

# Cyber Security Toolkit

---

## Introduction:

A **Cyber Security Tool** is a must for Cyber Security and Privacy of a business or individual. Cyber Security is the method that is *used to protect the network, system, or applications from the cyber-attacks*. It is used to avoid unauthorized data access, data loss, and identity theft.

So, being cyber security students, we decided to create a toolkit which contains some necessary tools for securing confidential information and accounts.

## Programming Language and Tool Used:

- **Language:** C++ (Concepts of Loops, Conditions, Arrays, Functions)
- **IDE:** Dev C++

## This toolkit contains 3 tools:

1. **Cryptography Tool**
2. **Password Analyzer**
3. **Password Generator**

Which are accessible by their respective number:

```
=====
====>>> Cyber Security Toolkit <<<====
=====

1) Cryptography Tool
2) Password Analyzer
3) Password Generator
4) Exit
=>
```

To understand this cryptography tool, we have to understand what **cryptography** means. Cryptography may be defined as an approach used to transform the data into a form that the client

can only understand to whom the data is sent. In simple terms, *it's the way to establish secure communication among peers.*

By entering “1”, we will enter into Cryptography Tool:

```
=====
====>>> Cryptography Tool <<<====
=====
What do you want to perform?
1) Encryption
2) Decryption
=>
```

This tool has two parts:

1. Encryption
2. Decryption

## Encryption:

Enter “1” for performing encryption, and then it will ask the user to enter his/her plain text / message.

```
=====
====>>> Cryptography Tool <<<====
=====
What do you want to perform?
1) Encryption
2) Decryption
=>1
Enter the Plain Text:
```

- When user enters his/her message, which he/she wants to encrypt, then it will ask for the secret key (Any remember able keyword).
- The secret key should be alphabetic and would not contain any number.

```

=====
====>>> Cryptography Tool <<<====
=====
What do you want to perform?
1) Encryption
2) Decryption
=>1
Enter the Plain Text:
stay home stay safe
Enter the Alphabetic Secret key: ah2nz
The Secret key shouldn't contain any number <=
Enter the Alphabetic Secret key:

```

- It would not accept any secret key having number in it.
- When it receives a *correct secret key*, it would generate the encrypted text of his/her message.

```

=====
====>>> Cryptography Tool <<<====
=====
What do you want to perform?
1) Encryption
2) Decryption
=>1
Enter the Plain Text:
stay home stay safe
Enter the Alphabetic Secret key: ah2nz
The Secret key shouldn't contain any number <=
Enter the Alphabetic Secret key: ahnnz

Encrypted Text: tbgf abig vcas ghom
Do you want to continue this tool? (y/n):

```

In this case,

**Secret Message:** “stay home stay safe”

**Secret Key:** “ahnnz”

**Encrypted Text:** “tbgf abig vcas ghom”

### Decryption:

Enter “2” for performing decryption, and then it will ask the user to input his/her encrypted message, that he/she encrypted using this tool.

```
=====
====>>> Cryptography Tool <<<====
=====
What do you want to perform?
1) Encryption
2) Decryption
=>2
Enter the Encrypted Text:
```

```
=====
====>>> Cryptography Tool <<<====
=====
What do you want to perform?
1) Encryption
2) Decryption
=>2
Enter the Encrypted Text:
tbgf abig vcas ghom
Enter the Secret key:
```

After entering encrypted message, the program will ask for the secret key that the user used at the time of encryption.

If the user enters wrong secret key, the plain message could also be incorrect.

If the user enters his/her correct secret, then the tool will generate the original decrypted plain message.

```
=====
====>>> Cryptography Tool <<<====
=====
What do you want to perform?
1) Encryption
2) Decryption
=>2
Enter the Encrypted Text:
tbgf abig vcas ghom
Enter the Secret key: ahnnz

Plain Text: stay home stay safe
Do you want to continue this tool? (y/n):
```

## Working Algorithm:

Actually, the program, after taking plain text message from the user, counts its number of words and stores these words as items in an array. Then it simply reverses the array.

- For counting number of words:

```
string plain_text;
plain_text.clear();
cout<<"Enter the Plain Text: "<<endl;
cin.ignore();
getline(cin, plain_text);
int len = 1;
    for (int i = 0; i<(int) plain_text.length() ; i++){
        plain_text[i] = tolower(plain_text[i]);
        if (plain_text[i]==' '){
            len++;
        }
    }
```

- For storing words in an array;

```
string word;
word.clear();
string arr[len];
int j = 0;
    for (int i=0 ; i<(int) plain_text.length();i++){
        if (plain_text[i] != ' '){
            word.push_back(plain_text[i]);
        }
        if (plain_text[i] == ' '){
            arr[j] = word;
            j++;
        }
```

```

        word.clear();
    }
    if (i == (int) plain_text.length()-1){
        arr[j] = word;
    }
}


```

- For reversing the array;

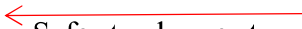
```

string reversed_arr[len];
int var1 = 0;
for (int index = len-1; index>=0 ; index-- ){
    reversed_arr[var1] = arr[index];
    var1++;
}

```

  
 Stay home stay safe

Will be revered to;

  
 Safe stay home stay

After reversing it, it will take the alphabetic secret key, and converts it into its respective numeric value. E.g.

**a to 1, b to 2, c to 3, d to 4 , and so on.**

**E.g.**

**a h n n z    →    1 8 14 14 26**

- For converting alphabetic key to its numeric value;

```

string the Alphabetic Secret key: ";
cin>>secret_ secret_key;
secret_key.clear();
cout<<"Enter key;

```

```

        int numeric_key[secret_key.length()-1];
        for (int i= 0 ; i < (int) secret_key.length() ; i++) {
            while(isdigit(secret_key[i])){
                cout<<"The Secret key should'nt contain any number <=\\n";
                cout<<"Enter the Alphabetic Secret key: ";
                cin>>secret_key;
                int numeric_key[secret_key.length()-1];
                i=0;
            }
            numeric_key[i] = int(secret_key[i]) - 96;
        }
        cout<<endl;

```

Then we will arrange the reverred plaintext and the numeric key as follows;

In this case,

safe	stay	home	stay
1	8	14	14

Then, it will shift each letter of each word according to the number written below to create its encrypted text.

**tbgf abig vcas ghom**

```

string alphabet = "abcdefghijklmnopqrstuvwxyz";
string var2;
int num = 0;
string cypher_text;
for (int l = 0 ; l < len ; l++){

    var2 = reversed_arr[l];
    int no = l;
    if (no>(int)(secret_key.length()-1)){

```

```

        no = no-secret_key.length();
    }
    for (int j = 0 ; j<= (int) var2.length() ; j++){
        if (isalpha(var2[j])){

            num = int(var2[j]) - 96;

            num += int(numeric_key[no]);
            if (num>26){
                num = num - 26;
            }
            var2[j] = alphabet[num-1];
        }

        var2[j] = var2[j];
        num = 0;
    }

    cypher_text.append(var2);
    cypher_text.append(" ");
}

cout<<"Encrypted Text: "<<cypher_text;

```

For decryption, this encrypted text is shifted in reverse order by using the same secret key. Then the string is reversed again to get the original plain text message.

- For counting the number of words;

```

string cypher_text;

cout<<"Enter the Encrypted Text: "<<endl;
cin.ignore();
getline(cin, cypher_text);
int len = 1;

```



```

        for (int i = 0; i<(int)cypher_text.length() ; i++){
            if (cypher_text[i]==' '){
                len++;
            }
        }

```

- Storing these words in the array;

```

string word;
string arr[len];
int j = 0;
for (int i=0 ; i<(int)cypher_text.length();i++){

    if (cypher_text[i] != ' '){
        word.push_back(cypher_text[i]);
    }

    if (cypher_text[i] == ' '){
        arr[j] = word;
        j++;
        word.clear();
    }

    if (i == (int) cypher_text.length()-1){
        arr[j] = word;
    }

}

```

- Getting the secret key and converting it into its numeric form;

```

string secret_key;
cout<<"Enter the Secret key: ";
cin>>secret_key;
int numeric_key[secret_key.length()-1];
for (int i= 0 ; i < (int) secret_key.length() ; i++) {

```

```

        numeric_key[i] = int(secret_key[i]) - 96;
    }
    cout<<endl;

```

- Shifting each word, in reverse, according to its respective number;

```

string alphabet = "abcdefghijklmnopqrstuvwxyz";
string var2;
int num;
string plain_text;
plain_text.clear();
string arr2[len];
for (int l = 0 ; l < len ; l++){

    var2 = arr[l];
    int no = l;
    if (no>(int) (secret_key.length()-1)){
        no = (no-secret_key.length());
    }
    for (int j = 0 ; j<=(int) var2.length() ; j++){
        if (isalpha(var2[j])){
            num = int(var2[j]) - 96;
            num -= int(numeric_key[no]);
            if (num<=0){
                num += 26;
            }
            var2[j] = alphabet[num-1];
        }

        var2[j] = var2[j];
        num = 0;
    }
    arr2[l] = var2;
}

```

```

    }
    • Reversing the array;
    string reversed_arr[len];
    int var1 = 0;
        for (int index = len-1; index>=0 ; index-- ){
            reversed_arr[var1] = arr2[index];
            var1++;
        }
    for (int index = 0; index<len ; index++ ){
        plain_text.append(reversed_arr[index]);
        plain_text.append(" ");
    }
    cout<<"Plain Text: "<<plain_text;
}

```

In this technique, every word is encrypted according to different shifts.

## 2- Password Analyzer

We need our users to create a strong password so that their data is more secure. A strong password is supposed to contain uppercase English letters, lowercase English letters, special characters, digits, etc.

Let's define a strong password for our users. A password is strong if:

- It contains at least two uppercase letter.
- It contains at least two lowercase letter.
- It contains at least two digit.
- It has special characters.
- Its length must be more than 8 characters.

We made an extension in our application named “**Password Analyzer**”.

When 2 is entered on the main menu, the program displays a new screen asking user to enter a password to analyze it.

Now user will enter a password/passphrase that needs to be analyzed and press enter. We take “pakistan” as an example for now.

```
=====
====>>> Password Analyzer <<<====
=====

Enter your password: _
```

```
=====
====>>> Password Analyzer <<<====
=====

Enter your password: pakistan

=====
====>>> Suggestions <<<====
=====
=> Password length is too SHORT
=> Add more UPPERcase characters to increase strength!
=> Add more DIGITS to increase strength!
=> Add more SPECIAL characters to increase strength!

=====
====>>> Password Analyzed <<<====
=====
=> Total Length: 8
=> Lowercase: 8
=> Uppercase: 0
=> Digits: 0
=> Special: 0

=====
====>>> Password Strength <<<====
=====
5% |==.....|

Do you want to continue this tool? (Y/N): _
```

First menu shows “**Suggestions**” to the user. There are checks in the code that will display a suggestion if:

- It doesn't contain at least two uppercase letter.
- It doesn't contain at least two lowercase letter.
- It doesn't contain at least two digit.
- It doesn't have special characters.
- Its length is less than 8 characters.

Second menu shows **total length, number of alphabets (uppercase/lowercase), digits and special characters.**

Third menu shows the strength score of password out of 100. In our example the score given is 5% which shows that password “pakistan” is very weak and requires more characters to be entered.

### Working of code:

We use **int** datatype for **lower case, upper case, digits, special** and to give **strength**. All these are initialized to zero except strength which is initialized to 100.

```
int l_case=0, u_case=0, digit=0, special=0, strength=100;
```

**String** datatype is used for entering password. **Int len** is used to enter length of password.

```
string password;  
cin.ignore();  
cout<<"\nEnter your password: ";  
getline(cin, password);  
int len = password.length();
```

When password is entered it gives suggestions according to the checks as mentioned in the code. For this process, “**for loop and if conditions**” are used. In the next code block, “**for loop**” will run for every character in the password entered by user and is used here to count its lowers, uppers, digits and special and will display their total number.

```
for (int i =0 ; i < len ; i++){  
    if(islower(password[i]))  
        l_case++;  
    if(isupper(password[i]))  
        u_case++;  
    if(isdigit(password[i]))  
        digit++;  
    if(!isalpha(password[i]) && !isdigit(password[i]) && password[i] != ' ')  
        special++;  
}  
if (l_case<=2){  
    cout<<"=> Add more LOWERcase characters to increase strength!"<<endl;  
    strength -= 15;
```

```

    }
    if (u_case<=2){
        cout<<"=> Add more UPPERcase characters to increase strength!"<<endl;
        strength -= 15;
    }
    if (digit<=2){
        cout<<"=> Add more DIGITS to increase strength!"<<endl;
        strength -= 15;
    }
    if (special<=2){
        cout<<"=> Add more SPECIAL characters to increase strength!"<<endl;
        strength -= 15;
    }

```

In the third and last menu of the code strength of the password out of 100 is display which was being calculated in the above two menus and a symbol of ‘+’ or ‘.’ is displayed.

```

cout<<strength<<"% |";
    for (int i = 0 ; i < int(strength/2) ; i++){
        cout<<"+";
    }
    for (int i = 0 ; i < (100-(strength) )/2; i++){
        cout<<".";
    }
    cout<<"|"<<endl<<endl;

```

### 3- Password Generator

Passwords are an important part of our lives. They secure our data and accounts. We have to set a strong password for healthy security system. Password is nothing but the combination of uppercase letters (A to Z), Lower case letters (a to z), numbers (0 to 9), and some special characters.

We made an extension in our application named “**Password Generator**”

Display page asks user to enter any number above 10.

```
=====
====>>> Password Generator <<<====
=====
Enter the Length (>10) of Password: _
```

#### Working Algorithm:

In case a number less than 10 is entered, it will display a message asking user to enter a number above 10. It is done since passwords with more than 10 length are more secure.

```
=====
====>>> Password Generator <<<====
=====
Enter the Length (>10) of Password: 2
Less than 10 is not allowed!!!
Enter the Length (>10) of Password:
```

Once a number above 10 is entered, it will display a password of user entered length which can be copied and used.

```
=====
====>>> Password Generator <<<====
=====
Enter the Length (>10) of Password: 14
=> Generated Password of Length 14 is : UabP1SoWkLrySQ
Do you want to continue this tool? (y/n):
```

## Working of code:

Array **alpha\_num** contains the number, special symbol and alphabets(in both cases) used for generating random passwords.

```
char alpha_num[] =
"67ghiST5j8^&:*a07WX@!ef9V2tv34d5wxQR14DO!pkJ8LZl2mMNBC9!noqYeruA#6$sy3zPbcU%F
GHK";

int req_length, string_length = (sizeof(alpha_num) - 1);

cout << "Enter the Length (>=10) of Password: ";

cin >> req_length;

while(req_length < 10){
    cout<<"Less than 10 is not allowed!!! \nEnter the Length (>=10) of Password:";
    cin >> req_length;
}
```

We can user-defined value for the length to generate the random passwords. **Srand()** is a function which is used to take the letters, numbers and special characters randomly.

```
srand(time(0));

cout << "\n=> Generated Password of Length "<<req_length<< " is : ";

for (int i = 0; i < req_length; i++)
    cout << alpha_num[rand() % string_length];
```

We have used **cstdlib** and **ctime** library for random function.

```
#include <cstdlib>
```

```
#include<ctime>
```

The program asks you to enter the length of the password. **For loop** iterates to take the random password for the particular length. Once a letter or number or special character is chosen for password then it will not take randomly again.

We can create as many as password because of our code. Now your new password is ready which suggests a secure and if not try another one.