Linnéuniversitetet
Kalmar Växjö

Linnaeus University

1DV700 - Computer Security
Assignment 1

Student: <Bakay Heysem> <Askan>
Personal number: 960206-7317
Student ID: ab224hv@student.lnu.se

Setup Premises

I used Windows 10 as OS and InteliJ as my text editor. I used Java as the computer language for the pratical parts.

Task 1

a)
Symmetric encryption – Asymmetric encryption.
In Symmetric encryption a single key is used to both decrypt and encrypt data, however in asymmetric encryption one key is used to encrypt and another key is used to decrypt. If A wants to send an encrypted message to B using asymmetric encryption, B has to give A his/her public key to encrypt it. B would already have a private key and would use it to decrypt the message [1].

Encryption algorithms – Hash algorithms
In encryption algorithm you retrieve the original data back however in hash algorithm you just confirm that the data is correct data. Hashing algorithms are used mainly in passwords. In encryption the size of the output is same as the size of the input, but in Hash the output size can be different than the input size [2].

Compression – Hashing
In compression the idea is to recover the original data even though the end product is smaller. However in hashing you can't recover the original data back.

b)
Steganography is used to hide the data in plain sight so people who are not supposed to see it doesn't discover its existence. Watermarking shows the ownership of the data, it also is hidden. Encryption protects the data by making it unreadable. Both steganography and watermarking are hidden however encryption is in plain sight but unreadable. Both steganography and encryption aims to make original data undiscoverable, however watermarking aims to show the ownership of the data.

Task 2

a)
The website has an example of digital image steganography. Website hides the image you want to be hidden inside another image. The tool the website is using is "the least significant bit" steganography. They replaced the least significant bits of the image with the most significant bits of the image that was going to be hidden. When you want to get the original image back you just take the bits you replaced and fill the rest of the byte with 0's. This way you could get almost the same thing you have hidden. The difference is not noticeable to the human eye since for both images the most significant bits haven't been replaced. For a person who knows there is another image hidden inside this image it would be easy to get to the secret image. Since all they have to do is just take the last few bits of each byte of the image and make that the header of the new byte of the hidden image and fill the rest of the new byte with 0s. Also they can detect if there is a hidden image under another image by looking at the intensity. Images with another image hidden in them will be more intense than originally. By intensity I mean they could take the last bits of the original image and the last bits of the image with hidden data, they will see that the image with hidden data is much more intense than it originally should be.

b)
Greek Steganography
When we saw the first steganography it was in Greece around 440BC. The Greek king Histaeus used steganography to send hidden messages. He did so by shaving the heads of the slaves and tattooing the hidden message on their scalps, then he would wait for the hair to grow so the hair would cover the secret message. He would send the slave to the recipient when the message was totally hidden under the hair. The recipient would shave the heads of the slaves again to read the hidden message [3].

Words hidden in Words
Null ciphers is another type of steganography method to pass on secret messages. The hidden message would be hidden in a paragraph character by character in a uniform pattern [3].

"This missing educator's id seems encrypted !"

For example if we take the second letter of each word on the above sentence we get the word "hidden".

Secret messages in Sound
Just like images, messages can be hidden in the LSB (Least Significant Bit) of the audio cover. How ever changes in audio cover is much more noticeable then an image since our ears are more sensitive to change and the extra bits we provide changes the audio even if slightly [4] .

c)
First of all I used a help of a website to translate the image into hex values. Then In java I built a program which took the last bit of each hex value and put it in a string. When I printed the string out, I saw lines of irregular 1's in the beginning and then all 0's. I copied the irregular part (until the last 1) and used this website https://www.binaryhexconverter.com/binary-to-ascii-text-converter to translate it to text. It printed out Congratulations!

Task 3

a)
The result is “Encrypted Message”.

b)
This message could be decrypted by someone who doesn’t have a key. They would have to try all the combination of letters but there are only 26 of them. It would not take long to try the combinations to get something which makes sense also in the message some alphabets are repeated one after another which could give them clues on the message if they do frequency analysis [5].

Task 4

I implemented Ceasers Cipher (Substitution) and Rail Fence Cipher (Transposition) for this assighnment.
For the Ceasers Cipher I ask the user to input a letter between a – y. This is used to generate a number which will be used for encryption later. I took the contents of the file and looped through it character by character. First thing I check is if the character is non alphabetic like "?" or " ", if it is I put that character without encrypting it. Then I check if the character is a capital letter or a small letter, this matters since capital letters and small letters have different numeric value ranges
(A-Z → 65-90) (a-z → 97-122). I do modulo (64 for capital, 96 for small) to get a proper aphabetic representation of the numbners place (1-26). Then to that I add the number generated from the key to find the letter which will substitute this letter. For decryption I do the same procedure but instead of adding I substract the generated number from the key.

For Rail Fence Cipher I ask the user to input a letter between e – n. This is used to decide amount of coloms the plain text will be distributed to. I basically generate a specific amount of Strings in an ArrayList and add the letters of the plaintext one by one to those strings. So when I combine all the Strings the end result is a ciphertext made out of the letters from the plain text scrambled. To decrypt it I use the key to find out how many characters each String had. When I do find that out I create Strings out of the substrings of the cipher text. Then I loop through those Strings and retreve the characters at the same position of each string into a single String this way I get my plain text. It is quite confusing while explaining but the code will be more clear.

Task 6

To try and decrypt I picked Jill Egel's file. I knew Jill used substitution method to encrypt the text since the spaces between words and sentences didn't seem irregular and there was a full stop after each sentence. So I took the first word in the text "Vhfuhw" as an example to use. In java I created a program which increased the Ascii number of each char with 1 in a loop which ran 26 times. When I could not find a coherent result I decreased the Ascii number of each char with 1 instead of increasing it and ran that loop for 26 times too. On the 3rd loop I realized that word "Secret" was created. So I decided to do this with all the words in the text file. The result is: Secret message – Top Secret May only be read by security passed personnel My name is Jill and I am form Germany. I arrived last Friday. And this is my secret message for the first assignment. (Repeated 5 times) Jill Egel

Task 7

I created two different hash functions for this exercise and ran the tests for both of them. They both have their advantages and disadvantages and neither are perfect. In both of my hash functions I am taking a String input from the user then I am translating each letter to corresponding decimal like 'A' to 65 and putting it in an int variable. I am then performing different logical and arithmetic functions on them. I made sure that the output was 16 bits each time.

Hash Function One:
In this function I tried to implement byte checksum. First I check the size of the String the user has entered. If the user has entered a single character, I add the character to itself, the output will be 8 bits then I take ones compliment (~) of the result. Turn them into String of bits and combine them together for a final result of 16 bits in a String. If the size of the input String is more then one, I iterate the String character by character and add them all with each other instead of adding a single character to itself, then I take the ones compliment of the result and combine them for a 16 bit String output.

Sometimes the addition might give a result of 8 bits where most significant bit is 0, in that case java disregards it while giving it as a bit String output. So the size of total sum might end up being 7 bits, To prevent that before combining the ones compliment with the total sum I check that the output String format is 8 chars if not I make the most significant bit 0 manually then I combine the ones compliment with the total sum.

Also if the entered String is large the total sum might be over 8 bits, in that case I take the least significant 8 bits from total sum and ones compliment as the final result.

Hash Function Two
In this hash function I am using logical function XOR (^). First I am making sure that there are at least 4 characters and there are even number of characters to be processed. To make sure of that I add a single char of padding as needed extra to what the user has entered. The char I use as padding is 'X'. Then I parse the string by two. So I have multiple sub-strings with 2 characters. This is to make sure that each sub-string is 16 bits in total. Then I XOR them to each other, as a result I get a 16 bit String.

Issues
For the first hashing algorithm, the result is not extremely unique since I have to use the least significant 8 bits of the total sum. If the assignment allowed to have a larger digest I would have gotten a much more unique results. Also if you enter the same word with different character order it still gives the same hash digest as the original word. For example "cat" would give the same hash digest as the word "act". This is because for addition the position does not matter, it still gives out the same result.

Even though second algorithm gives a much more unique result, it has its own setbacks too. For example of the user enters a single character 'X' or a sequence of Xs (XXX) the result will be a stream of 16 '0' bits. Or even if the user enters a single character that is not 'X', least significant 8 bits will always be 0.

b)
Tests
I found a list of 1000 most used English word list. The words vary in size and character content. I added them to a text file. I got the hash digest of each word, then I created a hash map. As the key I put the hash digests, which are supposed to be unique for each of them, and as the value I put the words themselves. When I conducted statistical testing I got the following results.

Hash One:
Out of 1000 entries 250 different hash digests were produced. This number is very low.
There was:
31 keys that corresponds to a single value
44 keys that corresponds to two values
43 keys that corresponds to three values
33 keys that corresponds to four values
38 keys that corresponds to five values
28 keys that corresponds to six values
12 keys that corresponds to seven values
12 keys that corresponds to eight values
8 keys that corresponds to nine values
1 keys that corresponds to ten values
So at most a key would correspond to 10 values.

Hash Two:
I got a much better results even though not perfect. I got 855 hash digests produced out of 1000.
There was:
730 keys that corresponds to a single value
108 keys that corresponds to two values
14 keys that corresponds to three values
3 keys that corresponds to four values

Now I ran the same tersts with very similar 1000 words with a character different and all same size.

Hash One:
For the first algorithm test again results were awefull. Out of 1000 words only 97 different hash digests were produsedd.
There was:
15 keys that corresponds to a single value
3 keys that corresponds to two values
4 keys that corresponds to three values

4 keys that corresponds to four values
2 keys that corresponds to five values
4 keys that corresponds to six values
4 keys that corresponds to seven values
4 keys that corresponds to eight values
4 keys that corresponds to nine values
3 keys that corresponds to ten values
15 keys that corresponds to eleven values
7 keys that corresponds to twelve values
2 keys that corresponds to thirteen values
2 keys that corresponds to fourteen values
2 keys that corresponds to fifteen values
2 keys that corresponds to sixteen values
2 keys that corresponds to seventeen values
2 keys that corresponds to eighteen values
2 keys that corresponds to nineteen values
2 keys that corresponds to twenty values
2 keys that corresponds to twenty one values
1 keys that corresponds to twenty two values
2 keys that corresponds to twenty three values
2 keys that corresponds to twenty four values
2 keys that corresponds to twenty five values
2 keys that corresponds to twenty six values
1 keys that corresponds to twenty seven values

Hash Two:
For the secend algorithms test the results were suprisingly even better than the test with 1000 random words. Out of 1000 similar words 973 different hash digests were produced.
There was:
937 keys that corresponds to a single value
36 keys that corresponds to two values

c)
To prove that my Hash Two is not secure all the attacker has to do is enter a single character to the hash function. He will see that the Most Significant 8 bits will be random but the Least Significant 8 bits will always be a stream of 0s. Also when the attacker inserts character X he will see that all 16 bits will be 0. He can than can make out that I am XOR ing 16 bits with each other and padding with the letter X. If the sender sends a hash digest of a single character the attacker can just XOR the Most Significant 8 bits of the digest with binary representation of X and the result will be the binary representation of the character sent.

Bibliography

[1] ssl2buy.com, "Symetrick vs. Asymmetrick Encryption – What are the differences?", [2020-02-09], url: [https://www.ssl2buy.com/wiki/symmetric-vs-asymmetric-encryption-what-are-differences]

[2] P.Nohe, "The difference between Encryption, Hashing and Salting", [2020-02-09], url: [https://www.thesslstore.com/blog/difference-encryption-hashing-salting/]

[3] A. Siper, R. Farley, C. Lombardo, "The Rise of Steganography", [2020-02-09], url: [http://csis.pace.edu/~ctappert/srd2005/d1.pdf]

[4] S.K. Arora, "Audio Steganography: The art of hiding secrets within earshot", [2020-02-09], url: [https://medium.com/@sumit.arora/audio-steganography-the-art-of-hiding-secrets-within-earshot-part-2-of-2-c76b1be719b3]

[5] 101computing.net "Frequency Analysis", [2020-02-09], url: https://www.101computing.net/frequency-analysis/]