Seguridad y Protección de Sistemas Informáticos

Práctica 5: Puzles Hash

Francisco Fernández Millán

Seguridad y Protección de Sistemas Informáticos

Práctica 5: Puzles Hash

1. Para la función H , realizad, en el lenguaje de programación que queráis, una función que tome como entrada un texto y un número de bits b . Creará un id que concatene una cadena aleatoria de n bits con el texto. Pegará a ese id cadenas aleatorias x de n bits hasta lograr que H(id||x) tenga sus primeros b bits a cero. La salida será el id , la cadena x que haya proporcionado el hash requerido, el valor del hash y el número de intentos llevados a cabo hasta encontrar el valor x apropiado.

Código de mi programa:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import uuid
import hashlib
import random
import string
import binascii


#Función que se encarga de cifrar mediante sha256 una id y un número aleatorio.
def hash_password(id, x):
	x = uuid.uuid4().hex

	return hashlib.sha256(x.encode() + id.encode()).hexdigest() + x

#Función que genera una cadena aleatoria con un tamaño dado.
def generar_cadena(size, chars=string.ascii_uppercase + string.digits):
	return ''.join(random.choice(chars) for _ in range(size))

#Función que pasa a binario una cadena dada.
def pasar_bin(h):
	scale = 16
	num_of_bits = 1
	h_binary = bin(int(h, scale))[2:].zfill(num_of_bits)

	return h_binary
```

```
#Función hash que dado un texto y un número de ceros introducido, junto
#con un cadena generada, obtenemos el número de iteraciones necesarias para
#alcanzar el número de ceros especificado.
def hash(texto, cadena, numCeros):
	id = cadena + texto #concatenación de la cadena generada + el texto
	cont = 0
	encontrado = False

	x = random.randrange(256) #generar valor aleatorio de x
	h = hash_password(id, x)
	h_binary = []
	h_binary = pasar_bin(h) #pasamos a binario el cifrado en hex
	lon = len(h_binary) #obtenemos la longitud para seleccionar posiciones
	print("\nValor de h: " + str(h_binary))
	n_ceros = str(numCeros)

	#Bucle con variable bool de parada cuando sea True
	#Según el número de ceros a buscar entra en un if u otro
	#Comprobamos las últimas posiciones en busca de los ceros
	while encontrado == False:
	if n_ceros == '1':
		print("Entra en el primer if")
		while h_binary[lon-1] != '0':
			cont += 1
			print("\n--------------------")
			print("Valor de x: " + str(x))
			print("\nValor de h: " + str(h_binary))
			x = random.randrange(256)
			h = hash_password(id, x)
			h_binary = pasar_bin(h)
			lon = len(h_binary)

		print("\n--------------------")
		print("Valor de x: " + str(x))
		print("\nValor de h: " + str(h_binary))
		encontrado = True

	elif n_ceros == '2':
		print("Entra en el segundo if")
		while h_binary[lon-1] != '0' or h_binary[lon-2] != '0':
			cont += 1
			print("\n--------------------")
			print("Valor de x: " + str(x))
			print("\nValor de h: " + str(h_binary))
			x = random.randrange(256)
			h = hash_password(id, x)
			h_binary = pasar_bin(h)
			lon = len(h_binary)

		print("\n--------------------")
		print("Valor de x: " + str(x))
		print("\nValor de h: " + str(h_binary))
		encontrado = True
```

```
        elif n_ceros == '3':
                print("Entra en el tercer if")
                while h_binary[lon-1] != '0' or h_binary[lon-2] != '0' or h_binary[lon-3] != '0':
                        cont += 1
                        print("\n--------------------")
                        print("Valor de x: " + str(x))
                        print("\nValor de h: " + str(h_binary))
                        x = random.randrange(256)
                        h = hash_password(id, x)
                        h_binary = pasar_bin(h)
                        lon = len(h_binary)

                print("\n--------------------")
                print("Valor de x: " + str(x))
                print("\nValor de h: " + str(h_binary))
                encontrado = True

        elif n_ceros == '4':
                print("Entra en el cuarto if")
                while h_binary[lon-1] != '0' or h_binary[lon-2] != '0' or h_binary[lon-3] != '0' or
h_binary[lon-4] != '0':
                        cont += 1
                        print("\n--------------------")
                        print("Valor de x: " + str(x))
                        print("\nValor de h: " + str(h_binary))
                        x = random.randrange(256)
                        h = hash_password(id, x)
                        h_binary = pasar_bin(h)
                        lon = len(h_binary)

                print("\n--------------------")
                print("Valor de x: " + str(x))
                print("\nValor de h: " + str(h_binary))
                encontrado = True

        elif n_ceros == '5':
                print("Entra en el quinto if")
                while h_binary[lon-1] != '0' or h_binary[lon-2] != '0' or h_binary[lon-3] != '0' or
h_binary[lon-4] != '0' or h_binary[lon-5] != '0':
                        cont += 1
                        print("\n--------------------")
                        print("Valor de x: " + str(x))
                        print("\nValor de h: " + str(h_binary))
                        x = random.randrange(256)
                        h = hash_password(id, x)
                        h_binary = pasar_bin(h)
                        lon = len(h_binary)
                        print("Longitud de la cadena: "+str(lon))

                print("\n--------------------")
                print("Valor de x: " + str(x))
                print("\nValor de h: " + str(h_binary))
                encontrado = True
        cont += 1

        return cont
```

```
text = input('Inserta el texto: ')
numCeros = input('Introduce el numero de ceros: ')
cadena = generar_cadena(256)
hash_out = hash(text, cadena, numCeros)

print("\nId: " + cadena + text)
print("\nCadena aleatoria: " + cadena)
print("\nNum. Iteraciones: " + str(hash_out))
```

Ejemplo de salida con número de ceros = 5.

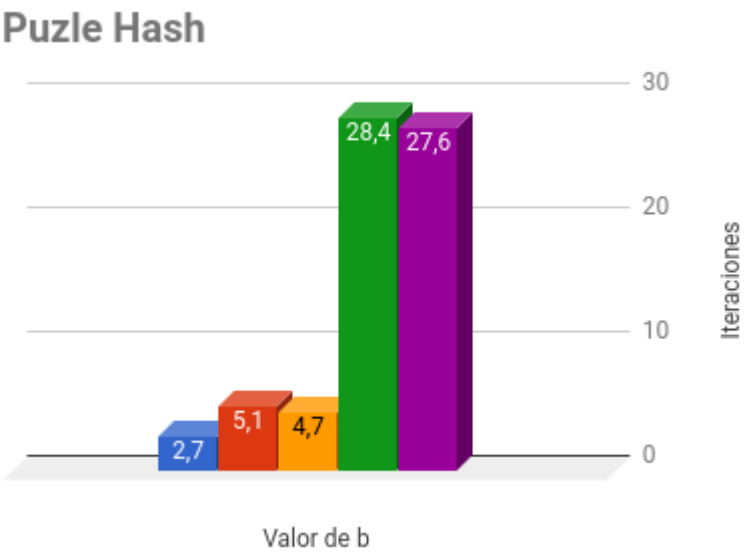
```
--------------------
Valor de x: 58

Valor de h: 1100100001000001111001011111111001000101110110111101011111100100001110000000110010
0001000000010001011000010000001001001111110001110010110001110010001110001000011000001100101011
1101110011000100101110110010111010101100110000100101101011001111110100111111110111001101010101
1001011111110000110111101010010101100110101001100001001001000001100110010011101010011010111011
1000100001001001010011100000

Id: C3J72GD4UAIBDDCJBFYI6Z4XZDLUAFB9XR523ME3FNLT6BG7RO2HR3UK9VXI5722YRAFZS1PKN4C2GD0G6GOZEK0
4JRBDSD5PM0R82ZFZNRC40U6HI8ET001LI902CIECYRN2AGNFATUUOVIGAKH7FZ7UM0VPLL8B72QX7JTFT956BV18QJ0
VUB9J2HS93GNWO9CEKRYPNZDVG0ITRBGR4V9351PE663TUEZD7WOY96YY6VTNCS0X79STSWAL788qwertyuiop

Cadena aleatoria: C3J72GD4UAIBDDCJBFYI6Z4XZDLUAFB9XR523ME3FNLT6BG7RO2HR3UK9VXI5722YRAFZS1PKN
4C2GD0G6GOZEK04JRBDSD5PM0R82ZFZNRC40U6HI8ET001LI902CIECYRN2AGNFATUUOVIGAKH7FZ7UM0VPLL8B72QX7
JTFT956BV18QJ0VUB9J2HS93GNWO9CEKRYPNZDVG0ITRBGR4V9351PE663TUEZD7WOY96YY6VTNCS0X79STSWAL788

Num. Iteraciones: 11
francisco@Fernandez-Ubuntu:~/Escritorio/SPSI/Prácticas/P5$
```

2. Calculad una tabla/gráfica que vaya calculando el número de intentos para cada valor de b. Con el objeto de que los resultados eviten ciertos sesgos, para cada tamaño b realizad el experimento 10 veces y calculad la media del número de intentos.

Tabla obtenida según el número de ceros y las iteraciones:

		Número de bits a 0			
b=1	**b=2**	**b=3**	**b=4**	**b=5**	**I**
2	3	7	69	27	**T**
8	5	2	6	17	**E**
2	2	5	29	1	**R**
1	1	1	20	121	**A**
2	2	8	52	8	**C**
3	4	5	9	2	**I**
1	10	6	41	29	**O**
1	2	6	8	29	**N**
3	8	2	13	19	**E**
4	14	5	37	23	**S**
2,7	**5,1**	**4,7**	**28,4**	**27,6**	**Media**

Gráfica obtenida:



3. Repetid la función anterior con el siguiente cambio:
Se toma un primer valor aleatorio x y se va incrementando de 1 en 1 hasta obtener el hash requerido.

Código de mi programa:

Para este apartado el único cambio que he realizado en mi código es, cambiar la generación del valor aleatorio de x de la línea “x = random.randrange(256)” por “x += 1” en cada uno de los bucles.

Ejemplo de salida con número de ceros = 5.

```
------------------
Valor de x: 32

Valor de h: 10101000101000100111111000100000110101100010100100011111011100101111101110100011100
11101010100111010100101010011100011000111011110100001000000101011100101000010101010101111000
11100010101011000000010110010110100010010111100010111110011100001110111001010100011101110000
11101110110110010100110010110000001000000100010000110111101111111101000010100000001001101110000
01011010011001111011111101101
Longitud de la cadena: 384

------------------
Valor de x: 33

Valor de h: 11100111101001111111001100010000100111011111001001010001100000101101011010101011111
11010111001101100011100010101101100011000001101000001000111010101001000111100000001110100111
11010111000010011000110010111110100100101101001001100100011111101110111111111111101101011110111
01110110010010111110010110000011010010110100011100000111101001100100001111101001110111100001
01001110001000101111111000000

Id: XV7QPKGCF3ZG3H67UYOVBEM8669DY7UKEI5QSM2ZLMLVKYH6TQ28OTUMRH4RYB2B58OOUDRQKZ4HVKZ8Y3FD6AJW
8CJL3L2DFZ80R5TQW6L2TDEW2RA29NBE2AEVAE6Z1GRI52RJ13UN2WMB4U0GX4MH1F96BV55F8E537P5LS32NZDT0G42
9821UFVXXUTVFE15FX4PZUCAJ984JKO02QH9OUQ1WTDEOHOB8E1TXP3R4Y3WYSDB40YXAO5UPVA0qwertyuiop

Cadena aleatoria: XV7QPKGCF3ZG3H67UYOVBEM8669DY7UKEI5QSM2ZLMLVKYH6TQ28OTUMRH4RYB2B58OOUDRQKZ
4HVKZ8Y3FD6AJW8CJL3L2DFZ80R5TQW6L2TDEW2RA29NBE2AEVAE6Z1GRI52RJ13UN2WMB4U0GX4MH1F96BV55F8E537
P5LS32NZDT0G429821UFVXXUTVFE15FX4PZUCAJ984JKO02QH9OUQ1WTDEOHOB8E1TXP3R4Y3WYSDB40YXAO5UPVA0

Num. Iteraciones: 2
francisco@Fernandez-Ubuntu:~/Escritorio/SPSI/Prácticas/P5$
```

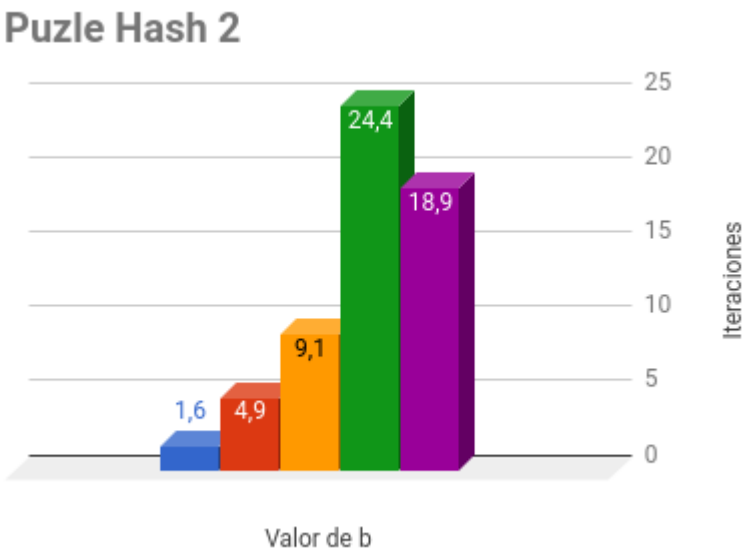
Como podemos observar, el valor de x es secuencial, a partir de un valor generado.

4. Calculad una nueva tabla/gráfica similar a la obtenida en el punto 2 pero con la función construida en 3.

Tabla obtenida según el número de ceros y las iteraciones:

		Número de bits a 0			
b=1	**b=2**	**b=3**	**b=4**	**b=5**	**I**
2	7	2	23	15	**T**
1	7	2	13	4	**E**
1	7	9	41	54	**R**
2	2	20	7	9	**A**
2	2	21	1	23	**C**
3	4	9	6	19	**I**
1	9	2	86	6	**O**
2	5	9	52	19	**N**
1	1	16	22	9	**E**
1	5	1	13	31	**S**
1,6	**4,9**	**9,1**	**24,4**	**18,9**	**Media**

Gráfica obtenida:



Si realizamos una comparativa con los resultados obtenidos en esta gráfica y en la anterior, podemos observar como en todos los valores de b, se obtiene una media menor excepto en el valor b=3 que obtenemos el doble.