# **CSE 350: Network Security Assignment 1 Report**

Utkrisht Sikka (2019215)

Tanya Gupta (2019119)

# Project Number - 1

# **Encryption**

We use the poly-alpha substitution algorithm for encryption. For encryption, we first call encrypt\_wrapper() that takes the key and hashed plaintext of the form (p1,p2) as its parameters. Then we make calls to the function encrypt() to encrypt the plaintext p1 and hash value p2 separately. The encrypted plaintext and hash value are concatenated to form the final ciphertext.

encrypt() takes two arguments : key and the string to encrypt(suppose str). First, all characters of str are mapped to numeric values with mapping as given below.

| **character** | **number** |
| --- | --- |
| a | 0 |
| b | 1 |
| … | … |
| z | 25 |

The key is replicated till the new key’s length is equal to length of str. Elongated key’s characters are mapped to numbers. Then, each numeric value of str is added to the corresponding numeric value of elongated key modulo 26.

Resulting numbers are mapped back to characters to obtain final cipher text.

# **Decryption**

For decryption, we first call decrypt\_wrapper() that takes the key and decrypts a ciphertext of the form (c1,c2) to (p,Hash(p)). For this, we make calls to the function decrypt() to decrypt the ciphertext c1 and encrypted hash value c2 separately. The decrypted plaintext and hash value are concatenated to the form (p,Hash(p)) and returned.

decrypt() takes two arguments : key and the string to be decrypted(suppose str). First, all characters of str are mapped to numeric values with mapping as given below.

| **character** | **number** |
| --- | --- |
| a | 0 |
| b | 1 |
| … | … |
| z | 25 |

The key is replicated till the new key’s length is equal to length of str. Elongated key’s characters are mapped to numbers. Then, from each numeric value of str we subtract the corresponding numeric value of elongated key modulo 26.

Resulting numbers are mapped back to characters to obtain final decrypted text and it is returned.

# **Text Recognizability**

# After getting a plaintext consisting of characters from a-z, we convert it into a tuple format string consisting of the original plain text and a hash corresponding to it. This is done to make it recognizable. The hash is calculated via the function **Hash**, which takes in the plaintext as an input and returns the recognizable plaintext consisting of the given plaintext and the hash value. The function **recognizable** takes in a string and returns true if it is in the recognizable format, i.e. of the form (string, Hash(string)). It basically splits the input to segregate the supposed original text and the corresponding hash, and then makes a call to the function Hash with the original text as an input and returns true if its output matches the corresponding hash extracted earlier. The hashing function used is the one described in Figure 11.5, Chapter 11 of Stallings book (7 th edition). The block size is of length 8, and the char ‘a’ has been padded to the original plaintext to make its length divisible by the block length. The characters a-z have been mapped to the integers 0 to 25, and then the respective integers from each of the blocks have been added together followed by a modulo 26 operation to help map the summed value back to a character.

# **Brute-Force Attack**

We enumerate all possible keys of length 4(made of lowercase english letters). For each key, we decrypt each of the 5 cipher texts one by one. Since each ciphertext is of the form (c1, c2), we decrypt c1 and c2 and obtain p1 and p2 respectively. Then the tuple (p1,p2) is checked for the property π using recognizable(). If a key decrypts all cipher texts such that the decrypted (p1,p2) string is recognizable, then that key is stored in a separate list ‘possible\_keys’. In the end, all such keys are printed.

# **Sample Input and Output**

Eg1:

* Plain text without hash: shutterislandisanamericanpsychologicalthriller
* After conversion to a recognizable plaintext : (shutterislandisanamericanpsychologicalthriller,pvrvtdsa)
* The resulting ciphertext after encryption with the key ‘hope’: (zvjxasgmzzprkwheuobiywreudhcjvdpvuxghzilywaplf,wjgzarhe)

Eg2:

* Plain text without hash: thefirstquizofcollaborativefilteringisnextweek
* After conversion to a recognizable plaintext : (thefirstquizofcollaborativefilteringisnextweek,qizueaai)
* The resulting ciphertext after encryption with the key ‘hope’: (avtjpfhxxixdvtrsszpfvfpxpjtjpziiywckpgciehlily,xwoylopm)

Eg3:

* Plain text without hash: iwanttofinishmyworkassoonaspossiblepleasehelp
* After conversion to a recognizable plaintext : (iwanttofinishmyworkassoonaspossiblepleasehelp,wssugtsp)
* The resulting ciphertext after encryption with the key ‘hope’: (pkprahdjpbxwoanavfzezgdsuohtvghmizttsspwlvtpw,dghynhht)

Eg4:

* Plain text without hash: whyisthemessfoodsobadpleasegivethemcookingclasses
* After conversion to a recognizable plaintext : (whyisthemessfoodsobadpleasegivethemcookingclasses,mbjtwxmq)
* The resulting ciphertext after encryption with the key ‘hope’: (dvnmzhwitshwmcdhzcqekdaihgtkpjtxosbgvczmuurphghiz,tpyxdlbu)

Eg5:

* Plain text without hash: idontknowwhatthefifthplaintextshouldbeorwhatitshouldnotbe
* After conversion to a recognizable plaintext : (idontknowwhatthefifthplaintextshouldbeorwhatitshouldnotbe,tppjmwwy)
* The resulting ciphertext after encryption with the key ‘hope’: (prdraycsdkweahwimwuxodaepbiiehhlviahisdvdvpxphhlviahucifl,adentklc)