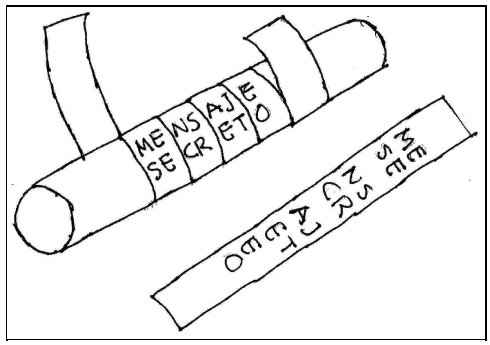
Instituto Tecnológico de Costa Rica
Ingeniería en Computación
IC-1801 Taller de Programación
Semestre I, 2016

Criptografía



La criptografía es la técnica, ciencia o arte de la escritura secreta. El principio básico de la criptografía es mantener la privacidad de la comunicación entre dos personas alterando el mensaje original de modo que sea incomprensible a toda persona distinta del destinatario.

Se puede decir que la criptografía es tan antigua como la civilización, cuestiones militares, religiosas o comerciales impulsaron desde tiempos remotos el uso de escrituras secretas; los antiguos egipcios usaron métodos criptográficos, mientras el pueblo utilizaba la lengua demótica, los sacerdotes usaban la escritura hierática (jeroglífica) incomprensible para el resto. Los antiguos babilonios también utilizaron métodos criptográficos en su escritura cuneiforme.

Tipos de criptografía:

1. *Criptografía Clásica*

La criptografía clásica viene desde la antigüedad hasta 1949. Su seguridad radica en el desconocimiento del algoritmo utilizado. El cifrado de textos es una actividad que ha sido ampliamente usada a lo largo de la historia humana, sobre todo en el campo militar y en aquellos otros en los que es necesario enviar mensajes con información confidencial y sensible a través de medios no seguros.

2. Criptografía Moderna

Los sistemas criptográficos clásicos presentaban una dificultad en cuanto a la relación complejidad-longitud de la llave / tiempo necesario para cifrar y descifrar

el mensaje. En la era moderna esta barrera clásica se rompió, principalmente por los siguientes factores:

- *Velocidad de cálculo*: con la aparición de los computadores se dispuso de una potencia de cálculo muy superior a la de los métodos clásicos.
- *Avance de las matemáticas*: que permitieron encontrar y definir con claridad sistemas criptográficos estables y seguros.
- *Necesidades de seguridad*: surgieron muchas actividades nuevas que precisaban la ocultación de datos, con lo que la Criptografía experimentó un fuerte avance.

A partir de estas bases surgieron nuevos y complejos sistemas criptográficos, que se clasificaron en los dos tipos o familias principales, los de llave simétrica y los de llave pública. Los modernos algoritmos de cifrado simétricos mezclan la trasposición y la permutación, mientras que los de llave pública se basan más en complejas operaciones matemáticas.

Cifrado

El cifrado es un procedimiento que utiliza un algoritmo para transformar un mensaje, sin atender a su estructura lingüística o significado, de tal forma que lo hace incomprensible o, al menos, difícil de comprender a toda persona que no tenga la clave o conozca el algoritmo de cifrado utilizado.

Descifrado

El descifrado es el proceso de convertir el texto cifrado en el texto en claro. Para realizar esta tarea se requiere conocer el algoritmo de cifrado y además la función inversa al cifrado.

Por lo tanto siendo $f( )$ la función de cifrado, debe existir una $f^{-1}( )$ como función de descifrado.

Algunas de las técnicas de criptografía clásica que serán implementados como parte de esta tarea programada corresponden a las categorías de:

- Cifrado por sustitución
- Cifrado por transposición
- Cifrado por código telefónico
- Cifrado por codificación binaria

Estos algoritmos se detallan a continuación:

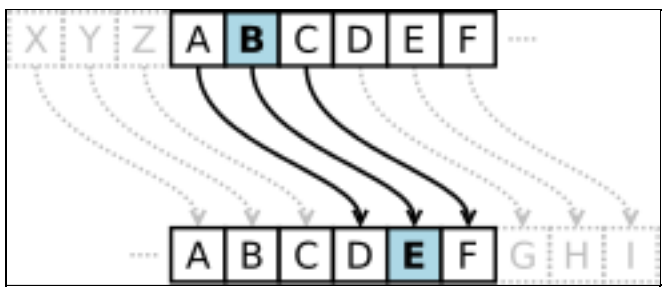
1. Cifrado por sustitución

En un cifrado por sustitución, las letras (o grupos de letras) son sistemáticamente reemplazadas en el mensaje por otras letras (o grupos de letras).

Una sustitución muy conocida en el cifrado es la del Cifrado César. Para cifrar un mensaje mediante el Cifrado César, cada letra del mensaje es reemplazada por la letra ubicada tres posiciones después en el abecedario. Por tanto, la A sería reemplazada por la D, la B por la E, la C por la F, etc. Por último la X, la Y y la Z serían reemplazadas por la A, la B y la C respectivamente.

1.1 Cifrado César

Establece las parejas de sustitución desplazando tres posiciones el orden del alfabeto del texto en claro. Cuando se acaban las letras por el final se empieza por el



principio. Por tanto en castellano la A será sustituida por la D, la B por la E,... y la Z por la C. Este tipo de cifrado se dice que es de alfabeto desplazado. En este algoritmo la clave está implícita en el mismo.

El alfabeto a usar sería:

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
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

No se contemplarán diferencias entre mayúsculas y minúsculas.

Ejemplo:

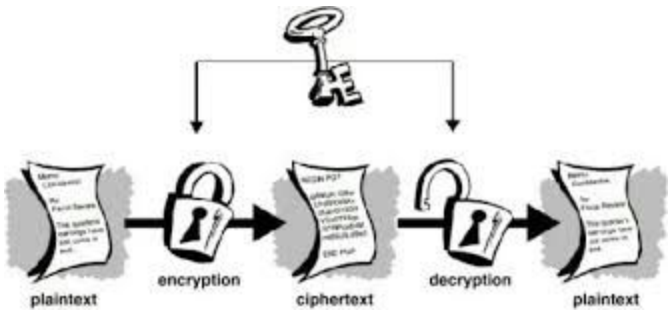
Codifica ®		**¬ decodifica**
Frases		**Frases**
t**a**re**a** programada criptografia de datos		WDSHD SURJUDPDGD FULSWRJUDILD GH GDWRV

1.2 Cifrado por llave

Una clave, palabra clave o clave criptográfica es una pieza de información que controla la operación de un algoritmo de criptografía. Habitualmente, esta información es una secuencia de números o letras mediante la cual, en criptografía, se especifica la transformación del texto plano en texto cifrado, o viceversa.

En sistemas informáticos, la clave sirve para verificar que alguien está autorizado para acceder a un servicio o un sistema.

Las claves también se utilizan en otros algoritmos criptográficos, como los sistemas de firma digital y las funciones de hash con clave (asimismo llamadas códigos de autentificación de mensajes).



En nuestro caso, se codificará cada palabra del texto en forma independiente. Se utilizará una palabra clave para realizar el proceso de cifrado, se utilizará la siguiente correspondencia de valores:

a	b	c	d	e	f	g	h	i	j	k	l	m
1	2	3	4	5	6	7	8	9	10	11	12	13
n	o	p	q	r	s	t	u	v	w	x	Y	z
14	15	16	17	18	19	20	21	22	23	24	25	26

Para cada palabra, debe sumar el valor de la primera letra de la palabra clave al valor de la primera letra de la palabra en cuestión. Sumar el valor de la segunda letra de la palabra clave al valor de la segunda letra de la palabra en cuestión. Así sucesivamente.

Si el tamaño de la palabra clave es menor que la palabra a codificar, debe utilizar nuevamente letras de la palabra clave hasta que ambas palabras sean del mismo tamaño, por el contrario, si el tamaño de la palabra clave es mayor que la palabra a codificar, se utilizarán en este caso, las letras que sean necesarias de la palabra clave.

Ejemplo:

Codifica ®			**¬ decodifica**	
Frases	**Clave**		**Frases**	**Clave**
tarea programada sobre codificacion	tango		nbflp jscngunokp xf wprpucdojxio	tango

t	a	n	g	o		t	a	n	g	o	t	a	n	g	o		t	a		t	a	n	g	o	t	a	n	g	o	t	a
t	a	r	e	a		p	r	o	g	r	a	m	a	d	a		d	e		c	o	d	I	f	i	c	a	c	i	o	n

N	b	f	l	p		j	s	c	n	g	u	n	o	k	P		x	f		w	p	r	p	u	c	d	o	j	x	i	o

La codificación de las letras va desde 0 a 25. En algunos casos la suma de dos letras puede resultar en un código inválido. Cuando eso suceda debe restar 26 al valor resultante. Por otro lado, cuando decodifica, debe hacer el proceso inverso, restar los valores de las letras según se muestra a continuación:

Codifica ®				**¬ decodifica**		
t	Valor: 20	Letra inicial de **tarea**		n	Valor: 14	Letra inicial de **nbflp**
t	Valor: 20	Letra inicial **tango**		t	Valor: 20	Letra inicial de **tango**
n	(20+20= 40) (40-26=14)	40 no es código válido 14 corresponde a la letra n		t	(14-20=-6) (-6+26=20)	-6 no es código válido 20 corresponde a la letra t de **t**area

Para todos los casos anteriores se trabajará con el siguiente alfabeto: (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) y el espacio en blanco.

No se contemplarán diferencias entre mayúsculas y minúsculas.

1.3 Sustitución Vigenére (Blaise de Vigenére, Siglo XVI)

La asignación de caracteres se realiza teniendo en cuenta la posición del carácter en el mensaje y el dígito que le corresponde según la clave.

Tabla a utilizar:

a	b	c	d	e	f	g	h	i	j	k	l	m
1	2	3	4	5	6	7	8	9	10	11	12	13
n	o	p	q	r	s	t	u	v	w	x	Y	z
14	15	16	17	18	19	20	21	22	23	24	25	26

Ejemplo:

Codifica ®			**¬ decodifica**	
Frases	**Cifra**		**Frases**	**Cifra**
t**a**re**a** programada criptografia de datos	23		v**d**th**c** ruqjtdodfd euksvriucikd fh fdvru	23

El mensaje cifrado se consigue adelantando 2 letras la primera que encontremos, 3 la segunda, 2 la tercera, 3 la cuarta y así sucesivamente para cada una de las palabras en forma independiente.

Si se desea decodificar, debe retroceder 2 letras la primera que encontremos, 3 la segunda, 2 la tercera, 3 la cuarta y así sucesivamente para cada una de las palabras en forma independiente.

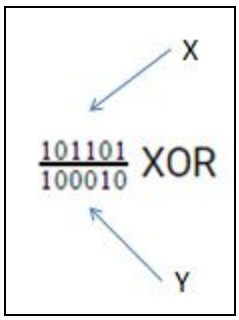
Como se ve, la letra "a" de la primera palabra aparece una vez como “d” y otra como “c”, entonces no hay una correspondencia uno a uno entre el alfabeto inicial y los símbolos del mensaje cifrado. Cuando inicia otra palabra se debe repetir el mismo proceso y finalizará una vez que no queden palabras por procesar.

Nota: La cifra para realizar el proceso de codificación/decodificación debe tener 2 dígitos. Algunos ejemplos válidos para la cifra: 36, 21, 48, etc.

1.4 Sustitución mediante XOR y llave

La idea general de este algoritmo es aplicar un XOR entre cada uno de los elementos del texto en claro con la llave indicada por el usuario.

Esta operación tiene el siguiente comportamiento para cada bit de salida:



- Es el mismo bit de X si en la misma posición de Y el valor es 0.
- Es el complemento del bit de X si el bit en la misma posición de Y es 1.

Ejemplo de la operación XOR

El número 45 se expresa en binario como 101101 y el número 34 se expresa en binario como 100010. Esto es verificable mediante la función en Python:

```
>>> bin(45)
'0b101101'
>>> bin(34)
'0b100010'
```

El resultado de la operacion anterior es 001111 y podemos conocer su representación decimal mediante otra función predefinida en Python:

```
>>> int('001111',2)
15
```

Ahora bien, solamente nos queda identificar con que elemento de la tabla ASCII guarda relación el número 15. Para este efecto, es posible utilizar la operación chr().

```
>>> print chr(15)
```

Codifica ®			**¬ decodifica**	
Frases	**Llave**		**Frases**	**Llave**
t**a**re**a** programada	secreto		\x07\x04\x11\x17\x04T\x1f\x01\n\x04\x00\x04\x19\x0e\x17\x04 (léase la frase anterior sin saltos de línea)	secreto

Para el caso de aplicar el XOR a la primera letra de la frase con la primera letra de la clave, el procedimiento es el siguiente:

t - Representación en tabla ASCII con el número 116

s - Representación en la tabla ASCII con el número 115

Ahora bien, aplicamos el XOR a estos valores:

```
>>> 116 ^ 115
7
```

El resultado, es decir el número decimal 7 tiene el siguiente valor asociado en la tabla ASCII:

```
>>> chr(7)
'\x07'
>>> print ('\x07')
•
```

Por lo tanto el alfabeto válido como entrada para **este** algoritmo corresponde a los 256 elementos de la tabla ASCII.

2. Cifrado por transposición

En un cifrado por transposición, las letras no se cambian por otras sino que se cambia el orden de estas. El orden es alterado de acuerdo con un esquema bien definido. Muchos cifrados por transposición se basan en un diseño geométrico.

2.1 Palabra inversa

El método de transposición consiste en una reordenación de los símbolos del mensaje original de modo que éste resulte ilegible. La reordenación se puede realizar desde un modo simple: escribiendo el mensaje letra a letra pero al revés, ya sea para codificación o decodificación. En este algoritmo la clave está implícita.

Ejemplo:

Codifica ®		**¬ decodifica**
Frases		**Frases**
esto es un secreto no lo puedo decir aserpros		otse se nu oterces on ol odeup riced sorpresa

2.2 Mensaje inverso

Ahora en cambio es la frase completamente invertida.
Nuevamente su clave es implícita.

Ejemplo:

Codifica ®		**¬ decodifica**
Frases		**Frases**

Hola mi nombre es Python		nohtyP se erbmon im aloH

3. Cifrado por código telefónico

A cada tecla del teléfono se le asignan letras en el siguiente orden 2-abc 3-def 4-ghi 5-jkl 6-mno 7-pqrs 8-tuv 9-wxyz. Cada letra se sustituye por el número al que está asignada + la posición que ocupa (que puede ser 1, 2, 3 ó 4); así la la letra e será 32, y la letra s será 74. Cada letra equivalente separada por un espacio en blanco y un * entre palabras

Para decodificar la información se debe obtener la letra correspondiente a cada número y dejar un espacio en blanco cuando ocurra un *

Ejemplo:

Codifica ®		**¬ decodifica**
Frase		**Frase**
tarea programada criptografia de datos zygalski Henryk		81 21 73 32 21 * 71 73 63 41 73 21 61 21 31 21 23 73 43 71 81 63 41 73 21 33 43 21 * 31 32 * 31 21 81 63 74 94 93 41 21 53 74 52 43 * 42 32 62 73 93 52

Nota: El alfabeto válido para este reto son solamente todas letras que se muestran en la figura del teclado telefónico y el simbolo de * como separador de cada palabra.

4. Cifrado por Codificación Binaria (Francis Bacon, Siglo XVI)

El código binario es el utilizado por los ordenadores del presente y a parte de su uso en informática, también podemos utilizar su mismo fin; codificar información; para transmitir mensajes. Para ello se propone un modelo de binario simple, tan solo con cinco bits, ya que sería inútil utilizar

combinaciones de 8 bits con solamente 26 letras del alfabeto.

Siguiendo este esquema construimos el alfabeto:

a	b	c	d	e	f	g	h	i	j	k	l	m
00000	00001	00010	00011	00100	00101	00110	00111	01000	01001	01010	01011	01100

n	o	p	q	r	s	t	u	v	w	x	y	z
01101	01110	01111	10000	10001	10010	10011	10100	10101	10110	10111	11000	11001

Cada letra se reemplaza por su equivalente binario, se separa de la otra con espacio. La separación entre palabras equivale a un *

Ejemplo:

Frase
tarea programada criptografia datos zygalski Henryk

Resultado:

10011 00000 10001 00100 00000 * 01111 10001 01110 00110 10001 00000 01100 00000 00011 00000
00010 10001 01000 01111 10011 01110 00110 10001 00000 00101 01000 00000 * 00011 00000 10011 01110 10010
11001 11000 00110 00000 01011 10010 01010 01000 * 00111 00100 01101 10001 11000 01010

¿ Por qué debo hacer a conciencia mi tarea programada?

- Practicar las habilidades de resolución de problemas
- Aumentar el conocimiento del estudiantes sobre el lenguaje de programación Python
- Practicar la experimentación y la resolución de problemas (divide y vencerás)
- Ejercitar la toma de decisiones
- Fomentar la investigación por parte del estudiante
    - Sobre los conceptos relacionados con temas de cifrado o criptografía de información
    - Implementación de estructuras de control básicas
    - Mecanismos para solicitar datos al usuario
    - Recursividad
    - Concatenación de cadenas de caracteres
    - Uso de las funciones matemáticas básicas de Python
    - Uso de listas en Python

Por hacer:

Implementar una solución computacional que inicialmente muestra en la consola un menú, que debe indicar al usuario los posibles algoritmos de criptografía clásica que implementa la solución.

Después que el usuario selecciona algún algoritmo**, se debe solicitar al usuario** los datos de entrada según sea necesario para cada mecanismo de codificación. La tarea debe permitir **codificar** o **decodificar** el texto según la explicación de cada uno de los mecanismos de codificación documentados en esta tarea. Debe tomar en cuenta que la entrada y salida de datos se realiza mediante la consola exclusivamente.

Es importante que realice la validación de los datos de entrada según las características requeridas por cada mecanismo de codificación. Debe proveer la robustez de su solución.

Puntos a ser evaluados:

1. Correctitud de la solución computacional - 65%

Algoritmo	Codifica	Decodifica
Cifrado César	5 puntos	5 puntos
Cifrado por llave	10 puntos	10 puntos
Sustitución Vigenére	5 puntos	5 puntos
Sustitución XOR y llave	10 puntos	10 puntos
Palabra inversa	5 puntos	5 puntos
Mensaje inverso	5 puntos	5 puntos
Cifrado telefonico	5 puntos	5 puntos
Cifrado binario	5 puntos	5 puntos

2. Robustez de la solución computacional (validaciones) - 10%

3. Evitar los síntomas de un diseño pobre “olores del software” - 5%
    a. Rigidez
    b. Fragilidad
    c. Inmovilidad
    d. Viscosidad
    e. Complejidad innecesaria
    f. Repetición innecesaria
    g. Opacidad

4. Entregar un documento con al menos los siguientes apartados: - 20%
    a. Manual de usuario - 25 puntos
    b. Pruebas de funcionalidad - 25 puntos

        Debe demostrar evidencia de todas las funcionalidades implementadas en la tarea programada. Para *esto* realice la ejecución de su tarea programada y tome **screenshots** dejando evidencia del resultado de la ejecución de cada transacción.
    c. Lecciones aprendidas - 15 puntos

        Debe hacer un listado de todas las lecciones aprendidas producto del desarrollo de la tarea programada. Las lecciones aprendidas pueden ser de carácter personal y/o técnico.
    d. Olores del software - 25 puntos

        Debe indicar cuál o cuáles estrategias se usaron para apartarse de cada uno de los olores del software vistos en clase. Las estrategias deben sustentarse con segmentos de código de la tarea programada (tome screenshots) o bien con la documentación de las decisiones que fueron tomadas durante el diseño y la implementación.
    e. Minutas y evidencias de asignación de responsabilidades - 10 puntos

        Debe indicar las fechas y tiempos de cada una de las reuniones que realiza el equipo asi como donde se realizan. También los temas tratados, las responsabilidades que son asignadas a los miembros del equipo y el seguimiento de los retos encontrados. Incluir un posible cronograma de cómo distribuir el trabajo a lo largo del tiempo asignado.

5. Debe entregar un CD/DVD, que contenga únicamente 2 carpetas llamadas: documentación y solución computacional, en la primera deberá incluir el documento de Word (no pdf) solicitado y en la segunda los archivos y/o carpetas necesarias para la implementación de esta tarea.

6. Entregue el CD/DVD en un sobre de manila sellado con sus calidades. La entrega es estrictamente personal en el momento que el profesor lo haya dispuesto.
    a. Las calidades deben incluir:
        i. Nombre del curso
        ii. Número de semestre y año lectivo
        iii. Nombre del Estudiante
        iv. Número de carnet
        v. Número de tarea programada
        vi. Fecha de entrega
        vii. Estatus de la entrega (definido por el responsable de la implementación de la tarea): [Deplorable|Regular|Buena|Muy Buena|Excelente|Superior]

7. La tarea será revisada en la versión de Python 3.5.1 únicamente.

Condiciones generales:

Esta tarea programada se rige por las siguientes condiciones:

Nota: El incumplimiento de alguna condición implicará una calificación de cero.

1. El desarrollo de la tarea es estrictamente en PAREJAS
2. La tarea NO debe implementarse con interfaz gráfica.
3. Debe cumplir con todo lo indicado en la sección “Puntos a ser evaluados”
4. Deberá entregarse en tiempo y forma según el plazo establecido por el profesor al momento la lectura de este documento.
5. El lenguaje de programación a utilizar es Python v3.5.1
6. Debe aplicar únicamente la programación recursiva para dar solución a esta tarea.
7. Se cuenta con 3 semanas a partir del día de entrega de la tarea.

Anexo 1: Manual de Usuario

1. Portada
2. Introducción
3. ¿Qué funcionalidades implementadas tiene el software? (estado general de la tarea)
4. Explicación paso a paso de cómo probar cada uno de los algoritmos implementados, esta explicación debe incluir el uso de casos de pruebas y fotos de pantalla.
5. Pendientes de implementar y su justificación.
6. Bibliografía y fuentes digitales utilizadas