Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №1**

По дисциплине «Криптографические методы защиты информации»

Тема: «Алгоритмы обмена ключами»

**Выполнил:**

Студент 3 курса

Группы ИИ-23

Макаревич Н.Р.

**Проверил:**

Хацкевич A. C.

Брест 2024

**Цель:** изучить алгоритмы обмена ключами. Практически реализовать алгоритмы обмена ключами.

Варианты алгоритмов обмена ключами:

5. Усилинный EKE

Реализовать алгоритм обмена сеансовым ключом. Сеансовый ключ используется для шифрования/расшифрования информации.

Шифровать/расшифровывать информацию используя стандарты RC4

**Ход работы:**

**Класс Client:**

class Client {

public:

std::string name;

Client(std::string name);

void syncSharedKey(Client&);

std::vector<unsigned char> sendEpK();

void receiveEpK(const std::vector<unsigned char>& EpK);

std::vector<unsigned char> sendEpEkK();

void receiveEpEkK(const std::vector<unsigned char>& EpEkK);

std::vector<unsigned char> sendEkRa();

std::vector<unsigned char> sendEkSa();

void receiveEkRaSa(const std::vector<unsigned char>& Ra, const std::vector<unsigned char>& Sa);

std::vector<unsigned char> sendEkRecRandM();

bool receiveEkRaRbSb(const std::vector<unsigned char>& Ra, const std::vector<unsigned char>& Rb, const std::vector<unsigned char>& Sb);

bool receiveEkRb(const std::vector<unsigned char>& Rb);

std::vector<unsigned char> sendEkRb();

void makeKey();

std::vector<unsigned char> encrypt(std::string message);

std::string decrypt(const std::vector<unsigned char>& cipher);

private:

const int keySize = 16; // AES-128

std::vector<unsigned char> publicKey; // K'

std::vector<unsigned char> sharedKey; // P

std::vector<unsigned char> sessionKey; // K

std::vector<unsigned char> randomMessage; //Ra or Rb

std::vector<unsigned char> secretPart; //Sa or Sb

std::vector<unsigned char> recRandMessage;

std::vector<unsigned char> recSecretMessage;

void generateRandVec(std::vector<unsigned char>& vec);

std::vector<unsigned char> encryptAES(const std::vector<unsigned char>& message, const std::vector<unsigned char>& key);

std::vector<unsigned char> decryptAES(const std::vector<unsigned char>& encryptedMessage, const std::vector<unsigned char>& key);

};

**Функция соединения клиента и сервера:**

bool establishConection(Client& server, Client& client) {

client.syncSharedKey(server);

client.receiveEpK(server.sendEpK());

server.receiveEpEkK(client.sendEpEkK());

client.receiveEkRaSa(server.sendEkRa(), server.sendEkSa());

if (!server.receiveEkRaRbSb(client.sendEkRecRandM(), client.sendEkRa(), client.sendEkSa()))

return false;

if (!client.receiveEkRb(server.sendEkRb()))

return false;

server.makeKey();

client.makeKey();

return true;

}

**Главная функция:**

int main() {

if (OPENSSL\_init\_crypto(0, nullptr) != 1) {

std::cerr << "OpenSSL initialization failed." << std::endl;

return 1;

}

Client server("Alice");

Client client("Bob");

if (establishConection(server, client)) {

std::cout << "\nSuccesfull conection\n\n";

std::cout << "Enter message to encrypt: ";

std::string message, result;

std::cin >> message;

std::vector<unsigned char> ciphertext = server.encrypt(message);

result = std::string(ciphertext.begin(), ciphertext.end());

std::cout << "Encrypted text: " << result << "\n";

result = client.decrypt(ciphertext);

std::cout << "Decrypted text: " << result << "\n";

}

else {

std::cerr << "Error with conection.\n";

}

return 0;

}

**Вывод:** в ходе лабораторной работы реализовал алгоритм 2обмена сеансовым ключом