

Міністерство освіти та науки України  
Харківський національний університет радіоелектроніки  
Кафедра програмної інженерії

Лабораторна робота №4  
з дисципліни: «Безпека програм та даних»  
на тему: «Алгоритм обміну ключами Діффі-Хелмана»

Виконав

ст. гр. ПЗПІ-20-1

Бабанін А.К.

Перевірив

доцент кафедри ПІ

Турута О.О.

2023

**Мета роботи:** Отримати навички безпечного обміну ключами по каналу зв'язку, які надалі можуть бути використані в якомусь алгоритмі шифрування. Реалізувати програмно (на будь-якій мові програмування) роботу алгоритму Діффі-Хеллмана.

### Хід роботи

1. Створити програмну реалізацію алгоритму обміну ключами Діффі-Хеллмана.
2. Реалізувати чат на трьох і більше користувачів.
3. Створити приємний та зрозумілий інтерфейс для перевірки зробленої роботи.

Процес обміну ключами між трьома, або більше учасниками чату виглядає наступним чином: нехай є  $m$  користувачів

0. Клієнти під'єднуються до чат-хабу
1. Хаб надсилає кожному клієнту пару  $g =$  відкрите просте число та  $p =$  відкрите просте число.
2. Кожен клієнт обчислює  $A =$  відкритий ключ клієнта, та повертає його до хабу
3. Для користувача  $U_n$  передається множина відкритих ключів  $K = \{k_1, k_2, \dots, k_m\} - \{k_n\}$  для обчислення спільного секрету.

Для взаємодії між клієнтами використовується технологія веб-сокетів та бібліотека. На серверній стороні додатку виконуються наступні операції:

Зберігається інформація про підключених клієнтів;

1. Ведеться облік кількості підключених клієнтів;

2. При підключенні нового клієнта відправляються ініціалізаційні дані (публічні ключі) клієнту
3. Видаляється відключений клієнт зі списку та оновлюється інформація про підключених клієнтів;
4. Дозволяє клієнтам надсилати та отримувати повідомлення.

Лістинг коду наведений далі:

```
function handleKeys(connection, connectionId, data){
    userPublicKeys[connectionId] = data.a;
    if (Object.keys(userPublicKeys).length < 2){

        console.log("Only one user was connected. Waiting...")
    }else{
        console.log("Minimum 2 users were connected. Proceed with keys exchange.")
        for(let key in connections){

            let publicKeys = []

            let toCollect = Object
                .keys(connections)
                .filter(x => x !== key)
                .forEach(x => publicKeys.push(userPublicKeys[x]));

            let data = {
                users: publicKeys,
            }

            console.log("Sending data to client" + data + " " + key)
            connections[key].send(JSON.stringify(data))
        }
    }
}
```

Чат є анонімний, тому кожен користувач має умовний UUID ідентифікатор у чаті. Для шифрування повідомлень використовується алгоритм AES. Спочатку в анонімному чаті присутні 2 користувачі. Результати роботи застосунку наведені далі:

```
Opening socket
Connection with server established!
  ▶ Object
received: {"g":5,"p":23}
18
  ▶ Object
received: {"users":[2]}
Shared key: 13
Original message:Hello guys!
Encrypted message:Urf9a6iPa1C2a1Cda02Faw==
Urf9a6iPa1C2a1Cda02Faw==
  ▶ Object
received:
{"message":"UqhNt1JAnbfvqP23tvn1twVQJTeWxow1XrA2A23Gejc=", "author":"4030f27b-0d10-407b-b2e4-ea15fc68ea36"}
Received: Hi! How are you doing, bro?
```

Рисунок 1 – Лог повідомлення першого користувача

```
Opening socket App.js:69
Connection with server established! App.js:22
  ▶ Object App.js:29
received: {"g":5,"p":23} App.js:30
2 App.js:38
  ▶ Object App.js:29
received: {"users":[18]} App.js:30
Shared key: 13 App.js:54
  ▶ {current: WebSocket} App.js:29
received: {"message":"Urf9a6iPa1C2a1Cda02Faw==", "author":"3489ab61-e19c-4992-a8e8-382848ac99b7"} App.js:30
Received: Hello guys! App.js:61
Original message:Hi! How are you doing, bro? App.js:83
Encrypted message:UqhNt1JAnbfvqP23tvn1twVQJTeWxow1XrA2A23Gejc= App.js:88
UqhNt1JAnbfvqP23tvn1twVQJTeWxow1XrA2A23Gejc= App.js:97
>
```

Рисунок 2 – Лог повідомлення другого користувача

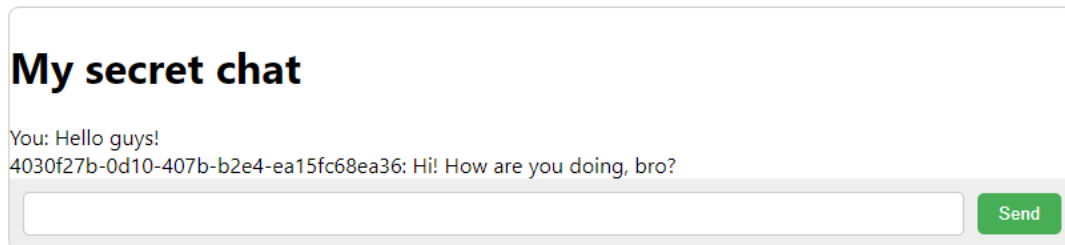


Рисунок 3 – Вигляд чату для першого користувача

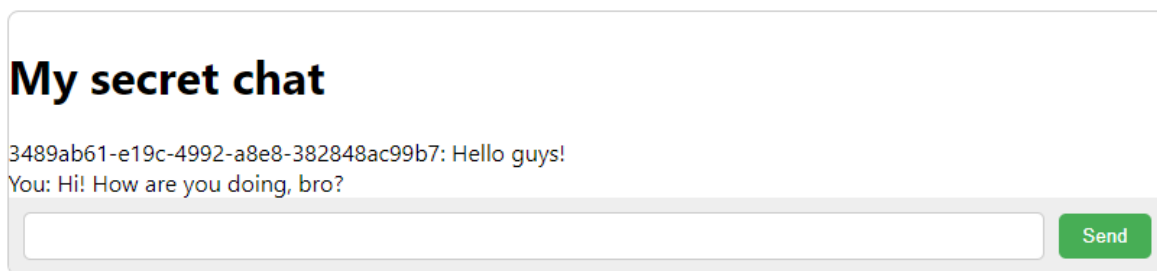


Рисунок 4 – Вигляд чату для другого користувача

Коли під'єднується новий користувач, то відбувається перерахунок ключів:

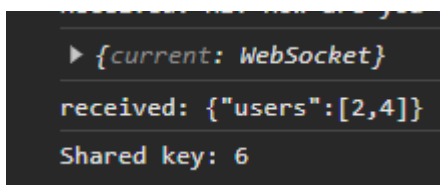


Рисунок 5 – Нові публічні ключі, які отримав перший користувач

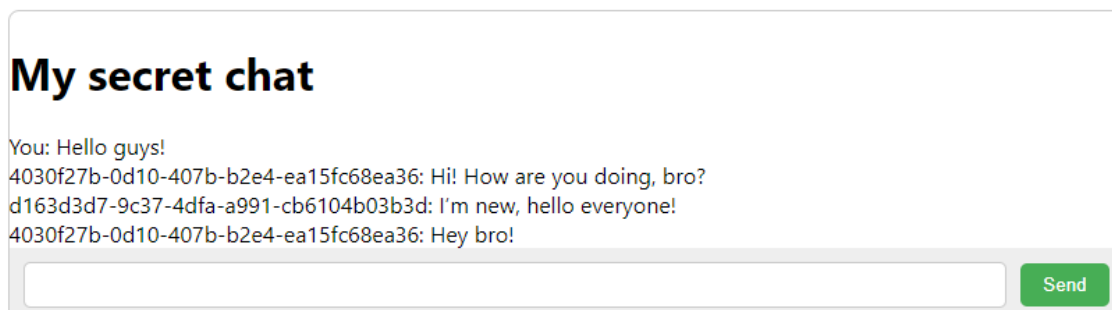


Рисунок 6 – Вигляд чату для першого користувача

**Висновки:** На даній лабораторній роботі отримав навички безпечного обміну ключами по каналу зв'язку, які надалі можуть бути використані в якомусь алгоритмі шифрування. Реалізував програмно (на мові програмування JavaScript) роботу алгоритму Діффі-Хеллмана