ASSIGNMENT 1

The Avalanche Effect

SEPTEMBER 26, 2020

Sukhman Atwal

Contents

Introduction

The objective of this assignment was to use a cipher of our choice that would read hashes generated by the tool and perform a bit-by-bit comparison, producing count value(s).

For the purposes of this assignment, I used SHA256 (Secure Hash Algorithm 256-bit) which was designed by the NSA and first published in 2001.

I created a script using Python 3 to convert two user inputted strings to sha256 hash and to binary. The script would then compare the binary outputs and count the difference between the two. All of this data would then be saved in a csv file.

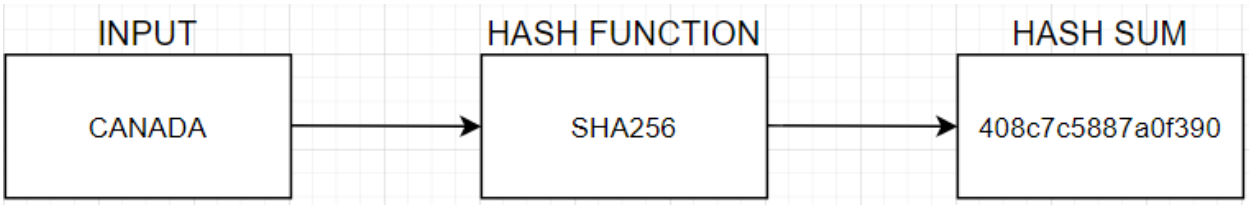
How to Use

You would run the script using the command line, which should then ask for user input twice.

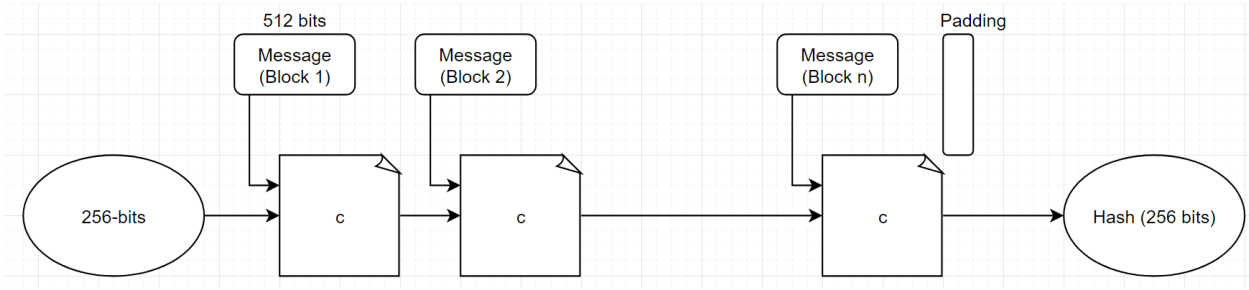
```
======= RESTART: C:\Users\Sukh\Desktop\Sukh_Atwal-A00907714\Assign1.py =======
Enter the first word: first
Enter the second word: second
```

Afterwards, all relevant information will be displayed and saved to a csv file located in the same directory as the script.

Design



A simplified version of how it works. You enter an input, in this case 'Canada.' It goes through using the hash function of your choice, SHA256 here and outputs the hash sum.

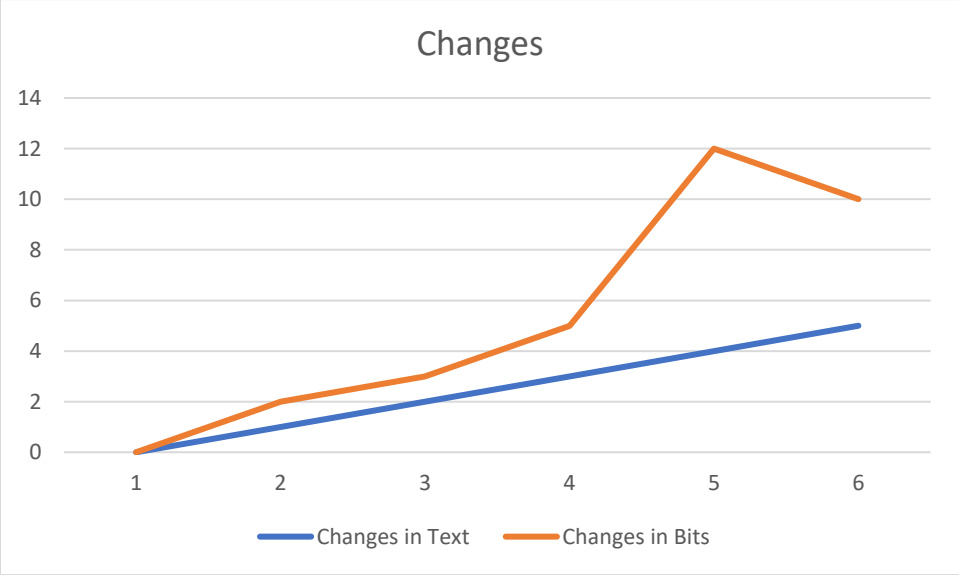


SHA256 turns inputs into 512 bits sized messages which are then fed to 'C' which is a compression function producing a 256-bit output which would be the input for the 'second' message. This continues until the end of the messages. (Kanumarlapudi, 2018)

Diagrams

I'll start by showing the changes in bits in the table below for the word 'test', changing the letters one by a time.

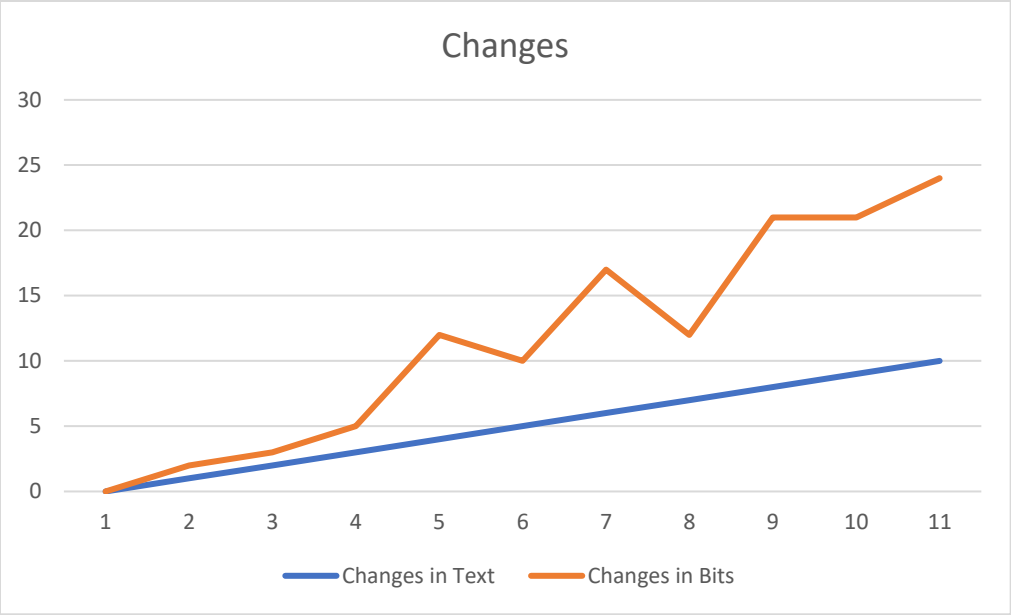
Word	Changes in Text	Binary	Changes in Bits
test	0	01110100011001010111001101110100	0
tess	1	01110100011001010111001101110011	3
tett	1	01110100011001010111010001110100	3
tets	2	01110100011001010111010001110011	6
trts	3	01110100011100100111010001110011	10
etts	4	01100101011101000111010001110011	10



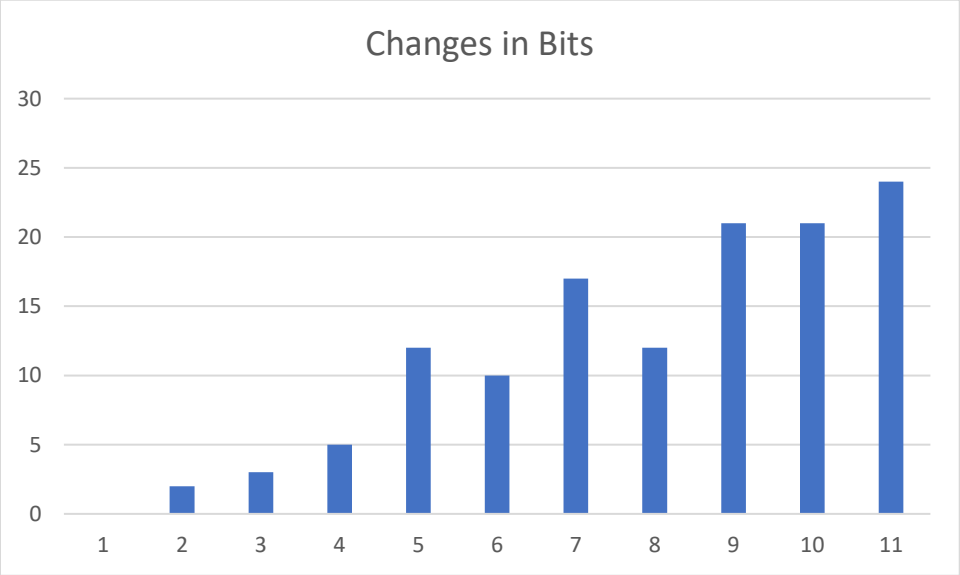
The graph above shows the changes in text and changes in bits over the 6 iterations.

I also did it with the word 'securities', changing letters one at a time.

Word	Changes in Text	Binary	Changes in Bits
securities	0	011100110110010101100011011101010111001001101001011101000110100101100101011100011	0
securitiea	1	011100110110010101100011011101010111001001101001011101000110100101100101011000011	2
securitiab	2	011100110110010101100011011101010111001001101001011101000110100101100001011000010	3
securitabc	3	011100110110010101100011011101010111001001101001011101000110000101100010011000011	5
securiabcd	4	011100110110010101100011011101010111001001101001011000010110001001100011011001100	12
securabcde	5	011100110110010101100011011101010111001001100001011000100110001101100100011001101	10
secuabcdfg	6	011100110110010101100011011101010110000101100010011000110110010001100110011001111	17
secabcdefg	7	011100110110010101100011011000010110001001100011011001000110010101100110011001111	12
seabcdefgh	8	011100110110010101100001011000100110001101100100011001010110011001100111011010000	21
sabcdefghi	9	011100110110000101100010011000110110010001100101011001100110011101101000011010001	21
abcdefghij	10	011000010110001001100011011001000110010101100110011001110110100001101001011010010	24



The graph below shows the changes in text and bits over the 11 iterations. We can see some dips taking place for the bits.



Here we specifically look at the bit's changes over time. We can see an avalanche effect taking place as the changes in bits are rising, falling, and then rising again.

Testing

Here we are checking to ensure we are getting the correct outputs from the user inputs.

Test #	Description	Tools Used	Expectations	Actuality	Pass/Fail
1	Users are able to enter anything into the user input section	IDLE	The user should be able to enter anything into the user input section and it should display back the hash and binary values.	The user is able to input any word or number with no errors and the hash and binary values are correctly displayed in the console.	Pass
2	The data is outputted to an csv file	IDLE, CSV	The data entered should be sent to an csv file located in the same directory as the python script.	The same data inputted and seen in the console is also displayed in the csv file located in the same directory.	Pass
3	Looking at the count value when entering inputs	IDLE	The count value should be the same if the inputs are the same.	The count values are the same if the inputs are the same, the counts would be 1 or higher only if the inputs are different.	Pass
4	The conversion from input to binary is correct	IDLE, Browser	The output in binary should be able to convert to the same word that we input at the start	After converting the binary output from the script, we got the same input as the one we initially entered.	Pass

Test 1

Here I ran the script using IDLE and used words, numbers, and random characters as my inputs.

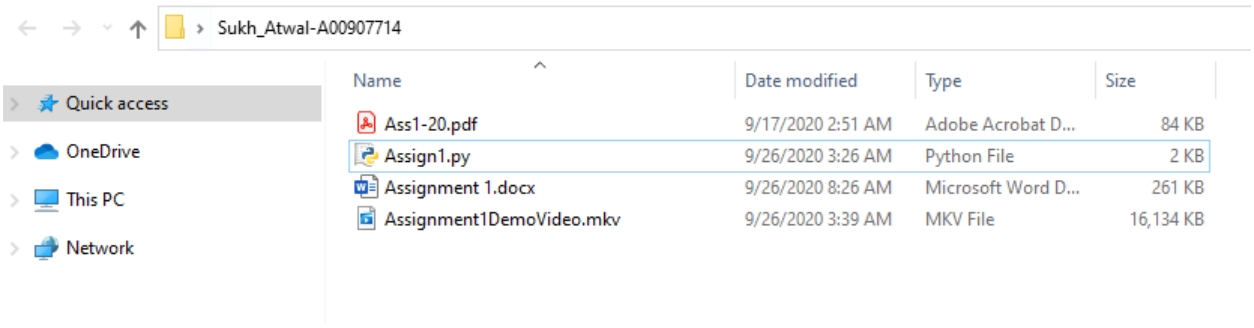
```
======= RESTART: C:\Users\Sukh\Desktop\Sukh_Atwal-A00907714\Assign1.py =======
Enter the first word: test test
Enter the second word: test test
03ffdf45276dd38ffac79b0e9c6c14d89d9113ad783d5922580f4c66a3305591
03ffdf45276dd38ffac79b0e9c6c14d89d9113ad783d5922580f4c66a3305591
011101000110010101110011011101000010000001110100011001010111001101110100
011101000110010101110011011101000010000001110100011001010111001101110100
0
>>>
======= RESTART: C:\Users\Sukh\Desktop\Sukh_Atwal-A00907714\Assign1.py =======
Enter the first word: test
Enter the second word: test
9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08
9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08
01110100011001010111001101110100
01110100011001010111001101110100
0
>>>
======= RESTART: C:\Users\Sukh\Desktop\Sukh_Atwal-A00907714\Assign1.py =======
Enter the first word: 2342342
Enter the second word: 234234
26d10a4296aabe4d38f53c4df663033eaaf3738772e95a7dd01dcef0a21e4426
49cbe57e9503aa0c6d0cf1f88ec2b8e1dfb870c346fb9c82bc0489915108d75a
00110010001100110011010000110010001100110011010000110010
001100100011001100110100001100100011001100110100
15
>>>
```

```
======= RESTART: C:\Users\Sukh\Desktop\Sukh_Atwal-A00907714\Assign1.py =======
Enter the first word: ttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttt
tttttttttttttttttttttttttttttttttajkofg;aksdhng;daksgnb;sdgkhnsphgisegp;wjng;lsekgnblsegknslkjgnslkegjnbfsdgjdsb
ngkjsbgksbgskdjgbskjgbsdkgj
Enter the second word: skldjghsdoilgfhsloghsdloighbslghnbhsrlgyhslgnblsehnglsbgnlksegblksebglstgbselgbselfhelig
jhsoilghbsdlkgjnsd;gksdjnlgnhsdlgnks
cc1b976bfebc5c893369522667d7f01ed9e37c837a382fb83f6667546416b3fc
32bfa4d15ee7d59dfe13fc49a6398adafd65705ac2bd6f5c04336b158ce054ce
0111010001110100011101000111010001110100011101000111010001110100011101000111010001110100011101000111010001110100011101000111010
0011101000111010001110100011101000111010001110100011101000111010001110100011101000111010001110100011101000111010001110100011101
0001110100011101000111010001110100011101000111010001110100011101000111010001110100011101000111010001110100011101000111010001110
1000111010001110100011101000111010001110100011101000111010001110100011101000111010001110100011101000111010001110100011101000111
0100011101000111010001110100011101000111010001110100011101000111010001110100011101000111010001110100011101000111010001110100011
1010001110100011101000111010001110100011101000111010001110100011101000111010001110100011101000111010001110100011101000111010001
1101000111010001110100011101000111010001110100011101000111010001110100011101000111010001110100011101000111010001110100011101000
1110100011101000111010001110100011101000111010001110100011101000111010001110100011101000111010001110100011101000111010001110100
0111010001110100011101000111010001110100011101000111010001110100011101000111010001110100011000010110101001101011011011 1
1011001100110011100111011011000010110101101110011011001000110100001101110011001110011101101100100011000010110 10
1101110011011001110110111001100010001110110111001101100100011001110110101101101000011011100111001101110000011 01
0000110011101101001011100110110010101100111011100000011101101110111011010100110111001100111001110110110110001 11
0011011001010110101101100111011011100110001001101100011100110110010101100111011010110110111001110011011011 00011
010110110101001100111011011100111001101101100011010110110010101100111011010100110111001100010011001100111001101
1001000110011101101010011001000111001101100010011011100110011101101011011010100111001101100010011001110110 10110
11100110110001001100111011100110110101101100100011010100110011101100010011100110110101101101010011001110110 0010
01110011011001000110101101100111011010 10
0111001101101011011011000110010001101010011001110110100001110011011001000110111101101001011011000110011101100 11
001101000011100110110110001101111011001110110100001110011011001000110110001101111011010010110011101101000011000
1001110011011011000110011101101000011011100110001001101000011100110111001001101100011001110111100101101000 01110
0110110110001100111011011100110001001101100011100110110010101101000011011100110011101101100011100110110001001 10
011101101110011011000110101101110011011001010110011101100010011011000110101101110011011001010110001001100111011
01100011100110111010001100111011000100111001101100101011011000110011101100010011100110110010101101100011001 1001
1010000110010101101100011010010110011101101010011010000111001101101111011010010110110001100111011010000110001 00
1110011011001000110110001101011011001110110101001101110011100110110010000111011011001110110101101110011011001 00
0110101001101110011011000110011101101110011010000111001101100100011011000110011101101110011010110111 0011
693
>>>
======= RESTART: C:\Users\Sukh\Desktop\Sukh_Atwal-A00907714\Assign1.py =======
Enter the first word: @@@@@@$$****
Enter the second word: @$($(()(!@)&
a2790a852edd08c8507d8efdc61a3a0679ae58fa2e7d9d026718d91eca8d7521
3bc833906817a89f8fce71e27b1832e2373aa986c6e238b23622a0f8816fb12a
01000000010000000100000001000000010000000100000000100100001001000010101000101010001010100010 1010
0100000000100100001010000001001000010100000101000000101001001010000001000010100000000010100100100110
31
```

You can see that the hash and binary values were outputted correctly for each attempt.

Test 2

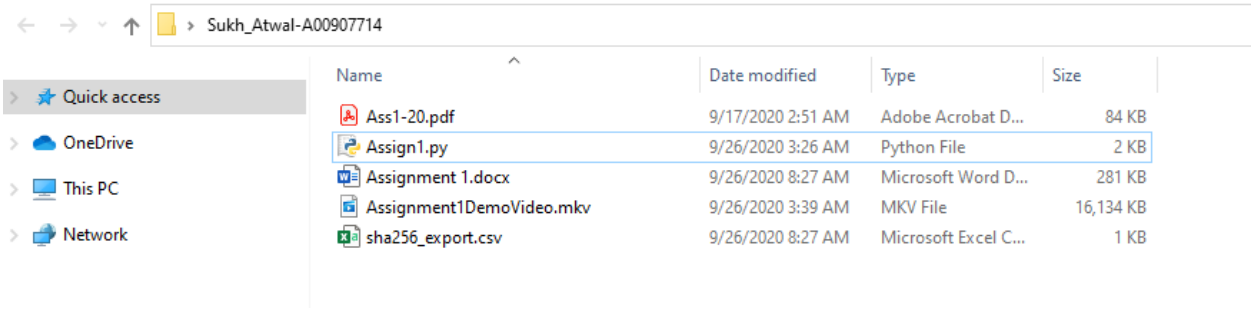
Here you can see that I have the python script in the following folder.



Now I'll run the script and enter the following inputs.

```
======= RESTART: C:\Users\Sukh\Desktop\Sukh_Atwal-A00907714\Assign1.py =======
Enter the first word: cryptography
Enter the second word: crytographe
e06554818e902b4ba339f066967c0000da3fcda4fd7eb4ef89c124fa78bda419
326b0301c1d86232eeadeb9a0d6b57d6686c7d369a0708572641a0744bd4a442
01100011011100100111100101110000011101000110111101100111011100100110000101110000011010000111100
01100011011100100111100101110100011011110110011101110010011000010111000001101000011001010
14
>>>
```

As you can see, I get the hash and binary values in the console.



There is also a sha256_export.csv file now in the directory.

A	B	C	D
	User Input	sha256 Output	Binary Format
0	cryptography	e06554818e902b4ba339f066967c0000da3fcda4fd7eb4ef89c124fa78bda419	1.10001E+94
1	crytographe	326b0301c1d86232eeadeb9a0d6b57d6686c7d369a0708572641a0744bd4a442	1.10001E+86

Here I open it and get both inputs, the hash value as well as the binary value as well which are the same as the one displayed in the console.

Test 3

For the purposes of this test, I commented out the hash and binary values so that the only output in the console would be the difference in bit counts.

```
======= RESTART: C:\Users\Sukh\Desktop\Sukh_Atwal-A00907714\Assign1.py =======
Enter the first word: test
Enter the second word: test
0
>>>
======= RESTART: C:\Users\Sukh\Desktop\Sukh_Atwal-A00907714\Assign1.py =======
Enter the first word: british columbia institute of technology
Enter the second word: british columbia institute of technology
0
>>>
```

Above we can see that when the inputs are the same, there is a count of 0, meaning there is no difference in bits between the two.

```
======= RESTART: C:\Users\Sukh\Desktop\Sukh_Atwal-A00907714\Assign1.py =======
Enter the first word: test
Enter the second word: tess
3
>>>
======= RESTART: C:\Users\Sukh\Desktop\Sukh_Atwal-A00907714\Assign1.py =======
Enter the first word: test
Enter the second word: tets
6
>>>
======= RESTART: C:\Users\Sukh\Desktop\Sukh_Atwal-A00907714\Assign1.py =======
Enter the first word: test
Enter the second word: etst
4
>>>
```

When we have different inputs, you can see the count value has gone up to 3, 6 and 4 respectively.

```
======= RESTART: C:\Users\Sukh\Desktop\Sukh_Atwal-A00907714\Assign1.py =======
Enter the first word: british columbia institute of technology
Enter the second word: british columbia institute oo technology
2
>>>
======= RESTART: C:\Users\Sukh\Desktop\Sukh_Atwal-A00907714\Assign1.py =======
Enter the first word: british columbia institute of technology
Enter the second word: british columbia insitute of technology
60
>>>
```

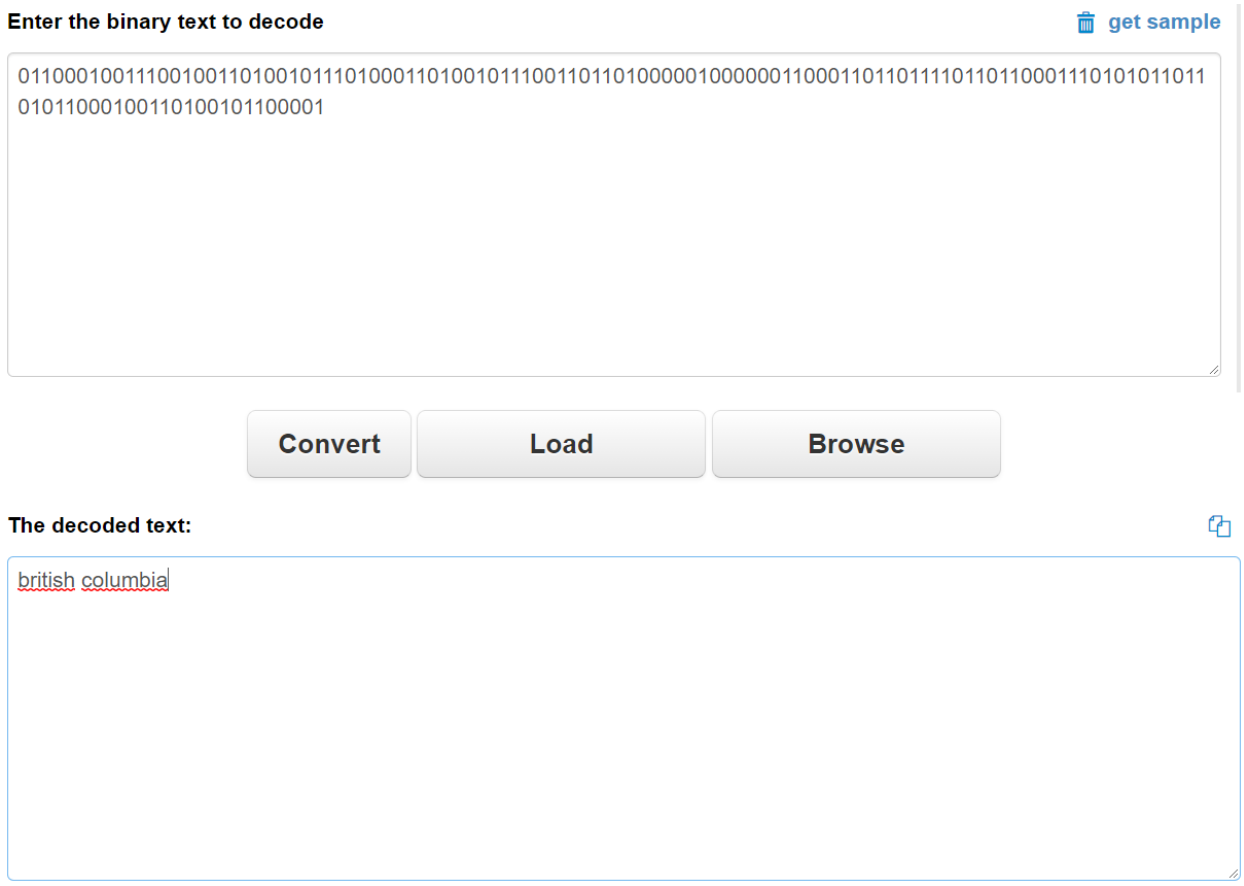
Using the second input example, we can see again the count value of bits changed is 2 and 60 for the following inputs.

Test 4

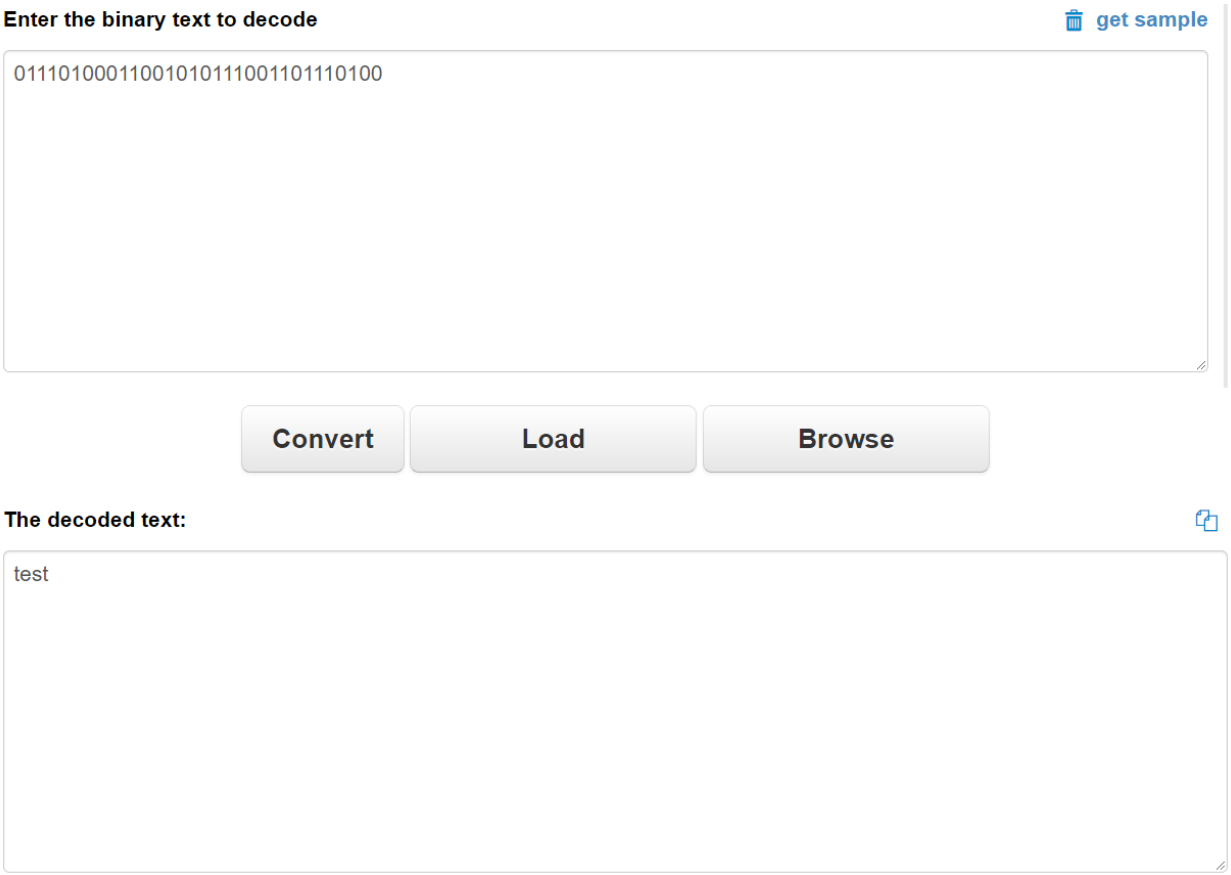
The two words I am using to test if the input to binary is working correctly are british columbia and test.

```
Enter the first word: british columbia
Enter the second word: test
3f5703ce38317bd7b4a59a58186b499b05abf9fc16fe52e9cbef0ce4a04d0d18
9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08
01100010011100100110100101110100011010010111001101101000001000000110001101101111011011000111010101101101100010011010010110000101110100011001010111001101110100
```

The website I am using is codebeautify for this scenario found at https://codebeautify.org/binary-to-text.



Starting off with british columbia which had a binary value of 01100010011100100110100101110100011010010111001101101000001000000110001101101111011011000111010101101101100010011010010110000101110100011001010111001101110100 in the script output. We can see it correctly converted it to the input ‘british columbia’.



Next, I looked at test which had a binary value of 01110100011001010111001101110100 in the script output. As we can see again, it correctly converted it to the input ‘test’.

Bibliography

Kanumarlapudi, P. (2018, February 20). *BlockChain for layman — Part 2*. Retrieved from medium: https://medium.com/@pkmar437/blockchain-for-layman-part-2-a8984fda0acc