DAT510-1 20H Network security and vulnerability. Assignment 1. Cryptanalysis of primitive ciphers

Asahi Cantu Moreno (student id: 253964)

September 15, 2020

Abstract

The study of encryption algorithms, techniques and potential vulnerability detection are very important for the understanding and detection of potential risks related to the information transmission over insecure channels. In this assignment a study of the cryptanalysis, the statistical techniques to detect encryption algorithm vulnerabilities and the potential threats of wrong or obsolete algorithms is performed in a two-part code development which involves:

1. **Part I. Polyalphabetic ciphers.** Divided in two different tasks where two ciphered text paragraphs are given to be deciphered by statistical techniques and brute force attacks. Text was ciphered with a poly-alphabetic substitution cipher. Techniques such as word frequency analysis and n-gram frequency analysis are performed to start retrieving the core of the text and eventually find the potential encryption key by knowing the initial parameters of the encryption algorithm. For the second task another ciphered text is analyzed and several keys found by brute-forcing and different decipher techniques which basically consist using the same tools from the first task and special text comparison to find the optimal encryption algorithm.
2. **Part II. Simplified DES.** The development of an SDES and triple SDES algorithm provides the insights and abilities to understand how encryption algorithms work under real implementations. Once developed it some keys, plain text and ciphered texts are required to be found, where also special statistical analysis is performed to decipher ciphered messages in order to demonstrate the vulnerability of such encryption mechanism and how these techniques can be easily bypassed by malicious third-parties. Finally the development of a mini web server is performed to analyse the potential vulnerabilities for the encryption algorithm on the back end.

For each element a deep explanation and case study is provided and finally the statistical results of the developed code is presented to provide perspective on the encryption mechanisms studied for this assignment. The developed code and implementations for each part of the assignment can be found together in the contents of this document repository.

1 Part I. Polyalphabetic ciphers.

Polyaphabetic ciphers are and extension of simple monoalphabetic ciphers where monoalphabetic substitution can be performed by any given letter of the English alphabet (26 letters). The following ciphered text was provided and ciphered with an implementation of the Vigenere Cipher[1]:

> BQZRMQ KLBOXE WCCEFL DKRYYL BVEHIZ NYJQEE BDYFJO PTLOEM EHOMIC UYHHTS GKNJFG EHIMK NIHCTI HVRIHA RSMGQT RQCSXX CSWTNK PTMNSW AMXVCY WEOGSR FFUEEB DKQLQZ WRKUCO FTPLOT GOJZRI XEPZSE ISXTCT WZRMXI RIHALE SPRFAE FVYORI HNITRG PUHITM CFCDLA HIBKLH RCDIMT WQWTOR DJCNDY YWMJCN HDUWOF DPUPNG BANULZ NGYPQU LEUXOV FFDCEE YHQUXO YOXQUO DDCVIR RPJCAT RAQVFS AWMJCN HTSOXQ UODDAG BANURR REZJGD VJSXOO MSDNIT RGPUHN HRSSSF VFSINH MSGPCM ZJCSLY GEWGQT DREASV FPXEAR IMLPZW EHQGMG WSEIXE GQKPRM XIBFWL IPCHYM OTNXYV FFDCEE YHASBA TEXCJZ VTSGBA NUDYAP IUGTLD WLKVRI HWACZG PTRYCE VNQCUP AOSPEU KPCSNG RIHLRI KUMGFC YTDQES DAHCKP BDUJPX KPYMBD IWDQEF WSEVKT CDDWLI NEPZSE OPYIW

It can be assumed that the cipher technique used only lowercase letters as key transformation and the spaces were removed during the encryption process. It was finally split into 8-char segments and transformed to uppercase for better readability. Therefore three basic factors are to be assumed:

1. The text was ciphered using only lower characters from the English Alphabet
2. The spaces for the encoded text were removed
3. There is no special char other than English alphabet letter present in the plain text.
4. A key no longer than 10 characters was used to encrypt the message

1.1 Ciphered text statistical analysis

Before any statistical analysis can be performed it is necessary to highlight the importance of statistical English alphabet such as [LettterFrequency(2020)]:

- English letter frequency distribution

  e t a o i n s r h l d c u m f p g w y b v k x j q z

- 2-gram frequency distribution

[1]The best known, and one of the simplest, polyalphabetic ciphers. The set of related monoalphabetic substitution rules consists of the 26 Caesar ciphers with shifts of 0 through 25. Each cipher is denoted by a key letter, which is the cipher text letter that substitutes for the plain text letter a. [Stallings(2017b)]

th he an in er on re ed nd ha at en es of nt ea ti to io le is ou ar as de rt ve

- 3-gram frequency distribution

  the and tha ent ion tio for nde has nce tis oft men

It is very important for a cryptanalysis to start with a strong basis and assumptions about the alphabet that might have been used and the language since it can give big insights for the pattern detection of ciphered words being together representing either a letter, a 2-gram and even a 3-gram.

With this pattern in mind it is feasible to perform a statistical analysis on based on the estimated frequency some words or n-grams can appear in a sentence such an in the following example:

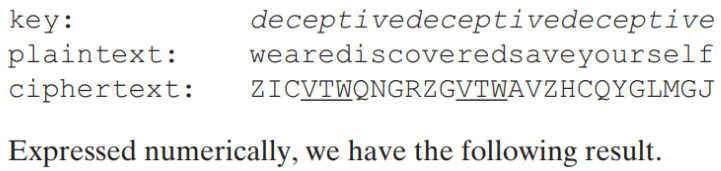

Figure 1: Cryptanalysis of a simple sentence showing the pattern repetition of ciphered words for a predefined phrase [Stallings(2017b)]

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Bi performing a statistical analysis with python code on the ciphered text it was possible to find the following patterns:

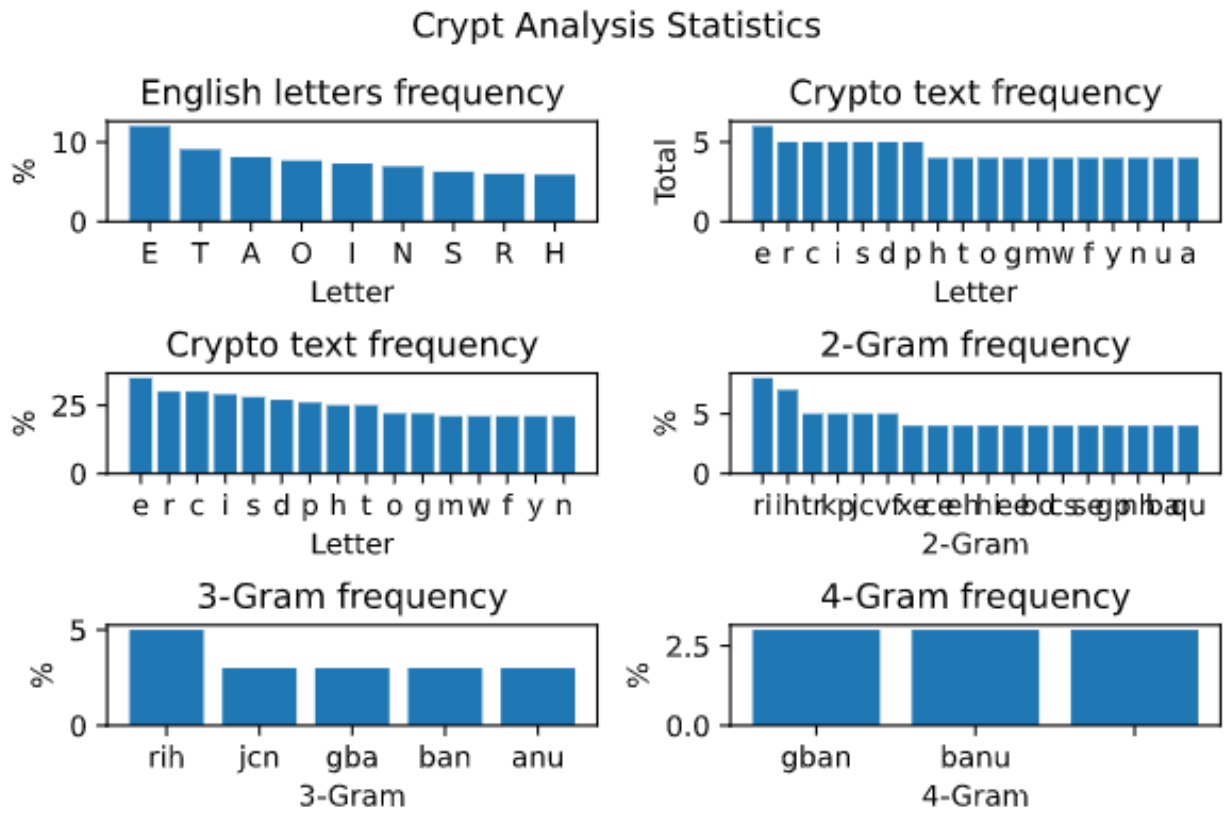

Figure 2: Statistical word, 2-gram and 3-gram distribution on the ciphered text

And then with such assumptions it was possible to create a dictionary with the candidate patterns set in the ciphered text:

Englsh alphabet with its indexes

Letter	**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**
Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

English 2-Gram Frequency	**TH HE IN ER AN RE**
Cipher	RI IH TR KP JC VF
Candidate Key	YB BD LE GY JP EB

English 3-Gram Frequency	**THE AND THA ENT ING ION TIO FOR**
Cipher	RIH JCN GBA BAN ANU
Candidate Key	YBD JPK NUA ZNU SAO

Figure 3: Potential 2-gram and 3-gram pattern detected in the ciphered text.

Taking into account the 2-gram 'TH'(Which is a candidate for ciphered pattern 'RI') and 3-gram 'THE' (which is also a candidate for ciphered pattern 'RIH') the candidate fragment of the key is found to be **'YBD'**.

From this approach and trying to find a repetitive pattern with the provided keys, a special fragment of the ciphered text (which contained the pattern 'RI') was chosen to perform a brute-force key analysis. The created algorithm intended to find first the key length by increasing the key size and trying to spot potential English fragments after running up to a 10 length key.

With an 8-length key 'YBDaaaaa' very plausible phrases were found.

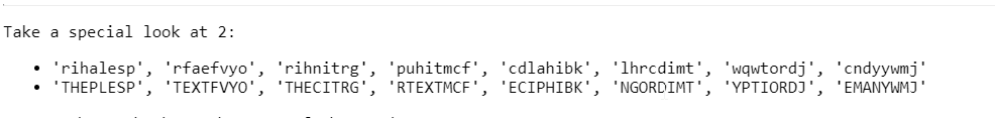

Figure 4: Final insight for a potential candidate pattern with key = 8-char length with repetitive english words.

For the rest of the analysis some candidate composed words were guessed to be present in the phrase and then the brute force algorithm was run several times to confirm the hypothesis whether a special key-char could be present and chosen on purpose as well to reduce the brute-force time consumption.

After several trials the key ''**bdlaekcy**'' was successfully found with th decipher text from the original message:

> AN ORIGINAL MESSAGE IS KNOWN AS THE PLAIN TEXT WHILE THE CODED MESSAGE IS CALLED THE CIPHERTEXT. THE PROCESS OF CONVERTING FROM PLAIN TEXT TO CIPHER TEXT IS KNOWN AS ENCIPHERING OR ENCRYPTION RESTORING. THE PLAIN TEXT FROM THE CIPHER TEXT IS DECIPHERING OR DECRYPTION. THE MANY SCHEMES USED FOR ENCRYPTION CONSTITUTE THE AREA OF STUDY KNOWN AS CRYPTOGRAPHY SUCH A SCHEME IS KNOWN AS A CRYPTOGRAPHIC SYSTEM OR A CIPHER TECHNIQUES USED FOR DECIPHERING A MESSAGE WITHOUT ANY KNOWLEDGE OF THE ENCIPHERING DETAILS FALLIN TO THE AREA OF CRYPTANALYSIS. CRYPTANALYS IS IS WHAT THE LAY PERSON CALLS BREAKING THE CODE. THE AREAS OF CRYPTOGRAPHY AND CRYPTANALYSIS TOGETHER ARE CALLED CRYPTOLOGY.

It is possible to check and confirm this is the key by running a polyalphabetic algorithm with the provided key and trying to match it with he initial ciphered pattern.

Finally it was required for the algorithm to encrypt the same text with an increasing key length and measure the time that the algorithm took to encrypt the phrase with longer keys. The results were very interesting:

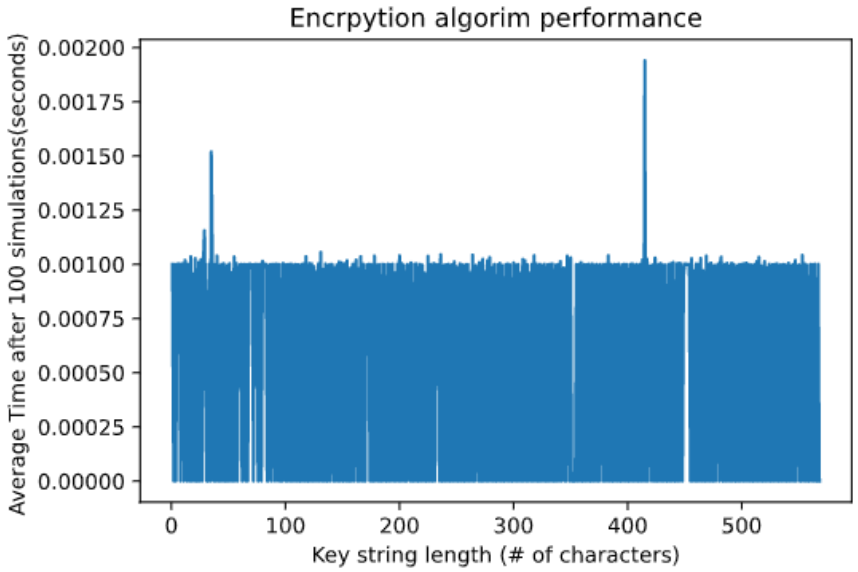

Figure 5: Performance measure of time taken to encrypt the same text with longer keys.

It was initially thought that the time would increase as the key-length was bigger, however, regardless the key size the algorithm took approximately the same amount of time.

1.2 Complex Ciphered text statistical analysis

Another ciphered text was provided. Same algorithm and key used to decipher the previous message were employed in the original encryption method.It was necessary to find the additional pattern done to the algorithm that allowed the decryption of the given ciphered sentence:

> BQZRMQ KLAYAV AYITET EOFGWT EALRRD HNIFML BIHHQY XXEXYV LPHFLW UOJILE GSDLKH BZGCTA LHKAIZ BIOIGK SZXLZS UTCPZW JHNPUS MSDITN OSKSJI EOKVIL BKMSZB XZOEHA KTAWXP WLUEJM AIWGLR TZLVHZ SATVQI HZWAXX ZXDCIV TMLBIQ RWZMLB VNGVQK AIZBXZ HVVMMA MJLRIW GKITZL VHZRRV YCBTVM FVOIYE FSKGKJ AVWHUV BUHZSA EFLHMQ HHVSGZ XIKYTS YZXUUC KBTOGU VABLDP BGJCGF NLIIYA HJFWGG PSCPVA ZEASME MLGOYR CGFXVG EJTTTW TSAAIL QFKEEP CPULXW WZRLVI VVYUMS MSILRI IBLWJI TKWUXZ GUZEJG DUCQEE QEOBTP SIHTGW UALVMA ILTAEZ TFLDPE IVEGYH PLZRTC YJVYGX ABFNPQ XLCEYA RGIFCC WHBGIF WSYLBZ MDWFPX KZSYCY APJTFR CKTYYU YICYLR ZALETS DWHMGR PTTGUW CGFNTB JTRNWR AADNPQ XLTBGP RZMJTF KGTSPV DTVAPE ZPRIP

It was very interesing to notice (not after decrypting unfortunately) that the first 8 characters of the decrypted text matched the first character from the previous deciphered message.

> **anorigin**zvpvwogvdqtobwuvdxarntfphcblxyfj...

It was then a matter to figure out how the key was used to encrypt the rest of the message. Finally and after several trials it was found that for this message the first 8 characters were encoded with the initial key, but the second segment could be decoded by using the previous encoded pattern as a key.

The final deciphered text turned out to be the same as the first one.

2 Part II. Simplified DES.

In this section it was required to implement an SDES and triple-DES algorithm to find the keys and ciphered text combinations.

2.1 SDES Tests

Raw Key	PlainText	CipherText
0000000000	10101010	00010001
0000011111	11111100	10011101
0010011111	10100101	10010000
0010011111	01010101	11001100
1111111111	11111111	00001111
1111111111	01100001	01000011
1000101110	00111000	00011100
1000101110	00001100	11000010

Figure 6: Test performed with the implemented SDES algorithm.

2.2 Triple SDES Tests

Raw Key 1	Raw Key 2	PlainText	CipherText
1000101110	0110101110	11010111	10111001
1000101110	0110101110	10101010	11100100
1111111111	1111111111	00000000	11101011
0000000000	0000000000	01010010	10000000
1000010110	0110101110	11111100	11100110
1011101111	0110101110	01001111	01010000
1111111111	1111111111	10101010	00000100
0000000000	0000000000	00000000	11110000

Figure 7: Test performed with the implemented Triple SDES algorithm.

2.3 SDES Text Cracking

CTX1.txt

01000111000000010100000011001101110010110000000101110100000000010110111001010111010
10111011011100100011100000001010001111011010010011111000100001000111011011100100 11
00101011111001011101101110011011101011101001001111101011110000100101001010100010000
10011111100110110010111010011110011001000000001010101110110111010010000010011111010
11110100011110101111011101000111010000000001010011000000000101101110101110101000100
00100011101101110010011001010111110010111000000011000100010010000

CTX2.txt

00000001101001110011001011000110011001001010011111010111010011100111000111010001 1
01001001110000000001101001110000000110011001101000011101101000000001100111001110 11
11011111100010010010011100100111001001100110100001011111101010000010110011111011 0
01000011000110001001001010000100100011101001110111010010011100010000011010000101 11
11100000000101111110110101111101011110100111111011111010011110011100100110011101101
00000000110011100111011110111111000100100101001111101101001000001

Figure 8: Provide texts to be deciphered with SDES and triple SDED respectively.

A ciphered text was provided and key required to be found given the same conditions as in part 1. To find he candidate key the same statistical method was applied with he intention to find all the potential ciphered segments that would math the character 'e' (since 'e' is the most common letter in the alphabet. The most frequent token 8-bit segments were chosen to be deciphered with an increasing key and then try to find if by deciphering such segment the result was and 'e' or 'E'. If that happens then decipher the rest of the message and check for non-alphabet characters, which in such case would be assumed to be an invalid deciphered text and discarded.

The increasing key would go from '0000000000' to '1111111111' (1024 permutations or 2^{10}). Also, first 5 most frequent 8-bit patterns were chosen to be run against the brute-fore attack, the maximum amount of instructions to run would be $1024 * 5 = 5120$ cycles to find the potential deciphered text, something meaningless in terms of resource consumption. The found key is **1111101110** and deciphered message

simplifieddesisnotsecureenoughtoprovideyousufficientsecurity

.

```
Candidate key 1111101010  => simplifieddesisnotsecureenoughtoprovideyousufficientsecurity
Elapsed time 0.3178083896636963


Key found for txt1 : 1111101110 decrypted text:
    simplifieddesisnotsecureenoughtoprovideyousufficientsecurity
```

Figure 9: Deciphered text and time consumption for SDES message deciphering.

2.4 Triple SDES Text Cracking

A new ciphered text message was provided and asked to find the ciphering keys. Same procedure is performed as in previous task with the difference that triple SDES uses two

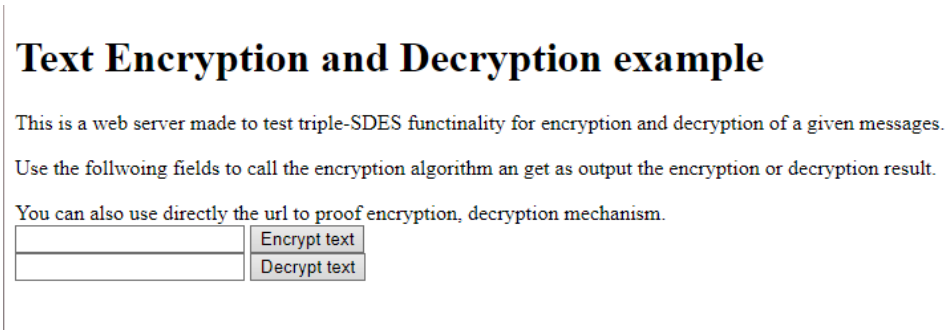

Figure 11: A demo of the server created with the provided keys and mounted as a live tool to cipher and decipher text.

10-bit key segments, so now the maximum amount of instructions to run would be $1024 * 1024 * 5 = 26214400$ cycles to find the potential deciphered text. This is 5120 more trials required to find candidate keys. The found key is **k1=1111101010, k2=0101011111** and deciphered message

simplifieddesisnotsecureenoughtoprovideyousufficientsecurity

.

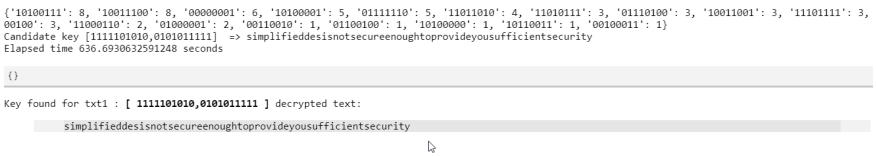

Figure 10: Deciphered text and time consumption for Triple SDES message deciphering.

2.5 Server implementation for messaging ciphering/deciphering

A mini web server was implemented with Flask-Python. Built SDES and Triple SDES algorithms were implemented to decrypt and encrypt a given message via a web page. The keys were provided (k1 ='1000101110', k2='0110101110') and store in raw text on the back-end side of the server, which means that a malicious user would not be able to spot the key unless he has access to the source code of the solution. It has also been highlighted from previous deciphering mechanisms that not much computational resources would be required to eventually crack and find the keys that are working in the mini web site.

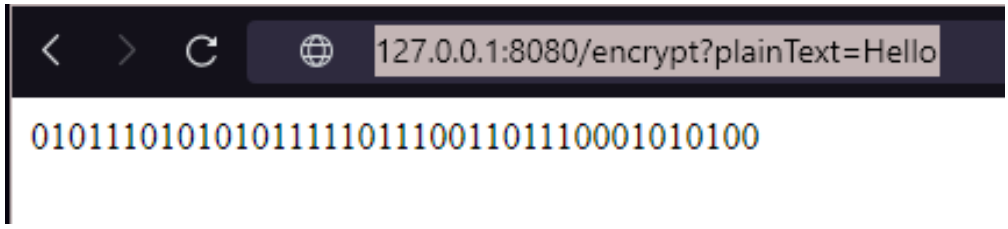

Figure 12: Server encryption example of the work 'Hello', obtaining the key '01011101010101111101110011011100010101000'.

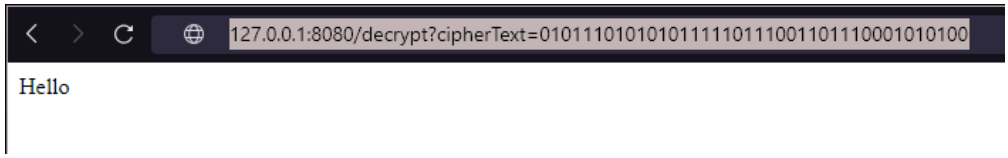

Figure 13: Server decryption example of the bit-word '01011101010101111101110011011100010101000', obtaining the word 'Hello'.

3 Conclusion

For this assignment different topics were analyzed and researched to produce insightful information and important understanding for the encryption algorithms for:

1. **Polyalphabetic encryption algorithms**. It was possible to understand the way polyalphabetic ciphers work and the potential vulnerabilities for its simplicity by word substitution. It was demonstrated that even if the text is encoded in a different language, it would just be necessary to understand the word frequency distribution of such language to start with the brute-force attack.

2. **SDES Algorithm**. Simple DES algorithm allowed a deep visualization, interpretation and understanding of encryption algorithms and schemes. Extremely important and useful being able to read and understand the technical SDES Sheet [Stallings(2017a)] and then translate it into code. Perform proper tests and finally give it a proper structure to make it part of a useful package.

3. **Triple DES Algorithm**. Very interesting to realize how by just repeating SDES algorithm and using different keys, the ciphering process can be strengthen 1000 times, making the deciphering mechanism even more complicated.

4. **Server encryption and test**. After the implementation of a SDES algorithm it was possible to take it into a single module and use it to create a web server. Very interesting to realize that depending on the internet protocol (http or https), the submit mechanism (GET or POST) and the way the source code and keys are stored on the server side play an important role for secure systems. Being naive in its configuration, protection and proper storage might end up being very critical for any organization.

5. **Statistical analysis, Cryptanalysis and brute-force algorithms**. Trying to find the different ways to analyze ciphered texts and perform statistical analysis involved

very interesting approaches and knowledge acquired to create all the code and implementations to perform the cracking instructions and server mounting.

All results have been packed and submitted into a github repository where all the information and research can be found.

A very important lesson learned from this analysis is the realization that no system is 100% secure, it is just matter of time and resources before one can access to relevant information, however, time and resources play a key role on it, so it is probable although unfeasible to break encryption algorithms.

References

[LettterFrequency(2020)] LettterFrequency. Eta: Letterfrequency.org, 2020. URL http://letterfrequency.org.

[Stallings(2017a)] William Stallings. *Cryptography And Network Security Principles And Practice*, volume 1, chapter Apendix G. Simplified DES. Pearson, Edinburgh Gate Harlow Essex CM20 2JE England, 7 2017a.

[Stallings(2017b)] William Stallings. *Cryptography And Network Security Principles And Practice*, volume 1, chapter 3.2, page 103. Pearson, Edinburgh Gate Harlow Essex CM20 2JE England, 7 2017b.