Linnéuniversitetet
Kalmar Växjö

Linnaeus University

1DV700 - Computer Security

Assignment 1

Student: Evan Huynh

Personal number: 970330-4130

Student ID: eh223im@student.lnu.se

Setup Premises

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

OS: OS X 10.13 on MacBook Pro.

IDE: IntelliJ IDEA 2019.3

Java version: 13

Web browser: Google Chrome (ver 79 as the time of writing)

Specific tool used:

Task 2: https://hexed.it

Task 3: https://quipqiup.com/

Task 6: https://www.dcode.fr/monoalphabetic-substitution

Task 1

1) The first task is to investigate different terms within cryptography and related areas.

    a. Different between pairs of methods:

Symmetric encryption: Alice wants to send Bob the message P. She encrypts the message using key K which results $E(K, P)$. If Bob wants to read P, he has to decrypt $E(K, P)$ using the decryption process which results $D(K, E(K, P))$ We can see that the same key is used to lock and unlock message. The algorithm to obtain the plaintext is $P = D(K, E(K, P))$ because Decryption and Encryption are the mirror image of the process using the same key K. [1]

Advantages: Using the same key K both to encrypt and decrypt, the algorithm is faster since only one shared key is used.

Disadvantages: If the key K is lost then all the messages are available to the attackers. The transmission of key K which can be both hard and dangerous, due to physical constrain or the communication method.

Asymmetric encryption: different keys are used to encrypt and decrypt the message. The process can be described using $P = D(K_D, E(K_E, P)$ [1]. This means that the encryption uses key K_E and the decryption process use key K_D, which is totally different than each other.

Advantages: Alice and Bob do not need to know each other private key in order to read the message. If the algorithm is lost, future messages would not be read since the interceptor will not know the key value. [1]

Disadvantages: The need to keep track of more keys since one key is used to encrypt the message and one key is used to decrypt it.

Encryption algorithms: the encryption algorithm is the process of encoding the message so that the message does not become obvious. [1] However, encryption algorithm is a two-way function since the message has to be readable by both party. If Alice send Bob something, Bob should be able to read it.

Hash algorithms: the hash algorithm digest the message and provide the value, [2] doing the similar thing with the encryption. The purpose of hash is to verify the authenticity of the message. [1] However, the algorithm should be a one-way function since there should be no way that the attacker can work backward the original message from the hashed value. This mean that once Alice hash something, Bob should NOT have the original message from Alice, just a value to ensure that the message has not been modified.

Compression: compression is the act of encoding to reduce the file size in order to save disk space and bandwidth transmission. [3] Compressed files must be recoverable.

Hashing: a hashing algorithm is a function to map data to a table. However, hashed value is not recoverable since it is a one way function.

b) What are the differences

Steganography: “Steganography permits data to be hidden in large, complex, redundant

data sets.” [1] This means steganographic data is only hidden within bigger data set, without scrambling it. They are used in case where encryption is illegal or to hide the fact that there are secret messages being sent. [4]

Encryption: “Encryption is the process of encoding a message so that its meaning is not obvious”. [1] Encrypted data must be recoverable. They are normally used to hide the message from unauthorized access. [1]

Digital watermarking: they are used to protect the digital file from being copied. [1]

Task 2

2) Steganography

a. According to the website [5], they use the classical technique of hiding the image within image using the least significant bit to achieve it.

Limitation: the hidden image quality is losses due to loss of information. Storing picture within pictures will results the recovered image quality is loss since there are not enough information that could be recovered from the stego file. Also, if the message is too large, the original picture quality might be modified too much that the attacker can notice something wrong with it. For example, hiding 7 bits in the original image might results the low-quality stego image, which indicates something is wrong with the data being sent. [5]

b. Not only it can be done using images, it could be performed similar way using audio files. The technique is the same, using LSB to hide the information within the audio file. [4]

c. Since the image is mostly black and white, those white regions are represented in 00 bit. However, I noticed there are a lot of irregularity starting from bit #54 (reference to my Java program). In this image, most of the black pixels are represented by FE, however there are some that are represented by FF which caught my attention. Most of the black are represented by FE, throughout the rest of the file.
So I decided to write some Java code. [6] [7] I was able to extract specific region that contain the message. The algorithm was pretty simple, take the bit number, convert to integer and take the modulus 2. If it is even, that bit has a 0 hidden in the LSB, otherwise 1.
The rest of the program is just for decorating and house-keeping purpose. I just simply split the string every 8th bit and convert binary string to ASCII. [8]

```java
public class Secret {
    public static void main(String[] args) throws Exception {
        // Source: http://www.java2s.com/Code/Java/File-Input-Output/Readbytesanddisplaytheirhexadecimalvalues.htm
        FileInputStream fin = new FileInputStream( name: "/Users/My2ndAngelic/Downloads/Secret.bmp");
        int len;
        byte data[] = new byte[1];
        // Read bytes until EOF is encountered.
        StringBuilder sb = new StringBuilder();
        StringBuilder sb2 = new StringBuilder();
        ArrayList<String> arrstr = new ArrayList<>();
        do {
            len = fin.read(data);
            for (int j = 0; j < len; j++) {
                String a = String.format("%02X",data[j]);
                    sb.append(a).append(" ");
                    arrstr.add(a);
            }
        } while (len != -1);
        // Get 1 and 0
        for (int i = 54; i<200; i++) {
            String a = arrstr.get(i);

            int i1 = Integer.parseInt(String.valueOf(a.charAt(1)), radix: 16) % 2;
            if (i1 == 1) {
                sb2.append("1");
            } else if (i1 == 0) {
                sb2.append("0");
            }
        }
        // Split them every 8th position
        for (int i = 0; i < sb2.length(); i++) {
            if (i % 9 == 0) {
                sb2.insert(i, str: " ");
            }
        }
        System.out.println(sb2);
    }
}
```

The final message is “Congratulation!”

Task 3

3) Message decryption
   a) With pen and paper: “encrypted message”.
   b) It could be decrypted without the key. Since this is a monoalphabetic cipher, it could be decrypted with ease. Although frequency analysis of single character might not work 100%, there are also some more popular bigrams (th, he, …), trigrams (the, and, …), not to mention double letters (ss, ee, ll, …) in English. [9]

Knowing I used the tool [10] (mentioned in the beginning) to help me to decrypt the message without the alphabet. It gave ‘exploited message’ as the highest possible result and ‘encrypted message’ as the second possible one.

Task 4

4) Substitution and Transposition cipher

Substitution cipher

```
static String encryption(String text, int key) {
    StringBuilder temp = new StringBuilder();
    for (int i = 0; i<text.length(); i++) {
        temp.append(Character.toString( codePoint: text.charAt(i) + key));
    }
    return temp.toString();
}

static String decryption(String text, int key) {
    StringBuilder temp = new StringBuilder();
    for (int i = 0; i<text.length(); i++) {
        temp.append(Character.toString( codePoint: text.charAt(i) - key));
    }
    return temp.toString();
}
```

For the substitution cipher program, it is fairly easy to understand. I just take every character, plus the key, which is an integer from 0 to 255 (exactly 256 possible keys). This function will perform the encryption on every letter and then return everything as a string. Since StringBuilder class is faster than string concatenation, I decide to use it as a way to optimize my program. However, to also save time, I will force exit the program in case of invalid key number (smaller than 0 or larger than 255), which is shown below.

```
if (key < 0 || key > 255) {
    System.err.println("Invalid key.");
    System.exit( status: -1);
}
```

Decrypting the cipher text is just a matter of take it and subtract with the provided key.

Transposition cipher

For transposition cipher, the problem becomes a little bit harder. There are two problems that need to be solved: the key and the encryption algorithm.

For the key, below are every invalid reasons for the key:

Key	Invalid reason	Note
2013	Contain 0	If 0 is in the beginning, it is still invalid since I treat

		the user input key as a String and read every digit.
214	Maximum digit is larger than the key	Should not work because we do not know where should the missing letter go
234	Does not contain 1	Same reason as above
2113	Contain duplicate	This exists only to save time
-1234	Contain digit smaller than 0 or larger than 9	This should not be possible if user enter such an integer since I will convert it into String and split every character into integer array.

The program, of course will exit if the user enter the invalid key or no input file found.

For the encryption/decryption algorithm, since they both works the same, I will just explain one of them.

At first, the algorithm checks for the length of the key (says k length) and length of the string (s length). If $s \not\equiv 0 \ (mod\ k)$, then the original string will be appended $k - (s \ mod \ k)$ whitespaces. Algorithmically speaking, this ensures that we have enough characters to swap around.

The second step, I will split the string into $\left\lceil \frac{s}{k} \right\rceil$ arrays, casting both of s and k into double then cast the ceiling result into integer again. For this, the string is now split into array with equal length, split each of those strings into subarrays containing characters and then the swapping begins.

For the swapping algorithm, I just get ever right character into their right position by just make an new empty array, read both the each character array and key array. Since the key array is the position of the original character array, every character in the new array is the character of the original array of their respected position. Do this for all characters in each subarray and then using StringBuilder to append all the characters in every subarray in ever array.

Here is the implementation of the swapping algorithm:

```
// Swap
static char[] swapChar(char[] arr, int[] key) {
    char[] temp = new char[arr.length];
    for (int i = 0; i<key.length; i++) {
        temp[i] = arr[key[i]-1];
    }
    return temp;
}
```

Decrypting the cipher text is just a matter of swapping back the encrypted one since swapping them twice will yield the original result. This algorithm is symmetric.

File handling

Reading file is just a matter of putting every line into a String array. Below is the algorithm I used [11]:

```
// Source: https://stackoverflow.com/questions/16265693/how-to-use-bufferedreader-in-java
static String[] read(String loc) throws Exception {
    BufferedReader in = new BufferedReader(new FileReader(loc));
    String line;
    ArrayList<String> arrlist = new ArrayList<>();
    while((line = in.readLine()) != null)
    {
        arrlist.add(line);
    }
    in.close();
    return arrlist.toArray(new String[0]);
}
```

For writing, I just write every line in the String array of to a new line.

Main program

```
package assign1;

import java.util.Scanner;

public class Main {
    public static void main(String[] args) throws Exception {
        System.out.print("Pick your method, [S]ubstitution (default) or [T]ransposition: ");
        Scanner sc = new Scanner(System.in);
        String a = sc.nextLine();
        if (a.toUpperCase().equals("T")) {
            Transposition.main(new String[0]);
        } else {
            Substitution.main(new String[0]);
        }
    }
}
```

Main program is just a method to ask the user which algorithm they want to run and then call the main method of the respected algorithm. The rest of each program is just asking users if they want encrypt or decrypt, key, input file, output file and then perform the main algorithm. Please note that the encryption/decryption in the transposition is just for decoration since both of them work the same no matter what they would enter.

I gave default options for every question since dealing other user inputs is not really interesting compared to the algorithm. Of course, this is not an option for the key or for the in/output file since there is no way that I would know which one of them they want to perform algorithm on or the key.

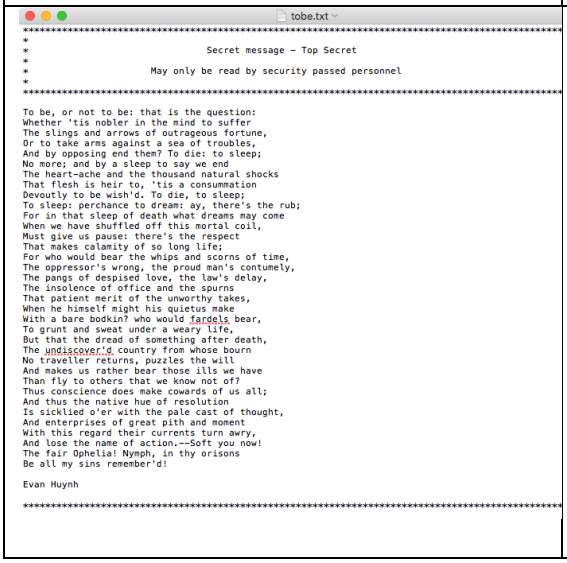
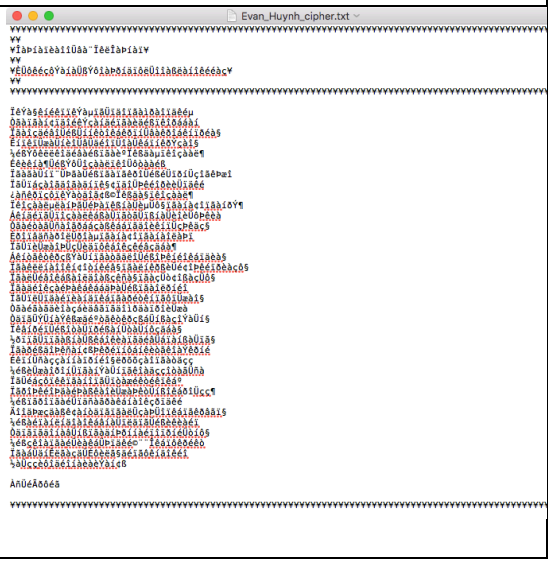
Calling the main program in each class is not a good practice. This is not a really big deal in this case.

Task 5

5) Encrypted file

In this task I added the soliloquy by Prince Hamlet in act 3, scene 1 of William Shakespeare's play Hamlet, mostly known as "To be or not to be" [12]. This file is encrypted using my substitution method with the key '123' and uploaded in my name. Below are the images showing what the file should looks like on my machine before and after encrypting it.

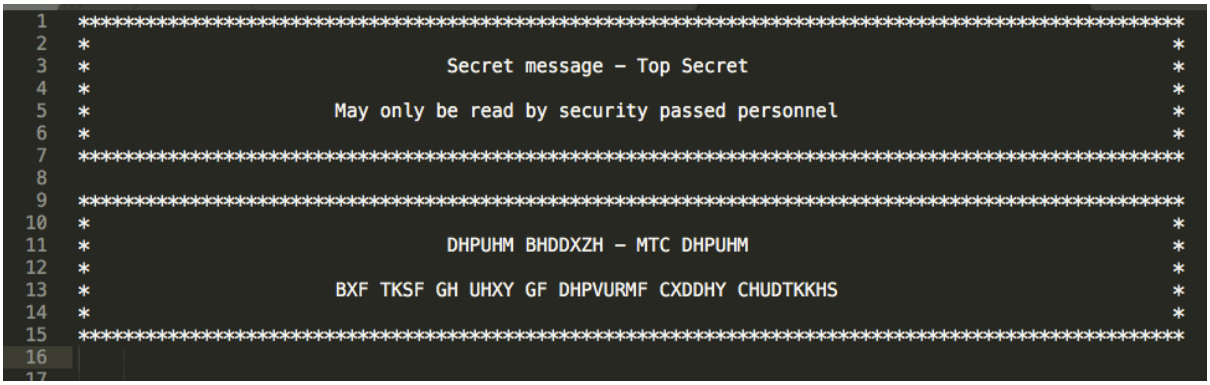
Of course, the encrypted file does contain some special characters and non-printable characters as well. Here is the result.

Original	Encrypted
	

Task 6

6) Cryptanalysis

In this task I picked the easy monoalphabetic cipher by Haofei Yan [13] (given from the name of the file). I immediately noticed that the first two lines give the most of information.



Which gives me this alphabet.

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

mih hxusrhdm **jktnk** cutzuxbbxgsh bxpirkh (**mixm** rd x **bxpirkh nitdh ghixertu** pxk gh ptkmutsshy gf **pixkzhd** mt x "cutzuxb") nxd xs-axoxur'd cutzuxbbxgsh ivbxktry utgtm rk 1206. xs-axoxur'd utgtm nxd turzrkxssf x gtxm nrmi **qtvu xvmtbxmrp bvdrprxkd mixm** qstxmhy tk x sxjh mt hkmhumxrk zvhdmd xm utfxs **yurkjrkz** cxumrhd. **ird** bhpixkrdb ixy x cutzuxbbxgsh yuvb bxpirkh nrmi chzd (pxbd) mixm gvbc rkmt srmmsh **shehud** mixm tchuxmh mih chupvddrtk. mih yuvbbhu ptvsy gh bxyh mt csxf yrqqhuhkm uifmibd xky yrqqhuhkm yuvb cxmmhukd gf bterkz mih chzd mt yrqqhuhkm stpxmrtkd. **xktmihu dtcirdmrpxmhy** cutzuxbbxgsh **bxpirkh** gf **xs-axoxur nxd mih** pxdmsh **pstpj**.	**t?e** earliest **?no?n** programmable mac?ine (**t?at** is a **mac?ine ??ose be?a?ior** can be controlled by **c?anges** to a "program") ?as al-?a?ari's programmable ?umanoid robot in 1206. al-?a?ari's robot ?as originally a boat ?it? **?our automatic musicians t?at** ?loated on a la?e to entertain guests at royal **drin?ing** parties. **?is mec?anism** ?ad a programmable drum mac?ine ?it? pegs (cams) t?at bump into little **le?ers** t?at operate t?e percussion. t?e drummer could be made to play di??erent r?yt?ms and different drum patterns by mo?ing t?e pegs to different locations. **anot?er sop?isticated** programmable **mac?ine** by **al-?a?ari ?as t?e** castle **cloc?**.

The next step is taking the first paragraph, replaced all the unknown with ? using MS Word 'Find and Replace' tool. This step ensures that later on I will not accidentally replace unsolved clues with another one. Then, I started using the same 'Find and Replace' tool to solve for the unknowns, which gives the following paragaph on the right hand side.

Because this text is in English (obviously), I underline and bolden some of the words which gives me some informations and the corresponding one on the left, like machine, another, sophisticated, clock, the, etc. This one also mentioned the castle clock machine, a quick Google search gives the name Al-Jazari, which is the inventor of such machine. This results in the bold and underlined text on the alphabet. By this state, only two clues left.

The next step is to search for paragraphs which contain Q and X, which is to search for words contain L, W in the cipher text. So I look at the first sentence in the second paragraph.

mih axplvxuy sttb, yhehstchy rk 1801, rd tqmhk lvtmhy xd x dtvuph tq curtu xum.	the jac?uard look, developed in 1801, is often ?uoted as a source of prior art.

This immediately gives Q encrypted to L, so X only encrypted to W. This is noted by the bold and underline text with background.

With the completed alphabet, the final step is to decode using tool. I used the website dcode.fr [14] and below is what I get. I copied all the note with no edit on spacing. Below are both cipher text and decrypted one.

```
**************************************************************************
*************************
*                                                                        *
*                    DHPUHM BHDDXZH - MTC DHPUHM                         *
*                                                                        *
*                    BXF TKSF GH UHXY GF DHPVURMF CXDDHY CHUDTKKHS
*
*                                                                        *
**************************************************************************
*************************


    PTBCVMHU CUTZUXBBRKZ, IRDMTUF TQ CUTZUXBBRKZ

QUTB NRJRCHYRX, MIH QUHH HKPFPSTCHYRX (081110)


MIH HXUSRHDM JKTNK CUTZUXBBXGSH BXPIRKH (MIXM RD X BXPIRKH NITDH GHIXERTU PXK GH PTKMUTSSHY GF PIXKZHD MT X "CUTZUXB") NXD XS-AXOXUR'D CUTZUXBBXGSH IVBXKTRY UTGTM RK 1206. XS-AXOXUR'D UTGTM NXD TURZRKXSSF X GTXM NRMI QTVU XVMTBXMRP BVDRPRXKD MIXM QSTXMHY TK X SXJH MT HKMHUMXRK ZVHDMD XM UTFXS YURKJRKZ CXUMRHD. IRD BHPIXKRDB IXY X CUTZUXBBXGSH YUVB BXPIRKH NRMI CHZD (PXBD) MIXM GVBC RKMT SRMMSH SHEHUD MIXM TCHUXMH MIH CHUPVDDRTK. MIH YUVBBHU PTVSY GH BXYH MT CSXF YRQQHUHKM UIFMIBD XKY YRQQHUHKM YUVB CXMMHUKD GF BTERKZ MIH CHZD MT YRQQHUHKM STPXMRTKD. XKTMIHU DTCIRDMRPXMHY CUTZUXBBXGSH
```

BXPIRKH GF XS-AXOXUR NXD MIH PXDMSH PSTPJ.

MIH AXPLVXUY STTB, YHEHSTCHY RK 1801, RD TQMHK LVTMHY XD X DTVUPH TQ CURTU XUM. MIH BXPIRKH VDHY X DHURHD TQ CXDMHGTXUY PXUYD NRMI ITSHD CVKPIHY RK MIHB. MIH ITSH CXMMHUK UHCUHDHKMHY MIH CXMMHUK MIXM MIH STTB IXY MT QTSSTN RK NHXERKZ PSTMI. MIH STTB PTVSY CUTYVPH HKMRUHSF YRQQHUHKM NHXEHD VDRKZ YRQQHUHKM DHMD TQ PXUYD. MIH VDH TQ CVKPIHY PXUYD NXD XSDT XYTCMHY GF PIXUSHD GXGGXZH XUTVKY 1830, MT PTKMUTS IRD XKXSFMRPXS HKZRKH.

MIRD RKKTEXMRTK NXD SXMHU UHQRKHY GF IHUBXK ITSSHURMI NIT, RK 1896 QTVKYHY MIH MXGVSXMRKZ BXPIRKH PTBCXKF (NIRPI GHPXBH RGB). IH RKEHKMHY MIH ITSSHURMI CVKPIHY PXUY, MIH PXUY UHXYHU, XKY MIH JHF CVKPI BXPIRKH. MIHDH RKEHKMRTKD NHUH MIH QTVKYXMRTK TQ MIH BTYHUK RKQTUBXMRTK CUTPHDDRKZ RKYVDMUF. MIH XYYRMRTK TQ X CSVZ-GTXUY MT IRD 1906 MFCH R MXGVSXMTU XSSTNHY RM MT YT YRQQHUHKM ATGD NRMITVM IXERKZ MT GH UHGVRSM (MIH QRUDM DMHC MTNXUY CUTZUXBBRKZ). GF MIH SXMH 1940D MIHUH NHUH X EXURHMF TQ CSVZ-GTXUY CUTZUXBBXGSH BXPIRKHD, PXSSHY VKRM UHPTUY HLVRCBHKM, MT CHUQTUB YXMX CUTPHDDRKZ MXDJD (PXUY UHXYRKZ). MIH HXUSF PTBCVMHUD NHUH XSDT CUTZUXBBHY VDRKZ CSVZ-GTXUYD.

X GTW TQ CVKPI PXUYD NRMI DHEHUXS CUTZUXB YHPJD.

MIH RKEHKMRTK TQ MIH ETK KHVBXKK XUPIRMHPMVUH XSSTNHY PTBCVMHU CUTZUXBD MT GH DMTUHY RK PTBCVMHU BHBTUF. HXUSF CUTZUXBD IXY MT GH CXRKDMXJRKZSF PUXQMHY VDRKZ MIH RKDMUVPMRTKD TQ MIH CXUMRPVSXU BXPIRKH, TQMHK RK GRKXUF KTMXMRTK. HEHUF BTYHS TQ PTBCVMHU NTVSY GH SRJHSF MT KHHY YRQQHUHKM RKDMUVPMRTKD MT YT MIH DXBH MXDJ. SXMHU XDDHBGSF SXKZVXZHD NHUH YHEHSTCHY MIXM SHM MIH CUTZUXBBHU DCHPRQF HXPI RKDMUVPMRTK RK X MHWM QTUBXM, HKMHURKZ XGGUHERXMRTKD QTU HXPI TCHUXMRTK PTYH RKDMHXY TQ X KVBGHU XKY DCHPRQFRKZ XYYUHDDHD RK DFBGTSRP QTUB (H.Z. XYY W, MTMXS). RK 1954 QTUMUXK, MIH QRUDM IRZIHU SHEHS CUTZUXBBRKZ SXKZVXZH, NXD RKEHKMHY. MIRD XSSTNHY CUTZUXBBHUD MT DCHPRQF PXSPVSXMRTKD GF HKMHURKZ X QTUBVSX YRUHPMSF (H.Z. F = W*2 + 5*W + 9). MIH CUTZUXB MHWM, TU DTVUPH, NXD PTKEHUMHY RKMT BXPIRKH RKDMUVPMRTKD VDRKZ X DCHPRXS CUTZUXB PXSSHY X PTBCRSHU. BXKF TMIHU SXKZVXZHD NHUH YHEHSTCHY, RKPSVYRKZ TKHD QTU PTBBHUPRXS CUTZUXBBRKZ, DVPI XD PTGTS. CUTZUXBD NHUH BTDMSF DMRSS HKMHUHY VDRKZ CVKPI PXUYD TU CXCHU MXCH. (DHH PTBCVMHU CUTZUXBBRKZ RK MIH CVKPI PXUY HUX). GF MIH SXMH 1960D, YXMX

DMTUXZH YHERPHD XKY PTBCVMHU MHUBRKXSD GHPXBH RKHWCHKDREH HKTVZI DT CUTZUXBD PTVSY GH PUHXMHY GF MFCRKZ YRUHPMSF RKMT MIH PTBCVMHUD. MHWM HYRMTUD NHUH YHEHSTCHY MIXM XSSTNHY PIXKZHD XKY PTUUHPMRTKD MT GH BXYH BVPI BTUH HXDRSF MIXK NRMI CVKPI PXUYD.

XD MRBH IXD CUTZUHDDHY, PTBCVMHUD IXEH BXYH ZRXKM SHXCD RK MIH XUHX TQ CUTPHDDRKZ CTNHU. MIRD IXD GUTVZIM XGTVM KHNHU CUTZUXBBRKZ SXKZVXZHD MIXM XUH BTUH XGDMUXPMHY QUTB MIH VKYHUSFRKZ IXUYNXUH. XSMITVZI MIHDH BTUH XGDMUXPMHY SXKZVXZHD UHLVRUH XYYRMRTKXS TEHUIHXY, RK BTDM PXDHD MIH IVZH RKPUHXDH RK DCHHY TQ BTYHUK PTBCVMHUD IXD GUTVZIM XGTVM SRMMSH CHUQTUBXKPH YHPUHXDH PTBCXUHY MT HXUSRHU PTVKMHUCXUMD. MIH GHKHQRMD TQ MIHDH BTUH XGDMUXPMHY SXKZVXZHD RD MIXM MIHF XSSTN GTMI XK HXDRHU SHXUKRKZ PVUEH QTU CHTCSH SHDD QXBRSRXU NRMI MIH TSYHU STNHU-SHEHS CUTZUXBBRKZ SXKZVXZHD, XKY MIHF XSDT XSSTN X BTUH HWCHURHKPHY CUTZUXBBHU MT YHEHSTC DRBCSH XCCSRPXMRTKD LVRPJSF. YHDCRMH MIHDH GHKHQRMD, SXUZH PTBCSRPXMHY CUTZUXBD, XKY CUTZUXBD MIXM XUH BTUH YHCHKYHKM TK DCHHY DMRSS UHLVRUH MIH QXDMHU XKY UHSXMREHSF STNHU-SHEHS SXKZVXZHD NRMI MTYXF'D IXUYNXUH. (MIH DXBH PTKPHUKD NHUH UXRDHY XGTVM MIH TURZRKXS QTUMUXK SXKZVXZH.)

MIUTVZITVM MIH DHPTKY IXSQ TQ MIH MNHKMRHMI PHKMVUF, CUTZUXBBRKZ NXD XK XMMUXPMREH PXUHHU RK BTDM YHEHSTCHY PTVKMURHD. DTBH QTUBD TQ CUTZUXBBRKZ IXEH GHHK RKPUHXDRKZSF DVGAHPM MT TQQDITUH TVMDTVUPRKZ (RBCTUMRKZ DTQMNXUH XKY DHUERPHD QUTB TMIHU PTVKMURHD, VDVXSSF XM X STNHU NXZH), BXJRKZ CUTZUXBBRKZ PXUHHU YHPRDRTKD RK YHEHSTCHY PTVKMURHD BTUH PTBCSRPXMHY, NIRSH RKPUHXDRKZ HPTKTBRP TCCTUMVKRMRHD RK SHDD YHEHSTCHY XUHXD. RM RD VKPSHXU ITN QXU MIRD MUHKY NRSS PTKMRKVH XKY ITN YHHCSF RM NRSS RBCXPM CUTZUXBBHU NXZHD XKY TCCTUMVKRMRHD.

IXTQHR FXK IF222XC

****
****

****
****

*

*

*

SECRET MESSAGE - TOP SECRET

*

*

*

*

MAY ONLY BE READ BY SECURITY PASSED PERSONNEL

*

*

*

****
****

COMPUTER PROGRAMMING, HISTORY OF PROGRAMMING

FROM WIKIPEDIA, THE FREE ENCYCLOPEDIA (081110)

THE EARLIEST KNOWN PROGRAMMABLE MACHINE (THAT IS A MACHINE WHOSE BEHAVIOR CAN BE CONTROLLED BY CHANGES TO A "PROGRAM" WAS AL-JAZARI'S PROGRAMMABLE HUMANOID ROBOT IN 1206. AL-JAZARI'S ROBOT WAS ORIGINALLY A BOAT WITH FOUR AUTOMATIC MUSICIANS THAT FLOATED ON A LAKE TO ENTERTAIN GUESTS AT ROYAL DRINKING PARTIES. HIS MECHANISM HAD A PROGRAMMABLE DRUM MACHINE WITH PEGS (CAMS) THAT BUMP INTO LITTLE LEVERS THAT OPERATE THE PERCUSSION. THE DRUMMER COULD BE MADE TO PLAY DIFFERENT RHYTHMS AND DIFFERENT DRUM PATTERNS BY MOVING THE PEGS TO DIFFERENT LOCATIONS. ANOTHER SOPHISTICATED PROGRAMMABLE MACHINE BY AL-JAZARI WAS THE CASTLE CLOCK.

THE JACQUARD LOOM, DEVELOPED IN 1801, IS OFTEN QUOTED AS A SOURCE OF PRIOR ART. THE MACHINE USED A SERIES OF PASTEBOARD CARDS WITH HOLES PUNCHED IN THEM. THE HOLE PATTERN REPRESENTED THE PATTERN THAT THE LOOM HAD TO FOLLOW IN WEAVING CLOTH. THE LOOM COULD PRODUCE ENTIRELY DIFFERENT WEAVES USING DIFFERENT SETS OF CARDS. THE USE OF PUNCHED CARDS WAS ALSO ADOPTED BY CHARLES BABBAGE AROUND 1830, TO CONTROL HIS ANALYTICAL ENGINE.

THIS INNOVATION WAS LATER REFINED BY HERMAN HOLLERITH WHO, IN 1896 FOUNDED THE TABULATING MACHINE COMPANY (WHICH BECAME IBM). HE INVENTED THE HOLLERITH PUNCHED CARD, THE CARD READER, AND THE KEY PUNCH MACHINE. THESE INVENTIONS WERE THE FOUNDATION OF THE MODERN INFORMATION PROCESSING INDUSTRY. THE ADDITION OF A PLUG-BOARD TO HIS 1906 TYPE I TABULATOR ALLOWED IT TO DO DIFFERENT JOBS WITHOUT HAVING TO BE REBUILT (THE FIRST STEP TOWARD

PROGRAMMING). BY THE LATE 1940S THERE WERE A VARIETY OF PLUG-BOARD PROGRAMMABLE MACHINES, CALLED UNIT RECORD EQUIPMENT, TO PERFORM DATA PROCESSING TASKS (CARD READING). THE EARLY COMPUTERS WERE ALSO PROGRAMMED USING PLUG-BOARDS.

A BOX OF PUNCH CARDS WITH SEVERAL PROGRAM DECKS.

THE INVENTION OF THE VON NEUMANN ARCHITECTURE ALLOWED COMPUTER PROGRAMS TO BE STORED IN COMPUTER MEMORY. EARLY PROGRAMS HAD TO BE PAINSTAKINGLY CRAFTED USING THE INSTRUCTIONS OF THE PARTICULAR MACHINE, OFTEN IN BINARY NOTATION. EVERY MODEL OF COMPUTER WOULD BE LIKELY TO NEED DIFFERENT INSTRUCTIONS TO DO THE SAME TASK. LATER ASSEMBLY LANGUAGES WERE DEVELOPED THAT LET THE PROGRAMMER SPECIFY EACH INSTRUCTION IN A TEXT FORMAT, ENTERING ABBREVIATIONS FOR EACH OPERATION CODE INSTEAD OF A NUMBER AND SPECIFYING ADDRESSES IN SYMBOLIC FORM (E.G. ADD X, TOTAL). IN 1954 FORTRAN, THE FIRST HIGHER LEVEL PROGRAMMING LANGUAGE, WAS INVENTED. THIS ALLOWED PROGRAMMERS TO SPECIFY CALCULATIONS BY ENTERING A FORMULA DIRECTLY (E.G. Y = X*2 + 5*X + 9). THE PROGRAM TEXT, OR SOURCE, WAS CONVERTED INTO MACHINE INSTRUCTIONS USING A SPECIAL PROGRAM CALLED A COMPILER. MANY OTHER LANGUAGES WERE DEVELOPED, INCLUDING ONES FOR COMMERCIAL PROGRAMMING, SUCH AS COBOL. PROGRAMS WERE MOSTLY STILL ENTERED USING PUNCH CARDS OR PAPER TAPE. (SEE COMPUTER PROGRAMMING IN THE PUNCH CARD ERA). BY THE LATE 1960S, DATA STORAGE DEVICES AND COMPUTER TERMINALS BECAME INEXPENSIVE ENOUGH SO PROGRAMS COULD BE CREATED BY TYPING DIRECTLY INTO THE COMPUTERS. TEXT EDITORS WERE DEVELOPED THAT ALLOWED CHANGES AND CORRECTIONS TO BE MADE MUCH MORE EASILY THAN WITH PUNCH CARDS.

AS TIME HAS PROGRESSED, COMPUTERS HAVE MADE GIANT LEAPS IN THE AREA OF PROCESSING POWER. THIS HAS BROUGHT ABOUT NEWER PROGRAMMING LANGUAGES THAT ARE MORE ABSTRACTED FROM THE UNDERLYING HARDWARE. ALTHOUGH THESE MORE ABSTRACTED LANGUAGES REQUIRE ADDITIONAL OVERHEAD, IN MOST CASES THE HUGE INCREASE IN SPEED OF MODERN COMPUTERS HAS BROUGHT ABOUT LITTLE PERFORMANCE DECREASE COMPARED TO EARLIER COUNTERPARTS. THE BENEFITS OF THESE MORE ABSTRACTED LANGUAGES IS THAT THEY ALLOW BOTH AN EASIER LEARNING CURVE FOR PEOPLE LESS FAMILIAR WITH THE OLDER LOWER-LEVEL PROGRAMMING LANGUAGES, AND THEY ALSO ALLOW A MORE EXPERIENCED PROGRAMMER TO DEVELOP SIMPLE APPLICATIONS QUICKLY. DESPITE THESE BENEFITS, LARGE COMPLICATED PROGRAMS, AND PROGRAMS THAT ARE MORE DEPENDENT ON SPEED STILL REQUIRE THE FASTER AND RELATIVELY LOWER-LEVEL LANGUAGES WITH

TODAY'S HARDWARE. (THE SAME CONCERNS WERE RAISED ABOUT THE ORIGINAL FORTRAN LANGUAGE.)

THROUGHOUT THE SECOND HALF OF THE TWENTIETH CENTURY, PROGRAMMING WAS AN ATTRACTIVE CAREER IN MOST DEVELOPED COUNTRIES. SOME FORMS OF PROGRAMMING HAVE BEEN INCREASINGLY SUBJECT TO OFFSHORE OUTSOURCING (IMPORTING SOFTWARE AND SERVICES FROM OTHER COUNTRIES, USUALLY AT A LOWER WAGE), MAKING PROGRAMMING CAREER DECISIONS IN DEVELOPED COUNTRIES MORE COMPLICATED, WHILE INCREASING ECONOMIC OPPORTUNITIES IN LESS DEVELOPED AREAS. IT IS UNCLEAR HOW FAR THIS TREND WILL CONTINUE AND HOW DEEPLY IT WILL IMPACT PROGRAMMER WAGES AND OPPORTUNITIES.

HAOFEI YAN HY222AP

****
****

Final note: it is actually easier to use quipqiup.com as in task 2. That also gives me the answer with no work at all.

Task 7

7) Hash function

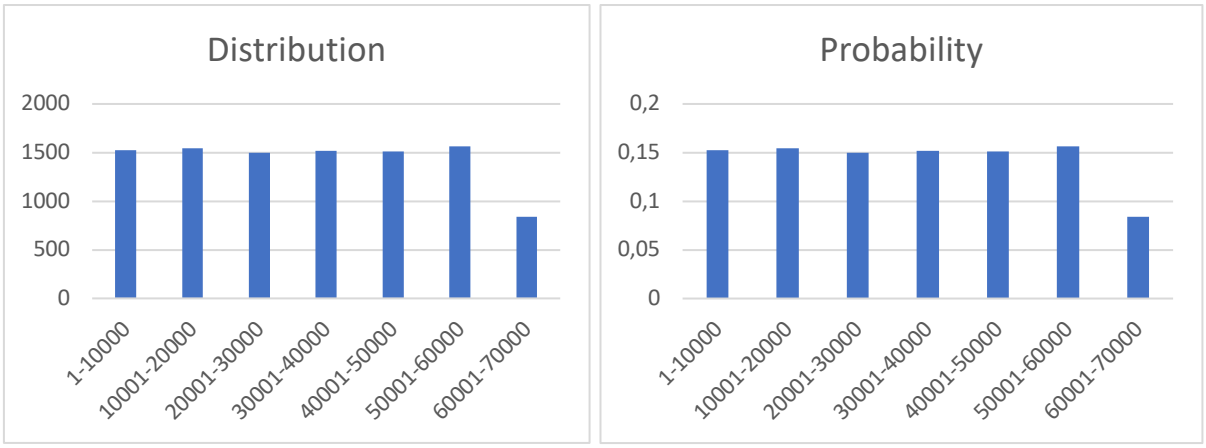
a) In this task I wrote the hash function that perform hashing on a byte array instead of a string. My hashing program takes the string, converts it into byte array and hashes it.

The hashing algorithm [15] takes the original vector (which is 101), XOR with the hash after shifted it 2 bits left, update it, then XOR with its value shift 3 bits right, update it, multiply with 31, update it and then XOR with the character at the first position of the string. The second turn is then using the current hash value, perform everything again but the vector is now the hash of the first character. After reaching the end of the string, this procedure starts hashing again with the vector is now the current hash value from the beginning of the string, and then repeat itself 11 times. The final step is to get the hash AND 0xFFFF to get a 16-bit string.

b) Analyzing of my hash function

At first glance, it seems like a good hash value since string 1 and 2 yield 12385 and 25280 in hash value. This makes this hash function potentially yields random enough results.

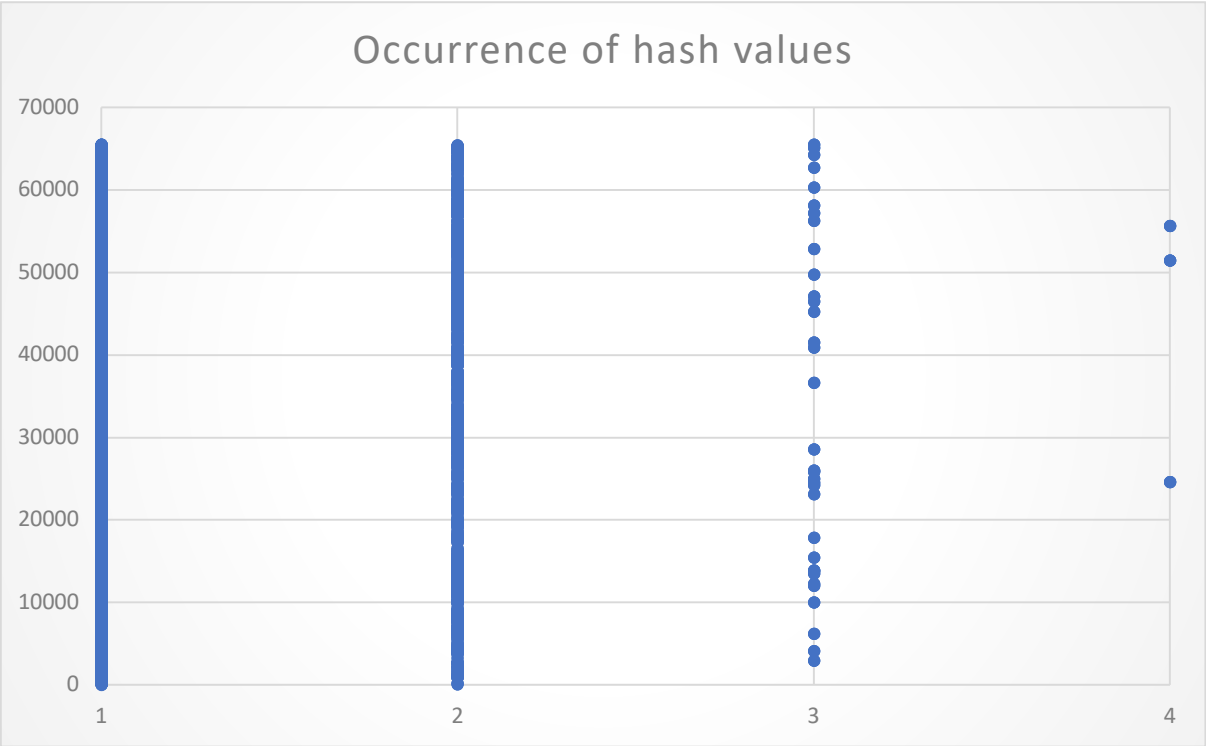
For analyzing it, I used all the number from 0 to 9999, convert it into String and hash them. Below is the figure showing the distribution of my hash values and probability of them.



This part shown that most of the distribution of this hash function is around 5-15% in each 10000 range, which is pretty good for a hash function, considering it is not a really good one.

For the uniformity and randomness, the chi-squared test is 289.5 (maybe).

c) For not so good part, my hash encryption has a lot of collision from my hash function. Just run the same statistic again but this time instead of grouping them, making them individual is a better idea to prove it.



Hashing 10000 strings above will give 1482 collisions. Also this algorithm is too easy to predict the output value. From a simple vertical line test, we can clearly see many collision from this hash function with just 10000 input strings (which is 2% of the maximum range possible of all 16-bit strings).

Bibliography

[1] C. P. Pfleeger, Security in Computing, Pearson Education, Inc, 2015.

[2] Wikipedia, "Hash function," [Online]. Available: https://en.wikipedia.org/wiki/Hash_function. [Accessed 08 Feb 2020].

[3] Wikipedia, "Data compression," [Online]. Available: https://en.wikipedia.org/wiki/Data_compression. [Accessed 08 Feb 2020].

[4] J. Granneman, "How does steganography work and does it threaten enterprise data?," Aug 2013. [Online]. Available: https://searchsecurity.techtarget.com/answer/How-does-steganography-work-and-does-it-threaten-enterprise-data.

[5] J. Stanley, "Image Steganography," [Online]. Available: https://incoherency.co.uk/image-steganography/. [Accessed 02 Feb 2020].

[6] "Read bytes and display their hexadecimal values. : FileInputStream « File Input Output « Java," java2s.com, [Online]. Available: http://www.java2s.com/Code/Java/File-Input-Output/Readbytesanddisplaytheirhexadecimalvalues.htm. [Accessed 09 Feb 2020].

[7] aragaer, "c - Format specifier %02x - Stack Overflow," Stack Exchange Inc, 26 Aug 2013. [Online]. Available: https://stackoverflow.com/questions/18438946/format-specifier-02x/18438992. [Accessed 08 Feb 2020].

[8] RapidTables.com, "Binary to Text Converter | Binary Translator," [Online]. Available: https://www.rapidtables.com/convert/number/binary-to-ascii.html. [Accessed 08 Feb 2020].

[9] A. Stanoyevitch, Introduction to Cryptography with Mathematical Foundations and Computer Implementations (Discrete Mathematics and Its Applications), Taylor & Francis Ltd, 2010.

[10] E. Olson, "quipqiup - cryptoquip and cryptogram solver," [Online]. Available: https://quipqiup.com/. [Accessed 09 Feb 2020].

[11] M. Herlitzius, "How to use BufferedReader in Java," Stack Exchange Inc, 28 Apr 2013. [Online]. Available: https://stackoverflow.com/questions/16265693/how-to-use-bufferedreader-in-java. [Accessed 08 Feb 2020].

[12] W. Shakespeare, "Hamlet," [Online]. Available: http://shakespeare.mit.edu/hamlet/hamlet.3.1.html. [Accessed 08 Feb 2020].

[13] H. Yan, "MyMoodle," [Online]. Available: https://mymoodle.lnu.se. [Accessed 08 Feb 2020].

[14] dCode, "Monoalphabetic Substitution Cipher - Cryptogram Decoder, Solver," [Online]. Available: https://www.dcode.fr/monoalphabetic-substitution. [Accessed 09 Feb 2020].

[15] jonathanasdf, "Good Hash Function for Strings," 12 Apr 2010. [Online]. Available: https://stackoverflow.com/questions/2624192/good-hash-function-for-strings. [Accessed 02 Feb 2020].