo es la fecha y hora en que fue enviado, y el tercero es el mensaje (que está encriptado con tu clave).
- `mensajeria.enviar_mensaje(mail_destinatario, mensaje_encriptado)`: Envía el `mensaje_encriptado` al estudiante cuya dirección de correo @puc.cl es `mail_destinatario`. Si el mensaje enviado tiene más de 45 caracteres u ocurre algún error, retornará `False`. Si el mensaje es enviado correctamente, retornará `True`.
- `mensajeria.mensajes_enviados_hoy()`: Retorna la cantidad de mensajes que has enviado en el último día.

Para evitar el spam, existe una cuota de mensajería. Cada usuario podrá enviar máximo 50 mensajes al día. Una vez que hayas enviado estos 50 mensajes, no podrás interactuar con el servidor hasta que acabe dicho día.

Funcionamiento interno

Habrás notado que en todo lo anterior hay algo extraño, y esto es que los mensajes que te envían tus compañeros son encriptados con sus claves, pero a ti te llegan encriptados con tu clave. ¿Cómo puede ser esto? Lo que ocurre es que el servidor al que se conecta la librería `mensajeria` conoce todas las claves. Luego, cuando envías un mensaje encriptado con tu clave, el servidor lo desencriptará con tu clave y lo encriptará con la clave del destinatario. Te darás cuenta de que esto nos permite a nosotros leer los mensajes originales, por lo que no es una buena idea enviar mensajes de mal gusto.

msgGUI

Esta librería te permitirá manejar el programa a través de una interfaz gráfica. Al iniciar tu programa se abrirá una ventana que tiene los elementos necesarios para redactar un mensaje (ver Figura 1).

Para que tu programa pueda ser manejado por una interfaz gráfica, la librería `msgGUI` ofrece las siguientes instrucciones:

- `pantalla_redactar()`: Muestra la patalla para redactar mensajes (ver Figura 1).

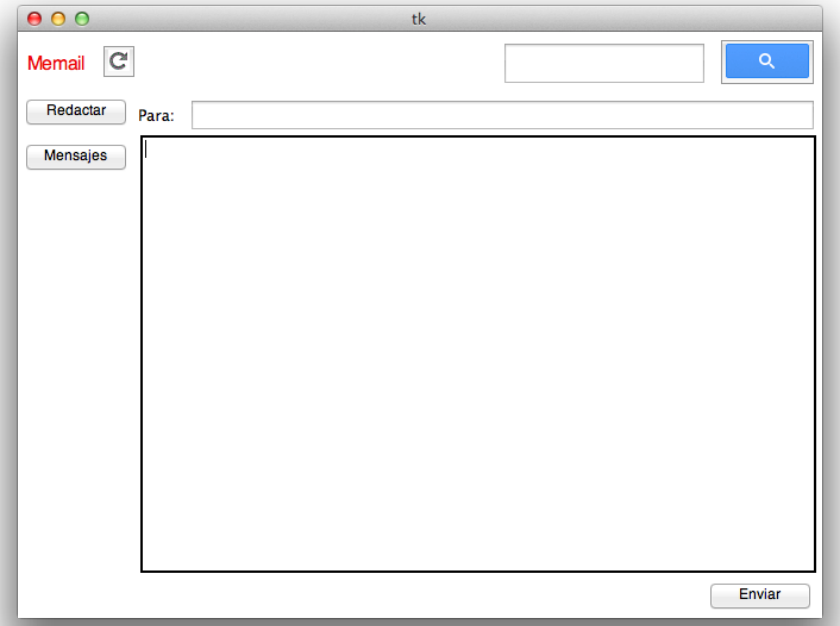

Figura 1: Ventana inicial de la librería `msgGUi`.

- `destinatario()`: Retorna el string que se encuentra en el cuadro de texto para ingresar el destinatario.
- `mensaje_redactado()`: Retorna el string que se encuentra en el cuadro de texto en el cual se escribe un mensaje (debajo del destinatario).
- `borrar_destinatario()`: Borra el string que se encuentra en el cuadro de texto para ingresar el destinatario.
- `borrar_mensaje_redactado()`: Borra el string que se encuentra en el cuadro de texto en el cual se escribe un mensaje.
- `pantalla_mensajes()`: Muestra la patalla ver mensajes (ver Figura 2).
- `poner_mensaje_al_final(msg)`: Pone el mensaje `msg` al final de todos los mensajes que se muestran.
- `poner_mensaje_al_principio(msg)`: Pone el mensaje `msg` al principio de todos los mensajes que se muestran.
- `borrar_mensajes()`: Borra todos los mensajes de la pantalla de mensajes.
- `preguntar(preg)`: muestra al usuario la pregunta `preg` y toma el valor True o False dependiendo de si el usuario responde si o no respectivamente.
- `alerta(msg)`: muestra una ventana con el mensaje `msg`, y espera a que el usuario presione Ok para seguir ejecutando el programa.

- `esperar_click()`: al llamar a esta instrucción el programa se detiene hasta que el usuario haga click en alguno de los botones de la interfaz. Luego, toma como valor el número del botón que se presionó. (Figura 3)
- `cambiar_titulo(s)`: cambia el título de la ventana por el string `s`. Por ejemplo, el título de la ventana en la Figura 2 es 'tk'.
- `salir()`: cierra el programa.

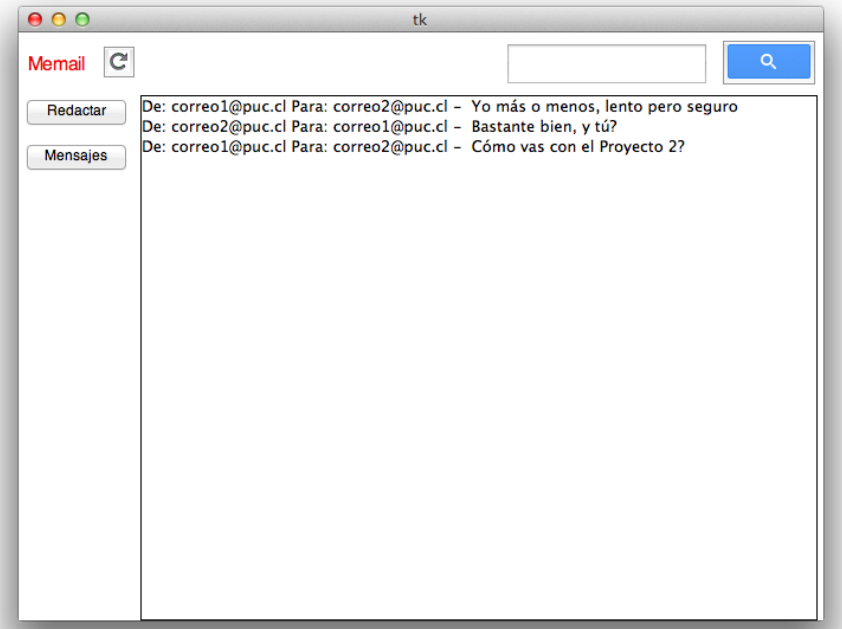

Figura 2: Ventana para mostrar mensajes.

Configuración

Para que tu proyecto funcione correctamente, debes tener en una misma carpeta tu proyecto junto con los archivos

- `mensajeria.py`
- `msgGUI.py`
- `search_icon.gif`
- `refresh_icon.gif`

que se encuentran junto a este enunciado en la página del curso. Además, el archivo de tu proyecto debe complir con el template que se muestra en el ejemplo de la próxima página, y que también podrás encontrar en la página del curso.

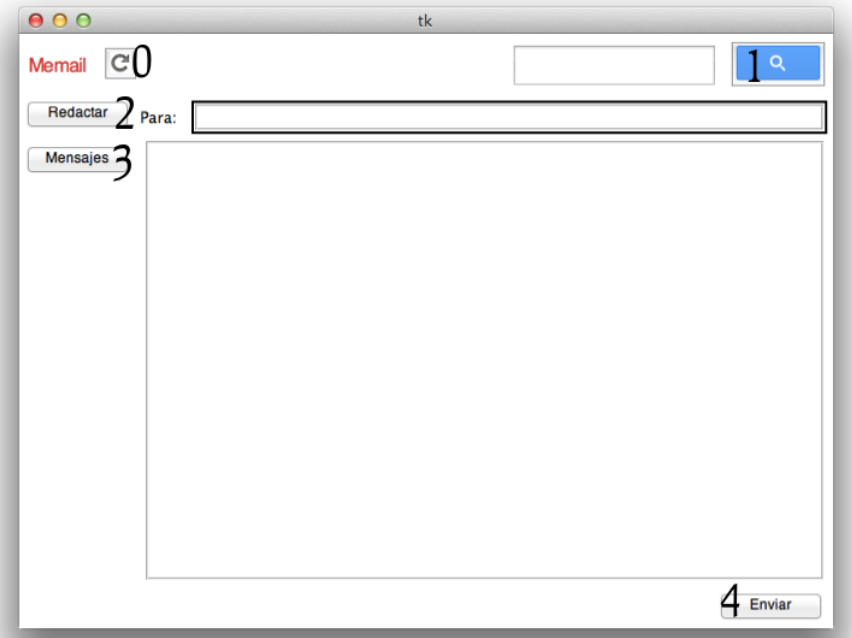

Figura 3: Códigos que retorna el método `esperar_click()`.

```
import msgGUI
import mensajeria

def tarea(gui):

    #INICIO TAREA


    gui.cambiar_titulo("Mensajería Segura")

    if mensajeria.conectar(mi_mail@puc.cl, mi_clave):
        gui.alerta("te has conectado correctamente al servidor!")
    else:
        gui.alerta("Imposible conectar con el servidor de mensajes")

    btn = gui.esperar_click()
    gui.alerta("has presionado el botón " + str(btn))
    gui.pantalla_mensajes()

    #FIN TAREA

app = msgGUI.Application(None)
app.cargar_programa(program)
app.iniciar()
```

Restricciones

En la corrección del proyecto se exigirá que existan las siguientes funciones:

- `string_a_binario(s)`: retorna la concatenación de los números binarios correspondientes a los códigos ASCII de los caracteres del string `s`. Por ejemplo, si recibe el string "st", debe retornar el string "0111001101110100", que es la concatenación de "01110011" y "01110100". Esto a su vez corresponde a la notación binaria de 8 dígitos de los números 115 y 116, codificaciones ASCII de las letras `s` y `t` respectivamente.
- `binario_a_string(s)`: es la función inversa de `string_a_binario(s)`. Por ejemplo, si recibe el string "0111001101110100", debe retornar "st".
- `OTP(msg,clave)`: recibe un mensaje `msg` en texto plano y lo encripta (o decripta, ya que son equivalentes) utilizando la clave `clave`, representada como un string de ceros y unos.

Además, no podrás utilizar el operado *bitwise xor*, ni la creación de listas explicada en la sección List Comprehensions de la documentación de Python 3.

Informe

Además de entregar el código de tu archivo, deberás entregar un informe llamado [numero_de_alumno]_informe_proyecto_1.pdf en el cual se explique, usando no más de una plana, qué fue lo más difícil a la hora de desarrollar el proyecto y cómo lo resolviste. También podrás agregar a este informe una nueva plana con una sección que explique en qué aspectos superaste lo que se pide en este enunciado.

Entrega

Antes del día 15 de octubre a las 23:59:00 debes entregar un archivo llamado [numero_de_alumno]_proyecto_2.zip en el sitio web del curso. Este archivo debe contener exclusivamente el archivo [numero_de_alumno]_proyecto_2.py y el informe en formato PDF. Las 10 primeras personas que, utilizando la librería `mensajeria`, envíen el mensaje "Mi tarea funciona excelente!" correctamente encriptado al correo "cuadrodehonor@puc.cl" formarán parte del cuadro de honor del sitio del curso.