Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Информационные сети. Основы безопасности

ОТЧЁТ

к лабораторной работе №3

на тему

**ИДЕНТИФИКАЦИЯ И АУТЕНТИФИКАЦИЯ ПОЛЬЗОВАТЕЛЕЙ. ПРОТОКОЛ KERBEROS**

Выполнил: студент гр. 253505

Сенько Н. С.

Проверил: ассистент кафедры

информатики

Герчик А. В.

Минск 2025

**СОДЕРЖАНИЕ**

[1 Цель работы 3](#__RefHeading___Toc1348_1888910669)

[2 Ход работы 4](#__RefHeading___Toc1350_1888910669)

[Заключение 5](#__RefHeading___Toc1352_1888910669)

[Приложение А (обязательное](#__RefHeading___Toc3241_1888910669)) и[сходный код программного продукта 6](#__RefHeading___Toc3243_1888910669)

# 1 ЦЕЛЬ РАБОТЫ

Целью данной лабораторной работы является изучение принципов идентификации и аутентификации пользователей с использованием протокола Kerberos. В ходе работы необходимо ознакомиться с архитектурой протокола, его основными компонентами и механизмами работы. Также предполагается реализовать простую симуляцию процесса аутентификации с использованием Kerberos, что позволит лучше понять его функционирование и применение в современных системах безопасности.

# 2 ХОД РАБОТЫ

Протокол Kerberos был разработан для обеспечения безопасной аутентификации пользователей в сетевых системах. Он использует централизованный подход, где Центр распределения ключей (KDC) играет ключевую роль в процессе аутентификации.

**2.1 Архитектура Kerberos**

Kerberos состоит из следующих основных компонентов:

– ­Клиент: Пользователь или приложение, запрашивающее доступ к сервису.

– Сервер: Сервис, к которому клиент хочет получить доступ.

– KDC (Центр распределения ключей): Доверенный сервер, который выдает ключи и билеты для аутентификации.

**2.2 Процесс аутентификации**

Процесс аутентификации в Kerberos включает несколько этапов:

– Запрос аутентификации (AS\_REQ): Клиент отправляет запрос на KDC с идентификационными данными.

– Ответ KDC (AS\_REP): KDC проверяет данные и отправляет клиенту зашифрованный сеансовый ключ и TGT (Ticket Granting Ticket).

– Запрос мандата (TGS\_REQ): Клиент использует TGT для запроса доступа к конкретному сервису.

– Ответ TGS (TGS\_REP): KDC отправляет клиенту мандат для доступа к запрашиваемому сервису.

Таким образом, в ходе выполнения лабораторной работы было реализовано и протестировано программное средство шифрования и дешифрования текстовых файлов двумя шифрами. В процессе разработки были изучены принципы работы выбранных алгоритмов шифрования, их особенности, преимущества и недостатки.

Программа была протестирована на различных входных данных, что позволило убедиться в её корректности и работоспособности. Кроме того, в ходе выполнения лабораторной работы была проведена отладка кода, исправлены возможные ошибки и оптимизированы некоторые участки программы для повышения её производительности.

# ЗАКЛЮЧЕНИЕ

В ходе работы была изучена архитектура и принципы работы протокола Kerberos, а также реализована его простая симуляция. Протокол Kerberos обеспечивает надежную аутентификацию пользователей в сетевых системах, используя централизованный подход и симметричное шифрование.

Были рассмотрены основные этапы аутентификации, включая запросы и ответы между клиентом и KDC. Реализованная симуляция позволила лучше понять механизмы работы протокола и его применение в современных системах безопасности.

Таким образом, проделанная работа способствовала углублению знаний в области аутентификации и идентификации пользователей, а также позволила применить на практике навыки программирования для реализации протоколов безопасности.

# Приложение А (обязательное)

# Исходный код программного продукта

import hashlib

import os

class KerberosServer:

def \_\_init\_\_(self):

self.clients = {}

self.services = {}

self.session\_keys = {}

def register\_client(self, client\_id, secret):

self.clients[client\_id] = secret

def register\_service(self, service\_id, secret):

self.services[service\_id] = secret

def authenticate(self, client\_id):

if client\_id in self.clients:

session\_key = os.urandom(16)

self.session\_keys[client\_id] = session\_key

# Здесь вместо настоящего шифрования мы просто возвращаем хеш от секретного ключа клиента

client\_secret = self.clients[client\_id]

return hashlib.sha256(client\_secret + session\_key).hexdigest()

else:

raise ValueError("Unknown client")

def validate\_request(self, client\_id, service\_id, authenticator):

if client\_id in self.clients and service\_id in self.services:

expected\_authenticator = hashlib.sha256(self.clients[client\_id] + self.session\_keys[client\_id]).hexdigest()

if authenticator == expected\_authenticator:

return True

return False

class Client:

def \_\_init\_\_(self, client\_id, secret, kerberos\_server):

self.client\_id = client\_id

self.secret = secret

self.kerberos\_server = kerberos\_server

def request\_access(self):

authenticator = self.kerberos\_server.authenticate(self.client\_id)

return authenticator

class Service:

def \_\_init\_\_(self, service\_id, secret, kerberos\_server):

self.service\_id = service\_id

self.secret = secret

self.kerberos\_server = kerberos\_server

def grant\_access(self, client\_id, authenticator):

if self.kerberos\_server.validate\_request(client\_id, self.service\_id, authenticator):

print("Access granted to client:", client\_id)

else:

print("Access denied for client:", client\_id)

# Пример успешной аутентификации

print("Пример успешной аутентификации:")

kerberos\_server = KerberosServer()

# Регистрация клиента и сервиса

kerberos\_server.register\_client("client1", b"client\_secret")

kerberos\_server.register\_service("service1", b"service\_secret")

# Успешный сценарий

client = Client("client1", b"client\_secret", kerberos\_server)

authenticator = client.request\_access()

service = Service("service1", b"service\_secret", kerberos\_server)

service.grant\_access("client1", authenticator) # Ожидается успешный доступ

print("\nПример неуспешной аутентификации:")

# Пример неизвестного клиента

try:

unknown\_client = Client("unknown\_client", b"client\_secret", kerberos\_server)

authenticator = unknown\_client.request\_access()

except ValueError as e:

print("Ошибка:", e) # "Unknown client"

# Пример неверного аутентификатора

fake\_authenticator = "invalid\_authenticator"

service.grant\_access("client1", fake\_authenticator) # Ожидается отказ в доступе

# Пример неизвестного сервиса

unknown\_service = Service("unknown\_service", b"service\_secret", kerberos\_server)

unknown\_service.grant\_access("client1", authenticator) # Ожидается отказ в доступе