

"Assignment-1"

A REPORT SUBMITTED TO

THE NATIONAL INSTITUTE OF ENGINEERING, MYSURU

(An Autonomous Institute under VTU, Belagavi)



In partial fulfilment of the requirements for Cryptography (CS6C02), Sixth Semester

**Bachelor of Engineering
in
Computer Science and Engineering**

Submitted by
Mausam Borah (4NI20CS051)

To

Ramya S

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

THE NATIONAL INSTITUTE OF ENGINEERING

Mysore - 570008

2022-2023

Question 1. Implement Affine Cipher.

Code:

Code

CryptographyAssignment / 1_affine.cpp

mausam30 Update 1_affine.cpp

516e256 · 47 minutes ago · History

Code Blame 72 lines (60 loc) · 2.08 KB

```
1 #include <iostream>
2 #include <string>
3 #include <cctype>
4
5 std::string affine_encrypt(const std::string& plain_text, int a, int b) {
6     std::string alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
7     std::string encrypted_text = "";
8
9     for (size_t i = 0; i < plain_text.length(); i++) {
10         char ch = plain_text[i];
11         if (std::isalpha(ch)) {
12             char uppercase_ch = std::toupper(ch);
13             int index = alphabet.find(uppercase_ch);
14             int encrypted_index = (a * index + b) % 26;
15             char encrypted_char = alphabet[encrypted_index];
16             encrypted_text += encrypted_char;
17         } else {
18             encrypted_text += ch;
19         }
20     }
21
22     return encrypted_text;
23 }
24
25 std::string affine_decrypt(const std::string& cipher_text, int a, int b) {
26     std::string alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
27     std::string decrypted_text = "";
28     int a_inverse = 0;
29
30     for (int i = 0; i < 26; i++) {
31         if ((a * i) % 26 == 1) {
32             a_inverse = i;
33             break;
34         }
35     }
36
37     for (size_t i = 0; i < cipher_text.length(); i++) {
38         char ch = cipher_text[i];
39         if (std::isalpha(ch)) {
40             char uppercase_ch = std::toupper(ch);
41             int index = alphabet.find(uppercase_ch);
42             int decrypted_index = (a_inverse * (index - b + 26)) % 26;
43             char decrypted_char = alphabet[decrypted_index];
44             decrypted_text += decrypted_char;
45         } else {
46             decrypted_text += ch;
47         }
48     }
49
50     return decrypted_text;
51 }
52
53 int main() {
54     std::string plain_text;
55     int a, b;
56
57     std::cout << "Enter the plain text: ";
58     std::getline(std::cin, plain_text);
59     std::cout << "Enter the multiplicative value: ";
60     std::cin >> a;
61     std::cout << "Enter the additive value: ";
62     std::cin >> b;
63     std::cin.ignore();
64
65     std::string encrypted_text = affine_encrypt(plain_text, a, b);
66     std::cout << "Encrypted text: " << encrypted_text << std::endl;
67
68     std::string decrypted_text = affine_decrypt(encrypted_text, a, b);
69     std::cout << "Decrypted text: " << decrypted_text << std::endl;
70
71     return 0;
72 }
```

Documentation · Share feedback

Code

CryptographyAssignment / 1_affine.cpp

Raw

Top

Code Blame 72 lines (60 loc) · 2.08 KB

```
35
36
37     for (size_t i = 0; i < cipher_text.length(); i++) {
38         char ch = cipher_text[i];
39         if (std::isalpha(ch)) {
40             char uppercase_ch = std::toupper(ch);
41             int index = alphabet.find(uppercase_ch);
42             int decrypted_index = (a_inverse * (index - b + 26)) % 26;
43             char decrypted_char = alphabet[decrypted_index];
44             decrypted_text += decrypted_char;
45         } else {
46             decrypted_text += ch;
47         }
48     }
49
50     return decrypted_text;
51 }
52
53 int main() {
54     std::string plain_text;
55     int a, b;
56
57     std::cout << "Enter the plain text: ";
58     std::getline(std::cin, plain_text);
59     std::cout << "Enter the multiplicative value: ";
60     std::cin >> a;
61     std::cout << "Enter the additive value: ";
62     std::cin >> b;
63     std::cin.ignore();
64
65     std::string encrypted_text = affine_encrypt(plain_text, a, b);
66     std::cout << "Encrypted text: " << encrypted_text << std::endl;
67
68     std::string decrypted_text = affine_decrypt(encrypted_text, a, b);
69     std::cout << "Decrypted text: " << decrypted_text << std::endl;
70
71     return 0;
72 }
```

Documentation · Share feedback

Output:

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

- (base) mausamborah@Mausams-MacBook-Air cry assign % g++ 1_affine.cpp
- (base) mausamborah@Mausams-MacBook-Air cry assign % ./a.out
Enter the plain text: CRYPTOMAUSAM
Enter the multiplicative value: 11
Enter the additive value: 15
Encrypted text: LUTYQNRPFPR
Decrypted text: CRYPTOMAUSAM
- (base) mausamborah@Mausams-MacBook-Air cry assign %

Question 2. Implement Extended Euclidean Algorithm.

Code:

Code

CryptographyAssignment / 2_exteuclidean.cpp

mausam30 Add files via upload 96ef8ba · 51 minutes ago History

Code Blame 55 lines (45 loc) · 1.31 KB

```
1 #include <iostream>
2 #include <stdexcept>
3
4 struct ExtendedEuclideanResult {
5     int gcd;
6     int x;
7     int y;
8 };
9
10 ExtendedEuclideanResult extended_euclidean_algorithm(int a, int b) {
11     if (b == 0) {
12         ExtendedEuclideanResult result;
13         result.gcd = a;
14         result.x = 1;
15         result.y = 0;
16         return result;
17     }
18
19     ExtendedEuclideanResult prev_result = extended_euclidean_algorithm(b, a % b);
20     ExtendedEuclideanResult result;
21     result.gcd = prev_result.gcd;
22     result.x = prev_result.y;
23     result.y = prev_result.x - (a / b) * prev_result.y;
24
25     return result;
26 }
27
28 int find_modular_inverse(int a, int m) {
29     ExtendedEuclideanResult result = extended_euclidean_algorithm(a, m);
30     if (result.gcd != 1) {
31         throw std::runtime_error("Inverse does not exist.");
32     }
33
34     int inverse = (result.x % m + m) % m;
35     return inverse;
36 }
```

Documentation · Share feedback

Code

CryptographyAssignment / 2_exteuclidean.cpp

Raw

Code Blame 55 lines (45 loc) · 1.31 KB

```
18
19     ExtendedEuclideanResult prev_result = extended_euclidean_algorithm(b, a % b);
20     ExtendedEuclideanResult result;
21     result.gcd = prev_result.gcd;
22     result.x = prev_result.y;
23     result.y = prev_result.x - (a / b) * prev_result.y;
24
25     return result;
26 }
27
28 int find_modular_inverse(int a, int m) {
29     ExtendedEuclideanResult result = extended_euclidean_algorithm(a, m);
30     if (result.gcd != 1) {
31         throw std::runtime_error("Inverse does not exist.");
32     }
33
34     int inverse = (result.x % m + m) % m;
35     return inverse;
36 }
37
38 int main() {
39     int a, m;
40
41     std::cout << "Enter a number to find inverse: ";
42     std::cin >> a;
43     std::cout << "Enter the number whose modulus is to be found: ";
44     std::cin >> m;
45
46     int inverse;
47     try {
48         inverse = find_modular_inverse(a, m);
49         std::cout << "Modular inverse of " << a << " mod " << m << " is: " << inverse << std::endl;
50     } catch (const std::runtime_error& e) {
51         std::cout << e.what() << std::endl;
52     }
53
54     return 0;
55 }
```

Documentation · Share feedback

Output:

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

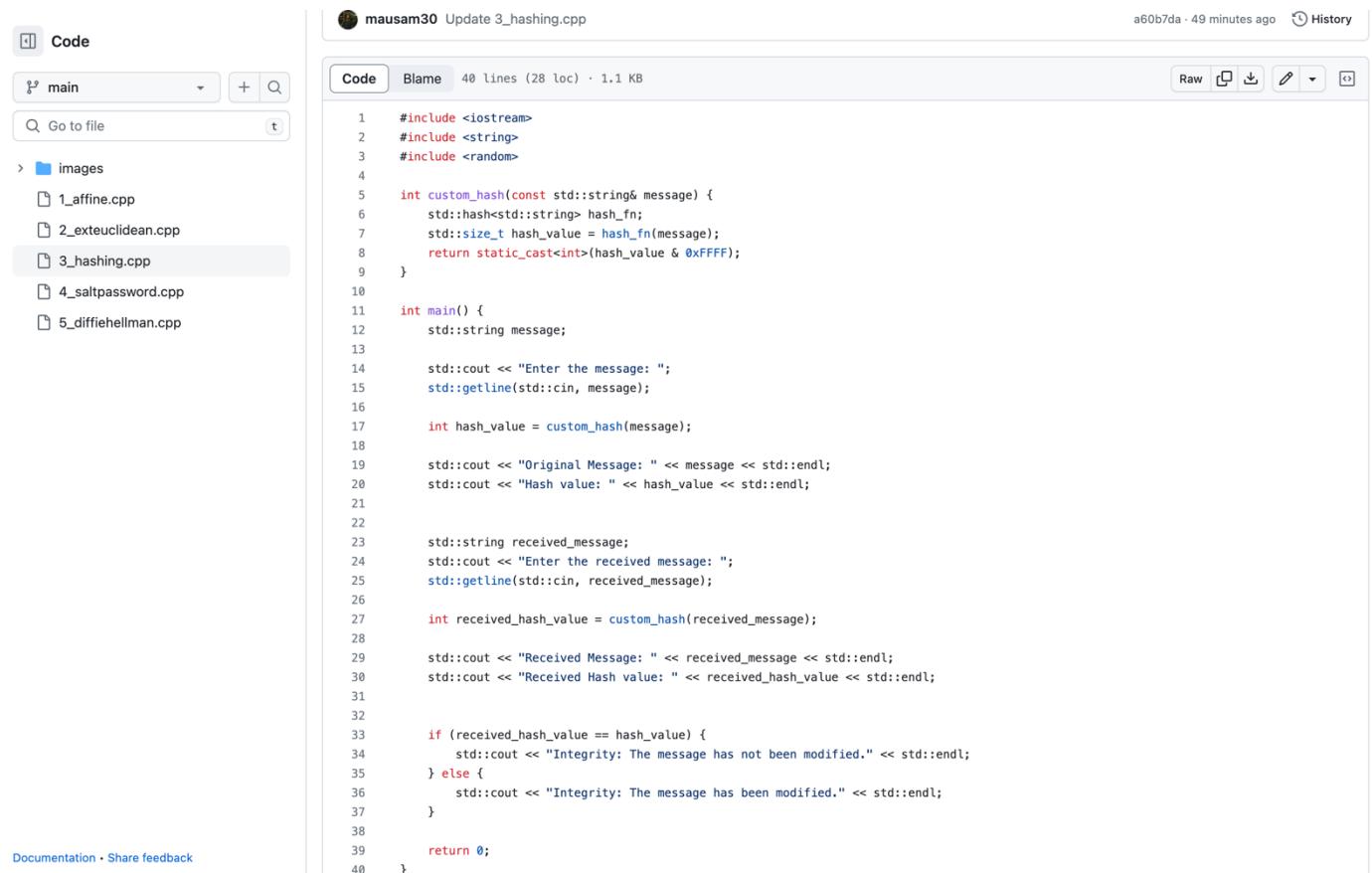
- (base) mausamborah@Mausams-MacBook-Air cry assign % g++ 2_exeuclidean.cpp
- (base) mausamborah@Mausams-MacBook-Air cry assign % ./a.out
Enter a number to find inverse: 7
Enter the number whose modulus is to be found: 26
Modular inverse of 7 mod 26 is: 15
- (base) mausamborah@Mausams-MacBook-Air cry assign % ./a.out
Enter a number to find inverse: 15
Enter the number whose modulus is to be found: 26
Modular inverse of 15 mod 26 is: 7
- (base) mausamborah@Mausams-MacBook-Air cry assign % ./a.out
Enter a number to find inverse: 13
Enter the number whose modulus is to be found: 26
Inverse does not exist.
- (base) mausamborah@Mausams-MacBook-Air cry assign % []

Question 3. Select a simple message. Perform a hash function on it.

i) Simulate a receiver computing the hash again and ensuring its integrity.

ii) Slightly change the message. Simulate a receiver computing hash and find it not matching.

Code:



```
#include <iostream>
#include <string>
#include <random>

int custom_hash(const std::string& message) {
    std::hash<std::string> hash_fn;
    std::size_t hash_value = hash_fn(message);
    return static_cast<int>(hash_value & 0xFFFF);
}

int main() {
    std::string message;

    std::cout << "Enter the message: ";
    std::getline(std::cin, message);

    int hash_value = custom_hash(message);

    std::cout << "Original Message: " << message << std::endl;
    std::cout << "Hash value: " << hash_value << std::endl;

    std::string received_message;
    std::cout << "Enter the received message: ";
    std::getline(std::cin, received_message);

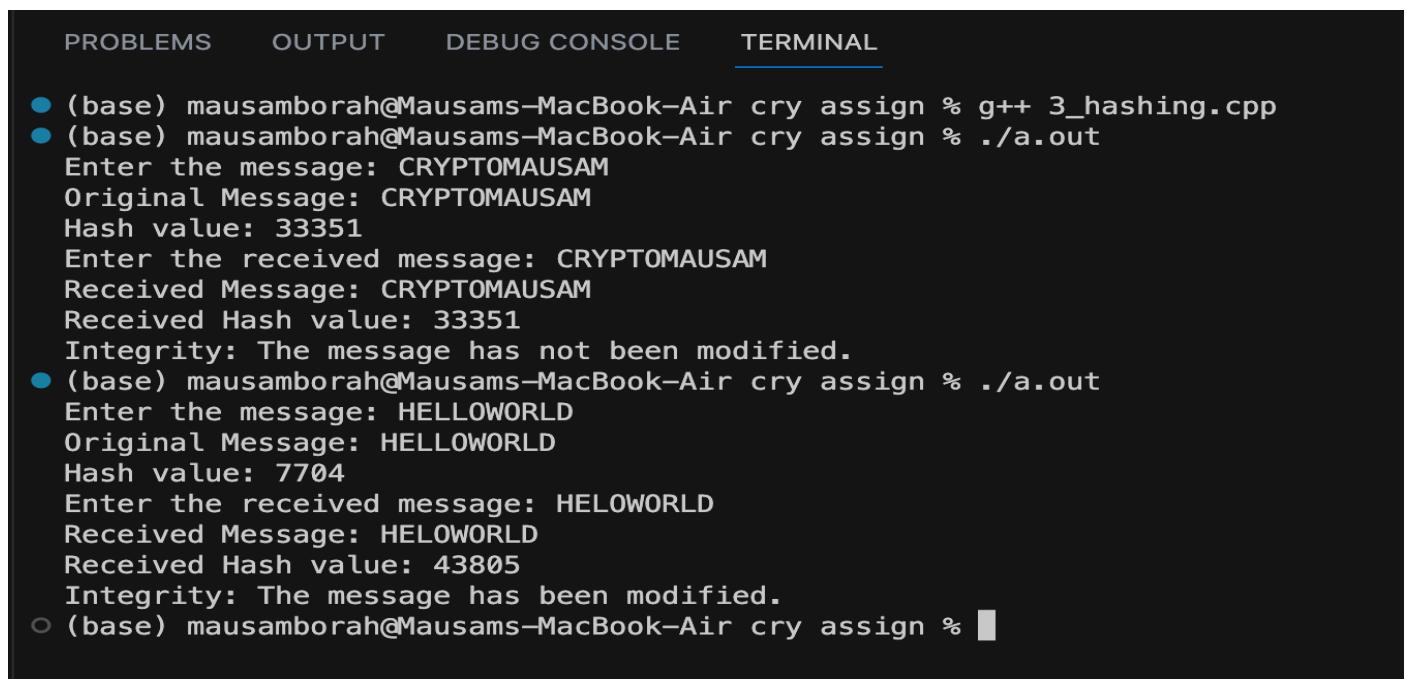
    int received_hash_value = custom_hash(received_message);

    std::cout << "Received Message: " << received_message << std::endl;
    std::cout << "Received Hash value: " << received_hash_value << std::endl;

    if (received_hash_value == hash_value) {
        std::cout << "Integrity: The message has not been modified." << std::endl;
    } else {
        std::cout << "Integrity: The message has been modified." << std::endl;
    }

    return 0;
}
```

Output:



PROBLEMS	OUTPUT	DEBUG CONSOLE	TERMINAL
● (base) mausamborah@Mausams-MacBook-Air cry assign % g++ 3_hashing.cpp			
● (base) mausamborah@Mausams-MacBook-Air cry assign % ./a.out	Enter the message: CRYPTOMAUSAM		
	Original Message: CRYPTOMAUSAM		
	Hash value: 33351		
	Enter the received message: CRYPTOMAUSAM		
	Received Message: CRYPTOMAUSAM		
	Received Hash value: 33351		
	Integrity: The message has not been modified.		
● (base) mausamborah@Mausams-MacBook-Air cry assign % ./a.out	Enter the message: HELLOWORLD		
	Original Message: HELLOWORLD		
	Hash value: 7704		
	Enter the received message: HELOWORLD		
	Received Message: HELOWORLD		
	Received Hash value: 43805		
	Integrity: The message has been modified.		
○ (base) mausamborah@Mausams-MacBook-Air cry assign %			

Question 4. a) Create a password file of 10 passwords and use it for identification.

b) Modify one to store the hash values of passwords & use it.

c) Optional: Create a salt file; add salt to password before storing in (b).

Code:

The screenshot shows a GitHub code editor interface. On the left, there's a sidebar with a 'Code' tab, a dropdown for 'main', and a search bar. Below that is a tree view of files: 'images', '1_affine.cpp', '2_exteuclidean.cpp', '3_hashing.cpp', '4_saltpassword.cpp' (which is selected), and '5_diffiehellman.cpp'. At the bottom of the sidebar, there are links for 'Documentation' and 'Share feedback'.

The main area is titled 'CryptographyAssignment / 4_saltpassword.cpp'. It shows a commit by 'mausam30' with the message 'Update 4_saltpassword.cpp'. The commit was made 49 minutes ago at '690a5d4'. There are buttons for 'Raw', 'Edit', 'Delete', and other options.

The code itself is as follows:

```
1 #include <iostream>
2 #include <fstream>
3 #include <string>
4 #include <random>
5 #include <algorithm>
6 #include <functional>
7 #include <cctype>
8 #include <unordered_map>
9
10 std::string generate_salt(std::size_t length = 8) {
11     std::string salt_characters = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^&*";
12     std::random_device rd;
13     std::mt19937 generator(rd());
14     std::shuffle(salt_characters.begin(), salt_characters.end(), generator);
15     salt_characters.resize(length);
16     return salt_characters;
17 }
18
19
20 std::string hash_password(const std::string& password, const std::string& salt) {
21     std::string salted_password = password + salt;
22     std::hash<std::string> hash_fn;
23     std::size_t hashed_password = hash_fn(salted_password);
24     return std::to_string(hashed_password);
25 }
26
27
28
29 bool check_password(const std::string& password, const std::string& hashed_password, const std::string& salt) {
30     std::string salted_password = password + salt;
31     std::hash<std::string> hash_fn;
32     std::size_t hashed_input_password = hash_fn(salted_password);
33     return hashed_password == std::to_string(hashed_input_password);
34 }
35
36 int main() {
```

Code

main Go to file

> images
1_affine.cpp
2_exteuclidean.cpp
3_hashing.cpp
4_saltpassword.cpp
5_diffiehellman.cpp

CryptographyAssignment / 4_saltpassword.cpp

Code Blame 106 lines (91 loc) · 3.87 KB

```
36 int main() {
37     std::size_t num_users;
38     std::cout << "Enter the number of users: ";
39     std::cin >> num_users;
40     std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
41
42     std::ofstream file("hashed_passwords.txt");
43     if (!file) {
44         std::cerr << "Failed to open the file." << std::endl;
45         return 1;
46     }
47
48     std::unordered_map<std::string, std::string> password_file;
49     for (std::size_t i = 1; i <= num_users; ++i) {
50         std::string username, password;
51         std::cout << "Enter the username for User" << i << ": ";
52         std::getline(std::cin, username);
53         std::cout << "Enter the password for User" << i << ": ";
54         std::getline(std::cin, password);
55         password_file[username] = password;
56     }
57
58     std::unordered_map<std::string, std::pair<std::string, std::string> > hashed_password_file;
59     for (std::pair<const std::string, std::string& pair : password_file) {
60         const std::string& username = pair.first;
61         std::string& password = pair.second;
62         const std::string salt = generate_salt();
63         const std::string hashed_password = hash_password(password, salt);
64         hashed_password_file.insert(std::make_pair(username, std::make_pair(hashed_password, salt)));
65         file << username << ":" << hashed_password << ":" << salt << '\n';
66     }
67     file.close();
68
69     std::string input_username, input_password;
70     std::cout << "Enter username: ";
71     std::getline(std::cin, input_username);
72     std::cout << "Enter password: ";
73     std::getline(std::cin, input_password);
74
75     std::ifstream hashed_passwords_file("hashed_passwords.txt");
76     ...
```

Code

main Go to file

> images
1_affine.cpp
2_exteuclidean.cpp
3_hashing.cpp
4_saltpassword.cpp
5_diffiehellman.cpp

CryptographyAssignment / 4_saltpassword.cpp

Code Blame 106 lines (91 loc) · 3.87 KB

```
69     std::string input_username, input_password;
70     std::cout << "Enter username: ";
71     std::getline(std::cin, input_username);
72     std::cout << "Enter password: ";
73     std::getline(std::cin, input_password);
74
75     std::ifstream hashed_passwords_file("hashed_passwords.txt");
76     if (!hashed_passwords_file) {
77         std::cerr << "Failed to open the file." << std::endl;
78         return 1;
79     }
80
81     std::string line;
82     bool login_successful = false;
83     while (std::getline(hashed_passwords_file, line)) {
84         std::size_t delimiter_pos_1 = line.find(':');
85         std::size_t delimiter_pos_2 = line.find(':', delimiter_pos_1 + 1);
86         if (delimiter_pos_1 != std::string::npos && delimiter_pos_2 != std::string::npos) {
87             std::string stored_username = line.substr(0, delimiter_pos_1);
88             std::string stored_hashed_password = line.substr(delimiter_pos_1 + 1, delimiter_pos_2 - delimiter_pos_1 - 1);
89             std::string stored_salt = line.substr(delimiter_pos_2 + 1);
90             if (stored_username == input_username &&
91                 check_password(input_password, stored_hashed_password, stored_salt)) {
92                 login_successful = true;
93                 break;
94             }
95         }
96     }
97     hashed_passwords_file.close();
98
99     if (login_successful) {
100         std::cout << "Login successful." << std::endl;
101     } else {
102         std::cout << "Login failed. Invalid username or password." << std::endl;
103     }
104
105     return 0;
106 }
```

Output:

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

- (base) mausamborah@Mausams-MacBook-Air cry assign % ./a.out
Enter the number of users: 10
Enter the username for User1: u1
Enter the password for User1: u1pass
Enter the username for User2: u2
Enter the password for User2: u2pass
Enter the username for User3: u3
Enter the password for User3: u3pass
Enter the username for User4: u4
Enter the password for User4: u4pass
Enter the username for User5: u5
Enter the password for User5: u5pass
Enter the username for User6: u6
Enter the password for User6: u6pass
Enter the username for User7: u7
Enter the password for User7: u7pass
Enter the username for User8: u8
Enter the password for User8: u8pass
Enter the username for User9: u9
Enter the password for User9: u9pass
Enter the username for User10: u10
Enter the password for User10: u10pass
Enter username: u7
Enter password: u7pass
Login successful.

- (base) mausamborah@Mausams-MacBook-Air cry assign % █

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

- (base) mausamborah@Mausams-MacBook-Air cry assign % ./a.out
Enter the number of users: 10
Enter the username for User1: u1
Enter the password for User1: u1pass
Enter the username for User2: u2
Enter the password for User2: u2pass
Enter the username for User3: u3
Enter the password for User3: u3pass
Enter the username for User4: u4
Enter the password for User4: u4pass
Enter the username for User5: u5
Enter the password for User5: u5pass
Enter the username for User6: u6
Enter the password for User6: u6pass
Enter the username for User7: u7
Enter the password for User7: u7pass
Enter the username for User8: u8
Enter the password for User8: u8pass
Enter the username for User9: u9
Enter the password for User9: u9pass
Enter the username for User10: u10
Enter the password for User10: u10pass
Enter username: u6
Enter password: u7pass
Login failed. Invalid username or password.

- (base) mausamborah@Mausams-MacBook-Air cry assign % █

Hashed password:

```
[hashed_passwords.txt]
1  u10:5719059039242858425:T$5@6Jpt
2  u6:4994163829975450158:xYheAw2#
3  u5:130566239097960841:bePAX2Ec
4  u8:5653666782158312540:4D9LXHpG
5  u4:16025903756261847435:xgGOJ74R
6  u3:11915412710416120854:pVFQ0EZR
7  u7:15622030458450873397:BIhmRuq1
8  u2:5512669019510138763:X%u09ECn
9  u9:16784909078117150250:IOU!zQLs
10 u1:11539106910029189730:jNIRl3MA
11
```

5. Generate a symmetric key by using Diffie-Hellman method given g, p, x and y

Code:

Code

CryptographyAssignment / 5_diffiehellman.cpp

mausam30 Add files via upload 96ef8ba · 54 minutes ago History

Code Blame 104 lines (94 loc) · 2.4 KB

```
1 #include <iostream>
2 #include <vector>
3 #include <cmath>
4
5 int prime_checker(int p) {
6     if (p < 1) {
7         return -1;
8     } else if (p > 1) {
9         if (p == 2) {
10             return 1;
11         }
12         for (int i = 2; i < p; ++i) {
13             if (p % i == 0) {
14                 return -1;
15             }
16         }
17         return 1;
18     }
19     return -1;
20 }
21
22 int primitive_check(int g, int p, std::vector<int>& L) {
23     for (int i = 1; i < p; ++i) {
24         int result = g;
25         for (int j = 1; j < i; ++j) {
26             result = (result * g) % p;
27         }
28         L.push_back(result);
29     }
30     for (int i = 1; i < p; ++i) {
31         if (std::count(L.begin(), L.end(), i) > 1) {
32             L.clear();
33             return -1;
34         }
35     }
36     return 1;
37 }
```

Documentation · Share feedback

Code

CryptographyAssignment / 5_diffiehellman.cpp

Raw

Code Blame 104 lines (94 loc) · 2.4 KB

```
37 }
38
39 int main() {
40     std::vector<int> L;
41     int P;
42     while (true) {
43         std::cout << "Enter P: ";
44         std::cin >> P;
45         if (prime_checker(P) == -1) {
46             std::cout << "Number Is Not Prime, Please Enter Again!" << std::endl;
47             continue;
48         }
49         break;
50     }
51
52     int G;
53     while (true) {
54         std::cout << "Enter The Primitive Root Of " << P << ": ";
55         std::cin >> G;
56         if (primitive_check(G, P, L) == -1) {
57             std::cout << "Number Is Not A Primitive Root Of " << P << ", Please Try Again!" << std::endl;
58             continue;
59         }
60         break;
61     }
62
63     int x1, x2;
64     while (true) {
65         std::cout << "Enter The Private Key Of User 1: ";
66         std::cin >> x1;
67         std::cout << "Enter The Private Key Of User 2: ";
68         std::cin >> x2;
69         if (x1 >= P || x2 >= P) {
70             std::cout << "Private Key Of Both The Users Should Be Less Than " << P << "!" << std::endl;
71             continue;
72         }
73         break;
74     }
75
76     int y1 = 1;
```

Output:

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

- (base) mausamborah@Mausams-MacBook-Air cry assign % g++ 5_diffiehellman.cpp
- (base) mausamborah@Mausams-MacBook-Air cry assign % ./a.out
Enter P: 37
Enter The Primitive Root Of 37: 19
Enter The Private Key Of User 1: 5
Enter The Private Key Of User 2: 2

Secret Key For User 1 Is 3
Secret Key For User 2 Is 3
Keys Have Been Exchanged Successfully
- (base) mausamborah@Mausams-MacBook-Air cry assign % ./a.out
Enter P: 47
Enter The Primitive Root Of 47: 23
Enter The Private Key Of User 1: 7
Enter The Private Key Of User 2: 4

Secret Key For User 1 Is 25
Secret Key For User 2 Is 25
Keys Have Been Exchanged Successfully
- (base) mausamborah@Mausams-MacBook-Air cry assign % ./a.out
Enter P: 48
Number Is Not Prime, Please Enter Again!
Enter P: 23
Enter The Primitive Root Of 23: 17
Enter The Private Key Of User 1: 4
Enter The Private Key Of User 2: 3

Secret Key For User 1 Is 6
Secret Key For User 2 Is 6
Keys Have Been Exchanged Successfully
- (base) mausamborah@Mausams-MacBook-Air cry assign %

Github Link : <https://github.com/mausam30/CryptographyAssignment>