En bild som visar Teckensnitt, text, vit, Grafik

Automatiskt genererad beskrivning

Linnaeus University

1DV700 - Computer Security

Assignment 1

Student: <Kinan> <Almasri>

Personal number: 030918-0479

Student ID: ka223mf@student.lnu.se

En bild som visar skiss, växt, rita, botanik

Automatiskt genererad beskrivning

Setup Premises

Explain your setup such as, OS, web browser, tools being used, development environment, and whatever else is necessary…

Task 1

a) The distinction between Symmetric and Asymmetric encryption lies in their key structures. Symmetric encryption utilizes a single key, often referred to as a secret key, for both the encryption and decryption of data. On the other hand, Asymmetric encryption employs two distinct keys: a public key for encrypting data and a private key for decrypting it. This summary is based on the information provided in your slides.

In the process of encryption, plaintext is converted into ciphertext using specific encryption algorithms and keys. This ciphertext is capable of being transformed back into its original, readable format with the aid of a decryption key, which may be identical to the encryption key. unlike hash algorithms are designed to take input data of any size and process it into a unique, fixed-length output composed of a string of characters and numbers. Notably, the output produced by hash algorithms is not reversible, meaning it cannot be decrypted [1].

Compression is a method used to reduce the size of the files while hash function can be used to hashing the data to value and it can be used as an encryption method.

b) Steganography is like hiding a secret note inside another normal message. Cryptography is different - it scrambles a message so only people with a special key can read it. With steganography, the secret is just tucked away inside something else. But watermarking it's like putting a special mark on something to show it's real and not copied. This mark can be hidden or easy to see [2].

Task 2

A: Decrypting "HKPUFCMHY BHDDXZH"

Using the provided substitution cipher, the decryption of "HKPUFCMHY BHDDXZH" is "SUBSTITUTION CIPHER". Each letter in the ciphertext is replaced with its corresponding letter in the plaintext alphabet as per the given key.

B: Attempting to Decrypt "NWSRC XQS JXB CWRGABY WKH VWUNWBA"

Approach: Without the exact key, a correct decryption is speculative. However the logical analysis like identifying common English words and patterns, should be applied. For example, "XQS" and "JXB" could be common three-letter words like "and", "the", "for", etc. However, without more context or a longer text, this remains a hypothesis.

C) It's hard to break codes made with simple substitution ciphers if you don't have the key, especially if the message is short. But if the message is longer, you might figure it out by looking at which letters show up a lot or by spotting patterns. To make this kind of cipher stronger, you could mix in some extra letters that don't mean anything or use different symbols for the same letter. But even then, these ciphers aren't as strong as the more complicated ones we have today.

Task 3

For this task I have made 5 methods.

1.substitution()

Here, trhe function is for both encrypting and decrypting the substitution cipher. Where it takes each letter of a text add or remove the key(depends on decrypt or encrypt) value to the letter’s ord() and then moduls with 256 to make sure that we get a character.   
  
2. Transposition\_encrypt()

3. Transposition\_decrypt()

Here in both functions rearranging the characters in the text based on the provided key. In encryption, the text is read row-wise and written column-wise. In decryption, this process is reversed.

4. File\_process()

This function where it handle a chosen file by the user and has the chipher’s method, key and cipher’s type together with the chosen file. And write a new decrypted/encrypted file.

5. Main() is to collect the user’s choices and call the file\_process()

Task4

I used substitution cipher and encrypt a text file using key = 3.

Task 5

I used <https://www.dcode.fr/monoalphabetic-substitution>

To decrypt oa222kq.txt and as226pr.txt

oa222kq.txt text has this cipher text

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 | Q | X | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Z | R | S | T | U | V | W |

By looking at the cipher text from the letter D in the decrypted text starts with alphabets that means the key is ‘3’, since the letter D comes after three letters.   
  
the same way in the other file I got this cipher text

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 | J | Q | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Z | R | S | T | U | V | W |

It is the same key ‘3’ as the previous one for the same reason.   
  
But the two cipher texts has a small issue with the letters Q, X and Y. this problem may caused by the functionality of the encryption method been used.

Task 6

1. I made a simple has function with two functions (add and get hash). Where get hash calculate the hash value of a word and the add function adds the word to the hash set.

list\_size = 256

buckets = [[] for i in range (list\_size)]

def get\_hash(word):

    hash\_value = 0

    for letter in word:

        hash\_value += ord(letter)

    return hash\_value % list\_size

def add(word):

    hash = get\_hash(word)

    if word not in buckets[hash]:

        buckets[hash].append(word)

1. For this task, I used two different files as inputs for first\_test and second\_test, both tests use the plot(freq) function to draw a result of each test.

In the first test, I added the words of the random\_file words in the simple hash provided in the previous task. The idea is to draw each bucket in the hash set and the length of it [3].   
  
Here is the result:

En bild som visar linje, Graf, diagram, text

Automatiskt genererad beskrivning

The result shows that almost all of buckets have the length value between 2-4 words. While only one bucket has 12 words, and four have 10 words.   
the result shows that almost the words are well distributed, but it could much better if all the buckets have 4-5 words.

For the second test, I only used the get\_hash function to analyse the its functionality.   
dic = {}

    for i in range(len(words) -1): #we don't use the last item (will be used already as a pair with the previous one)

        diff\_value = abs(get\_hash(words[i]) - get\_hash(words[i +1])) #only intressted in the value of the difference

        dic.setdefault(i, diff\_value)

    plot(dic)

By calculating the difference of the hash value between each word and the next similar word in the file.   
The matplotlib draw this diagram of the second test.

The result for the similar words file:

En bild som visar linje, diagram, Parallell, Graf

Automatiskt genererad beskrivning

Almost all the differences between two similar words are one.

1. The previous simple function is not secure, since we don’t have 100% distributed words through the hash set, and it makes it easy to track the address of a word/data since the differences between similar words/data are almost nothing.   
   That tells that the secure hash function should makes a unique and unexpectable values of the hash value. That ensure that the all data even if similar will be well distributed and not trackable.

Bibliography

* [1] <https://www.baeldung.com/cs/hashing-vs-encryption>
* [2] <https://www.ijert.org/digital-watermarking-and-steganography#:~:text=Steganography%20is%20used%20for%20secret,content%20authe>
* [3] <https://www.tutorialspoint.com/matplotlib/matplotlib_pylab_module.htm>