Network Security

Assignment 1 - Project 2

B N Jain

Submitted by:

Sarvagya Kaushik - 2021350

Kunal Sharma - 2021331

The programming language used in the assignment is C++. In our code we have in total **7 functions** including the main() function. Below is the explanation of each function one by one.

1. string hashfunction(const string &data) : This function takes **1 argument** named data which is the plaintext and the return type of the function is string. It returns the hash of the plaintext.

We have used the **CRC32** (Cyclic Redundancy Check) algorithm to generate the hash. This function repeats through every byte of the input string before setting a CRC variable's underlying worth to 0xFFFFFFFF. For each byte, it performs a **bitwise XOR** operation and updates the CRC using a polynomial (0xEDB88320) in a loop. After processing all bytes, the function XORs the CRC with 0xFFFFFFFF and converts the resulting CRC value into an 8-character string representation using base-26 encoding, where each digit represents a letter from 'a' to 'z'. The function guarantees the result string is precisely 8 characters in length, padding with 'a' if necessary or shortening it if longer.

2. string encrypt(string plaintext, string key) : This function takes **2 arguments** named plaintext and key. The return type of the function is string. It returns the encrypted plaintext which is called a ciphertext.

We have used a **transposition cipher** to encrypt the plaintext. In this function first we added the hash to the end of plaintext using the function hashfunction(). The function then determines the number of rows needed for the transposition based on the length of the key and the size of the extended plaintext. The function creates a **2D vector v** to store the transposed plaintext. It iterates through the plaintext and **fills the vector row-wise**. If the vector is not fully filled then we have added the **hyphens '-'** at the end of the vector. Subsequently, it constructs the ciphertext by rearranging the characters in the vector based on the key's order. At last the final encrypted string is returned.

3. string decrypt(string ciphertext, string key) : This function takes **2 arguments** named ciphertext and key. The return type of the function is string and It returns the decrypted plaintext.

In this function, the **reverse of the transposition cipher** is implemented to decrypt the ciphertext. In light of the length of the key and the ciphertext, the capability decides the number of columns that are required for decryption. It then constructs a 2D vector v and populates it by rearranging the characters in the ciphertext based on the order specified by the key. The final step involves reconstructing the original plaintext by **reading the vector column-wise**. At last, the decrypted plaintext is returned.

4. string removechar(string& word,char& ch) : This function takes **2 arguments** named word and ch. The return type of the function is string and it returns the updated string after removing all the occurrences of the character ch in the string word.

5. void string_permutation(vector<string> cipher_strings, string &orig, string &perm, string &ans) : This function takes **4 arguments** named cipher_strings, orig, perm and ans. The return type of the function is void.

This function **recursively generates permutations** of a given random key and checks if any permutation satisfies the condition based on the results of the decrypt and hashfunction functions. The function repeatedly removes a character from the key, appends it to the permutation string, and recursively calls itself with the updated keys. When the key becomes empty, the function decrypts each cipher string using the generated permutation and **removes a specific character '-'** from the resulting plaintext using the function removechar(). It then checks if the hash of the modified plaintext matches the last 8 characters of the plaintext. If this condition holds for all the 5 cipher strings, the permutation is considered a valid solution, and it is stored in the ans variable.

6. string bruteforceattack(vector cipher_strings, int key_size) : This function takes **2 arguments** named cipher_strings and key_size. The return type of the function is string. It returns the final key we get after a brute force attack.

This function does a brute-force attack on a set of cipher strings **to discover the encryption key** used to encrypt the plaintext. In the function first we initialized a string key with a sequential numeric pattern based on the key size which is being incremented in a for loop from range 1 to 9. It then calls the string_permutation function, which recursively generates permutations of the key and decrypts the cipher strings using each permutation. If a permutation is found that satisfies the specified hash conditions for all cipher strings, it is **stored in the ans variable.** At last, the function returns the discovered key, representing a successful outcome of the brute-force attack.

7. int main() : This is the main function of the code. We have **two text files**, one containing the key and the other file containing the 5 input plaintexts. In this function we are reading from the input and the key from these files and key is stored in the string variable named key and the plaintexts are stored in a vector named input_strings. Then we are calling the encryption function to encrypt the plaintexts and then we are doing a brute force attack to find the key used to encrypt the plaintext. At last the **key is printed on the console**.

sample inputs and outputs

Inputs: These are taken from the two files named input.txt and key.txt

input.txt:

```
sarvagyakaushik
kunalsharma
hellohowareyou
iamfinethankyou
letsgotopark
```

key.txt:

```
385164927
```

Output:

```
Key found after Brute force attack: 385164927
```