МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра теоретических основ компьютерной безопасности и криптографии

Протокол Ньюмана - Стабблбайна

ОТЧЁТ
ПО ДИСЦИПЛИНЕ
«КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ»

студента 5 курса 531 группы
специальности 10.05.01 Компьютерная безопасность
факультета компьютерных наук и информационных технологий
Никитина Арсения Владимировича

Преподаватель
аспирант _____ Р. А. Фарахутдинов
подпись, дата

Саратов 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В данной лабораторной работе поставлена задача рассмотреть протокол аутентификации и обмена ключами, на основе полученного материала реализовать Протокол Ньюмана – Стабблбайна.

1 Цель работы и порядок ее выполнения

Цель работы – изучение протокола Ньюмана – Стабблбайна и его реализация.

Порядок выполнения работы:

1. Разобрать, что такое симметричный протокол аутентификации и обмена ключами с использованием доверенной стороны;
2. Разобрать протокол Ньюмана – Стабблбайна;
3. Произвести программную реализацию.

2 Теоретическая часть

Протокол Ньюмана – Стабблбайна – симметричный протокол аутентификации и обмена ключами с использованием доверенной стороны. Является усовершенствованной версией протокола Yahalom. Особенностью протокола является отсутствие необходимости синхронизации часов у сторон, а также возможность повторной аутентификации без использования промежуточной стороны.

Криптографический протокол Ньюмана-Стабблбайна для удостоверения подписи и обмена ключами был впервые опубликован в 1993 году. Протокол является модификацией протокола Yahalom и разработан в Массачусетском технологическом институте (MIT) Клифордом Ньюманом и Стюартом Стабблбаном.

При симметричном шифровании, предполагается, что секретный ключ, который принадлежит клиенту, известен только ему и некоторой третьей доверенной стороне – серверу аутентификации. В процессе сеанса протокола клиенты Алиса и Боб получают от сервера аутентификации Трента новый секретный сессионный ключ для шифрования взаимных сообщений в текущем сеансе связи. Данный протокол «перекладывает» генерацию нового сессионного ключа на сторону доверенного центра.

Описание алгоритма:

Алиса и Боб хотят безопасно обмениваться сообщениями, находясь на различных концах сети. Предполагается, что каждому пользователю Трент

выделяет отдельный секретный ключ, и перед началом работы протокола все ключи уже находятся у пользователей.

1. Первым сообщение Алиса инициирует сеанс, пересылая Бобу свой идентификатор A и некоторое случайное число R_A: $Alisa \rightarrow \{A, R_A\} \rightarrow Bob$
2. Боб объединяет идентификатор Алисы, ее случайное число и метку времени, шифрует сообщение общим с Трентом ключом и посылает его Тренту, добавив свой идентификатор и случайное число Боба: $Bob \rightarrow \{B, R_B, E_B(A, R_A, T_B)\} \rightarrow Trent$
3. Трент генерирует сеансовый ключ K, а затем создает два сообщения. Первое включает идентификатор Боба, случайное число Алисы, случайный сеансовый ключ, метку времени и шифруется общим для Трента и Алисы ключом. Второе состоит из идентификатора Алисы, сеансового ключа, метки времени и шифруется общим для Трента и Боба ключом. Трент добавляет к ним случайное число Боба и отправляет Алисе: $Trent \rightarrow \{E_A(B, R_A, K, T_B), E_B(A, K, T_B),\ R_B\} \rightarrow Alice$
4. Алиса извлекает K и убеждается, что R_A совпадает с тем, что было послано на этапе 1. Алиса отправляет Бобу два сообщения. Первое – это второе сообщение от Трента, зашифрованное ключом Боба. Второе – это случайное число Боба, зашифрованное сеансовым ключом: $Alice \rightarrow \{E_B(A, K, T_B), E_K(R_B)\} \rightarrow Bob$
5. Боб расшифровывает сообщение своим ключом и убеждается, что значения T_B и R_B не изменились. Если оба случайных числа и метка времени совпадают, то Алиса и Боб убеждаются в подлинности друг друга и получают секретный ключ.

Нет необходимости синхронизировать часы, так как метка времени определяется только по часам Боба и только Боб проверяет созданную им метку времени.

3 Практическая часть

На рисунках 1 – 2 приведены примеры работы программы.

Рисунок 1 – пример запуска программы

Рисунок 2 – пример запуска программы

ПРИЛОЖЕНИЕ А

Листинг программы

```
from cryptography.fernet import Fernet
import uuid
import time
import random

def encrypt(key, x):

    values = ''
    for i in x:
        if type(i) == bytes:
            values += str(i, 'utf-8')
        else:
            values += str(i)
        values += ','

    values = values[:-1]
    cipher = Fernet(key)
    return cipher.encrypt(bytes(values, "utf-8"))

def decrypted(key, enc):
    cipher = Fernet(key)
    decrypted = str(cipher.decrypt(enc), "utf-8")
    return decrypted.split(',')


def main():

    # Протокол
    # ШАГ 1
    a_number_len = int(input('Введите количество бит в случайном числе
Алисы: '))
    alice_random_number = random.randint(2 ** a_number_len, 2 **
(a_number_len + 1)- 1)
    alice_UUID = uuid.uuid4()
    print(f"Алисе выдан идентификатор: {alice_UUID}")

    A_key = Fernet.generate_key()
    print('1. Алиса генерирует R_a =', alice_random_number)
    step_1 = (str(alice_UUID), str(alice_random_number))
    print("\tАлиса отсылает Бобу {A, R_a} =", step_1, '\n')

    # ШАГ 2
    b_number_len = int(input('Введите количество бит в случайном числе
Боба: '))
    bob_random_number = random.randint(2 ** b_number_len, 2 **
(b_number_len + 1)- 1)
    bob_UUID = uuid.uuid4()
    print(f"Бобу выдан идентификатор: {bob_UUID}")

    bob_key = Fernet.generate_key()
    bob_salt = time.time()
    print('2. Боб генерирует R_b =', bob_random_number)
    print('\tМетка времени Боба T_b =', bob_salt)
```

```
    step_2 = (str(bob_UUID), str(bob_random_number), encrypt(bob_key,
(step_1[0], step_1[1], bob_salt)))
    print("Боб отсылает Тренту {B, R_b, E_b(A, R_a, T_b)} =", step_2,
'\n')

    # ШАГ 3
    new_key = Fernet.generate_key()
    decrypted_step_2 = decrypted(bob_key, step_2[2])
    print('3. Трент расшифровывает E_b(A, R_a, T_b)')
    print('\tA =', decrypted_step_2[0])
    print('\tR_a =', decrypted_step_2[1])
    print('\tT_b =', decrypted_step_2[2])
    print('Трент генерирует K =', new_key)
    step_3_1 = encrypt(A_key, (step_2[0], decrypted_step_2[1], new_key,
decrypted_step_2[2]))
    step_3_2 = encrypt(bob_key, (decrypted_step_2[0], new_key,
decrypted_step_2[2]))
    s3 = (step_3_1, step_3_2, step_2[1])
    print("Трент отсылает Алисе {E_a(B, R_a, K, T_b), E_b(A, K, T_b),
R_b} =", s3, '\n')

    # ШАГ 4
    decrypted_step_3 = decrypted(A_key, s3[0])
    print('4. Алиса расшифровывает E_a(B, R_a, K, T_b)')
    print('\tB =', decrypted_step_3[0])
    print('\tR_a =', decrypted_step_3[1])
    print('\tK =', decrypted_step_3[2])
    print('\tT_b =', decrypted_step_3[3])
    s4 = (s3[1], encrypt(decrypted_step_3[2], {s3[2]}))
    print("Алиса отсылает Бобу {E_b(A, K, T_b), E_k(R_b)} =", s4, '\n')

    decrypted_s4_1 = decrypted(bob_key, s4[0])
    decrypted_s4_2 = decrypted(bytes(decrypted_s4_1[1], 'utf-8'), s4[1])
    print('Боб расшифровывает E_b(A, K, T_b)')
    print('\tA =', decrypted_s4_1[0])
    print('\tK =', decrypted_s4_1[1])
    print('\tT_b =', decrypted_s4_1[2])
    print('Боб расшифровывает E_k(R_b)')
    print('\tR_b =', decrypted_s4_2[0], '\n', '\n')


    #Проверка подлинности#
    # ШАГ 1
    print('Проверка подлинности')

    a_number_len = int(input('Введите количество бит в случайном числе
Алисы: '))
    alice_random_number = random.randint(2 ** a_number_len, 2 **
(a_number_len + 1)- 1)

    print('1. Алиса генерирует R2_a =', alice_random_number)
    step_1 = (step_3_2, str(alice_random_number))
    print("\tАлиса отсылает Бобу {E_b(A, K, T_b), R2_a} =", step_1)

    # ШАГ 2

    b_number_len = int(input('Введите количество бит в случайном числе
```

```
Боба: '))
    bob_random_number = random.randint(2 ** b_number_len, 2 **
(b_number_len + 1)- 1)

    print('2. Боб генерирует R2_b =', bob_random_number)
    step_2 = (str(bob_random_number), encrypt(decrypted_s4_1[1],
{step_1[1]}))
    print("\tБоб отсылает Алисе {R2_b, E_k(R2_a)} =", step_2, '\n')

    # ШАГ 3
    decrypted_step_2 = decrypted(decrypted_step_3[2], step_2[1])
    print('3. Алиса расшифровывает E_k(R2_a)')
    print('\tR2_a =', decrypted_step_2[0])
    ss3 = encrypt(decrypted_s4_1[1], {step_2[0]})
    print("\tАлиса отсылает Бобу {E_k(R2_b)} =", ss3, '\n')

    decrypted_ss3 = decrypted(bytes(decrypted_s4_1[1], 'utf-8'), ss3)
    print('\tБоб расшифровывает E_k(R2_b)')
    print('\tR2_b =', decrypted_ss3[0])


if __name__ == "__main__":
    main()
```