Linnaeus University

1DV700 - Computer Security
Assignment 1

Student: <Melat> <Getachew>
Personal number: 001214-7880
Student ID: mh225ic@student.lnu.se

Setup Premises

I used windows 10 operating system, python 3.8(visual studio code), hex editor, website that converts hex codes to binary, website that converts binary codes to ascii characters, and I used different sources from the internet.

Task 1

Cryptography is the practice and study of techniques for secure communication in the presence of third parties called adversaries [1]. It is the science of protecting information by transforming it into a secure format.

a)

The difference between symmetric encryption and asymmetric encryption is that symmetric encryption is a type of encryption where only one key (a secret key) is used both to encrypt and decrypt electronic information whereas asymmetric encryption is a type of encryption that uses public key for encryption and a secret(private) key for decryption. For standard encrypt/decrypt functions, symmetric algorithms generally perform much faster than their asymmetrical counterparts.

The difference between encryption algorithms and hashing algorithms is that encryption algorithm is a two way function; what is encrypted can be decrypted with the proper key whereas hashing algorithm is a one way function that scrambles a plain text to produce a unique message digest and with a properly designed algorithm, there is no way to reverse the hashing process to reveal the original password [2].

The difference between compression function and hash function is that compression function transforms a large, fixed length input into a shorter fixed length output whereas a hash function can process an arbitrary length message into a fixed length output. Hash function is one-way compression function which is easy to compute but hard to invert [3].

b)

Definition

Steganography is the practice of concealing a file, message, image, or video within another file, message, image, or video [4].

A digital watermark is a kind of marker covertly embedded in a noise-tolerant signal such as audio, video, or image data. It is typically used to identify ownership of the copyright of such signal [5].

Encryption is the process of encoding information. It is a process which converts the original representation of the information, known as plain text, into an alternative form known as cipher text [6].

Difference

Steganography and encryption are both used to ensure data confidentiality and digital watermarking is used to identify ownership and copyright. However, the main difference steganography and encryption between them is that with encryption anybody can see that both parties are communicating in secret. Steganography hides the existence of a secret message and in the best case nobody can see that both parties are communicating in secret [7].

Purposes

The purpose of steganography is to hide the existence of a message from a third party.

The purpose of digital watermarking is to make it more difficult for the original image to be copied or used without permission.

The purpose of encryption is to protect the confidentiality of a digital data and to conceal the content of the message by translating it into a code.

When are they used?

Steganography is used when someone wants to carry out hidden exchanges.

Digital watermarking is used when someone wants to verify the authenticity or integrity of the carrier signal or to show the identity of the owners.

Encryption is used when someone wants to make sensitive data more secure and less likely to be intercepted by those unauthorized to view it.

Task 2

a)

From the given website:
I used the 'hide image' tool then selected Wikipedia tree for the cover image and Wikipedia cat for the secret image.

- When the hidden bit was from 1 to 3, the secret image was undetectable.
- When the hidden bit was 4 and 5, it was suspicious that there was something behind the cover image.
- When the hidden bit was 6, we could see both the cover image and secret image in a faded version.
- When the hidden bit was 7, we could see what the secret message is, and we could also see some part of the cover image.
- The limitation was that we could not go further than 7 (hidden bits). If we would have, the image would be clear as the original (secret image).

I used the 'unhide image' tool and selected Walrus from the example.

- When the hidden bit was 1, the image was dark black and white.
- When the hidden bit was 2 and 3, the image was dark black and white but there were different colors in the upper part of the image.
- When the hidden bit was 4, the image was lighter than when the hidden bit was 3.
- When the hidden bit was 5, the image was lighter and colorful than when the hidden bit was 4.
- When the hidden bit was 6, the middle part of the image was colored. We can guess by seeing the image, but it is difficult to know what exactly the image is.
- When the hidden bit was 7, it was the same as when the bit was 6 but clearer and more visible.
- The limitation was that we could not go further than 7 (hidden bits). If we would have, the image would be clear and visible as the original.

The images were hidden using LSB method. Each pixel has three values (RGB) in LSB, each RGB value is 8-bit (it means we can store 8 binary values) and the rightmost bits are less significant. So, if we change the rightmost bits it will have a small visual impact on the final image. This is the steganography key to hide an image inside another. The LSB method works by changing the least significant bits from an image and including the most significant bits from the other image. The disadvantage of Least Significant Bit is that it is vulnerable to stegganalysis and is not secure at all.

b)

Techniques of hiding information inside image

The image steganography is the process in which we hide the data within an image so that there will not be any perceived visible change in the original image. The conventional image steganography algorithm is LSB embedding algorithm.

Masking and filtering techniques, usually restricted to 24 bits and grey scale images, hide information by marking an image, in a manner similar to paper watermarks [8]. Paper watermarks are designs or patterns put into paper during its production, by making thinner (line watermarks) or thicker (shadow watermarks) the layer of pulp when it is still wet.

Transform techniques embed the message by modulating coefficients in a transform domain, such as the Discrete Cosine Transform (DCT) used in JPEG compression, Discrete Fourier Transform, or Wavelet Transform. These methods hide messages in significant areas of the cover-image, which make them more robust to attack [8].

Hiding an image within a sound file

An image or a text can be converted into a sound file, which is then analysed with a spectrogram to reveal the image. Various artists have used this method to conceal hidden pictures in their songs [9].
Echo hiding, a form of data hiding, is a method for embedding information into an audio signal. It seeks to do so in a robust fashion, while not perceivably degrading the host signal (cover audio) [10].

Steganography in streaming media

Since the era of evolving network applications, steganography research has shifted from image steganography to steganography in streaming media such as Voice Over Internet Protocol (VoIP) [11]. Streaming media can be used as a carrier of hidden information where hiding in new network-based services is supported.

Printed

Digital steganography output may be in the form of printed documents. A message, the plain text, may be first encrypted by traditional means, producing a cipher text. Then, an innocuous cover text is modified in some way so as to contain the ciphertext, resulting in the stegotext. For example, the letter size, spacing, typeface, or other characteristics of a cover text can be manipulated to carry the hidden message. Only a recipient who knows the technique used can recover the message and then decrypt it [12].

Text steganography

Content steganography is a process which hides a secret data behind other content.

c) Before I started doing this task, I studied how LSB method works and how a BMP file is organized. First, I downloaded the secret.bmp from mymoodle. Second, I used a hex editor to get the hex code of the secret bitmap. Then, I started changing the hex codes randomly by guessing where the text is. I change the hex codes into binary and used least significant bit (LSB) to convert it to ascii. Then, I found out that there was c when I converted 8 hex codes and used their LSB to find out the ascii character, at the beginning of the hex on the fourth line after 00. Then, I continued doing the same thing and found out that text stops at line 12. The hidden text is found on the footer of the bitmap file.

The text I found: congratulations!

The main point is to convert the hex codes to binary and use LSB method to obtain their least significant bit and using their least significant bit to obtain the ascii character. 8 bit = 1 ascii character

Task 3

a) in vino veritas

b)

In brute-force attacks, we attempt every conceivable key to check whether it can unscramble the ciphertext. On the off chance that the key is right, the unscrambling brings about coherent English. But by analyzing the ciphertext first, we can diminish the quantity of potential keys to attempt and perhaps locate a full or halfway key.

For instance, if we had the plaintext MISSISSIPPI SPILL, the comparing ciphertext may be RJBBJBBJXXJ BXJHH. The quantity of letters in the primary expression of the plaintext and the first cipher word are equivalent. The equivalent is valid for the second plaintext word and the second cipher word. The plaintext and ciphertext share similar pattern of letters and spaces. Likewise notice that letters that repeat in the plaintext repeat similar number of times and in similar spots as the ciphertext.

We could hence expect that a cipher word compares to a word in the English word dictionary and that their word patterns would match. At that point, on the off chance that we can discover which word in the dictionary the cipher word decodes to, we can sort out the unscrambling of each cipher letter in that word. Also, on the off chance that we sort out enough cipher letter decodings utilizing this procedure, we might have the option to unscramble the whole message.

c) Yes.

QMJ BPZ B XPJZ RZWJPAXQ LAD

1) B is one word so it could be either I or a.
2) By seeing BPZ I guessed that B is a and BPZ is are.
3) By seeing the second and third word, I guessed that QMJ would be the pronoun you.
4) Then, I substituted the alphabets which I guessed.
5) After substituting, XPJZ is _rue then I guessed that X would be t and RZWJPAXQ is _e_ur_ty then I guessed that the word would be security.
6) Then, I knew that A was i so I tried each and every remaining alphabet to know what LAD is.
7) Then, I found out that the word is wiz.

The decrypted message is: You are a true security wiz.

Task 4

Main program

First, the program asks the user to choose a method (Substitution or Transposition). Next, the program asks the user what to perform (Encryption and Decryption). Then, asks the user to enter the path which he/she wants to encrypt or decrypt and key to encrypt or decrypt the file. Finally, it stores the encrypted or decrypted text in another file.

Code:

```
# Program
Method = int(input("1 for Substitution or 2 for Transposition: ", ))
Perform = int(input("1 to Encrypt or 2 to Decrypt: ", ))

# if method = 1 substitution and method = 2 transposition
# if perform = 1 encrypt and perform = 2 decrypt

if Method == 1 and Perform == 1:
    path = input("Path: ",)
    key = int(input("Key: ",))
    Encrypted_Text = encrypt_text(read_file(path),key)
    # Create a file to store the encrypted text(substitution)
    with open('outfile.csv', 'w') as file_handler:
        file_handler.write("{}\n".format(Encrypted_Text))

elif Method == 1 and Perform == 2:
    path = input("Path: ",)
    key = int(input("Key: ",))
    Decrypted_text = decrypt_text(read_file(path),key)
    # Create a file to store the decrypted text(substitution)
    with open('outfile.csv', 'w') as file_handler:
        file_handler.write("{}\n".format(Decrypted_text))


elif Method == 2 and Perform == 1:
    path = input("Path: ",)
    key = int(input("Key: ",))
    Encrypted_Tex = encrypt_text2(read_file(path),key)
    Encrypted_Text=Encrypted_Tex[1:]
    outFile = open("myOutFile.txt", "w")
    outFile.writelines("{}\n".format(Encrypted_Text))
    outFile.close()

elif Method ==  2 and Perform == 2:
    path = input("Path: ",)
    key = int(input("Key: ",))
    fd=open("myOutFile.txt","r")
    d=fd.read()
    fd.close()
    m=d.split("\n")
```

```
    s="\n".join(m[:-1])
    fd=open("myOutFile.txt","w+")
    for i in range(len(s)):
        fd.write(s[i])
    fd.close()
    Decrypted_text = decrypt_text2(read_file(path),key)
    # Create a file to store the decrypted text(Transposition)
    outFile = open("myOutFile.txt", "w")
    outFile.writelines("{}\n".format(Decrypted_text))
    outFile.close()
```

First function
Reading the file from the specified path.
Code:

```
# Read the file
def read_file(path):

    with open(path, "r") as file:
            full_text= " "
            for line in file:
                full_text += line
            return full_text
```

First and second function
Encryption and decryption (Substitution)
How I did substitution cipher.

- Create a list of all ascii characters from a to z (both lowercase and uppercase).
- Create a dictionary to store the substitution for all the characters.
- Then substitute each character according to the rule and it depends whether we want to encrypt or decrypt the text.
- Prints the new encrypted or decrypted text.

I did the substitution in two ways one with string.ascii_letters and string.printable.

I wrote comments on the code so that it is easier to understand.
Code:

```
#Substitution cipher

def encrypt_text(text,key):
    # List of all characters which are considered to be printable
    letters= string.ascii_letters

    #An empty dictionary to store the substitution for the letters
    #Based on the key
    sub = {}
    for i in range(len(letters)):
        sub[letters[i]] = letters[(i+key)%len(letters)]
```

```
    encrypted_txt=[]
    # Generating the cipher text
    for letter in text:
        if letter in letters:
            l = sub[letter]
            encrypted_txt.append(l)
        else:
            l = letter
            encrypted_txt.append(l)

    return ("".join(encrypted_txt))


def decrypt_text(text,key):
    # List of all characters which are considered to be printable
    letters= string.ascii_letters

    #An empty dictionary to store the substitution for the letters
    #Based on the key
    sub = {}
    for i in range(len(letters)):
        sub[letters[i]] = letters[(i-key)%(len(letters))]

    # Iterate to get the original text
    decrypt_txt = []
    for letter in text:
        if letter in letters:
            l = sub[letter]
            decrypt_txt.append(l)
        else:
            l = letter
            decrypt_txt.append(l)

    return ("".join(decrypt_txt))
```

Third function Encryption (Transposition)

Steps to encrypt using transposition.

1. Identifying the length of the message and the key.
2. The total number of rows equals to the key. (Number of boxes equals to row/key)
3. Begin filling in the crates from left to right, entering one character for each container.
4. At the point when you run out of boxes yet have more characters, add another column of boxes.
5. At the point when you arrive at the last character, conceal in the unused boxes in the last column.
6. Beginning from the upper left and going down every segment, work out the characters. At the point when you get to the lower part of a segment, move to the following segment to one side. Skirt any concealed boxes. This will be the ciphertext.

I wrote comments on the code so that it is easier to understand.

Code:

```
# Transposition
```

```
# Encryption
def encrypt_text2(text,key):
    # Generating the encrypted text
    encrypted_text = [""] * key
    # Iterating through each column in the encrypted text
    for coloum in range(key):
        c_index = coloum
        # Loops until the current index passes the text's length
        while c_index < len(text):
            # Puts the character in the current index
            encrypted_text[coloum] += text[c_index]
            # Moves the current index
            c_index += key
    # Converts the list(text) to a single string and it will return
    return "".join(encrypted_text)
```

Fourth function Decryption (Transposition)

Steps to decrypt using transposition.

1. Ascertain the quantity of sections you need by partitioning the length of the message by the key and afterward gathering together.
2. Draw boxes in rows and columns. Utilize the quantity of segments you determined in step 1. The quantity of lines is equivalent to the key.
3. Figure the quantity of boxes to conceal in by taking the complete number of boxes (the quantity of lines increased by the quantity of sections) and taking away the length of the ciphertext message.
4. Shade in the quantity of boxes you determined in sync 3 at the lower part of the furthest right section.
5. Fill in the characters of the ciphertext beginning at the top column and going from left to right. Skirt any of the concealed boxes.
6. Get the plaintext by perusing the furthest left segment through and through and proceeding to do likewise in every section.

I wrote comments on the code so that it is easier to understand.

Code:

```

# Decryption
def decrypt_text2(text,key):
    # Obtaining the number of columns
    total_col = int(math.ceil(len(text) /float(key)))
    # The number of rows is equal to the key
    total_row = key

    # Generating the plain text
    decrypt_text = [''] * total_col
    # col and row variables indicate where in the grid the character will enter
    col = 0
    row = 0
    for char in text:
        decrypt_text[col] += char
        col += 1
        # If no more columns or last shaded box go back to the first col
        if (col == total_col) or (col == total_col -
1 and row >= total_row - ((total_col*total_row) - len(text))):
```

```
            col = 0
            row += 1
    # Converts the list(text) to a single string and it will return
    return ''.join(decrypt_text)
```

Task 5

I have written a text and encrypted it with substitution.

For substitution:

- The key is 5.
- The file name is Melat_Getachew_Substitution.txt

Task 6

1. Cipher text from Hilda_Melander_Substitution:

- By seeing the cipher text (encrypted text), I guessed that he/she used substitution method.
- I guessed that he/she used substitution method because the punctuation marks and referencing numbers are in the right place.
- Referencing numbers are directly followed by a punctuation mark.
- The punctuation marks and referencing numbers would have been mixed if he/she used transposition method to encrypt the text.
- I tried to decrypt the text with key=2 and it was a failure.
- I tried to decrypt the text with key=3 and it was right. The deciphered text was readable.

- **KEY=3**

- Every ascii characters were shifted by 3.
- The plain text is:

```
**********************************************************************************
*                                                                                *
*                          Secret message - Top Secret                           *
*                                                                                *
*                 May only be read by security passed personnel                  *
*                                                                                *
*                                                                                *
************************************************** *********************************
```

English is a West Germanic language that was first spoken in early medieval England and is now the third most widespread native language in the world, after Standard Chinese and Spanish, as well as the most widely spoken Germanic language. Perhaps the most striking characteristic of present-day English is that it is spoken by many more non-native speakers than native.[4] This use of English as a lingua franca, a coincidental result of English-speaking colonial spread, sets English apart from any other Western European language. With the proliferation of international trade, global travel, and the Internet, its adoption in this role is occurring at an explosive rate: until the 1960s, native speakers were in the majority, but since then the atio has shifted quickly. Inn 2003, "the ratio of native to non-native [was] around 1:3."[5] Today, the ratio may well be 1:6; for every native speaker are six non-native speakers of reasonable competence.[6] English developed from West Germanic dialects from the 5th century AD and was named after the people of the Angles, one of the Germanic tribes that migrated to England at that time. It is closely related to the other West Germanic languages of Frisian, Low German/Low Saxon, German, Dutch, and Afrikaans. The English vocabulary has been significantly influenced by French (a Romance language), Norse (a North Germanic language), and by Latin. English has developed over the course of more than 1,400 Years. The earliest forms of English, a set of Anglo-Frisian dialects brought to Great Britain by Anglo-Saxon settlers in the 5th century, are called Old English. Middle English began in the late 11th century with the Norman conquest of England and was a period in which the language was influenced by French.[7]ly Modern English began in the late 15th century with the introduction of the printing press to London and the King James Bible, and the start of the Great Vowel Shift.[8] Through the worldwide influence of the British Empire, modern English spread around the world from the 17th to mid-20th centuries. Through all types of printed and electronic media, as well as the emergence of the

United States as a global superpower, English has become the leading language of international discourse and the lingua franca in many regions and in professional contexts such as science, navigation and law.[9] English is the most widely learned second language and is either the official language or one of the official languages in almost 60 sovereign states. There are more people who have learned it as a second language than there are native speakers. English is the most commonly spoken language in the United Kingdom, the United States, Canada, Australia, Ireland and New Zealand, and it is widely spoken in some areas of the Caribbean, Africa and South Asia.[10] It is a co-official language of the United Nations, of the European Union and of many other world and regional international organisations. It is the most widely spoken Germanic language, accounting for at least 70% of speakers of this Indo-European branch. English has a vast vocabulary and counting exactly how many words it has is impossible.[11][12] Hilda Melander

2. Cipher text from Michael Racette Olsen - substitution:

- By seeing the cipher text (encrypted text), I guessed that he/she used substitution method.
- I guessed that he/she used substitution method because the punctuation marks and referencing numbers are in the right place.
- Referencing numbers are directly followed by a punctuation mark.
- The punctuation marks and referencing numbers would have been mixed if he/she used transposition method to encrypt the text.
- I tried to decrypt the text with key=2,3,4,5,6,7 and it was a failure.
- I tried to decrypt the text with key=8 and it was right. The deciphered text was readable.

- **KEY=8**

- Every ascii characters are shifted by 8.
- The plain text is:

```
**********************************************************************************
*                                                                                *
*                                                                                *
*                              secret message - top secrett                      *
*                                                                                *
*                          May only be read by security passed personnel         *
*                                                                                *
**********************************************************************************
```

Gaius Julius Caesar[b] 13 July 100 BC[1] 15 March 44 BC),[2] known as Julius Caesar, was a Roman politician, general, and notable author of Latin prose. He played a critical role in the events that led To The demise of The Roman Republic and The rise of The Roman Empire. In 60 BC, Caesar, Crassus, and Pompey formed a political alliance that dominated Roman politics for several years. their attempts to amass power as populares were opposed by The Optimates within The Roman senate, among them Cato the younger with The frequent support of Cicero. Caesar's Victories in The Gallic wars, completed by 51 BC, extended Rome's territory to The English Channel and The Rhine. Caesar became the first Roman general to cross both when he built a bridge across the Rhine and conducted the first invasion of Britain. These achievements granted him unmatched military power and threatened to eclipse the standing of pompey, who had realigned himself with the senate after the death of Crassus in 53 BC. with The Gallic wars concluded, the senate ordered Caesar to step down from his military command and return to Rome. Caesar refused the order, and instead marked his defiance in 49 BC by crossing the rubicon with the 13th Legion, leaving his province and illegally entering Roman Italy Under arms.[3] Civil war resuited, and Caesar's victory in the war put him in an unrivalled position of power and influence. After assuming control of government, Caesar began a programme of Social and

governmental reforms, including the creation of The Julian calendar. He centralised the bureaucracy of The republic and was eventually proclaimed "dictator in perpetuity", giving him additional authority. But the underlying political conflicts had not been resolved, and on the Ides of March (15 March) 44 BC, Caesar was assassinated by a group of rebellious senators led by Marcus Junius Brutus. A new series of civil wars broke out, and the constitutional government of the republic was never fully restored. Caesar's adopted heir octavian, later known as Augustus, rose to sole power after defeating his opponents in the civil war. Octavian set about solidifying his power, and the era of The Roman Empire began. Much of Caesar's life is known from his own accounts of his military campaigns, and from other contemporary sources, mainly the letters and speeches of Cicero and the historical writings of sallust. The later biographies of Caesar by suetonius and piutarch are also major sources. Caesar is considered by many historians to be one of the greatest military commanders in history.[4]

SOURCE: wikipedia.

Michael RaceTTe OlSén

****

3. Cipher text from Fredric_Eriksson_Sepulveda:

- By seeing the cipher text (encrypted text), I guessed that he/she used transposition method.
- The reason I said he used transposition because I saw punctuation marks in place where they shouldn't be, and this '*' signs were scattered, and they were not in the proper place.

Task 7

a) A hash function is any function that can be used to map data of arbitrary size to fixed-size values I did task 7a and 7b together and I didn't have much time to explain it on the report, but I wrote comments on code so that it is understandable.

1. The first function is a function which converts each character to their ascii value by using "ord".
2. The second function is a function that returns the hash value for a given text.
   First, I created an empty list to store the hash value for a given text. Second, I add every "ascii value % 67" of each character and store the sum in one variable. Then, I divided the sum of the whole text by 10 to get a unique hash value for the text.
3. The third function is a function that generates random strings.
   I imported a random module that deals with random generation.

```
return ''.join(random.choices(string.ascii_letters + string.digits, k=50))
```

Return a k(50) sized list of elements which are mixed(string.ascii_letters and string.digits)

4. The fourth function is a function that asks the user the number of times he/she wants to generate random strings.
   It loops over n number of random strings and store the string and their hash values in another file.

```
import string

import random

hash_value = []
# This function returns the ascii value for each character
def get_ascii_value(character):
    return ord(character)

# Returns the hash value for a given text
def text_hash(text):
    hash_lst = []    # An empty array
    hash_value_for_text = 0

    for char in text:

        # Returns ascii value for each character
        ascii_value = get_ascii_value(char)
        hash_value_for_text += ascii_value % 67  # 67 = A

    # if the hash vlue is greater than 255 it will divide it by 10
    while hash_value_for_text > 255:
        hash_value_for_text = int(hash_value_for_text / 10)

    # Prints the hash value for the given text
    print(" Hash for text: ", hash_value_for_text)
    # hash_lst = the hash value and the text itself
    hash_lst = [int(hash_value_for_text), text]
```

```
    return hash_lst

# Generating random strings
def random_string():

    # Contains every ascii letters and digits
    # k=50 is the length of the random strings is 50
    return ''.join(random.choices(string.ascii_letters + string.digits, k=50))

# Asking the user the number times to test(to test the random strings)
def generate_hashings(num_of_strings):

    for i in range(num_of_strings):
        # The random strings generated in (def random_string()) function
        string = random_string()

        # To get the hash values of the random strings
        hash_values = text_hash(string)

        # Appending the generated hash values to the hash_value=[]
        hash_value.append(hash_values)

    # A file to store the text and their hash values
    with open('outfile.csv', 'a+') as file_handler:
        file_handler.write("{}\n".format(hash_value))


# Main program
# For task 7a
text = input("\n Text to hash: ", )
hash_lst = text_hash(str(text))
print("Hash value and the hashed text:", hash_lst)
# For task 7b
num_hashes = int(input("How many hashes should be generated?", ))
print("Generating {0} hashes:".format(num_hashes))
generate_hashings(num_of_strings=num_hashes)
```

b) The matplotlib module in my laptop is not working and I have tried to fix the problem for weeks, but I could not fix it. So, I just did it manually.

A good hash function should map the expected inputs as evenly as possible over its output range. That is, every hash value in the output range should be generated with roughly the same probability. The reason for this last requirement is that the cost of hashing-based methods goes up sharply as the number of collisions—pairs of inputs that are mapped to the same hash value—increases. If some hash values are more likely to occur than others, a larger fraction of the lookup operations will have to search through a larger set of colliding table entries. I tried entering the same kind of text several times and it returned the same output. I also tried to enter strings that are very similar but differ by one bit and it sometimes returned the same output which it shouldn't have.

When I run 1000 tests with random input strings (both in terms of size and content).

- It returned the same output which is right.

- Example: Text to hash: melat
  Hash for text: 196

  Text to hash: melat
  Hash for text: 196

  Text to hash: melat
  Hash for text: 196

When I run 1000 tests with input strings that are very similar (only differ in one bit)

- It returned the same output sometimes when it should have been different.

When I generate 10 random strings with the size of 50, it gave me same outputs for two different strings.

```
[[174, 'NFaPLjRsW6i6MS4Bpv9iGmRHgAP4tgn9hTx0H0dLBjqoYkjkdf']
[159, 'yrvsNGBXCUNd5xOr3DhtQBzzyHVRmtQmgpgqgLhaDecFOSi4WV']
[164, 'eQ0rUh0HUlrpgYSzFG6WSWn09Tce93EZ1AH7EWCAHw5En1tSyu']
[172, 'BwA2Ot4Yvojrie9RDqHfdIgP7doBSxjShJkuvBNUDdd2KGBMDs']
[157, 'QYZsFcomWIpmk5TLyD6eNkUIxhUR9XDXBeD5lbaZVotG8ZbqA9']
[171, 'mbDJoWokesaq6spLXP6kYVgdgJ7hz9Ou700OCaNErlOicw70cB']
[171, 'kDNmOsYtcR4zlf6lvm4Gql2udD3RQPfHmZgeWGVr17AWpfvNzt']
[164, 'RNdMxFbI67UIvTwgJELxehTN81kxqsTfb6XjAuAzk0IJdhbzYX']
[165, 'ClS3ihzv79NLQdwEy2HpI9Heg7zS3rzHHLg6wAdSaWL4D1FxJB']
[161, 'laNoKztOyLEF8T3o8rvYZdgqWDYv2sQMomTnE733iA1kFSNEak']]
```

c) A secure hash function has the following properties:

- The hash value is fully determined by the data being hashed.
- The hash function uses all the input data.
- The hash function "uniformly" distributes the data across the entire set of possible hash values.
- The hash function generates quite different hash values for similar strings.

A normal hash function usually guarantees a weaker extent of the ‘properties’ for a secure hash function. A secure hash function has properties that makes it impossible to deliberately find such collisions. A weak function does not. The easiest way to prove that my hash function is not secure is that it has collisions. A hash collision occurs when you can find two different documents (inputs) for which the hash algorithm will produce the same hash (output). Collision is considered insecure because there is a vulnerability that two different strings/messages can make the same hash.

References

[1] https://en.wikipedia.org/wiki/Cryptography

[2] https://gcn.com/articles/2013/12/02/hashing-vs-encryption.aspx

[3] https://en.wikipedia.org/wiki/One-way_compression_function

[4] https://en.wikipedia.org/wiki/Steganography

[5] https://en.wikipedia.org/wiki/Digital_watermarking

[6] https://en.wikipedia.org/wiki/Encryption

[7] https://www.cs.bham.ac.uk/~mdr/teaching/modules03/security/students/SS5/

[8] https://www.researchgate.net/publication/4008787

[9] https://en.wikipedia.org/wiki/Steganography

[10] https://arxiv.org/ftp/arxiv/papers/0802/0802.3746.pdf

[11] https://en.wikipedia.org/wiki/Steganography

[12] https://en.wikipedia.org/wiki/Steganography