|  |
| --- |
| Описание: https://lh6.googleusercontent.com/QcftzNtI05T0Y6fjdSh1Rr2rt8oqZ1IvnLvbn1jLJ7CCyteVir3k-xBLv4SL1wAgWJsRhmmJSR0UW-RP63_GQenE4vVWv05BRoZTsmIcBccVTnfxwmsnNMvjg599x9SqZd8E3dkd |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждениевысшего образования"МИРЭА - Российский технологический университет"РТУ МИРЭА |

Институт информационных технологий

Кафедра вычислительной техники

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3**

**по дисциплине**

**«Защита информации»**

Студент группы ИВБО-07-15 Титовский К. Э.

Принял Воронков С. О.

Москва 2018

Постановка задачи:

Разработать программу, реализующую протокол парольной аутентификации SRP.

Ход выполнения работы:

1. С помощью OpenSSL генерируется большое простое число N, при этом параметр g равен 2. В данной реализации используется SRP версии 6a, поэтому параметр k = H(N, g). В качестве хэш-функции используется SHA-256.
2. Создается объект класса Сервер, в его конструктор передаются N, g и k.
3. Чтобы войти в систему, клиенту нужно сначала зарегистрироваться. Он вводит логин (I) и пароль (p), который передаются в новый объект класса Клиент вместе с N, g и k.
4. Клиент генерирует s («соль», случайная строка), x (закрытый ключ), v (верификатор пароля) и отсылает I, s, v серверу. На этом регистрация заканчивается.
5. Начинается аутентификация. Клиент вводит логин (I) и пароль (p), который передаются в новый объект класса Клиент вместе с N, g и k.
6. Клиент генерирует a (случайное число), с помощью которого он сгенерирует A (публичный ключ).
7. Клиент отправляет I и A серверу. Сервер проводит проверку – если A не равно нулю, то он сохраняет его себе, иначе соединение прерывается.
8. Сервер генерирует b (случайное число), с помощью которого он сгенерирует B (публичный ключ).
9. Сервер отсылает клиенту s и B. Клиент проводит проверку – если B равно нулю, то соединение прерывается.
10. И клиент, и сервер вычисляют параметр u. Если u равен нулю, то соединение прерывается.
11. Клиент вычисляет свой закрытый ключ, потом общий ключ сессии S и его хэш K.
12. Сервер вычисляет общий ключ сессии S и его хэш K.
13. У обеих сторон появился общий ключ сессии, но для завершения аутентификации им необходимо свериться, что их ключи совпадают. Поэтому, сначала клиент вычисляет свое подтверждение M и отсылает его серверу. Сервер вычисляет свое подтверждение M и сравнивает его с клиентским M. Если они равны, то сервер вычисляет хэш R и отсылает его клиенту.
14. Клиент вычисляет свой хэш R и сравнивает его с серверным R.

Вывод:

Алгоритм работает корректно. Если пользователь при аутентификации вводит правильные логин и пароль, то R сервера и R клиента совпадают, соответственно клиент и сервер те, за кого себя выдают.