Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2025 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи захищеної мережі на основі PPTP-протоколу”

Виконав здобувач вищої освіти
IV курсу, групи КІ-21-1
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Назаревський М.В.
«____» _____ 2025 р.

Керівник проекту
доктор філософії (PhD)
_____ Усік П.С.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

Галузь знань ._12 _____*“*Інформаційні технології*”*_____

Спеціальність *123 “*Комп’ютерна інженерія*”*

Освітньо-професійна (освітньо-наукова) програма _*“*Комп’ютерна інженерія*”*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2025 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Назаревському Максиму Володимировичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи захищеної мережі на основі PPTP-протоколу*

2. Керівник роботи *Усік Павло Сергійович, доктор філософії (PhD)*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 46-02 від 17.01.2025 року

3. Строк подання студентом роботи до захисту *23.05.2025 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи захищеної мережі на основі PPTP-протоколу*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи в промислову експлуатацію.

6. Висновки

6. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Дата видачі завдання « 17 » січня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2025 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2025 р.	
3.	Розробка моделі компонента	20.03.2025 р.	
4.	Розробка структур даних	25.03.2025 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2025 р.	
6.	Програмування алгоритмів	10.04.2025 р.	
7.	Оформлення ПЗ	17.04.2025 р.	
8.	Попередній захист роботи	23.05.2025 р.	

Дата видачі завдання
« 17 » січня 2025 р.

Підпис керівника
_____ Усік П.С.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Назаревський М.В.
(прізвище та ініціали)

АНОТАЦІЯ

Назаревський М.В. Програмне забезпечення системи захищеної мережі на основі PPTP-протоколу. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи захищеної мережі на основі PPTP-протоколу.

Метою розробки є програмне забезпечення системи захищеної мережі на основі PPTP-протоколу.

Результат роботи – програмна реалізація системи захищеної мережі на основі PPTP-протоколу.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ з ОС Windows 10/11.

Програму розроблено в середовищі Python.

Ключові слова: комп'ютерна інженерія, PPTP-протокол

ABSTRACT

Nazarevsky M.V. Software for a secure network system based on the PPTP protocol. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed that is intended for a secure network system based on the PPTP protocol.

The purpose of the development is software for a secure network system based on the PPTP protocol.

The result of the work is a software implementation of a secure network system based on the PPTP protocol.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software are provided.

The program can be used on a PC with Windows 10/11 OS.

The program was developed in the Python environment.

Keywords: computer engineering, PPTP protocol

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

AH	–	автентифікуючий заголовок
CA	–	сертифікаційне співтовариство
DES	–	Data Encription Standard
DoS	–	атака "Відмова в обслуговуванні"
DOI	–	область інтерпретації
ESP	–	Інкапсуляція зашифрованих даних
HTTPS	–	зашифрований http
IAB	–	координаційна рада мережі Internet
IDS	–	система, яка автоматизує процес перегляду подій
IETF	–	проблемна група проектування Internet
IKE	–	протокол обміну ключами за замовчуванням для ISAKMP
IKMP	–	протоколу керування ключами прикладного рівня
IPsec	–	комплект протоколів захисту інформації по IP
ISAKMP	–	механізми узгодження атрибутів використовуваних протоколів
ISP	–	постачальник послуг Internet
MAC	–	коди на перевірку цілісності
MD5	–	дайджест повідомлення
Oakley	–	сесійні ключі на комп'ютери мережі Інтернет
PFS	–	ідеальна пряма безпека
PRF	–	псевдовипадкова функція
SA	–	Security Association
SKIP	–	команда підготовки наступної команди
SPI	–	індекс параметрів безпеки
SPD	–	база даних політики безпеки
SSL	–	протокол захищених сокетів
TCP	–	транспортний протокол
VPN	–	віртуальні приватні мережі

ВСТУП

Актуальність теми. Протокол PPTP (Point-to-Point Tunneling Protocol) дозволяє створювати безпечні канали для обміну даними за різними мережевими протоколами – IP, IPX або NetBEUI. Ці протоколи об’єднуються за допомогою протоколу PPTP у пакети протоколу IP, за допомогою яких вони передаються в зашифрованому вигляді через будь-яку мережу TCP/IP. Вихідний кадр PPP інкапсульований, тому протокол PPTP можна віднести до класу протоколів інкапсуляції канального рівня на мережевому рівні.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи захищеної мережі на основі PPTP-протоколу.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем захищеної мережі на основі PPTP-протоколу.

– Дослідження системи захищеної мережі на основі PPTP-протоколу.

– Програмна реалізація системи захищеної мережі на основі PPTP-протоколу.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі захищеної мережі на основі PPTP-протоколу.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи захищеної мережі на основі PPTP-протоколу, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

PPTP працює на основі технології PPP (протокол точка-точка), яка створює безпечний тунель між вами та віддаленим сервером. Після безпечного встановлення з'єднання ваші дані шифруються в IP-конвертах і передаються через тунель до кінцевої точки. Потім ваші дані розшифровуються, щоб одержувач міг їх прочитати.

PPTP має два типи потоку даних:

– Пакети даних.

– Контрольні повідомлення.

Ваші передані дані перетворюються на IP-пакети, стару технологію шифрування, розроблену Microsoft – MPPE (Microsoft Point-to-Point Encryption) – 128-бітне шифрування. Керуючі повідомлення використовуються для запуску та завершення зашифрованого з'єднання.

1.2 Область застосування

Переваги PPTP:

– Висока швидкість з'єднання: PPTP забезпечує високу швидкість завдяки легшому шифруванню, що дозволяє швидко переміщати дані по мережі.

– Просте налаштування: протокол простий у налаштуванні, тому багато компаній використовують його для забезпечення віддаленого доступу до корпоративних ресурсів.

– Широка сумісність: PPTP, спочатку розроблений Microsoft, сумісний з більшістю пристроїв і операційних систем, включаючи Windows, Linux, Android і старіші версії iOS і macOS.

– Обмежена підтримка на нових системах: через обмеження безпеки останні версії iOS і macOS більше не підтримують PPTP. Однак якщо ви встановите VPN на своєму маршрутизаторі та підключите його до свого iPhone або Mac, ви зможете використовувати PPTP з цими пристроями.

Недоліки PPTP:

– Слабке шифрування та автентифікація: застарілі шифрування та автентифікація PPTP роблять його вразливим до атак, тому багато постачальників VPN припинили його використання.

– Обмежений обхід брандмауера: PPTP не має надійних функцій захисту від брандмауера, і його легко заблокувати, оскільки він покладається на порт TCP 1723, який можна виявити. За допомогою цього протоколу VPN ви не зможете обійти блокування та брандмауери VPN.

– Сумісність із старими маршрутизаторами: для протоколу PPTP потрібна функція «Passthrough», яка є в основному в старих моделях маршрутизаторів, що обмежує його сумісність із сучасними маршрутизаторами.

– Численні недоліки безпеки: PPTP має відомі вразливості, включаючи випадки, коли державні установи успішно розшифрували його трафік.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи захищеної мережі на основі PPTP-протоколу, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

NordVPN

Швидкий, безпечний і чудовий для потокового передавання – найкращий VPN для більшості людей:

– Кількість серверів: 7300+.

– Розташування серверів: 154 у 118 країнах.

– Максимальна кількість підтримуваних пристроїв: 10.

– Цілодобовий живий чат: так.

– 30-денна гарантія повернення грошей: так.

Ціни:

– NordVPN 2 роки: 3,09 доларів США/міс.

– NordVPN 1 рік: 4,99 доларів США/міс.

– NordVPN 1 місяць: 12,99 доларів США/міс.

Переваги:

– Дуже швидкі з'єднання.

– Відмінна конфіденційність.

– Розблоковує більшість потокових сайтів.

– Чудова підтримка клієнтів.

– Програми можуть бути складними для абсолютних новачків.

– Подорожчання при поновленні.

NordVPN охоплює всі бази та має відмінні облікові дані щодо конфіденційності. Нещодавно було вп’яте перевірено політику заборони реєстрації, яка підтверджує, що NordVPN не збирає дані про вашу активність.

NordVPN використовує галузевий стандарт шифрування, яке фактично неможливо зламати. Це означає, що ніхто не може бачити, що ви робите в Інтернеті – навіть уряд чи ваш інтернет-провайдер.

NordVPN є хорошим співвідношенням ціни та якості, враховуючи кількість функцій, які ви отримуєте. Довші плани обходяться набагато дешевше. Найдешевшим є 2-річний план, який коштує 3,09 доларів на місяць (83 долари всього). Це трохи дорожче, ніж Surfshark – найкращий вибір, якщо у вас обмежений бюджет, – але дешевше, ніж заклятий конкурент ExpressVPN.

NordVPN показав дуже хороші результати в процесі тестування швидкості, досягнувши понад 950 Мбіт/с. Більш цікавим є той факт, що під час використання в реальному світі рідко, якщо взагалі коли-небудь, виявляли, що це має помітний вплив на наш досвід перегляду.

Це йде рука об руку з передовою в своєму класі потоковою потужністю NordVPN. Під час нашого тестування VPN він отримав доступ до кожної глобальної бібліотеки Netflix, що робить його найкращим VPN для Netflix, а також Disney+, Amazon Prime Video та регіональних служб, як-от BBC iPlayer.

Вищі плани передплати включають деякі якісні додаткові функції, найкориснішою з яких є Threat Protection Pro. Це інструмент захисту від зловмисного програмного забезпечення та веб-переглядача, який працює, навіть коли VPN вимкнено. Він визнаний найкращим у своєму роді, і під час нашого повсякденного використання NordVPN він виявився дуже зручним для захисту від шкідливих сайтів.

Що можна покращити: тут небагато, на що скаржитися, але якщо ви зовсім не технічний і хочете мати найпростіший досвід, безліч функцій NordVPN може вас відштовхнути. Якщо ви просто хочете бачити кнопку ввімкнення/вимкнення, виберіть натомість ExpressVPN.

Наша єдина справжня скарга на NordVPN – це величезне підвищення цін, яке ви побачите після закінчення початкової підписки. Якщо ви дозволите автоматичне поновлення підписки, ваш 2-річний план, за який ви заплатили 83

долари США, перетвориться на 1-річний поточний план, який коштує 139 доларів США. Це приблизно 11,60 доларів на місяць. Наша порада? Не дозволяйте цьому статися. Скасуйте та знову підпишіться за розумною ціною.

Купуйте NordVPN, якщо:

– Ви хочете мати багато додаткових функцій. Окрім основної VPN, NordVPN пропонує захист від зловмисного програмного забезпечення за допомогою Threat Protection Pro, Meshnet, темного веб-монітора тощо.

– Ви любите стрімінг. NordVPN розблокував усі потокові сайти, на яких його тестували, і це гарантує вам безпеку під час перегляду.

– Ви бажаєте найкращого з найкращого. Згідно з нашим тестуванням, NordVPN – це найповніша доступна VPN. Ви не можете зробити краще, ніж це.

Не купуйте NordVPN, якщо:

– У вас обмежений бюджет. Є кілька дешевших VPN, які дають NordVPN хороші результати за свої гроші – Surfshark і Private Internet Access є двома дешевими видатними.

– Ти технофоб. NordVPN не надто складно використовувати, але ExpressVPN є кращим вибором, якщо вам потрібна простіша альтернатива.

Surfshark

Найкращий недорогий VPN:

– Кількість серверів: 3200+.

– Розташування серверів: 141 у 100 країнах.

– Максимальна кількість підтримуваних пристроїв: необмежена.

– Цілодобовий живий чат: так.

– 30-денна гарантія повернення грошей: так.

Ціни:

– Surfshark 24 місяці: 1,99 доларів США/міс.

– Surfshark 12 місяців: 3,19 доларів США/міс.

– Surfshark 1 місяць: 15,45 доларів США/міс.

Переваги:

– Доступна вступна пропозиція.

– Простий у використанні.

– Чудові функції конфіденційності.

– Дуже швидкий і хороший для потокової передачі.

– Блокування зловмисного програмного забезпечення та фішингу працює погано.

– Не так багато налаштувань.

Що нам подобається: Surfshark – це найдешевший VPN, включений у цей посібник, і, на відміну від деяких інших вигідних постачальників, тут було зроблено небагато жертв. Завдяки поточній акції до 30 квітня вартість Surfshark починається від 1,99 дол. США на місяць (53 дол. США за все), що значно дешевше, ніж NordVPN (3,09 дол. США на місяць) і ExpressVPN (4,99 дол. США на місяць). Це якісний VPN за напрочуд низькою ціною.

Surfshark належить Nord Security, материнській компанії NordVPN, тож ви можете бути впевнені, що він дотримується таких же високих стандартів, коли мова йде про конфіденційність. У тестуванні все було в порядку. Його політика конфіденційності повністю перевірена. Цікаво, що нещодавно компанія зареєструвала патент на вдосконалення зашифрованих повідомлень. Якщо у вас є кришталева куля, будь ласка, повідомте нас, якщо побачите додаток, схожий на Signal, від Surfshark у майбутньому.

У планах з вищою ціною ви отримуєте такі додаткові функції, як антивірус, альтернативний ідентифікатор, альтернативний номер та інструмент для видалення особистих даних із Incogni, але справжня назва гри Surfshark – це швидкість і потокове передавання.

Surfshark був найшвидшим VPN загалом у останньому раунді тестування, і, хоча він може коливатися, Surfshark ніколи не працює млявим. також виявили, що це швидко при використанні старішого протоколу OpenVPN. Нас це ніколи не тримало в реальному світі.

З точки зору потокового передавання Surfshark не поступається NordVPN. Іншими словами, у вас не виникне проблем дивитися все, що завгодно, де б ви не були.

Що можна покращити: Як і у випадку з NordVPN, ціни на Surfshark стрімко зростають, якщо ви дозволите йому автоматичне поновлення. Якщо бути точним, він підскочить до 79 доларів США за 1 рік, тобто приблизно 6,60 доларів США на місяць. Нам не подобається ця практика, але майже кожен VPN так використовує. Не дозвольте цьому статися з вами.

Якщо ви експерт, вам може знадобитися VPN з більшою можливістю налаштування, як-от Private Internet Access або Proton VPN. Surfshark орієнтований на більш звичайних користувачів, що робить його простим у використанні, але трохи спрощеним для досвідчених користувачів.

Купуйте Surfshark, якщо:

– Ви любите вигідні пропозиції. Surfshark пропонує найкращу вартість серед усіх розглянутих нами послуг VPN.

– У вас дуже швидкий інтернет. Як найшвидший постачальник у останньому раунді тестування швидкості, Surfshark захистить вас, не сповільнюючи роботу.

– Ти великий стример. Як і NordVPN, Surfshark провів чисту перевірку в останньому раунді потокових тестів.

Не купуйте Surfshark, якщо:

– Ви досвідчений користувач. Орієнтація на зручність для початківців означає, що Surfshark залишив деякі додаткові технічні налаштування.

– Ви хочете покладатися на вбудовану технологію блокування шкідливих програм. Інструмент CleanWeb від Surfshark не такий ефективний, як Threat Protection від NordVPN або Threat Manager від ExpressVPN.

ExpressVPN

Простий і ефективний – ExpressVPN – найкращий VPN для початківців:

– Кількість серверів: 3000+.

– Розташування серверів: 160 у 105 країнах.

– Максимальна кількість підтримуваних пристроїв: 8.

– Цілодобовий живий чат: так.

– 30-денна гарантія повернення грошей: так

Ціни:

– ExpressVPN 24 міс: 4,99 доларів США/міс.

– ExpressVPN 12 міс: 6,67 доларів США/міс.

– ExpressVPN 1 місяць: 12,95 доларів США/міс.

Переваги:

– Провідна у своєму класі конфіденційність.

– Розблоковує безліч потокових сайтів.

– Дуже простий у використанні.

– Безліч додаткових інструментів.

– Досить дорого.

– Обмежене налаштування.

Що нам подобається: ExpressVPN добре відомий тим, що займає передові позиції у сфері конфіденційності VPN. Це був один із перших постачальників, який запровадив постквантове шифрування, і навіть створив власний протокол Lightway, щоб усунути недоліки в існуючих варіантах. Програми мають усі необхідні функції, такі як перемикач вимкнення та розділене тунелювання. Їх багато разів перевіряли, і, загалом, вони відчувають себе дуже добре складеними.

ExpressVPN швидкий, але не такий швидкий, як Surfshark, а також чудовий для потокового передавання. Однак справжній козир ExpressVPN полягає в тому, що він поєднує все це найпростішим способом. Відкрийте програму, і ви побачите лише кнопку ввімкнення та вимкнення. Ось і все. Для більшості людей це все, що потрібно, і прибирання додаткових речей вносить ясність у досвід.

Говорячи про це, він має масу додатків. Хоча більшість конкурентів пропонують рівні передплати та встановлюють ціну за кожен додатковий інструмент, ExpressVPN надає вам майже все одразу. Включено Identity Defender,

який включає видалення особистих даних, страхування від крадіжки посвідчення особи та сканер кредитних даних. Ви також отримаєте менеджер паролів і блокувальник реклами та зловмисного програмного забезпечення. Це справжній універсальний магазин конфіденційності в Інтернеті.

Дещо не пов'язане конкретно з VPN – але все ж цікаве – це розширення ExpressVPN на ринок eSIM з holiday.com. Це явно мається на увазі Saily від конкурента NordVPN і свідчить про те, що команда Red все ще впроваджує інновації.

Що можна покращити: Усі ці додаткові переваги мають свою ціну. Незважаючи на те, що нещодавно ExpressVPN трохи знизив ціни до 4,99 доларів США на місяць (139 доларів США за все), він усе ще дорожчий. Якщо ви будете використовувати все, що постачається з ним, це набагато краще, ніж NordVPN, але якщо вам потрібна лише VPN і нічого більше, є дешевші варіанти.

Як і Surfshark, ExpressVPN зосереджується на простоті. Це означає, що ви не можете внести багато змін у роботу VPN. Не чекайте багатьох технічних додаткових можливостей, таких як переадресація портів.

Нещодавно бачили повідомлення на форумах і Reddit про те, що деякі оновлення, зокрема Lightway Turbo, мають помилки. Це зовсім не схоже на колишній ExpressVPN. Хоча залишаємося без суджень, оскільки самі з ними не стикалися, уважно стежимо за процесом, особливо після останнього раунду звільнень.

Купуйте ExpressVPN, якщо:

– Ви будете використовувати всі бонусні функції. ExpressVPN поставляється в комплекті з менеджером паролів, блокувальником реклами, відстеженням і зловмисним програмним забезпеченням, моніторингом витоку особистих даних, страхуванням від крадіжки ідентифікаторів і видаленням брокера даних.

– Ви шукаєте найпростіший VPN. ExpressVPN робить захист в Інтернеті неймовірно простим.

– Вам потрібна передова конфіденційність. ExpressVPN незмінно є однією з перших VPN, яка займається «наступною великою справою» у сфері конфіденційності.

Не купуйте ExpressVPN, якщо:

– Вам просто потрібна VPN. Безліч додаткових можливостей ExpressVPN виправдовує його ціну, але якщо ви не скористаєтеся ними, ви витратите понад шанси.

– Ти любиш майструвати. Так само, як Surfshark, ExpressVPN не пропонує багато варіантів налаштування.

Proton VPN

Чудова конфіденційність, добре підходить для потокового передавання, але мало нових інструментів для гри:

– Кількість серверів: 12 000+.

– Розташування серверів: 144 у 117 країнах.

– Максимальна кількість підтримуваних пристроїв: 10.

– Підтримка живого чату: так.

– 30-денна гарантія повернення грошей: так

Ціна:

– Proton VPN 24 місяці ТГ: 3,59 доларів США/міс.

– Proton VPN 12 місяців ТГ: 3,99 доларів США/міс.

– Proton VPN 1 місяць TG: 9,99 доларів США/міс.

Переваги:

– Чудові розширені функції конфіденційності.

– Корисна безкоштовна версія.

– Дуже широке поширення серверів.

– Відносно дорого.

– Трохи застій в плані оновлень.

Що нам подобається: Proton VPN – це добре відома та заслужена довіра VPN, яка є частиною ширшого набору продуктів, орієнтованих на

конфіденційність, від швейцарської компанії Proton. Його партнерами є Proton Mail, Proton Pass, Proton Drive і Proton Calendar. Загалом це ефективна мережа VPN, і ті, хто надає перевагу конфіденційності, оцінять такі унікальні функції, як Secure Core та Alternative Routing – остання з яких розроблена для уникнення цензури в репресивних країнах.

Для звичайних користувачів Proton VPN також чудово підходить для потокового передавання. Це розблокувало все, що пробували під час тестування. Це також швидко. Він забезпечує швидкість, рівну NordVPN і Surfshark при використанні найближчого сервера. Нещодавнє оновлення програми також вирішило деякі наші проблеми, пов'язані зі складністю.

Найбільшою перевагою Proton VPN є вищезгадана екосистема супутніх продуктів, які доступні зі знижкою, якщо ви підписалися на тарифний план «Необмежений». Це коштує 9,99 доларів на місяць за 2-річним планом (240 доларів за все), і це значна економія порівняно з ціною лише за VPN у 4,99 доларів на місяць. Читачі Tom's Guide можуть отримати додаткову знижку на VPN, зменшивши її до 3,59 доларів США на місяць (86 доларів США).

На відміну від інших постачальників VPN, Proton VPN дуже відкритий і часто співпрацює з іншими компаніями, що займаються захистом конфіденційності. Наприклад, нещодавно він співпрацює з безпечним браузером Vivaldi, а його програми з відкритим кодом означають, що він нічого не приховує від зовнішнього світу.

Що можна покращити: У вакуумі Proton VPN є дуже хорошим продуктом, але для більшості людей немає нічого, що робить його кращим вибором, ніж конкуренти далі на цій сторінці. Швидкість хороша, але на міжміських з'єднаннях вона не така швидка, як NordVPN або Surfshark. Потокове передавання майже те саме – воно здатне, але не краще, ніж дешевші альтернативи.

Також слід зазначити, що хоча багато провайдерів вищого рівня часто додають нові функції, Proton VPN не оновлював те, що пропонує протягом

тривалого часу. Можна стверджувати, що він має все необхідне, але ніхто не заперечує, що інші VPN пропонують більше за меншу суму.

Купуйте Proton VPN, якщо:

– Ви живете в країні з жорсткою цензурою. Proton VPN здобув собі ім'я як прихильник безкоштовного Інтернету для всіх і дуже ефективний в ухиленні від цензурних заходів.

– Ви використовуєте інші продукти Proton. Не дивно, що Proton VPN добре інтегрується з усіма іншими програмами Proton.

– Ви цінуєте конфіденційність понад усе. Унікальні функції конфіденційності Proton VPN, такі як Secure Core, чудово підійдуть, якщо ви прагнете залишатися максимально конфіденційними.

Не купуйте Proton VPN, якщо:

– Вам подобаються блискучі нові функції. Proton VPN у жодному разі не має недостатньої потужності, але він рідко отримує нові іграшки, якими можна грати.

– Ви шукаєте вигідну пропозицію. Proton VPN коштує так само, як ExpressVPN, що робить його одним із найдорожчих варіантів на цій сторінці (хоча читачі Tom's Guide отримують знижку).

Private Internet Access

Найкращий VPN для технарів:

– Кількість серверів: 10 000+.

– Розташування серверів: 151 у 91 країні.

– Максимальна кількість підтримуваних пристроїв: необмежена.

– Цілодобовий живий чат: так.

– 30-денна гарантія повернення грошей: так

Ціни:

– Private Internet Access 24 місяці: 2,03 долара США/міс.

– Private Internet Access 1 місяць: 11,99 доларів США/міс.

Переваги:

– VPN з широкими можливостями налаштування.

– Чудово підходить для завантаження торрентів.

– Розумна ціна з автоматичним оновленням.

– Забагато для початківців.

– Не так швидко, як інші.

Що нам подобається: Private Internet Access створено для більш досвідчених користувачів VPN. Порівняно з іншими чотирма провайдерами тут, PIA пропонує набагато більше налаштувань із перенаправленням портів, спеціальним MTU, спеціальним DNS, регульованим шифруванням, автоматизацією підключення та багатьма іншими. Якщо нічого з цього не мало сенсу, швидше за все, PIA не підходить для вас, але якщо так, це єдина високоякісна VPN, яка пропонує таку глибину досвіду.

Потокове передавання також є сильною стороною – воно розблокувало всі сторони, з якими його тестували, – і оцінюємо його як найкращий торрент-сервер VPN завдяки безлічі функцій, згаданих вище. Програми, хоча й дещо складні, досить доступні. Існує спеціальне програмне забезпечення PIA для більшості поширених пристроїв.

Дивно, але PIA також дуже дешевий, коштуючи 2,19 доларів США на місяць (57 доларів за все), що дивно, враховуючи, скільки ви отримуєте. Ще кращим є той факт, що ваш план автоматично поновлюватиметься за подібною ціною. Отже, не потрібно хвилюватися про те, що вас спіймають.

Що можна покращити: Зрозуміло, що PIA не підійде більшості людей. Це просто занадто технічно. Хоча додатки чудово справляються з роботою, роблячи його зручним, вони далеко не такі досконалі, як NordVPN або ExpressVPN. Додаткові функції, ймовірно, налякають менш досвідчених людей, і для них є набагато кращий вибір далі на цій сторінці.

Швидкість підключення також не надто вражаюча. У той час як усі інші VPN на цій сторінці під час нашого тестування досягли принаймні 900 Мбіт/с,

максимальна швидкість PIA – 436 Мбіт/с. Це все ще набагато швидше, ніж базове підключення до Інтернету більшості людей, але якщо ви платите за гігабітний Wi-Fi, PIA не зможе встигати.

Купуйте PIA, якщо:

– Ви знаєте, що робите. Private Internet Access має найбільше настроюваних функцій з усіх представлених тут VPN.

– Ви часто торрентуєте. Такі функції, як перенаправлення портів, дозволяють налаштувати торрент-файли.

– Ви хочете автоматизувати свою VPN. PIA дозволяє налаштувати правила автоматизації, тобто вмикається щоразу, коли ви відкриваєте певну програму.

Не купуйте PIA, якщо:

– Вам потрібен простий VPN. Налаштування коштує дорого, а PIA складніша, ніж багато інших VPN.

– У вас супершвидкий інтернет. PIA не встигає за швидкістю з'єднання порівняно з іншими VPN на цій сторінці.

Які мережі VPN найкраще підходять для конфіденційності:

1. NordVPN – найкращий універсальний пакет конфіденційності.

2. Proton VPN – розширені функції для тих, хто в групі ризику.

3. ExpressVPN – чудова конфіденційність, проста у використанні.

Коли перевіряємо VPN, більшу частину нашого часу витрачаємо на перевірку їх конфіденційності. Вибрати найприватніший VPN серед провайдерів на цій сторінці складно, оскільки ніколи не рекомендуємо VPN, який не пропонує чудового захисту. Тим не менш, є кілька видатних.

NordVPN має гарну історію захисту своїх користувачів. Єдина чорна пляма – злом сервера ще в 2018 році. Однак жодні дані користувача не були скомпрометовані, і це налаштувало NordVPN на шлях, щоб подібне більше ніколи не повторилося.

Майже кожна програма та частина її інфраструктури були перевірені незалежною сторонньою стороною, а використовуване шифрування є

першокласним. Він також запровадив квантове шифрування (дізнайтеся більше в поясненні квантового шифрування) та протокол NordWhisper, що руйнує цензуру.

Proton VPN також сильний у цій сфері та має низку унікальних інструментів конфіденційності. Найцікавіша система Secure Core. Proton використовує надзахищені сервери в трьох місцях: Ісландія, Швейцарія та Швеція. Використовуючи Secure Core, ви спочатку пройдете один із них, перш ніж підключитися до вибраного вами розташування сервера. Це гарантує, що навіть якщо кінцевий сервер буде зламано – малоймовірно, але можливо – вашу справжню IP-адресу бачитиме лише сервер Secure Core.

Нарешті, ExpressVPN може бути дорогим, але ці гроші повертаються в продукт. Хоча він має меншу кількість серверів, близько 3000, кожен із цих серверів має високу якість і об'єктивно кращий, ніж декілька серверів нижчої якості. Він також провів найбільше зовнішніх перевірок серед усіх VPN, які перевірили, з чудовими результатами в кожному.

Які мережі VPN найкраще підходять для потокового передавання:

1. NordVPN – найкращий потоковий VPN загалом.
2. Surfshark – найкращий дешевий потоковий VPN.
3. ExpressVPN – найпростіший потоковий VPN.

У тестуванні потокової мережі VPN кожна VPN на цій сторінці змогла розблокувати кожну перевірену службу потокової передачі. Однак у цих результатах є невеликий нюанс.

У кількох тестах Netflix виявив, що використовуємо ExpressVPN. Це означало, що не могли транслювати те, що хотіли, і довелося змінити сервер. Отже, хоча ExpressVPN все ще здатний розблокувати ті самі сайти, NordVPN і Surfshark є трохи надійнішими.

Однак все одно рекомендуємо ExpressVPN замість приватного доступу до Інтернету та Proton VPN. Це пов'язано з тим, що PIA не може зрівнятися зі швидкістю ExpressVPN, NordVPN і Surfshark, а Proton VPN просто не має тієї ж

історії пріоритезації потокового передавання. У випадку Proton VPN цей рейтинг цілком може змінитися, якщо побачимо, що ці відмінні результати потокового передавання збережуться протягом наступного року або близько того.

Які мережі VPN найкраще підходять для уникнення цензури:

1. Proton VPN – довга історія захисту від цензури.

2. Astrill VPN – спеціально створений для розблокування Інтернету в Китаї.

3. ExpressVPN – найкращий універсальний варіант для доступу до безкоштовного Інтернету.

Перевірити, наскільки VPN ефективні для уникнення цензури, може бути складно. Проте, обговоривши контакти з усього світу та розуміючи, як працює цензура, можемо дати корисні поради.

ExpressVPN і Proton VPN вважаються найефективнішими з «великих» VPN для обходу державних обмежень. Це пов’язано з тим, що вони обидва вклали значні кошти в технологію блокування VPN. Протокол Lightway від ExpressVPN розроблено таким чином, щоб завжди бути обфускаваним, а протокол Stealth від Proton VPN так само спеціально створено, щоб уникнути блокування.

Проте Proton VPN, мабуть, має перевагу тут завдяки безкоштовному тарифному плану VPN. Це дозволяє будь-кому завантажувати та бути захищеним, не сплачуючи нічого, і хоча це обмежено, це чудовий інструмент. Він також надає дані для VPN-обсерваторії Proton, яка може діяти як провідник суспільних змін і вхідної цензури в усьому світі.

Однак у деяких країнах, зокрема в Китаї, стандартних VPN іноді недостатньо. Якщо ви вважаєте, що це так, Astrill VPN є шанованою службою, яка бачить велику користь для обходу Великого брандмауера. Зазвичай не рекомендуємо його як хороший універсальний вибір, оскільки він дорогий і поганий для потокового передавання. Але в огляді Astrill VPN знайшли його цілком придатним для використання. Це одна з найкращих VPN для Китаю.

Тестування швидкості VPN – складна справа, і завжди буде знімком у часі, а не вимірюванням з точністю до секунди. Однак наше тестування швидкості VPN є комплексним, і регулярно перевіряємо кожного постачальника, щоб переконатися, що не рекомендуємо VPN через відмінні результати.

Ми тестуємо на лінії 1 Гбіт/с (1000 Мбіт/с). Для довідки Speedtest.net повідомляє, що середня швидкість широкосмугового з'єднання в США становить близько 274 Мбіт/с, а середня швидкість мобільного інтернет-з'єднання в США – близько 164 Мбіт/с.

Усі провайдери на цій сторінці забезпечили максимальні значення вище цього – і враховуючи, що для потокової передачі 4K Netflix потрібно лише 25 Мбіт/с, жоден не сповільнить вас. Однак якщо ви платите за дуже швидке з'єднання, ви можете помітити зниження швидкості, якщо використовуєте Private Internet Access.

Які мережі VPN найкраще підходять для ігор:

1. NordVPN – найкраща в своєму класі конфіденційність і висока швидкість.

2. Surfshark – надшвидкий, із надійними з'єднаннями.

3. ExpressVPN – швидкий і простий у роботі.

Ігрові VPN мають бути всебічними – швидкість є важливою, але також абсолютна надійність і хороша конфіденційність для захисту від DDoS-атак.

Через це наші рекомендації майже такі ж, як і наш загальний рейтинг на цій сторінці: NordVPN – це золотий стандарт, Surfshark – чудова дешева альтернатива, а ExpressVPN – найкращий безпроблемний варіант.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Python – високорівнева мова програмування, яку називають другою за популярністю в світі. Її використовують для розробки вебзастосунків,

програмного забезпечення, машинного навчання. Python застосовують для вирішення робочих завдань у компаніях Google, Instagram, Facebook, IBM, NASA, Dropbox, Netflix та інших. Розробники цінують цю мову програмування за простоту у вивченні, ефективність та мультиплатформність.

Python – скриптова мова програмування з досить простим синтаксисом. Для розуміння достатньо порівняти принципи написання найпростішої програми, яка виводить на екран текстове повідомлення. Саме тому мова програмування Python більш доступна для новачків, а професіонали встигли адаптувати її для вирішення великої кількості завдань. Це мультиплатформне рішення, тому знання Python дає можливість працювати у різних сферах: від розробки мобільних застосунків до ігрової індустрії та штучного інтелекту.

У мови програмування динамічна типізація: є можливість передавати до функцій будь-який тип даних без попереднього вказання. Інтерпретованість дозволяє знаходити помилки у коді ще до повної збірки у робочий застосунок. При цьому Python дуже чітко дає зрозуміти, де та через що виникла помилка.

Це мова об'єктноорієнтованого програмування (ООП). Програмне забезпечення на Python оформлене у вигляді моделей, які можуть бути зібраними у пакети. Тип та структуру кожного об’єкта можна запитати під час виконання програми. Для кожного з об’єктів можна отримати всю інформацію щодо його внутрішньої структури. Окрім того:

– у мови логічний синтаксис, завдяки чому вихідний код легко читати та розуміти;

– гнучкість та масштабованість Python дозволяє адаптувати високорівневу логіку та розширяти складні застосунки, як тільки виникне така необхідність;

– розробка на Python у більшості випадків проходить швидше, ніж на інших мовах програмування;

– Python – інтерпретована мова програмування. Це значить, що код можна написати у будь-якому текстовому файлі на будь-якій платформі, і потім успішно запустити;

– у Python – колосальна спільнота однодумців. Тож будь-які складнощі конкретних розробників вирішуються колективно.

Проте є декілька особливостей, які можна віднести до недоліків. Це повільність (ця мова програмування хоч і універсальна, проте повільніша за інші), велика кількість ресурсів, необхідних для роботи та «прив'язаність» до системних бібліотек.

Мова програмування Python використовується у наступних сферах:

1. Розробка програмних застосунків будь-якого напрямку.

2. Розробка серверної частини мобільних застосунків (найпопулярніший напрямок).

3. Ігри. Багато сучасних ігор для комп'ютерів (наприклад, World of Tanks) частково чи повністю написані на Python.

4. Вбудовані системи для різних пристроїв. Дуже часто Python використовують для написання внутрішніх платформ управління банкоматами.

5. Скрипти та плагіни до уже реалізованих програм для автоматизації процесів чи створення інших рішень.

6. Тестування (автоматизація цього процесу).

7. Машинне навчання. – основна мова для написання алгоритмів і аналітичних застосунків у сфері Machine Learning.

Бібліотеки Python

Різні бібліотеки Python використовують для виконання конкретних завдань. Наприклад, Matplotlib підходить для відображення даних у двовимірній та тривимірній графіці. Pandas підходить для зручної роботи з даними. NumPy дозволяє створювати масиви та керувати ними. Requests використовується для веброзробки. OpenCV-Python відкриває можливості для обробки зображень з метою оптимізації систем «машинного зору».

Найвідоміші фреймворки для мови програмування Python

Фреймворки Python допомагають створити зручне та функціональне середовище для розробки. У них міститься набір інструментів, модулів та

бібліотек, корисних для виконання конкретних завдань. Це значно полегшує роботу: наприклад, дає змогу не витрачати час на розписування дій, які повторюються, а використати релевантний інструмент. Тож є можливість позбутися рутинних процесів та сконцентруватися на логіці проєкту.

Серед найпопулярніших фреймворків для Python:

– Django – найстаріший та найвідоміший. Створений для реалізації великих інтерактивних проєктів;

– Pyramid – зручний у налаштуваннях, і дає можливість реалізувати складні нестандартні ідеї;

– Web2py – підходить в першу чергу для вебзастосунків і може використовуватись на будь-яких архітектурах.

Популярні Python IDE

IDE або інтегровані середовища розробки – це програмне забезпечення, яке надає розробникам необхідні інструменти для написання, редагування, тестування та налаштування коду. Для розробки на Python найчастіше використовують IDE PyCharm, IDLE, Spyder та Atom.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи захищеної мережі на основі PPTP-протоколу.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі.

Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Протокол PPTP може бути дещо корисним, оскільки його слабке шифрування дозволяє підвищити швидкість мережі, але при цьому можна розблокувати геообмежений вміст. Однак наполегливо рекомендуємо використовувати інші протоколи VPN (наприклад, OpenVPN, IKEv2 або нещодавно випущений протокол WireGuard) під час роботи з онлайн-платежами та конфіденційними логінами (електронна пошта, соціальні мережі тощо).

Тим не менш, давайте перейдемо до того, що ви хочете знати. Далі наведено наш посібник, який пояснює, як налаштувати PPTP у Windows 10 вручну.

Існує багато причин, чому ваш комп’ютер може відмовлятися підключатися до вибраного PPTP VPN. Тому існує також багато можливих рішень. Ось що ви можете спробувати.

Перевірте підключення до Інтернету

Перш ніж спробувати щось інше, переконайтеся, що ваше веб-з’єднання працює належним чином. Переконайтеся, що ви відключені від PPTP VPN, і спробуйте переглянути веб-сторінки. Чи працює як треба? Ви можете відкривати сайти? Це повільніше, ніж зазвичай?

Якщо у вас виникли проблеми, рекомендуємо перезавантажити маршрутизатор. Потрібно відключити його від джерела живлення, зачекати близько 30 секунд, а потім знову увімкнути. Зачекайте кілька хвилин, поки ваш маршрутизатор знову завантажиться, і спробуйте знову підключитися до PPTP VPN.

Перевірте час свого комп’ютера

У деяких випадках перевірка того, що на комп’ютері встановлено правильно час і дату, може вирішити проблеми PPTP.

Відновіть реєстр комп’ютера

Іноді реєстр вашого комп’ютера може пошкоджуватися, що спричиняє різноманітні проблеми з програмним забезпеченням.

Перевстановіть мережеві драйвери

Якщо ви раніше використовували VPN, швидше за все, у вас встановлено кілька мережевих дисків. Це викликає різні конфлікти.

Перевірте брандмауер свого комп’ютера

Брандмауер вашого комп’ютера може блокувати ваше з’єднання PPTP. Ось чому ваше веб-з’єднання може не працювати, коли ви під’єднуєтеся до VPN.

Яке рішення найкраще підходить тій або іншій організації? Вибір визначається трьома факторами: розмір мережі, технічні навички, якими володіють співробітники організації, і обсяг трафіка, що планується обробляти. Процес шифрування даних вимагає істотних обчислювальних ресурсів і може перевантажити комп'ютер, коли декілька VPN-з'єднань одночасно беруть участь у передачі даних. У цьому випадку, щоб розвантажити центральний процесор, можливо, прийде встановити спеціальні пришвидчуючі плати.

Який би шлях не був обраний, однаково прийде зштовхнутися із проблемою керування VPN-пристроями й підтримки погоджених правил безпеки для VPN і міжмережних екранів у масштабах всієї організації. Якщо співробітники не мають достатні навички в цій області, можна довірити створення віртуальної приватної мережі незалежної компанії, що робить відповідні послуги. І необхідно згадати ще одну особливість VPN – використання цієї технології знижує продуктивність мережі, що обумовлено затримками встановлення захищеного з'єднання між VPN-пристроями, затримками шифрування даних, затримками контролю їхньої цілісності й збільшеним трафіком через використання більше довгих заголовків пакетів.

3.2 Розробка структурної схеми

Структурна схема системи зображена на рисунку 3.1.

PPTP працює шляхом інкапсуляції "рідних" пакетів локальної мережі – наприклад, пакетів IPX – усередину пакетів TCP/IP. Весь пакет IPX, включаючи його керуючу інформацію, стає корисним навантаженням для пакета TCP/IP, що потім можна передавати по Internet. Програмні засоби на іншому кінці лінії зв'язку витягають пакет IPX і направляють його для нормальної обробки відповідно до його власного протоколу. Цей процес називається тунелюванням, імовірно, тому, що створюється коридор в Internet, що з'єднує два вузли.

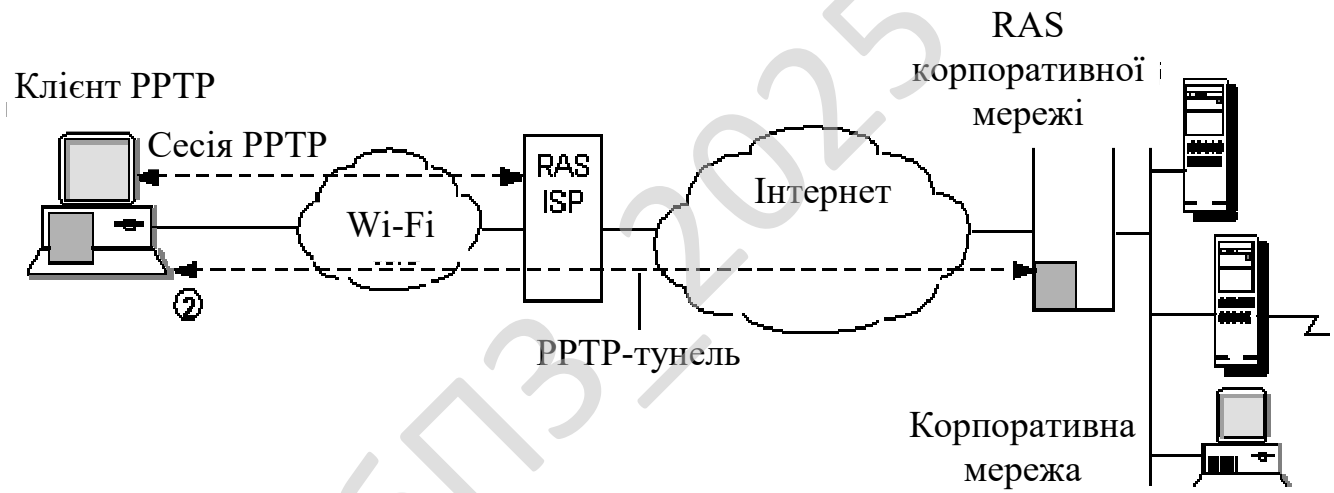

Рисунок 3.1 – Структурна схема системи

У перший раз клієнт дзвонить на сервер RAS ISP і встановлює з ним зв'язок за протоколом PPP, проходячи автентифікацію одним зі способів, підтримуваних провайдером – по протоколах PAP, CHAP або за допомогою термінального діалогу.

Після автентифікації в провайдеру, користувач удруге "дзвонить", цього разу в сервер віддаленого доступу корпоративної мережі. Цей "дзвінок" відрізняється від звичайного тем, що замість телефонного номера вказується IP-адреса RAS Windows Server 2025, підключеного до Internet з боку корпоративної

мережі. Між клієнтським комп'ютером і RAS корпоративної мережі встановлюється сесія за протоколом PPTP. Клієнт ще раз автентифікується, тепер на сервері RAS його мережі, а потім починається передача даних, як і в першому варіанті.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно. У системі використовуються наступні протоколи. В основу програмного забезпечення створення VPN підключень покладені протоколи забезпечення безпеки інформації IPsec та SSL.

Заголовок ESP – інкапсуляція зашифрованих даних

У випадку використання інкапсуляції зашифрованих даних заголовок ESP є останнім у ряді опціональних заголовків, "видимих" у пакеті. Оскільки основною метою ESP є забезпечення конфіденційності даних, різні види інформації можуть вимагати застосування істотно різних алгоритмів шифрування. Отже, формат ESP може перетерплювати значні зміни залежно від використовуваних криптографічних алгоритмів. Проте, можна виділити наступні обов'язкові поля: SPI (SPI – Secutity Parameter Index – індекс параметра безпеки), що вказує на контекст безпеки, поле порядкового номера, що містить послідовний номер пакета, і контрольна сума, призначена для захисту від атак на цілісність зашифрованих даних. Крім цього, як правило, у тілі ESP присутні параметри (наприклад, режим використання) і дані (наприклад, вектор ініціалізації) застосовуваного алгоритму шифрування. Частина ESP заголовка може бути зашифрована на відкритому ключі одержувача або на спільному ключі пари відправник-одержувач. Одержувач пакета ESP розшифровує ESP заголовок і використовує параметри й дані застосовуваного алгоритму шифрування для декодування інформації транспортного рівня.

Розрізняють два режими застосування ESP – транспортний і тунельний.

Транспортний режим – використовується для шифрування поля даних IP пакета, що містить протоколи транспортного рівня (TCP, UDP, ICMP), який, у свою чергу, містить інформацію прикладних служб. Прикладом застосування транспортного режиму є передача електронної пошти. Всі проміжні вузли на маршруті пакета від відправника до одержувача використовують тільки відкриту інформацію мережного рівня й, можливо, деякі опціональні заголовки пакета (в IPv6). Недоліком транспортного режиму є відсутність механізмів приховання конкретних відправника й одержувача пакета, а також можливість проведення аналізу трафіку. Результатом такого аналізу може стати інформація про об'єми й напрямки передачі інформації, області інтересів абонентів, розташування керівників.

Тунельний режим – припускає шифрування всього пакета, включаючи заголовок мережного рівня. Тунельний режим застосовується якщо буде потреба приховання інформаційного обміну організації із зовнішнім миром. При цьому, адресні поля заголовка мережного рівня пакета, що використовує тунельний режим, заповнюються міжмережним екраном організації й не містять інформації про конкретного відправника пакета. При передачі інформації із зовнішнього миру в локальну мережу організації як адреса призначення використовується мережна адреса міжмережного екрана. Після дешифрування міжмережним екраном початкового заголовка мережного рівня пакет направляється одержувачеві.

DOI – область інтерпретації

Протокол ISAKMP/Oakley не був спеціально розроблений для спільного використання із протоколом IPsec, тому виникає необхідність у так званій області інтерпретації (Domain Of Interpretation – DOI), що забезпечила б спільну роботу протоколів IPsec і ISAKMP/Oakley. Щоб інші протоколи також могли використовувати ISAKMP/Oakley, вони повинні мати власні DOI-області. У даний момент таких областей для інших протоколів не існує, але ситуація може змінитися на черговій конференції групи IETF або в тому випадку, якщо

приватний розроблювач, наприклад фірма Netscape, вирішить використовувати цей механізм. Більш докладно про це можна прочитати в документі "The Internet Key Exchange (IKE)", розробленому робочою групою IP Security Protocol Working Group (ftp://ftp.ietf.org/ internet-draft/draft-ietf-IPsec-isakmp-oakley-06.txt).

В основному режимі між сторонами погоджуються методи шифрування, хешування, автентифікації й так звана група DH (їх усього чотири), що визначає криптографічну стійкість алгоритму відкритого розподілу ключів. Перша група DH характеризується високою стійкістю й дозволяє використовувати стандарт DES, у той час як для другої й третьої груп варто застосовувати Triple DES. Оскільки в основному режимі іноді потрібно передавати до шести пакетів, то, наприклад, при використанні космічного сегмента з великою тимчасовою затримкою, DES краще застосовувати з більш сильною групою DH. Тоді перед виконанням чергового основного режиму, сполученого з інтенсивними обчисленнями й обміном пакетами, вам удасться виконати більше обмінів у швидкому режимі.

Коли SA-угода для обміну по протоколу Oakley установлюється в основному режимі, створюється ланцюжок випадкових біт, що використовують для генерації ключів. Також визначається тривалість (за часом або кількістю переданих даних) "життя" SA-угоди Oakley і дані для генерації ключів до того, як буде потрібно наступний обмін в основному режимі.

Швидкий режим простіше основного, і узгодження SA для IPsec здійснюється за допомогою трьох пакетів. IPsec-ключі створюються за допомогою простих операцій піднесення в ступінь переданих в основному режимі даних. У швидкому режимі погодяться також алгоритми шифрування й строки існування SA для IPsec-сеансів.

Згідно із цими строками визначається, як незабаром, залежно від часу або об'єму переданих даних, буде потрібно нове узгодження у швидкому режимі. Помітьте, є два різних строки існування SA-угоди. Основний режим задає його для протоколу Oakley, а швидкий – для обміну по протоколу IPsec. Як приклад

пропонуємо значення цих параметрів для шифрування IPsec-сеансів за допомогою алгоритму DES: 15 хв або 10 Мбайт для швидкого режиму, і 60 хв або 40 Мбайт для основного. Ці числа варто збільшити для Triple DES і зменшити для ARCFour (в ARCFour застосовується 40-бітний, а в TripleDES – 112-бітний ключ). Такий підхід дозволяє збалансувати криптографічну стійкість сервісів IPsec і вартість накладних витрат на передачу пакетів ISAKMP/Oakley.

При генерації ключів в основному режимі сеанс можна примусово перервати на підставі відкликання сертифіката. Сертифікати кінцевих вузлів використовуються тільки під час основного режиму. Таким чином, при анулюванні одного із сертифікатів обмін перерветься тільки в основному режимі. Тимчасові обмеження, погоджені в основному й швидкому режимах, значно відрізняються друг від друга й залежать від типу даних і транзакцій, що використовують IPsec-з'єднання. Для правильного визначення цих обмежень із обліком, з одного боку, об'єму обчислень і навантаження на мережу, а з іншого боку – імовірності порушення захисту даних, потрібно деякий аналіз.

Сполучення різних IPsec-механізмів забезпечує цілком безпечні з'єднання як між мережами, так і між кінцевими станціями. Оскільки практично всі постачальники підтримують ці стандарти, рано або пізно це приведе до виникнення середовища для реалізації безпечних з'єднань через Інтернет. Таким чином, протокол IPsec стане основним для безпечної е-комерції в Інтернет.

Заголовок AH

Автентифікуючий заголовок (AH) є звичайним опціональним заголовком і, як правило, розташовується між основним заголовком пакета IP і полем даних. Наявність AH ніяк не впливає на процес передачі інформації транспортного й більш високого рівнів. Основним і єдиним призначенням AH є забезпечення захисту від атак, пов'язаних з несанкціонованою зміною вмісту пакета, і в тому числі від підміни вихідної адреси мережного рівня. Протоколи більш високого рівня повинні бути модифіковані з метою здійснення перевірки автентичності отриманих даних. Формат AH досить простий і складається з 96-бітового

заголовка й даних змінної довжини, що складаються з 32-бітових слів. Назви полів досить ясно відбивають їхній зміст: Next Header указує на наступний заголовок, Payload Len представляє довжину пакета, SPI є покажчиком на контекст безпеки й Sequence Number Field містить послідовний номер пакета.

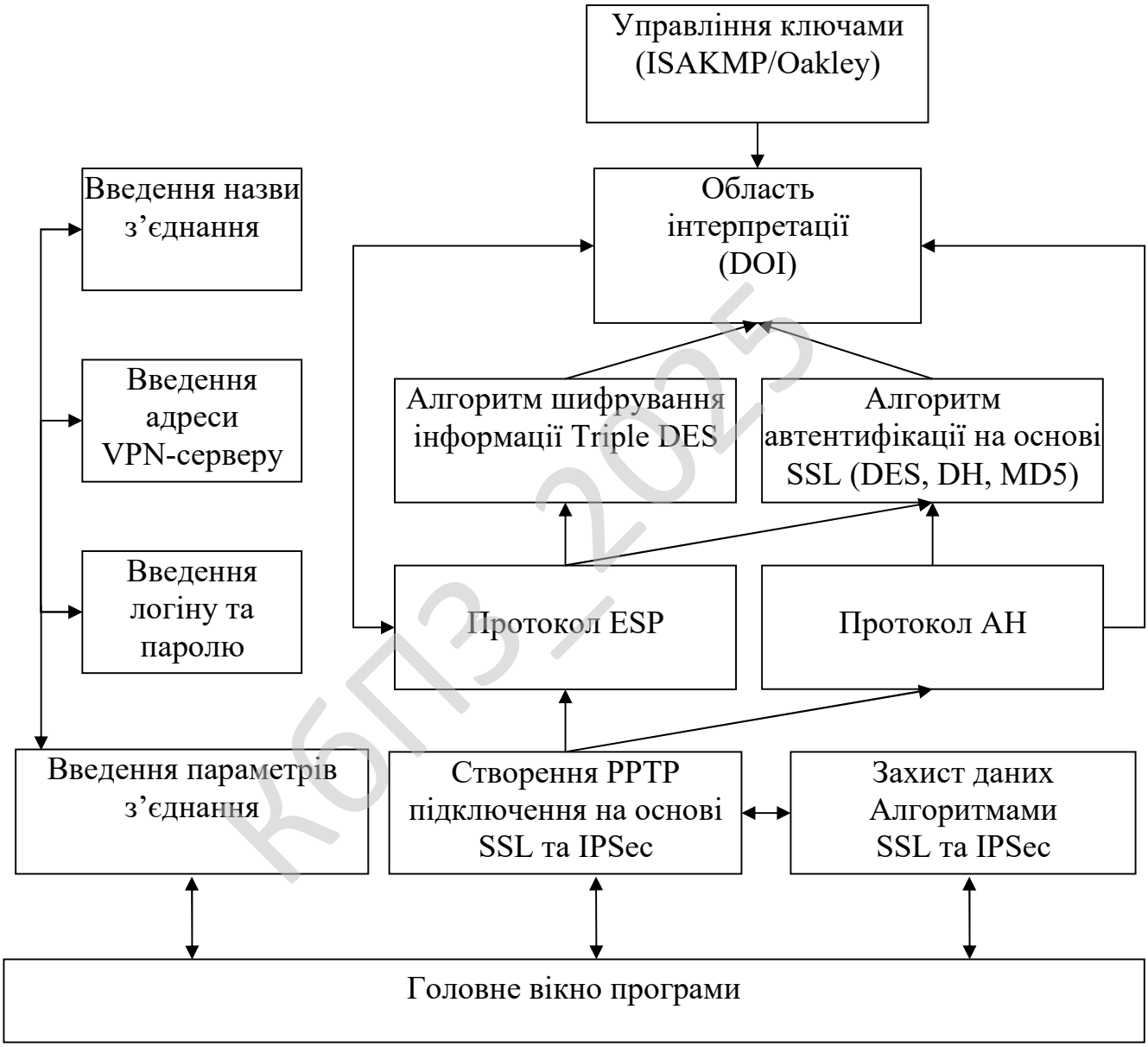

Рисунок 3.2 – Функціональна схема системи

Послідовний номер пакета був введений в AH в 1997 році в ході процесу перегляду специфікації IPsec. Значення цього поля формується відправником і

служить для захисту від атак, пов'язаних з повторним використанням даних процесу автентифікації. Оскільки мережа Інтернет не гарантує порядок доставки пакетів, одержувач повинен зберігати інформацію про максимальний послідовний номер пакета, що пройшов успішну автентифікацію, і про одержання деякого числа пакетів, що містять попередні послідовні номери (звичайно це число дорівнює 64). На відміну від алгоритмів обчислення контрольної суми, застосовуваних у протоколах передачі інформації з лініями зв'язку, що комутуються або по каналах локальних мереж і орієнтованих на виправлення випадкових помилок середовища передачі, механізми забезпечення цілісності даних у відкритих телекомунікаційних мережах повинні мати засоби захисту від внесення цілеспрямованих змін. Одним з таких механізмів є спеціальне застосування алгоритму MD5: у процесі формування AH послідовно обчислюється хеш-функція від об'єднання самого пакета й деякого попередньо погодженого ключа, а потім від об'єднання отриманого результату й перетвореного ключа. Даний механізм застосовується за замовчуванням з метою забезпечення всіх реалізацій IPv6, принаймні, одним загальним алгоритмом, не підданим експортним обмеженням.

Протокол ISAKMP/Oakley

Завдання алгоритмів IPsec – справа непроста, для цього потрібен протокол керування сеансом. Протокол ISAKMP (Internet Security Association Key Management Protocol) є рамковою основою для такого протоколу, а протокол Oakley – це вже конкретна реалізація його на цій основі, призначена для спільного використання з IPsec.

Протокол Oakley має більш широкий набір функціональних можливостей, ніж необхідно для керування IPsec-сеансами. Реалізація ISAKMP/Oakley являє собою функціональну підмножину, достатню, щоб забезпечити безпечний спосіб повідомлення автентифікованих даних для генерації ключів і SA-параметрів. Обмін по протоколу ISAKMP/Oakley відбувається у двох режимах (фазах): основному й швидкому. Відповідно до протоколу Oakley, обмін починається в основному й триває у швидкому режимі. У першому режимі встановлюються

угоди SA для обміну даними по протоколу Oakley, а в другому – по протоколу IPsec.

На один обмін в основному режимі може доводитися кілька обмінів у швидкому, так як час існування SA-угоди для протоколу Oakley може бути більш тривалим, ніж для протоколу IPsec. Завдяки обмеженому строку існування SA-угод комбінування в сеансі основного й швидкого режимів забезпечує дуже потужний захисний механізм обміну ключами.

Обмін ключами в основному режимі здійснюється по методу Діффі-Хелмана (DH), що вимагає інтенсивного використання обчислювальних ресурсів. Цей метод є механізмом розподілу відкритих ключів для безпечного обміну секретною інформацією без застосування якої-небудь інформації, заздалегідь відомим обом сторонам. Тому ним активно користуються для встановлення безпечних сеансів зв'язку в тих випадках, коли необхідний динамічний захист і коли кіцеві системи не належать однієй й тій же системі адміністративного керування. Наприклад, метод DH можна використовувати в електронній комерції при встановленні з'єднання для передачі транзакцій між двома компаніями.

Хоча цей метод і вимагає більших обчислювальних ресурсів, при його застосуванні можливий компроміс між криптостійкістю алгоритму (при використанні менш довгих відкритих ключів) і необхідним об'ємом обчислень. Обмін ключами у швидкому режимі не вимагає великого об'єму обчислень, так як тут використовується набір простих математичних операцій. Існує обмеження припустимого числа швидких фаз, перевищення якого веде до того, що ключі, згенеровані в основній фазі, а потім використовувані у швидких фазах, виявляться під погрозою розкриття. На сьогоднішній день немає твердого правила, що визначає число швидких фаз на одну основну фазу; криптографи діють, керуючись загальними міркуваннями й з огляду на оперативну обстановку.

В основному режимі обоє учасника обміну встановлюють SA-угоди для безпечного спілкування один з одним по протоколу Oakley. У швидкому режимі SA-угоди встановлюються вже "від імені" протоколу IPsec або будь-якої іншої служби, який необхідні дані для генерації ключів або узгодження параметрів. Протокол Oakley розроблений таким чином, що він ніяк не пов'язаний з IPsec.

Наприклад, для підвищення безпеки процесу встановлення сеансів його цілком можна використовувати разом із протоколом SSL (Secure Sockets Layer) версії 4.0 замість механізму обміну ключами SSL 3.0.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. Після початку роботи розробленого ПЗ ми потрапляємо до головного блоку системи звідки через ланку дій відбувається наступне:

– Виведення інтерфейсу ПЗ.

– Система VPN-з’єднань.

– Введення даних VPN-сервера.

– Введення логіну та паролю VPN-з’єднання.

– БД даних VPN.

– Моніторинг VPN підсистеми.

– Відключитися від VPN-з’єднання.

– Видалити VPN-з’єднання.

– Модифікувати VPN-з’єднання.

– Зміна назви VPN-з’єднання.

– Зміна назви адреси VPN-сервера.

– Підключитися до VPN-з’єднання.

– Введення логіну та паролю.

– Сеанс обміну даними (PPTP).

– Захист даних на основі PPTP-протоколу.

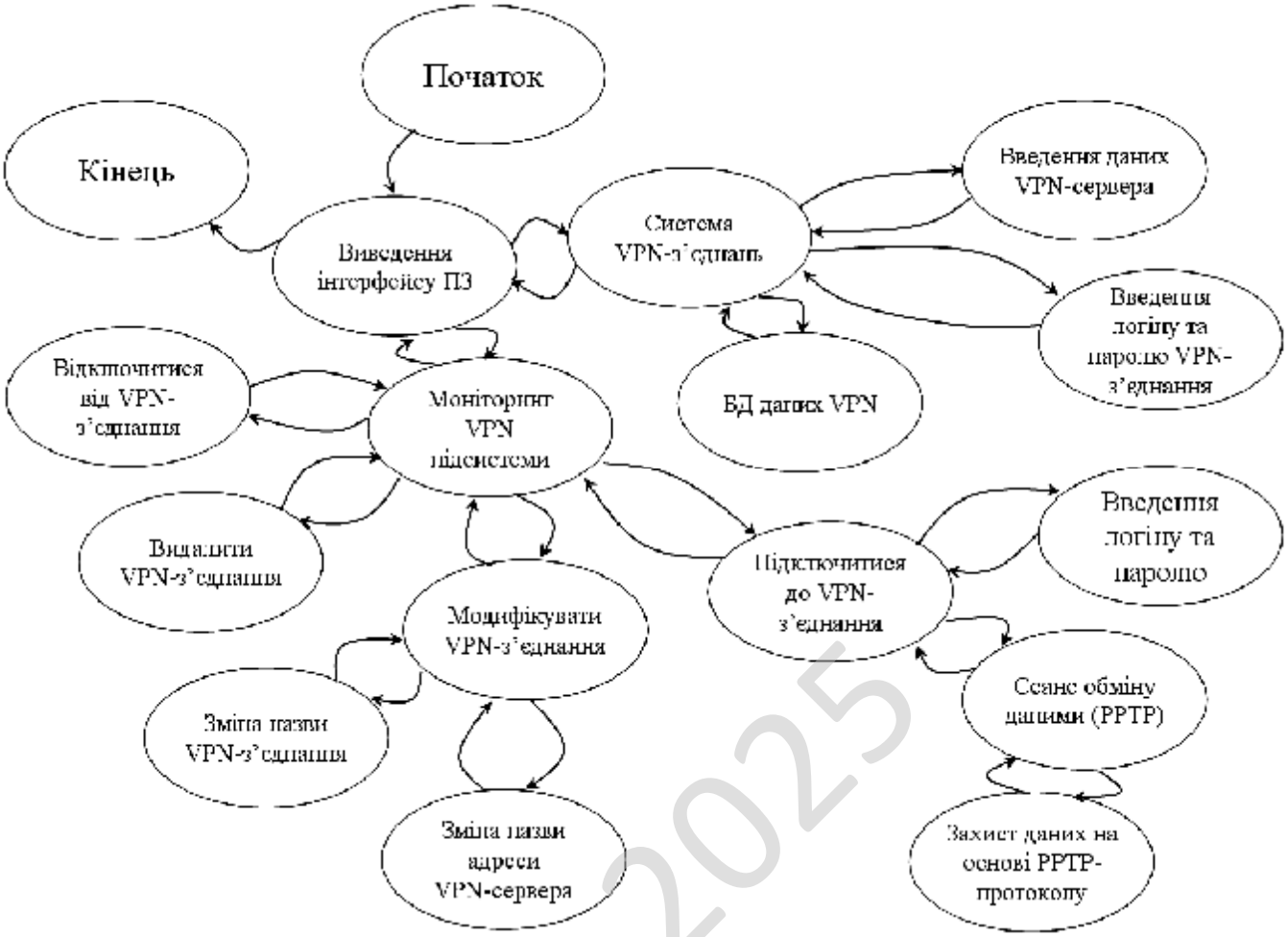

Рисунок 3.3 – Діаграма взаємодії процесів

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно розробка блок-схем. На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми. З якої видно що робота основної програми складається з початкових етапів ініціалізації ПЗ.

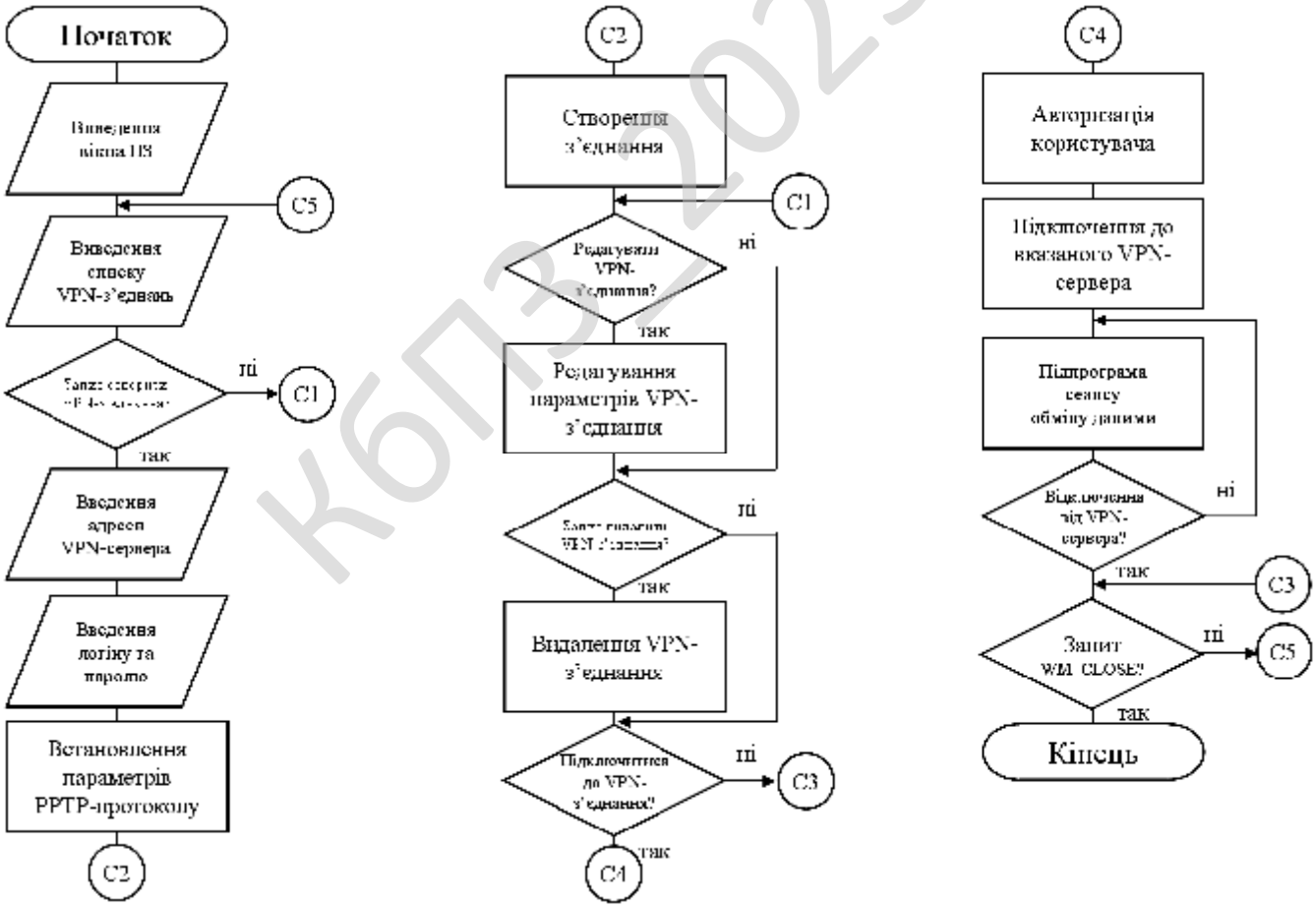

Рисунок 4.1 – Блок схема основної програми

Перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ.

При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Опис алгоритмів функціонування системи

Практично всі механізми мережної безпеки можуть бути реалізовані на третьому рівні еталонної моделі ISO/OSI.

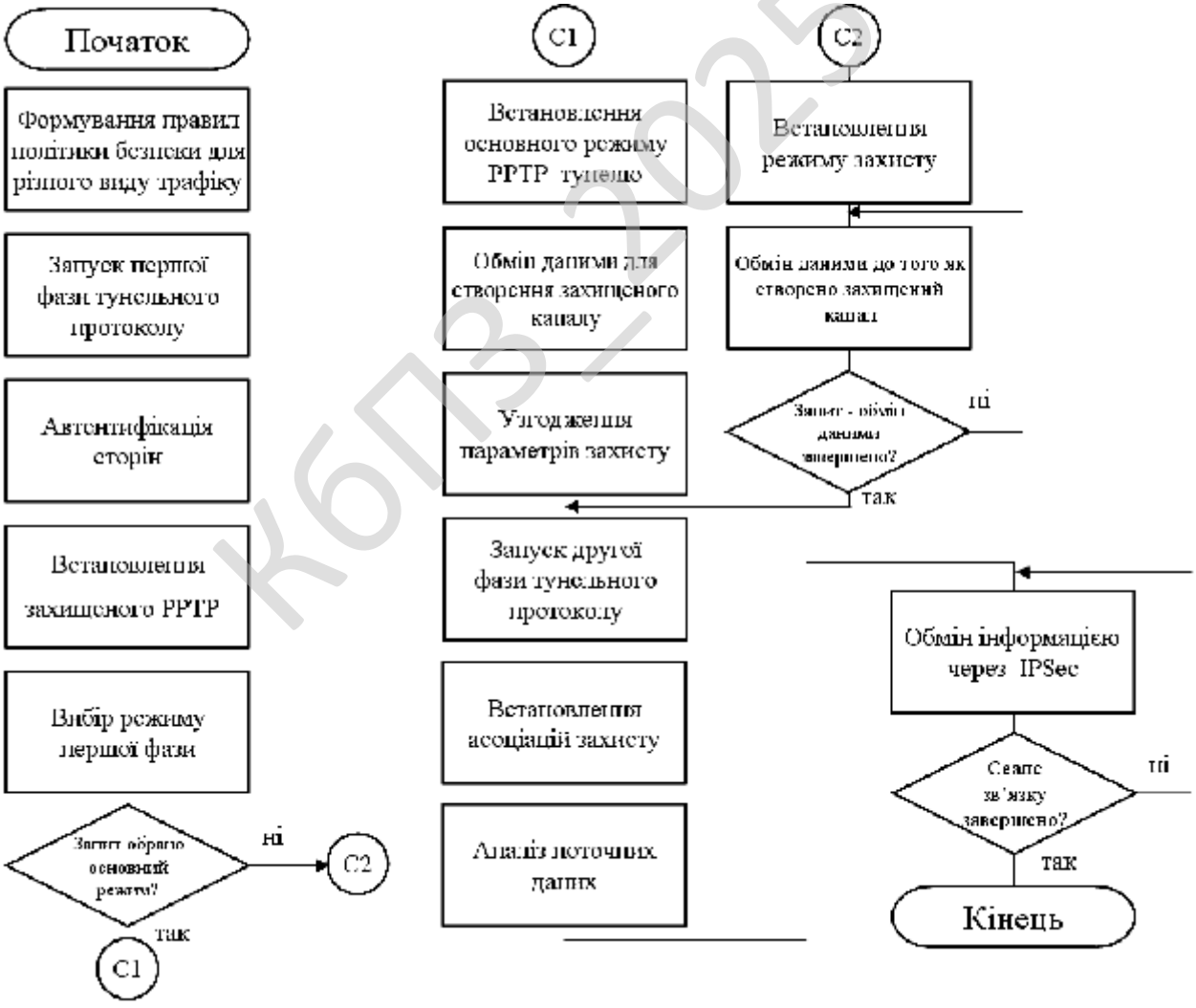

Рисунок 4.2 – Блок схема підпрограми

Більше того, IP-рівень можна вважати оптимальним для розміщення захисних засобів, оскільки при цьому досягається вдалий компроміс між захищеністю, ефективністю функціонування й прозорістю для додатків.

Опис системи

Система захищеної мережі базується на PPTP протоколі і реалізується за допомогою мови програмування Python вона працює в режимі реального часу з використанням окремих модулів що відповідають за встановлення з'єднання шифрування передачу даних логування та аналіз продуктивності мережі.

Система організовується у вигляді клієнтської частини серверної частини та модуля керування з'єднанням що забезпечує прозорий зв'язок між віддаленими вузлами.

Архітектура проекту дозволяє проводити роботи захищеного тунелю мережі з використанням простих алгоритмів шифрування даних що підтверджується проведенням розрахунків продуктивності мережі.

У випадку тестової передачі текстових повідомлень розрахунки показують що при передачі даних розмір яких становить приблизно 40 символів і часі моделювання 0.05 секунди система забезпечує пропускну здатність близько 800 одиниць даних за секунду що відповідає вимогам проектного завдання.

Система впроваджується з використанням модулів що виконують такі функції встановлення з'єднання шифрування повідомлень дешифрування отриманих даних моделювання передачі інформації і обчислення показників продуктивності ці функції інтегруються у модулі роботи мережі.

Це дозволяє отримати результат роботи системи та провести аналіз правильності обраних рішень застосування Python дозволяє створити гнучку систему з можливістю розширення функціоналу системи в режимі реального часу.

Інтеграція всіх компонентів забезпечує безперебійну роботу тунелювання даних виявлення можливих збоїв і оперативну діагностику роботи системи що підтверджується виконанням розрахунків продуктивності мережі.

Результати моделювання демонструють ефективність системи при передачі зашифрованих даних через забезпечений тунель зв'язку.

Система має простий механізм керування ключами шифрування що дозволяє у майбутньому розширити функціональні можливості проекту наступний вихідний текст містить приклад вихідного коду

```
# Функція встановлює з'єднання між сервером і клієнтом
def establish_connection(server_address, client_address):
    connection_status = True
    return connection_status
# Функція виконує шифрування даних за допомогою
# простого симетричного алгоритму
def encrypt_data(data, key):
    encrypted_data = ''.join(chr((ord(ch) + key) % 256) for ch in data)
    return encrypted_data

# Функція виконує дешифрування даних з використанням заданого ключа
def decrypt_data(encrypted_data, key):
decrypted_data = ''.join(chr((ord(ch) - key) % 256) for ch in
encrypted_data)
    return decrypted_data

# Функція виконує передачу зашифрованих даних через захищений канал
def send_data(connection, data):
    if connection:
        encrypted = encrypt_data(data, 3)
        return encrypted
    return None

# Функція обчислює продуктивність мережі шляхом
# поділу розміру даних на час передачі
def calculate_performance(data_size, transmission_time):
    performance = data_size / transmission_time
    return performance

# Функція симулює роботу всієї системи
# шляхом встановлення з'єднання передачі даних та аналізу продуктивності
def simulate_network():
    server_ip = "192.168.1.1"
    client_ip = "192.168.1.2"
    connection = establish_connection(server_ip, client_ip)
```

```
    transmitted_data = "Приклад тестового тексту для передачі"
    encrypted = send_data(connection, transmitted_data)
    throughput = calculate_performance(len(transmitted_data), 0.05)
    results = {"encrypted_data": encrypted, "throughput": throughput}
    return results
if __name__ == "__main__":
    simulation_results = simulate_network()
    print("Результати симуляції", simulation_results)
```

Система працює у реальному часі вона встановлює з'єднання між сервером і клієнтом система виконує шифрування отриманих даних шляхом зміщення кодових значень символів. Це дозволяє захищати інформацію під час передачі отримання зашифрованих даних здійснюється через виклик функції передачі повідомлень після чого система виконує розрахунок продуктивності мережі шляхом ділення кількості символів.

На час передачі результат розрахунку підтверджує ефективність обраних проектних рішень що демонструється отриманням високої пропускної здатності. Система перевіряє правильність роботи шляхом дешифрування даних і порівняння їх з початковими повідомленнями що забезпечує контроль коректності роботи всіх функцій проекту. Система має можливість бути модифікованою для розширення функціональних можливостей у майбутньому вона інтегрується з іншими компонентами мережевої інфраструктури що сприяє використанню даної розробки для практичних завдань у сфері забезпечення інформаційної безпеки.

Стандартизованими механізмами IP безпеки можуть (і повинні)користуватися протоколи більше високих рівнів і, зокрема, що управляють протоколи, протоколи конфігурування й маршрутизації. Засоби безпеки для IP описуються сімейством специфікацій IPsec, розроблених робочою групою IP Security. Протоколи IPsec забезпечують керування доступом , цілісність поза з'єднанням, автентифікацію джерела даних, захист від відтворення, конфіденційності, частковий захист від аналізу трафіку. Архітектура засобів безпеки для IP-рівня специфікована в документі.

Це насамперед протоколи забезпечення автентичності (протокол автентифікуючого заголовка – Authentication Header, AH) і конфіденційності (протокол інкапсулюючий захист вмісту- EncaPsulating Security Payload, ESP), а також механізми керування криптографічними ключами. На більше низькому архітектурному рівні розташовуються конкретні алгоритми шифрування, контролю цілісності й автентичності.

Нарешті, роль фундаменту виконує так званих домен інтерпретації (Domain of Interpretation, DOI), що є, по суті, базою даних, що зберігає відомості про алгоритми, їхніх параметрах, протокольних ідентифікаторах і т.п.

Розподіл на рівні важливий для всіх аспектів інформаційних технологій. Там же, де бере участь ще й криптографія, важливість зростає подвійно, оскільки доводиться вважатися не тільки із чисто технічними факторами, але й з особливостями законодавства різних країн, з обмеженнями на експорт і/або імпорт криптозасобів.

Протоколи забезпечення автентичності й конфіденційності в IPsec не залежать від конкретних криптографічних алгоритмів.

Більше того, саме розподіл на автентичність і конфіденційність надає й розроблювачам, і користувачам додатковий ступінь волі в ситуації, коли до криптографічного відносять тільки шифрувальні засоби.

У кожній країні можуть застосовуватися свої алгоритми, що відповідають національним стандартам, але для цього, як мінімум, потрібно подбати про їхню реєстрацію в домені інтерпретації.

Алгоритмічна незалежність протоколів, на жаль, має й зворотний бік, що складається в необхідності попереднього узгодження набору застосовуваних алгоритмів і їхніх параметрів, підтримуваних сторонами, що спілкуються. Іншими словами, сторони повинні виробити загальний контекст безпеки (Security Association, SA) і потім використовувати такі його елементи, як алгоритми і їхні ключі. За формування контекстів безпеки в IPsec відповідає особливе сімейство протоколів, що буде розглянуто в наступних розділах.

Протоколи забезпечення автентичності й конфіденційності можуть застосовуватися у двох режимах: транспортному й тунельному.

У першому випадку захищається тільки вміст пакетів і, бути може, деякі поля заголовків. Як правило, транспортний режим використовується хостами.

У тунельному режимі захищається весь пакет – він інкапсулюється в інший IP-пакет. Тунельний режим звичайно реалізують на спеціально виділених захисних шлюзах.

Забезпечення автентичності IP-пакетів

Протокол автентифікуючого заголовка (Authentication Header, AH) служить в IPsec для забезпечення цілісності пакетів і автентифікації джерела даних, а також для захисту від відтворення раніше посланих пакетів. AH захищає дані протоколів більше високих рівнів і ті поля IP-Заголовків, які не міняються на маршруті доставки або міняються передбачуваним образом. (Відзначимо, що число "непередбачених" полів невелике – це Prio. (Traffic Class), Flow Label і Hop Limit. Передбачувано міняється цільова адреса при наявності додаткового заголовка вихідної маршрутизації) .

Пояснимо зміст полів, специфічних для AH:

– індекс параметрів безпеки (SP) – 32-бітне значення, обиране одержувачем пакетів з AH-Заголовками як ідентифікатор протокольного контексту (див. вище розділ "Протокольні контексти й політика безпеки");

– порядковий номер – беззнакове 32-бітне ціле, нарощуване від пакета до пакета. Відправник зобов'язаний підтримувати цей лічильник, у той час як одержувач може (але не зобов'язаний) використовувати його для захисту від відтворення. При формуванні протокольного контексту обидві взаємодіючі сторони роблять свої лічильники нульовими, а потім погодженим образом збільшують їх. Коли значення порядкового номера стає максимально можливим, повинен бути сформований новий контекст безпеки;

– автентифікаційні дані – поле змінної довжини, що містить імітовставку (криптографічну контрольну суму, Integrity Check Value, ICV) пакета; спосіб його обчислення визначається алгоритмом автентифікації.

Для обчислення автентифікованих імітовставок можуть застосовуватися різні алгоритми. Специфікаціями пропонується обов'язкова підтримка двох алгоритмів, заснованих на застосуванні хеш-функцій із секретними ключами:

– HMAC-MD5 (Hashed Message Authentication Code – Message Digest version 5);

– HMAC-SHA-1 (Hashed Message Authentication Code – Secure Hash Algorithm version 1).

Забезпечення конфіденційності мережного трафіку

Протокол інкапсулюючий захисту вмісту (Encapsulating Security Payload, ESP) надає три види сервісів безпеки:

– забезпечення конфіденційності (шифрування вмісту IP-пакетів, а також частковий захист від аналізу трафіку шляхом застосування тунельного режиму);

– забезпечення цілісності IP-пакетів і автентифікації джерела даних;

– забезпечення захисту від відтворення IP-пакетів.

Можна бачити, що функціональність ESP ширше, ніж в AH (додається шифрування); взаємодія цих протоколів ми докладніше розглянемо пізніше. Тут же відзначимо, що ESP не обов'язково надає всі сервіси, але або конфіденційність, або автентифікація повинні бути задіяні. Формат заголовка ESP виглядає трохи незвичайно. Причина в тім, що це не стільки заголовок, скільки обгортка (інкапсулююча оболонка) для зашифрованого вмісту. Наприклад, посилання на наступний заголовок не можна виносити в початок, у незашифровану частину, тому що вона втратиться конфіденційності.

Поля "Індекс параметрів безпеки (SP)", "Порядковий номер" і "Автентифікационные дані" (останнє є присутнім тільки при включеної автентифікації) мають той же зміст, що й для AH. Правда, ESP автентифікує лише зашифровану частину пакета (плюс два перші поля заголовка).

Застосування протоколу ESP до вихідних пакетів можна уявляти собі в такий спосіб. Назвемо залишком пакета ту його частину, що міститься після передбачуваного місця вставки заголовка ESP. При цьому не важливо, який режим використовується – транспортний або тунельний. Кроки протоколу такі:

– залишок пакета копіюється в буфер;

– до залишку приписуються байти, що доповнюють, їхнє число й номер (тип) першого заголовка залишку, для того щоб номер був притиснутий до границі 32-бітного слова, а розмір буфера задовольняв вимогам алгоритму шифрування;

– поточний уміст буфера шифрується;

– у початок буфера приписуються поля "Індекс параметрів безпеки (SP)" і "Порядковий номер" з відповідними значеннями;

– поповнений уміст буфера автентифікується, у його кінець міститься поле "Автентифікаційні дані";

– у новий пакет листуються початкові заголовки старого пакета й кінцевий уміст буфера.

Таким чином, якщо в ESP включені й шифрування, і автентифікація, те автентифікується зашифрований пакет. Для вхідних пакетів дії виконуються у зворотному порядку, тобто спочатку виробляється автентифікація. Це дозволяє не витрачати ресурси на розшифровку підроблених пакетів, що в якімсь ступені захищає від атак на доступність.

Два захисних протоколи – AH і ESP – можуть комбінуватися різними способами. При виборі транспортного режиму AH повинен використовуватися після ESP (аналогічно тому, як у рамках ESP автентифікація йде слідом за шифруванням). У тунельному режимі AH і ESP застосовуються, строго говорячи, до різного (вкладеним) пакетам, число припустимих комбінацій тут більше (хоча б тому, що можливо багаторазову вкладеність тунелів з різними початковими й/або кінцевими крапками).

Сукупність механізмів, пропонована в рамках IPsec, є досить потужною й гнучкою. IPsec – це основа, на якій може будуватися реалізація віртуальних приватних мереж (VPN), забезпечуватися захищена взаємодія мобільних систем з корпоративною мережею, захист прикладних потоків даних і т.п.

Контексти безпеки й керування ключами

Формування контекстів безпеки в IPsec розділено на дві фази. Спочатку створюється керуючий контекст, призначення якого – надати довірений канал, тобто автентифікований, захищений канал для вироблення (у рамках другої фази) протокольних контекстів і, зокрема, для формування криптографічних ключів, використовуваних протоколами AH і ESP.

У принципі, для функціонування механізмів IPsec необхідні тільки протокольні контексти; керуючий відіграє допоміжну роль. Більше того, явне виділення двох фаз ускладнює формування ключів, якщо розглядати останнє як однократну дію. Проте, з архітектурних міркувань керуючі контексти не тільки можуть, але й повинні існувати, оскільки обслуговують всі протокольні рівні стека TCP/IP, концентруючи в одному місці необхідну функціональність.

Перша фаза починається в ситуації, коли взаємодіючі сторони не мають загальних секретів (загальних ключів) і не впевнені в автентичності один одного. Якщо із самого початку не створити довірений канал, то для виконання кожної керуючої дії із ключами (їхня модифікація, видача діагностичних повідомлень і т.п.) у кожному протоколі (AH, ESP, TLS і т.д.) цей канал прийде формувати заново.

Загальні питання формування контекстів безпеки й керування ключами висвітлюються в специфікації – "Контексти безпеки й керування ключами в Internet" (Internet Security Association and Key Management Protocol, ISAKMP). Тут уводяться дві фази вироблення протокольних ключів, визначаються види керуючих інформаційних обмінів і використовувані формати заголовків і даних. Іншими словами будується протокольно-незалежний каркас.

Існує кілька способів формування керуючого контексту. Вони розрізняються двома показниками:

– використовуваним механізмом вироблення загального секретного ключа;

– ступенем захисту ідентифікаторів сторін, що спілкуються.

У найпростішому випадку секретні ключі задаються заздалегідь (ручний метод розподілу ключів). Для невеликих мереж такий підхід цілком працездатний, але він не є масштабованим.

Остання властивість може бути забезпечене при автоматичному вироблення й розподілі секретних ключів у рамках протоколів, заснованих на алгоритмі Діффі-Хелмана. Приклад тому – "Протокол для обміну ключами в Internet" (The Internet Key Exchange, IKE).

При формуванні керуючого контексту ідентифікатори сторін, що спілкуються (наприклад, IP-адреси) можуть передаватися у відкритому виді або шифруватися. Оскільки ISAKMP передбачає функціонування в режимі клієнт/сервер (тобто ISAKMP-сервер може формувати контекст для клієнта), приховання ідентифікаторів деякою мірою підвищує захищеність від пасивного прослуховування мережі. Послідовність переданих повідомлень, що дозволяють сформувати керуючий контекст і забезпечують захист ідентифікаторів, виглядає в такий спосіб.

У першому повідомленні (1) ініціатор направляє пропозиції по наборі захисних алгоритмів і конкретних механізмів їхньої реалізації. Пропозиції впорядковуються по ступені переваги (для ініціатора). У відповідному повідомленні (2) партнер інформує про зроблений вибір – які алгоритми й механізми його влаштовують. Для кожного класу захисних засобів (генерація ключів, автентифікація, шифрування) вибирається тільки один елемент.

У повідомленнях (3) і (4) ініціатор і партнер відправляють свої частини ключового матеріалу, необхідні для вироблення загального секретного ключа (ми опускаємо деталі, специфічні для алгоритму Діффі-Хелмана). Одноразові номери (nonce) являють собою псевдовипадкові величини, що служать для захисту від відтворення повідомлень.

За допомогою повідомлень (5) і (6) відбувається обмін ідентифікаційною інформацією, підписаної (з метою автентифікації) секретним ключем відправника й зашифрованої виробленим на попередніх кроках загальним секретним ключем.

Для автентифікації передбачається використання сертифікатів відкритих ключів. Відзначимо, що в число даних, що підписуються, входять одноразові номери.

У представленому виді протокол формування керуючого контексту захищає від атак, вироблених нелегальним посередником, а також від нелегального перехоплення з'єднань.

Для захисту від атак на доступність, для яких характерно насамперед нав'язування інтенсивних обчислень, властиві криптографії з відкритим ключем, застосовуються так звані ідентифікуючі ланцюжки (cookies). Ці ланцюжки, формовані ініціатором і його партнером з використанням поточного часу (для захисту від відтворення), насправді присутні у всіх ISAKMP-повідомленнях і в сукупності ідентифікують керуючий контекст (у першому повідомленні, по зрозумілих причинах, фігурує тільки ланцюжок ініціатора). Якщо зловмисник намагається "завалити" кого-небудь запитами на створення керуючого контексту, підробляючи при цьому свою IP-адресу, то в повідомленні (3) він не зможе пред'явити ідентифікуючий ланцюжок партнера, тому до вироблення загального секретного ключа й, тим більше, електронного підпису й повномасштабної перевірки автентичності справа попросту не дійде.

Керуючі контексти є двонаправленими в тому розумінні, що кожна зі сторін, що спілкуються, може ініціювати з їхньою допомогою виробіток нових протокольних контекстів або інші дії.

Для передачі ISAKMP-повідомлень використовується будь-який протокол, однак у якості стандартного прийнятий UDP з номером порту 500.

Протокольні контексти й політика безпеки

Системи, що реалізують IPsec, повинні підтримувати дві бази даних:

– базу даних політики безпеки (Security Policy Database, SP);

– базу даних протокольних контекстів безпеки (Security Association Database, SAD).

Всі IP-пакети (вхідні й вихідні) зіставляються з упорядкованим набором правил політики безпеки. При зіставленні використовується селектор,

що фігурує в кожному правилі, – сукупність аналізованих полів мережного рівня й більше високих протокольних рівнів. Перше підходяще правило визначає подальшу долю пакета:

– пакет може бути ліквідований;

– пакет може бути оброблений без участі засобів IPsec;

– пакет повинен бути оброблений засобами IPsec з урахуванням набору протокольних контекстів, асоційованих із правилом.

Таким чином, системи, що реалізують IPsec, функціонують як міжмережні екрани, фільтруючи й перетворюючи потоки даних на основі попередньо заданої політики безпеки.

Далі детально розглянемо контексти й політику безпеки, а також порядок обробки мережних пакетів.

Протокольний контекст безпеки в IPsec – це односпрямоване "з'єднання" (від джерела до одержувача), що надає обслуговуються потокам, що, даних набір захисних сервісів у рамках якогось одного протоколу (AH або ESP).

У випадку симетричної взаємодії партнерам прийде організувати два контексти (по одному в кожному напрямку). Якщо використовуються й AH, і ESP, буде потрібно чотири контексти.

Елементи бази даних протокольних контекстів містять наступні поля (у кожному конкретному випадку деякі значення полів будуть порожніми):

– використовуваний у протоколі AH алгоритм автентифікації, його ключі й т.п.;

– використовуваний у протоколі ESP алгоритм шифрування, його ключі, початковий вектор і т.п.;

– використовуваний у протоколі ESP алгоритм автентифікації, його ключі й т.п.;

– час життя контексту;

– режим роботи IPsec: транспортний або тунельний;

– максимальний розмір пакетів;

– група полів (лічильник, вікно, прапори) для захисту від відтворення пакетів.

Користувачами протокольних контекстів, як правило, є прикладні процеси. Загалом кажучи, між двома вузлами мережі може існувати довільне число протокольних контекстів, тому що число додатків у вузлах довільно. Відзначимо, що як користувачів керуючих контекстів звичайно виступають вузли мережі (оскільки в цих контекстах бажано зосередити загальну функціональність, необхідну сервісам безпеки всіх протокольних рівнів еталонної моделі для керування криптографічними ключами).

Керуючі контексти – двосторонні, тобто кожний з партнерів може ініціювати новий ключовий обмін. Пара вузлів може одночасно підтримувати кілька активних керуючих контекстів, якщо є додатки з істотно різними криптографічними вимогами.

Наприклад, припустимо вироблення частини ключів на основі попередньо розподіленого матеріалу, у той час як інша частина породжується по алгоритму Діффі-Хелмана.

Протокольний контекст для IPsec ідентифікується цільовим IP-адресом, протоколом (AH або ESP), а також додатковою величиною – індексом параметрів безпеки (Security Parameter Index, SP). Остання величина необхідна, оскільки можуть існувати кілька контекстів з однаковими IP-адресами й протоколами. Далі буде показано, як використовуються індекси SP при обробці вхідних пакетів.

IPsec зобов'язує підтримувати ручне й автоматичне керування контекстами безпеки й криптографічних ключів. У першому випадку всі системи заздалегідь забезпечуються ключовим матеріалом і іншими даними, необхідними для захищеної взаємодії з іншими системами.

У другому – матеріал і дані виробляються динамічно, на основі певного протоколу – IKE, підтримка якого обов'язкова.

Протокольний контекст створюється на базі керуючого з використанням ключового матеріалу й засобів автентифікації й шифрування останнього.

Коли вироблявся керуючий контекст, для нього було створено три види ключів:

– SKEYID_d – ключовий матеріал, використовуваний для генерації протокольних ключів.

– SKEYID_a – ключовий матеріал для автентифікації.

– SKEYID_e – ключовий матеріал для шифрування.

Всі перераховані види ключів задіяні в обміні. Ключем SKEYID_e шифруються повідомлення. Ключ SKEYID_a служить аргументом хеш-функцій і тим самим автентифікує повідомлення. Нарешті, протокольні ключі – результат застосування псевдовипадкової (хеш) функції до SKEYID_d з додатковими параметрами, у число яких входять одноразові номери ініціатора й партнера. У результаті створення протокольного контексту виявляється автентифікованим, захищеним від несанкціонованого ознайомлення, від відтворення повідомлень і від перехоплення з'єднання.

Повідомлення (1) і (2) можуть нести додаткове навантаження, наприклад, дані для вироблення "зовсім нових" секретних ключів або ідентифікатори клієнтів, від імені яких ISAKMP-сервери формують протокольний контекст. Відповідно до протоколу IKE, за один обмін (щоскладається із трьох повідомлень) формується два односпрямованих контексти – по одному в кожному напрямку. Одержувач контексту задає для нього індекс параметрів безпеки (SP), що допомагає знаходити контекст для обробки прийнятих пакетів IPsec.

Строго говорячи, протокольні контексти відіграють допоміжну роль, будучи лише засобом проведення в життя політики безпеки; вона повинна бути задана для кожного мережного інтерфейсу із задіяними засобами IPsec і для кожного напрямку потоків даних (вхідні/вихідні).

Відповідно до специфікацій IPsec, політика розраховується на безконтекстну (незалежну) обробку IP-пакетів, у дусі сучасних фільтруючих маршрутизаторів. Зрозуміло, повинні існувати засобу адміністрування бази даних SP, так само, як і засобу адміністрування бази правил міжмережного екрана, однак цей аспект не входить до числа стандартизуємих.

Із зовнішньої точки зору, база даних політики безпеки (SP) являє собою впорядкований набір правил. Кожне правило задається як пара:

– сукупність селекторів;

– сукупність протокольних контекстів безпеки.

Селектори служать для відбору пакетів, контексти задають необхідну обробку. Якщо правило посилається на неіснуючий контекст, воно повинне містити достатню інформацію для його (контексту) динамічного створення. Очевидно, у цьому випадку потрібна підтримка автоматичного керування контекстами й ключами.

У принципі, функціонування системи може починатися із завдання бази SP при порожній базі контекстів (SAD); остання буде наповнюватися в міру необхідності.

Дифференційованість політики безпеки визначається селекторами, ужитими в правилах. Наприклад, пари взаємодіючих хостів може використовувати єдиний набір контекстів, якщо в селекторах фігурують тільки IP-адреси; з іншого боку, набір може бути своїм для кожного додатка, якщо аналізуються номери TCP- і UDP-портів.

Аналогічно, два захисних шлюзи здатні організувати єдиний тунель для всіх що обслуговуються хостів або ж розщепити його (шляхом організації різних контекстів) по парах хостів або навіть додатків.

Всі реалізації IPsec повинні підтримувати селекцію наступних елементів:

– вихідна й цільова IP-адреси (адреси можуть бути індивідуальними й груповими, у правилах допускаються діапазони адрес і метасимволи "будь-який";

– ім'я користувача або вузла у форматі DNS або X.500;

– транспортний протокол;

– номера вихідного й цільового портів (тут також можуть використовуватися діапазони й метасимволи).

Обробка вихідні й вхідного трафіку не є симетричною. Для вихідних пакетів проглядається база SP, перебуває підходяще правило, витягають асоційовані з ним протокольні контексти й застосовуються відповідні механізми безпеки. У вхідних пакетах для кожного захисного протоколу вже проставлене значення SP, однозначно визначальний контекст.

Перегляд бази SP у такому випадку не потрібно; можна вважати, що політика безпеки враховувалася при формуванні відповідного контексту. (Практично це означає, що ISAKMP-пакети мають потребу в особливому трактуванню, а правила з відповідними селекторами повинні бути включені в SP.)

Відзначена асиметрія, на наш погляд, відбиває певну незавершеність архітектури IPsec. У більше ранньому документі RFC 1825 поняття бази даних політики безпеки й селекторів були відсутні. У новій редакції специфікований перегляд бази SP як обов'язковий для кожного вихідного пакета, але не змінена обробка вхідних пакетів. Звичайно, добування контексту по індексі SP ефективніше, ніж перегляд набору правил, але при такому підході, щонайменше, утрудняється оперативна зміна політики безпеки. Що стосується ефективності перегляду правил, те її можна підвищити методами кешування, широко використовуваними при реалізації IP. Можливо, ще більш серйозним недоліком є неможливість узагальнення запропонованих процедур формування контекстів (керуючих і протокольних) на багатоадресний випадок. У поточних специфікаціях IPsec змішуються дві різні речі – область дії контексту (зараз це однобічний або двосторонній потік даних) і спосіб його ідентифікації (по індексі SP або парі ідентифікуючих ланцюжків). Виходить, що спосіб ідентифікації (іменування) нав'язує трактування області дії, що представляється невірним. На наш погляд, питання іменування можуть вирішуватися локально, а область дії контексту потенційно повинна поширюватися на довільне число партнерів.

4.2 Захист розробленого програмного забезпечення

Дані які використовуються у даній роботі захищаються алгоритмом ДСТУ 7624:2014 («Калина»). Одним із основних алгоритмів симетричного блокового шифрування, що використовуються в Україні, є ДСТУ 7624:2014 («Калина»), який визначає сучасний алгоритм симетричного блокового перетворення для забезпечення конфіденційності й цілісності інформації при її обробці та встановлює режими його роботи.

В алгоритмі шифрування даних «Калина» використовуються криптографічні перетворення, які відповідають сучасним вимогам до рівня криптостійкості та швидкодії.

Даний стандарт розроблено з урахуванням існуючих та потенційних загроз, подальшого інтенсивного розвитку інформаційних технологій та необхідності активного використання протягом кількох наступних десятиліть.

Стандарт блокового симетричного шифрування ДСТУ 7624:2014 визначає десять різних режимів роботи, що широко поширені відповідно до міжнародних стандартів ISO/IEC 10116:2006.

Це спрямовано на забезпечення широкого застосування ДСТУ 7624:2014, у тому числі для захисту інформації, що передається комп'ютерними мережами, прозорого шифрування жорстких дисків і змінних носіїв, електронних документів, ключових даних.

Ефективність реалізації систем, засобів та протоколів криптографічного захисту інформації в інформаційно-телекомунікаційних системах різного призначення може бути забезпечена саме наявністю такої кількості режимів роботи алгоритму.

До блокового шифру «Калина» ставляться такі вимоги: високий рівень криптографічної стійкості з достатнім запасом у разі появи нових атак протягом тривалого часу; висока швидкодія програмної реалізації на сучасних та перспективних платформах; компактність програмної та програмно-апаратної

реалізації; можливість ефективної інтеграції декількох алгоритмів в одному засобі криптографічного захисту; прозорість проектування, консервативний підхід до забезпечення стійкості; вища (або однакова) ефективність порівняно з найкращими світовими рішеннями.

Криптографічні алгоритми, які визначаються стандартами ДСТУ 7624:2014 і ДСТУ 7564:2014, є гнучкими, підтримують розмір блоку і довжину ключа від 128 до 512 біт.

Стандарт симетричного блокового шифрування «Калина» є результатом багаторічної плідної співпраці Державної служби спеціального зв'язку та захисту інформації України та провідних українських вчених.

Даний алгоритм шифрування враховує досвід і результати проведення міжнародних та відкритих національних конкурсів криптографічних алгоритмів.

Алгоритм ДСТУ 7624:2014 забезпечує досить високий рівень криптостікості порівняно з міжнародним стандартом AES (ISO/IEC 18033-3:2010), оскільки дає можливість застосовувати блок даних і ключ шифрування розміром аж до 512 біт.

Крім того, він має аналогічну або навіть більш високу швидкодію на сучасних і перспективних програмних та програмно-апаратних платформах.

На даний момент продовжуються роботи зі стандартизації вітчизняних криптографічних алгоритмів та протоколів.

При цьому не обмежується застосування гармонізованих стандартів у сфері захисту конфіденційної інформації.

Також зусилля зосереджено на використанні кращих практик застосування стандартів шифрування даних для захисту інформації в інформаційно-комунікаційних системах [10, 11].

Оскільки в стандартах симетричного блокового шифрування «Калина» та AES використовуються аналогічні криптографічні перетворення, на наш погляд, буде доцільним порівняти ці два алгоритми.

Основними відмінностями «Калина» від «Rijndael» (AES) є: збільшена кількість циклів шифрування (запас стійкості); використання додавання за модулем 264 і за модулем 2 для введення ключової інформації (захист від алгебричних атак, лінійного та диференціального криптоаналізів, інтерполяційної атаки тощо); використання чотирьох блоків нелінійного перетворення (S-блоків) замість одного (додатковий захист від алгебричних атак, поліпшення властивостей розсіювання алгоритму – покращені статистичні властивості, відповідно, більш високий рівень стійкості до диференціального та лінійного криптоаналізів тощо); використання випадково сформованих чотирьох блоків, відібраних критеріями стійкості до диференціального, лінійного криптоаналізів, ступені нелінійності булевих функцій (на відміну від S-блоку Rijndael/Camellia та інших шифрів, що використовують звернення в полі та, відповідно, квадратичні залежності між входом і виходом, – захист від алгебричних атак); принципово нова схема створення підключів (захист від усіх відомих атак на схеми створення підключів); досить висока продуктивність; можливість відновлення сеансового ключа за окремим підключем (додатковий захист від атак, що виконують відновлення підключів).

Усі поліпшення спрямовані на збільшення стійкості та запобігання потенційним вразливостям відносно Rijndael, виявленим в останні роки [12].

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ яке зображено на рисунку 5.1. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні розділи:

– Блок кнопок швидкого доступу до функцій програми.

– Вікно відображення існуючих підключень.

– Блок меню.

– Блок закладок.

Блок меню складається з наступних елементів: Файл; З’єднання; Безпека; Журнал; Параметри; Довідка. Блок закладок складається з наступних елементів: З’єднання; Безпека; Журнал.

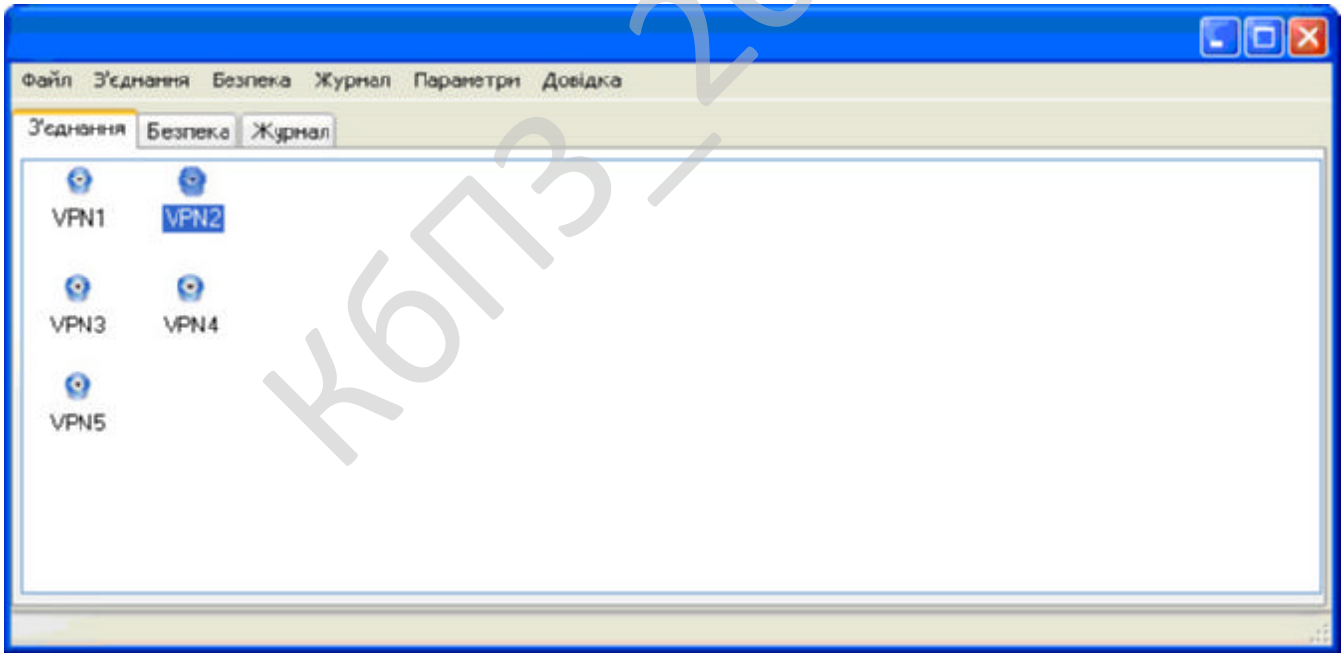

Рисунок 5.1 – Головне вікно програми (створення VPN-з’єднань)

Розроблена програма має дуже простий і інтуїтивно зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним

середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий.

Обрано умови розповсюдження – proprietary software. Програмне забезпечення, на яке зберігаються як немайнові, так і майнові авторські права. Отримавши або придбавши таке програмне забезпечення, користувач отримує обмежені права користування ним: може бути заборонено або закрито доступ до коду (вивчення), внесення змін, тиражування, розповсюдження та перепродаж. Програмне забезпечення вважається власницьким, якщо наявне хоча б одне з перелічених обмежень. Найчастіше основним методом захисту майнових прав на власницьке ПЗ, поза ліцензійною угодою, власник обирає закриття сирцевого коду, захищаючи свій продукт від модифікації і вбудовуючи системи обмеження користування через авторизацію.

Таке програмне забезпечення називається закритим. Проте, код власницького продукту може бути і відкритим, але власник може обмежити права користувача умовами користувацької ліцензії.

Власницьке програмне забезпечення та комерційне програмне забезпечення не є синонімами – власницьким може бути і безплатне (тобто, некомерційне) програмне забезпечення.

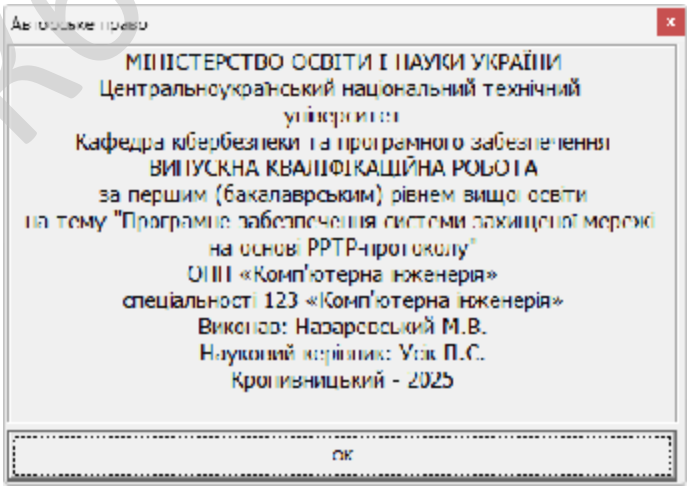

Рисунок 5.2 – Авторське право

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи захищеної мережі на основі PPTP-протоколу.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем захищеної мережі на основі PPTP-протоколу.

– Досліджена система захищеної мережі на основі PPTP-протоколу.

– На основі отриманих результатів досліджень створена програмна реалізація системи захищеної мережі на основі PPTP-протоколу.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання захищеної мережі на основі PPTP-протоколу.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об’єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи захищеної мережі на основі PPTP-протоколу. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне

програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм ДСТУ 7624:2014.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Loren Kohnfelder. Designing Secure Software. No Starch Press. 2022. 332 p.

2. Samir Kumar Rakshit. Ethical Hacker's Penetration Testing Guide. BPB Online. 2022. 509 p.

3. Corey J. Ball. Hacking APIs. No Starch Press. 2022. 353 p.

4. Kevin Beaver. Hacking for Dummies. John Wiley & Sons. 2022. 419 p.

5. Mark S. Merkow. Practical Security for Agile and DevOps. CRC Press. 2022. 236 p

6. Derek Fisher. Application Security Program Handbook. Manning Publications. 2021. 155 p.

7. Cameron Wyatt PH.D. Kali Linux Tutorial. Independently published. 2021. 60 p.

8. Alex Matrosov, Eugene Rodionov, Sergey Bratus. Rootkits and Bootkits. No Starch Press. 2019. 450 p.

9. Lakhno, V., Malyukov, V., Smirnov, O., Bebeshko, B., Chubaievskiy, V., Zhumadilova, M., Malyukova, I., Smirnov, S. «Multifactorial Model for Targeted Attacks Counteracting Within the Framework of a Multi-Step Quality Game with Fuzzy Information». *8th International Symposium on Intelligent Informatics, ISI 2023*, 2025. vol 389. pp 377-389. Springer, Singapore.

10. Kuznetsov O., Frontoni E., Kuznetsova Y., Smirnov O., Moskovchenko I. «Trust-Based Security Architecture for Edge Computing: A Simulation Study of Dynamic Trust Evolution and Attack Detection». *CEUR Workshop Proceedings,* 2024, 3909, pp. 227–241.

11. Kuznetsov, O., Frontoni, E., Kryvinska, N., Chevardin, V., Smirnov, O. «Wireless Network Encryption Stream Ciphers, Computational Modeling, and Security Analysis». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 379–402.

12. Kuznetsov, O., Frontoni, E., Kryvinska, N., Smirnov, O., Imoize, G.L. «Computational Modeling of Enhanced Spread Spectrum Codes for Asynchronous Wireless Communication». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 403–447.

13. Ткаченко, О., Ільєнко, А., Улічев, О., Мелешко, Є., Смірнов, О. «Правові засади поширення інформаційних впливів в соціальних мережах». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»,* 2024. № 2(26), С. 170–188.

14. Смірнова Т.В., Конопліцька-Слободенюк О.К., Буравченко К.О., Смірнов С.А., Кравчук О.В., Козірова Н.Л., Смірнов О.А. «Дослідження технологій забезпечення кібербезпеки хмарних сервісів IaaS, PaaS та SaaS». *Кібербезпека: освіта, наука, техніка.* 2024. №4(24), С. 6-27.

15. Вінтенко, Б., Миронець, І., Смірнов, О., Кравчук, О., Козірова, Н., Савеленко, Г., Коваленко, А. «Дослідження вимог та аналіз кібербезпеки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Кібербезпека: освіта, наука, техніка.* 2024. №3(23), С. 111-131.

16. Батрак О., Смірнова Т., Гнатюк В., Одарченко Р., Смірнов О. «Дослідження показників ефективності функціонування та перспектив розвитку систем IP-телефонії». *Підводні технології,* 2024, № 13, с. 28-35.

17. Kuznetsov, O., Kryvinska, N., Ilchenko, O., Smirnova, T., Ulianovska, Y. «Comparative Analysis of Cryptocurrency Trading Platforms Using the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, 2023, 3628, pp. 106-115.

18. Akhalaia, G., Iavich, M., Iashvili, G., Prysiazhnyy, D., Smirnova, T. «Secure Encrypted Connection on Georgian Website». *CEUR Workshop Proceedings,* 2023, 3550, pp. 313-320.

19. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56

20. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

21. Kuznetsov, O., Kandiy, S., Frontoni, E., Smirnov, O. «Trade-offs in Post-Quantum Cryptography: A Comparative Assessment of BIKE, HQC, and Classic McEliece». *CEUR Workshop Proceedings,* Volume 3504, 2023, pp. 1-11.

22. Smirnov, O., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebeshko, B. «Selection of a Rational Composition of İnformation Protection Means Using a Genetic Algorithm». *In: Rajakumar, G., Du, KL., Vuppalapati, C., Beligiannis, G.N. (eds) Intelligent Communication Technologies and Virtual Mobile Networks. Lecture Notes on Data Engineering and Communications Technologies*, vol 131. 2023. Springer, Singapore. pp. 21-34.

23. Смірнова Т.В., Гнатюк С.О., Бердибаєв Р.Ш., Сидоренко В.М., Жигаревич О.К., «Система корелювання подій та управління інцидентами кібербезпеки на об'єктах критичної інфраструктури». *Кібербезпека: освіта, наука, техніка*, №*3*(19), 2023, С. 176-196.

24. Смірнов О.А. Козлов Я.О., Смірнова Т.В. «Дослідження застосування SIEM-систем для забезпечення кібербезпеки та захисту інформації». *ІІ Міжнародна науково-практична Інтернет-конференція «Інновації та перспективні шляхи hозвитку інформаційних технологій (ІПШРІТ-2023)»* м.Черкаси 6 грудня 2023 року – Черкаси: ЧДТУ.– 2023. – С.251-252.

25. Козлов Я.О., Смірнова Т.В., Смірнов О.А. «Дослідження SIEM-систем для забезпечення кібербезпеки». *VII міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології" до 30-ти річчя кафедри кібербезпеки та програмного забезпечення,* м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 26.

26. Козлов Я.О., Козірова Н.Л., Смірнов О.А. «Дослідження структури та принципу роботи SIEM-системи». *VII міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології" до 30-ти річчя*

кафедри кібербезпеки та програмного забезпечення, м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 59.

27. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А., Коваленко А.С. «Дослідження нормативних документів та галузевих стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». *Системи управління, навігації та зв'язку*, 2023, вип. 2(72), С. 170-178.

28. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» *CEUR Workshop Proceedings,* Volume 3187, 2022,

29. Smirnov O.A., Al-Oraiqat A.M., Ulichev O.S., Meleshko Ye.V., Al-Rawashdeh H.S., Polishchuk L.I. «Modeling strategies for information influence dissemination in social networks». *Journal of Ambient Intelligence and Humanized Computing* Volume 13, Issue 5. Springer, Cham. 2022, pp. 2463-2477.

30. Смірнова Т.В., Гнатюк С.О., Сидоренко В.М., Юдін О.Ю., Сидоренко С.Ю., «Модель визначення критичності галузевих інформаційно-телекомунікаційних систем». *Проблеми інформатизації та управління,* № 2(70). 2022. С. 28-37.

31. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку,* 2022, № 3(69). С. 93-98.

32. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»,* № 2 (307). С. 46-52. 2022.

33. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 1(67). С. 84-89.

34. Smirnov O., Kuznetsov A., Zhora V., Onikiychuk A., Pieshkova O. «Hiding Messages in Audio Files Using Direct Spread Spectrum». *11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021,* Cracow, Poland, 22-25 September 2021. P. 414-418

35. Smirnov O., Kuznetsov A., Lokotkova I., Kuznetsova T., Florov S., Lebid O. «Using Orthogonal Signals to Hide Information in Images». *4 IEEE International Conference on Advanced Information and Communication Technologies (AICT) - 2021*, Lviv, Ukraine, September 21-25, 2021. P. 255-260.

36. Smirnov O., Kuznetsov A., Girzheva O., Kiian A., Nakisko O., Kuznetsova T. «Advanced Code-Based Electronic Digital Signature Scheme». *2020 IEEE International Conference on Problems of Infocommunications Science and Technology, PIC S and T 2020,* Kharkiv, 6 October 2020-9 October 2020, P. 358-362.

37. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova K. «Data hiding scheme based on spread sequence addressing». *CEUR Workshop Proceedings* Volume 2805, 2020, Pages *44-58.*

38. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». *International Journal of Computing*; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

39. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». *CEUR Workshop Proceedings*. Volume 2740, 2020, Pages 102-114.

40. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». *Journal of theoretical and applied information technology* Vol.98. No 21, 2020, P. 3334-3346.

41. Smirnov O., Kuznetsov A., Arischenko A., Chepurko I., Onikiychuk A., Kuznetsova T. «Pseudorandom sequences for spread spectrum image steganography». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 122-131.

42. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New technique for data hiding in cover images using adaptively generated pseudorandom sequences». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 1-14.

43. Smirnov O., Lutsenko M., Kuznetsov A., Kiian A., Kuznetsova T., «Biometric cryptosystems: overview, state-of-the-art and perspective directions». *Lecture Notes in Networks and Systems*, vol 152. Springer, Cham. 2021, pp 66-84.

44. Smirnov O., Kuznetsov A., Onikiychuk A., Makushenko T., Anisimova O., Arischenko A. «Adaptive pseudo-random sequence generation for spread spectrum image steganography». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Ukraine, Kyiv, May 14-18. 2020. P. 161-165.

45. Smirnov O., Kuznetsov A., Kiian A., Babenko V., Perevozova I.,Chepurko I. «New Approach to the Implementation of Post-Quantum Digital Signature Scheme». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 166-171.

46. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

47. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In:

Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

48. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 125-136.

49. Smirnov, O., Shekhanin, K., Kuznetsov, A., Krasnobayev, V. «Detecting Hidden Information in FAT». *International Journal of Computer Network and Information Security (IJCNIS)*. Vol. 12, No. 3, 2020. PP.33-43.

50. Smirnov, O., Kuznetsov, A., Gorbacheva, L., Babenko, V., «Hiding data in images using a pseudo-random sequence», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 646-660.

51. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

52. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

53. Smirnov, O., Ulichev, O., Meleshko, Y., Khokh, V., Goncharenko, I. «Method of Choosing Objects for Informational Influence in Social Networks during Information Campaign Based on the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, Vol 2588, P. 215-227, 2019.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи захищеної мережі на основі PPTP-протоколу.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 46-02 від 17.01.2025 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи захищеної мережі на основі PPTP-протоколу.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи захищеної мережі на основі PPTP-протоколу;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Python.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 67 аркушів.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2025 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 6.06.2025 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

_____ Усік П.С.

Програмне забезпечення системи захищеної мережі на основі PPTP-протоколу

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 25

Літера: РП

Кропивницький – 2025 року

Основна програма

```
import socket
import ssl
import threading
import struct
import hashlib
import base64
import time
import logging
import os
import random

# Ініціалізація логування
logging.basicConfig(filename='pptp_secure.log',              level=logging.DEBUG,
format='%(asctime)s - %(levelname)s - %(message)s')

# Параметри сервера
SERVER_HOST = '0.0.0.0'
SERVER_PORT = 1723

# Ключі для авторизації
USERNAME = "client"
PASSWORD = "password123"

# Конфігурація шифрування
CERT_FILE = 'server.crt'
KEY_FILE = 'server.key'

# Мапа активних клієнтів
clients = {}

# Буфер розміру
BUFFER_SIZE = 1024

# Встановлення шифрування SSL
def create_ssl_context():
    context = ssl.create_default_context(ssl.Purpose.CLIENT_AUTH)
    context.load_cert_chain(certfile=CERT_FILE, keyfile=KEY_FILE)
    return context

# Обробка авторизації клієнта
def authenticate(connection):
    try:
        connection.send(b"Username: ")
        username = connection.recv(BUFFER_SIZE).decode().strip()
        connection.send(b"Password: ")
        password = connection.recv(BUFFER_SIZE).decode().strip()

# Перевірка автентичності
        if username == USERNAME and password == PASSWORD:
            connection.send(b"Authenticated\n")
            return True
        else:
            connection.send(b"Authentication Failed\n")
            return False
    except Exception as e:
        logging.error(f"Authentication error: {e}")
        return False

# Імітація PPTP пакету
def generate_pptp_packet(payload):
    header = struct.pack('!HH', 1, len(payload))
```

```
    return header + payload

# Обробка даних PPTP
def handle_pptp_data(data):
    decoded = base64.b64decode(data)
    return decoded[::-1]

# Шифрування даних
def encrypt_data(data):
    m = hashlib.sha256()
    m.update(data)
    return m.hexdigest().encode()

# Генерація пакету відповіді
def respond_packet(message):
    payload = message.encode()
    packet = generate_pptp_packet(payload)
    return packet

# Потік для обробки клієнта
def client_thread(client_conn, address):
    logging.info(f"New connection from {address}")
    try:
        if not authenticate(client_conn):
            client_conn.close()
            return

        clients[address] = client_conn
        while True:
            data = client_conn.recv(BUFFER_SIZE)
            if not data:
                break

# Обробка отриманих даних
            response = handle_pptp_data(data)
            encrypted = encrypt_data(response)
            packet = respond_packet(encrypted.decode())
            client_conn.send(packet)
    except Exception as e:
        logging.error(f"Error with client {address}: {e}")
    finally:
        logging.info(f"Connection closed: {address}")
        client_conn.close()
        del clients[address]

# Створення SSL-серверного сокету
def start_ssl_server():
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.bind((SERVER_HOST, SERVER_PORT))
    sock.listen(5)

    context = create_ssl_context()

    logging.info(f"SSL PPTP server started on port {SERVER_PORT}")

    while True:
        client_sock, addr = sock.accept()
        ssl_conn = context.wrap_socket(client_sock, server_side=True)
        threading.Thread(target=client_thread, args=(ssl_conn, addr)).start()

# Генерація сертифікату якщо відсутній
def generate_self_signed_cert():
    if not os.path.exists(CERT_FILE) or not os.path.exists(KEY_FILE):
```

```
        os.system('openssl req -new -x509 -days 365 -nodes -out server.crt -
keyout server.key -subj "/C=UA/ST=Kyiv/L=Kyiv/O=VPN/OU=Dev/CN=localhost"')

# Основна функція запуску сервера
def main():
    generate_self_signed_cert()
    start_ssl_server()

# Виклик головної функції
if __name__ == '__main__':
    main()

# Резервна функція тестування пакету
def test_packet_creation():
    msg = "TestPPTP"
    packet = generate_pptp_packet(msg.encode())
    print(f"Packet: {packet}")

# Функція перевірки хешування
def test_encryption():
    data = b"SampleData"
    encrypted = encrypt_data(data)
    print(f"Encrypted: {encrypted}")

# Функція для демонстрації декодування
def test_handle_pptp():
    original = b"HelloWorld"
    encoded = base64.b64encode(original[::-1])
    decoded = handle_pptp_data(encoded)
    print(f"Decoded: {decoded}")

# Додаткові функції для роботи з PPTP (імітація реального протоколу)
def simulate_gre_tunnel():
    time.sleep(random.randint(1,3))
    logging.info("GRE tunnel simulation active")

def maintain_keepalive():
    while True:
        for addr in list(clients):
            try:
                clients[addr].send(b'KEEPALIVE\n')
            except:
                continue
        time.sleep(10)

# Запуск keepalive потоку
def start_keepalive_thread():
    threading.Thread(target=maintain_keepalive, daemon=True).start()

# Функція підрахунку активних клієнтів
def get_active_clients_count():
    return len(clients)

# Ініціалізація при імпорті
start_keepalive_thread()
```

Файл FlasQ.py

```
import sqlite3
from flask import Flask, request, render_template_string
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes
import base64
import threading
import time
import os

DATABASE = 'users.db'
KEY = get_random_bytes(16)
BLOCK_SIZE = 16

app = Flask(__name__)

def pad(data):
    length = BLOCK_SIZE - (len(data) % BLOCK_SIZE)
    return data + (chr(length) * length).encode()

def unpad(data):
    return data[:-ord(data[len(data)-1:])]

def encrypt_aes(data):
    cipher = AES.new(KEY, AES.MODE_ECB)
    return base64.b64encode(cipher.encrypt(pad(data)))

def decrypt_aes(enc):
    cipher = AES.new(KEY, AES.MODE_ECB)
    return unpad(cipher.decrypt(base64.b64decode(enc)))

def init_db():
    conn = sqlite3.connect(DATABASE)
    c = conn.cursor()
    c.execute('''CREATE TABLE IF NOT EXISTS users (
                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                    username TEXT NOT NULL UNIQUE,
                    password TEXT NOT NULL,
                    role TEXT NOT NULL,
                    login_attempts INTEGER DEFAULT 0,
                    locked INTEGER DEFAULT 0)''')
    conn.commit()
    conn.close()

def add_user(username, password, role):
    conn = sqlite3.connect(DATABASE)
    c = conn.cursor()
    c.execute("INSERT OR IGNORE INTO users (username, password, role) VALUES (?,
?, ?)", (username, password, role))
    conn.commit()
    conn.close()

def get_user(username):
    conn = sqlite3.connect(DATABASE)
    c = conn.cursor()
    c.execute("SELECT * FROM users WHERE username = ?", (username,))
    user = c.fetchone()
    conn.close()
    return user

def update_login_attempts(username, attempts):
    conn = sqlite3.connect(DATABASE)
```

```
    c = conn.cursor()
    c.execute("UPDATE users SET login_attempts = ? WHERE username = ?",
(attempts, username))
    conn.commit()
    conn.close()

def lock_user(username):
    conn = sqlite3.connect(DATABASE)
    c = conn.cursor()
    c.execute("UPDATE users SET locked = 1 WHERE username = ?", (username,))
    conn.commit()
    conn.close()

def is_locked(username):
    conn = sqlite3.connect(DATABASE)
    c = conn.cursor()
    c.execute("SELECT locked FROM users WHERE username = ?", (username,))
    result = c.fetchone()
    conn.close()
    return result and result[0] == 1

def authenticate_user(username, password):
    user = get_user(username)
    if not user:
        return False, "No user"
    if is_locked(username):
        return False, "Locked"
    stored_password = user[2]
    if stored_password == password:
        update_login_attempts(username, 0)
        return True, user[3]
    else:
        attempts = user[4] + 1
        if attempts >= 5:
            lock_user(username)
        update_login_attempts(username, attempts)
        return False, "Wrong"

@app.route('/')
def home():
    return '<h2>PPTP Secure Admin Panel</h2><a href="/users">View Users</a>'

@app.route('/users')
def list_users():
    conn = sqlite3.connect(DATABASE)
    c = conn.cursor()
    c.execute("SELECT username, role, login_attempts, locked FROM users")
    users = c.fetchall()
    conn.close()
    html = "<h3>Registered Users</h3><table
border='1'><tr><th>Username</th><th>Role</th><th>Attempts</th><th>Locked</th></t
r>"
    for u in users:
        html += f"<tr><td>{u[0]}</td><td>{u[1]}</td><td>{u[2]}</td><td>{'Yes' if
u[3] else 'No'}</td></tr>"
    html += "</table>"
    return html

def monitor_login_activity():
    while True:
        conn = sqlite3.connect(DATABASE)
        c = conn.cursor()
        c.execute("SELECT username, login_attempts FROM users")
```

```
        all_users = c.fetchall()
        for user in all_users:
            if user[1] > 3:
                print(f"Suspicious activity: {user[0]} has {user[1]} attempts")
        conn.close()
        time.sleep(15)

def admin_cli():
    while True:
        cmd = input("Admin> ")
        if cmd == "exit":
            break
        elif cmd == "list":
            conn = sqlite3.connect(DATABASE)
            c = conn.cursor()
            c.execute("SELECT username, role FROM users")
            for row in c.fetchall():
                print(row)
            conn.close()
        elif cmd.startswith("add"):
            _, uname, pwd, role = cmd.split()
            add_user(uname, pwd, role)
        else:
            print("Unknown command")

def main():
    init_db()
    add_user("admin", "adminpass", "admin")
    add_user("user1", "userpass", "user")
    threading.Thread(target=monitor_login_activity, daemon=True).start()
    threading.Thread(target=admin_cli, daemon=True).start()
    app.run(port=5000)

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

Файл listX.py

```
import secrets
import time
import hmac
import hashlib
from flask import Flask, request, jsonify
from collections import defaultdict
import threading
import socket
import select

app = Flask(__name__)

whitelist = set()
blacklist = set()

session_tokens = {}
session_expiry = {}
MAX_IDLE_TIME = 30
MAX_RATE = 1024
user_traffic = defaultdict(int)
last_active = {}

user_2fa_secrets = {}
active_tokens = {}

def add_to_whitelist(ip):
    whitelist.add(ip)

def add_to_blacklist(ip):
    blacklist.add(ip)

def is_ip_allowed(ip):
    if ip in blacklist:
        return False
    if len(whitelist) > 0:
        return ip in whitelist
    return True

def generate_token(username):
    token = secrets.token_hex(16)
    session_tokens[token] = username
    session_expiry[token] = time.time() + 3600
    return token

def validate_token(token):
    if token in session_tokens:
        if time.time() < session_expiry[token]:
            return session_tokens[token]
    return None

def expire_sessions():
    while True:
        now = time.time()
        expired = [t for t in session_expiry if session_expiry[t] < now]
        for t in expired:
            del session_expiry[t]
            del session_tokens[t]
        time.sleep(10)

def create_2fa_secret(username):
    secret = secrets.token_hex(8)
```

```
    user_2fa_secrets[username] = secret
    return secret

def generate_2fa_code(secret):
    ts = int(time.time() / 30)
    return              hmac.new(secret.encode(),              str(ts).encode(),
hashlib.sha256).hexdigest()[:6]

def verify_2fa_code(username, code):
    if username in user_2fa_secrets:
        secret = user_2fa_secrets[username]
        return code == generate_2fa_code(secret)
    return False

def mark_active(username):
    last_active[username] = time.time()

def disconnect_inactive_users():
    while True:
        now = time.time()
        to_remove = []
        for user, last in last_active.items():
            if now - last > MAX_IDLE_TIME:
                to_remove.append(user)
        for u in to_remove:
            if u in active_tokens:
                del active_tokens[u]
            if u in last_active:
                del last_active[u]
        time.sleep(5)

def enforce_rate_limit(user, size):
    user_traffic[user] += size
    if user_traffic[user] > MAX_RATE:
        return False
    return True

@app.route('/login', methods=['POST'])
def login():
    data = request.json
    ip = request.remote_addr
    if not is_ip_allowed(ip):
        return jsonify({'status': 'denied'}), 403
    username = data.get('username')
    password = data.get('password')
    if username == "admin" and password == "admin":
        secret = create_2fa_secret(username)
        return jsonify({'2fa_secret': secret})
    return jsonify({'status': 'fail'}), 401

@app.route('/verify', methods=['POST'])
def verify():
    data = request.json
    username = data.get('username')
    code = data.get('code')
    if verify_2fa_code(username, code):
        token = generate_token(username)
        active_tokens[username] = token
        mark_active(username)
        return jsonify({'token': token})
    return jsonify({'status': 'fail'}), 401

@app.route('/data', methods=['POST'])
```

```
def data():
    token = request.headers.get('Authorization')
    username = validate_token(token)
    if not username:
        return jsonify({'status': 'invalid'}), 401
    data_size = len(request.data)
    if not enforce_rate_limit(username, data_size):
        return jsonify({'status': 'rate-limit'}), 429
    mark_active(username)
    return jsonify({'status': 'ok'})

def simple_tcp_server():
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.bind(('0.0.0.0', 8081))
    sock.listen(5)
    inputs = [sock]
    connections = {}
    while True:
        readable, _, _ = select.select(inputs, [], [], 1)
        for s in readable:
            if s is sock:
                conn, addr = sock.accept()
                ip = addr[0]
                if not is_ip_allowed(ip):
                    conn.close()
                    continue
                inputs.append(conn)
                connections[conn] = (addr, time.time())
            else:
                data = s.recv(1024)
                if data:
                    user_traffic[s] += len(data)
                    if user_traffic[s] > MAX_RATE:
                        s.send(b'Rate limit exceeded\n')
                        inputs.remove(s)
                        s.close()
                    else:
                        s.send(b'OK\n')
                        connections[s] = (connections[s][0], time.time())
                else:
                    inputs.remove(s)
                    del connections[s]
                    s.close()

threading.Thread(target=expire_sessions, daemon=True).start()
threading.Thread(target=disconnect_inactive_users, daemon=True).start()
threading.Thread(target=simple_tcp_server, daemon=True).start()

if __name__ == '__main__':
    app.run(port=6000)
```

Файл render_activity.py

```
import logging
import shutil
import time
import os
from flask import Flask, jsonify, redirect, url_for, request,
render_template_string
from threading import Thread
import requests
import json
import datetime

app = Flask(__name__)

LOG_FILE = 'user_activity.log'
CONFIG_FILE = 'server.conf'
BACKUP_DIR = 'backups'

logging.basicConfig(filename=LOG_FILE, level=logging.INFO,
format='%(asctime)s - %(message)s')

activity_data = []

@app.before_request
def log_request_info():
    ip = request.remote_addr
    path = request.path
    method = request.method
    log_entry = f'{ip} {method} {path}'
    logging.info(log_entry)
    activity_data.append({
        'time': datetime.datetime.now().isoformat(),
        'ip': ip,
        'method': method,
        'path': path
    })

def render_activity():
    html = "<h2>Live Network Activity</h2><table
border='1'><tr><th>Time</th><th>IP</th><th>Method</th><th>Path</th></tr>"
    for entry in reversed(activity_data[-50:]):
        html +=
f"<tr><td>{entry['time']}</td><td>{entry['ip']}</td><td>{entry['method']}</td><t
d>{entry['path']}</td></tr>"
    html += "</table>"
    return html

@app.route('/activity')
def activity():
    return render_activity()

def backup_config():
    while True:
        if not os.path.exists(BACKUP_DIR):
            os.makedirs(BACKUP_DIR)
        timestamp = time.strftime('%Y%m%d%H%M%S')
        dst = os.path.join(BACKUP_DIR, f'config_backup_{timestamp}.conf')
        if os.path.exists(CONFIG_FILE):
            shutil.copy(CONFIG_FILE, dst)
        time.sleep(60)

OAUTH_CLIENT_ID = 'your-client-id'
OAUTH_CLIENT_SECRET = 'your-client-secret'
```

```
        OAUTH_REDIRECT_URI = 'http://localhost:5000/callback'
        OAUTH_AUTH_URL = 'https://accounts.google.com/o/oauth2/auth'
        OAUTH_TOKEN_URL = 'https://oauth2.googleapis.com/token'
        OAUTH_USERINFO_URL = 'https://www.googleapis.com/oauth2/v1/userinfo'

        @app.route('/login/oauth')
        def login_oauth():
            params = {
                'client_id': OAUTH_CLIENT_ID,
                'redirect_uri': OAUTH_REDIRECT_URI,
                'response_type': 'code',
                'scope': 'email profile',
                'access_type': 'offline'
            }
            url = OAUTH_AUTH_URL + '?' + '&'.join([f"{k}={v}" for k, v in
params.items()])
            return redirect(url)

        @app.route('/callback')
        def callback():
            code = request.args.get('code')
            data = {
                'code': code,
                'client_id': OAUTH_CLIENT_ID,
                'client_secret': OAUTH_CLIENT_SECRET,
                'redirect_uri': OAUTH_REDIRECT_URI,
                'grant_type': 'authorization_code'
            }
            token_res = requests.post(OAUTH_TOKEN_URL, data=data)
            token_json = token_res.json()
            access_token = token_json.get('access_token')
            headers = {'Authorization': f'Bearer {access_token}'}
            userinfo_res = requests.get(OAUTH_USERINFO_URL, headers=headers)
            userinfo = userinfo_res.json()
            return jsonify(userinfo)

        def renew_certificates():
            while True:
                os.system("certbot renew --quiet")
                time.sleep(86400)

        @app.route('/')
        def home():
            return '<h2>PPTP System Home</h2><a href="/activity">View Network
Activity</a> | <a href="/login/oauth">Login via OAuth</a>'

        Thread(target=backup_config, daemon=True).start()
        Thread(target=renew_certificates, daemon=True).start()

        if __name__ == '__main__':
            app.run(port=5000)
```

Файл pptp_a.py

```
import argparse
import logging
import sys
import socket
import subprocess
import time
from .logger import setup_logger
from .pptp import PPTPAutomaton, PPTPInfo
from .ppp import LCPEnumAuthMethodAutomaton
from .ppp_eap import EAPNegotiateAutomaton
from .ppp_chap import CHAPAutomaton
from .capture import PacketRecorder
from .authmethods import EAPAuthMethodSet, AuthMethodSet, PAP, CHAP_MD5,
CHAP_SHA1, MSCHAP, MSCHAPv2, EAP,\
                         get_all_eap_authmethods, EAPTLS, EAPPEAP, EAPCHAP,
EAPMSEAP


def check_raw_sock_perm():
    """
    Check if user has permissions to create RAW sockets
    :return:
        bool: True if raw socket can be succesfully created, False otherwise
    """
    from scapy.config import conf
    try:
        sck = conf.L3socket()
    except:
        return False
    sck.close()
    return True


def enabled_state_to_string(state):
    """
    Translate state to string
    :param state: bool or None
    :return:
        basestring:
    """
    if state is None:
        return 'Unknown'
    else:
        return 'Enabled' if state else 'Disabled'


def set_iptables_drop_icmp_protocol_unreachable():
    """
    Setup iptables firewall rule to drop ICMP protocol-unreachable packets
    """
    cmd = 'iptables -I OUTPUT -p icmp --icmp-type protocol-unreachable -j DROP
2>&1 1>/dev/null &&'\
          'iptables -I FORWARD -p icmp --icmp-type protocol-unreachable -j DROP
2>&1 1>/dev/null'
    ret_val = subprocess.call(cmd, shell=True)
    if ret_val != 0:
        print >> sys.stderr, 'Failed to add iptables ICMP protocol-unreachable
dropping rule'


def restore_iptables_drop_icmp_protocol_unreachable():
    """
    Remove iptables rule to drop ICMP protocol-unreachable packets
    :return:
    """
    cmd = 'iptables -D OUTPUT -p icmp --icmp-type protocol-unreachable -j DROP
2>&1 1>/dev/null &&' \
```

```
          'iptables -D FORWARD -p icmp --icmp-type protocol-unreachable -j DROP
2>&1 1>/dev/null'
    ret_val = subprocess.call(cmd, shell=True)
    if ret_val != 0:
        print >> sys.stderr, 'Failed to restore iptables rules'


def get_target_address_info(target):
    """
    Return hostname, alias list and ip address for ip address or domain
    :param target: ip address or domain
    :return:
        (hostname, alias list, ip)
    """
    target_hostname = None
    target_alias_list = None
    target_ip = None
    try:
        (target_hostname, target_alias_list, target_ip) =
socket.gethostbyaddr(target)
    except socket.herror:
        target_ip = [target]
        # TODO check that target IP is proper IP address
    return target_hostname, target_alias_list, target_ip


def print_header(str):
    """
    Print simple header/title
    :param str: text of title
    """
    print '{0}\n{1:^50}\n{0}'.format('='*50, str)


def print_property(property_name, value):
    """
    Simple wrapper to print named property
    :param property_name:   name of property
    :param value:           value of property
    """
    print '{0:25} {1}'.format(property_name, value)


def print_cert_str(cert_str):
    """
    Parse and print info from certificate
    :param cert_str: info from certificate formatted like
PROPERTY1=VALUE1/PROPERTY2=VALUE2
    :return:
    """
    for s in cert_str.split('/'):
        if '=' not in s:
            continue
        kv = s.split('=')
        print_property(kv[0]+':', kv[1])


def print_cert_info(method):
    """
    Print certificate info of EAP method
    :param method: EAPAuthMethod instance
    """
    if method is not None and method.get_enabled_state():
        print_header(str(method) + ' Certificate')
        if method.cert is not None:
            print_property('Serial:', str(method.cert.serial))
            print 'Issuer'
            print_cert_str(method.cert.issuer_str)
            print 'Subject'
```

```
            print_cert_str(method.cert.subject_str)
            print_property('Validity:', '%s to %s' % (method.cert.notBefore_str,
method.cert.notAfter_str))


def print_results(target_hostname, alias_list, target_ip, lcp_auth_methods,
eap_auth_methods, pptp_info, args):
    """
    Print formated test results
    :param target_hostname:     hostname of target server
    :param alias_list:          alias list of target server
    :param target_ip:           ip of target server
    :param lcp_auth_methods:    LCPAuthMethodSet instance with states of PPP
auth methods
    :param eap_auth_methods:    EAPAuthMethodSet instance with states of EAP
auth methods
    :param pptp_info:           PPTPInfo instance with info from control
connection
    :param args:                command line arguments from ArgumentParser
    """
    print_header('PPTP info')
    print_property('PPTP server domain:', target_hostname if target_hostname is
not None else 'Unknown')
    aliases = alias_list if alias_list is not None and len(alias_list) > 0 else
['Unknown']
    print_property('PPTP server aliases:', aliases[0])
    for alias in aliases[1:]:
        print_property('', alias)
    print_property('PPTP server IP:', target_ip[0] if target_ip[0] is not None
else 'Unknown')
    print_property('PPTP server port:', args.port)
    if pptp_info is not None:
        assert isinstance(pptp_info, PPTPInfo)
        print_property('Protocol version:',
pptp_info.get_protocol_version_str())
        print_property('Maximum channels:', pptp_info.get_maximum_channels())
        print_property('Firmware revision:', pptp_info.get_firmware_revision())
        print_property('Framing capabilities:',
pptp_info.get_framing_capabilities())
        print_property('Bearer capabilities:',
pptp_info.get_bearer_capabilities())
        print_property('Host name:', pptp_info.get_host_name())
        print_property('Vendor string:', pptp_info.get_vendor_string())
        print_property('Connection speed:', pptp_info.get_connection_speed())
        print_property('GRE window size:', pptp_info.get_window_size())
        print_property('Packet processing delay:', pptp_info.get_window_size())
        print_property('Physical channel id:',
pptp_info.get_physical_channel_id())


    if lcp_auth_methods is not None:
        print_header('PPP Authentication')
        ppp_methods = [PAP, CHAP_MD5, CHAP_SHA1, MSCHAP, MSCHAPv2, EAP]
        for ppp_method in ppp_methods:
            method_state = lcp_auth_methods.get_method_enabled_state(ppp_method)
            extra =
lcp_auth_methods.get_method(ppp_method).get_extra_as_string()
            if extra:
                print_property(str(ppp_method()),
enabled_state_to_string(method_state) + ',' +

lcp_auth_methods.get_method(ppp_method).get_extra_as_string())
            else:
                print_property(str(ppp_method()),
enabled_state_to_string(method_state))

    if eap_auth_methods is not None:
        print_header('EAP Authentication (Identity
\'{0}\')'.format(args.identity))
```

```
        if eap_auth_methods.is_disabled_for_identity():
            print 'EAP is disabled for identity \'{0}\''.format(args.identity)
        else:
            for eap_method in eap_auth_methods.get_methods():
                if isinstance(eap_method, EAPTLS) or isinstance(eap_method,
EAPPEAP):
                    print_property(eap_method,
enabled_state_to_string(eap_method.get_enabled_state()))
                else:
                    extra = eap_method.get_extra_as_string()
                    if extra == '':
                        print_property(eap_method,
enabled_state_to_string(eap_method.get_enabled_state()))
                    else:
                        print_property(eap_method,
enabled_state_to_string(eap_method.get_enabled_state()) + ',' + extra)

        print_cert_info(eap_auth_methods.get_method(EAPTLS))
        print_cert_info(eap_auth_methods.get_method(EAPPEAP))

    print_header('Warning')

    if lcp_auth_methods is not None:
        if lcp_auth_methods.get_method_enabled_state(PAP):
            print 'PAP Authentication is enabled. User credentials are sent in
plaintext, no encryption is used.'
        if lcp_auth_methods.get_method_enabled_state(CHAP_MD5) or
lcp_auth_methods.get_method_enabled_state(CHAP_SHA1):
            print 'CHAP Authentication is enabled. Connection is vulnerable to
MitM attacks, no encryption is used.'
        if lcp_auth_methods.get_method_enabled_state(MSCHAP) or
lcp_auth_methods.get_method_enabled_state(MSCHAPv2):
            print 'MSCHAP/MSCHAPv2 Authentication is enabled. NTHash of user
password can be recovered by sniffing' \
                  'network traffic.'
    if eap_auth_methods is not None:
        if eap_auth_methods.get_method_enabled_state(EAPPEAP):
            print 'PEAP Authentication is enabled. Make sure all clients are
validating server certificate.'
        if eap_auth_methods.get_method_enabled_state(EAPCHAP):
            print 'EAP-MD5 Authentication is enabled. Connection is vulnerable
to MitM attacks, no encryption si used.'
        if eap_auth_methods.get_method_enabled_state(EAPMSEAP):
            print 'MS-EAP (MSCHAPv2) Authentication is enabld. NTHash of user
password can be recovered by sniffing'\
                  'network traffic'
    print 'You are using PPTP. The PPTP protocol is not considered to be really
secure, even when configured properly.'


def main():
    parser = argparse.ArgumentParser('PPTP Auditing tool')
    parser.add_argument('target', help='Adress of PPTP server')
    parser.add_argument('-p', '--port', help='PPTP port', type=int,
default=1723, dest='port')
    parser.add_argument('-l', '--log', help='Filename for log',
default='log.txt',
                        dest='logfile')
    parser.add_argument('-i', '--identity', help='Identity to use with EAP',
default='user',
                        dest='identity')
    parser.add_argument('-c', '--dump_cert_file', help='File to dump server TLS
certificate to', default=None,
                        dest='cert_file')
    parser.add_argument('-e', '--test_all_eap_methods', help='Test all EAP auth
methods', default=False,
                        action='store_true')
    parser.add_argument('-r','--record_pcap', help='Record communication with
target to pcap file', default=None,
```

```
                        dest='pcap_file')
    parser.add_argument('-di', '--dont_drop_icmp', help='Dont drop ICMP
protocol-unreachable packets', default=True,
                        dest='drop_icmp', action='store_false')
    parser.add_argument('-d', '--log-debug', help='Log debug information',
                        action='store_const', dest='loglevel',
                        const=logging.DEBUG, default=logging.INFO)
    parser.add_argument('-a', '--log-append', help='Append output to logfile
instead of truncating it',
                        action='store_const', dest='logfile_mode',
                        const='a', default='w')
    args = parser.parse_args()

    setup_logger(args)

    print 'PPTP Auditor'

    if not check_raw_sock_perm():
        print >> sys.stderr, 'You don\'t have sufficient permission to create
raw sockets.\n'\
                             'Try running pptp_auditor as root.'
        sys.exit(-1)

    if args.drop_icmp:
        set_iptables_drop_icmp_protocol_unreachable()

    (target_hostname, alias_list, target_ip) =
get_target_address_info(args.target)

    pkt_recorder = None
    if args.pcap_file is not None:
        pkt_recorder = PacketRecorder(args.target, args.pcap_file)
        pkt_recorder.start()
        time.sleep(0.5)

    print 'Probing enabled LCP authentication methods'
    lcp_auth_methods = AuthMethodSet()
    pptp_automaton = PPTPAutomaton(args.target, LCPEnumAuthMethodAutomaton,

ppp_automaton_kwargs={'lcp_auth_methods':lcp_auth_methods},
                               port=args.port)

    pptp_info = None
    eap_auth_methods = None
    try:
        pptp_info = pptp_automaton.run()

        for chap_method in [CHAP_MD5, CHAP_SHA1, MSCHAP, MSCHAPv2]:
            if pptp_info is not None and
pptp_info.ppp_info.get_method_enabled_state(chap_method):
                pptp_automaton = PPTPAutomaton(args.target, CHAPAutomaton,

ppp_automaton_kwargs={'chap_method': pptp_info.ppp_info.get_method(chap_method),

'lcp_auth_methods': lcp_auth_methods},
                                               port=args.port)
                pptp_automaton.run()

        if pptp_info is not None and
pptp_info.ppp_info.get_method_enabled_state(EAP):
            if args.test_all_eap_methods:
                eap_auth_methods =
EAPAuthMethodSet(methods=get_all_eap_authmethods())
            else:
                eap_auth_methods = EAPAuthMethodSet()

            while not eap_auth_methods.is_state_of_all_methods_known():
                assert (isinstance(eap_auth_methods, EAPAuthMethodSet))
                print 'Probing enabled EAP authentication methods {0}/{1}' \
```

```
                        .format(eap_auth_methods.get_number_of_known_methods(),
len(eap_auth_methods.get_methods()))
                pptp_automaton = PPTPAutomaton(args.target,
EAPNegotiateAutomaton,

ppp_automaton_kwargs={'cert_file': args.cert_file,
                                                                    'identity':
args.identity,

'eap_auth_methods': eap_auth_methods},
                                                port=args.port)
                pptp_automaton.run()
    except socket.error as sock_err:
        print >> sys.stderr, 'Unexpected connection error: {0}'.format(sock_err)
    except Exception as error:
        print >> sys.stderr, 'Unexpected error: {0}'.format(error)
    finally:
        if pkt_recorder is not None:
            pkt_recorder.stop()
        if args.drop_icmp:
            restore_iptables_drop_icmp_protocol_unreachable()

    print_results(target_hostname, alias_list, target_ip, lcp_auth_methods,
eap_auth_methods, pptp_info, args)
```

Файл authmethods.py

```
from scapy.layers.l2 import EAP as EAP_pkt, eap_types
from scapy.layers.ppp import PPP_LCP_Auth_Protocol_Option


class AuthMethod:
    """
    Base class for holding authentication method info and state
    """

    def __init__(self):
        self.enabled_state = None
        self.extra = {}
        self.cert = None

    def set_enabled(self):
        """Mark authentication method as enabled"""
        self.enabled_state = True

    def set_disabled(self):
        """Mark authentication method as disabled"""
        self.enabled_state = False

    def is_state_known(self):
        """
        Returns value based on whether the state of authentication
        method is known
        :return:
            True if authentication method is known
            False if authentication method is not known
        """
        return self.enabled_state is not None

    def get_enabled_state(self):
        """
        Returns value based on authentication method
        known state
        :return:
            True if authentication method is enabled
            False if authentication method is disabled
            None if state of authentication method is not known
        """
        return self.enabled_state

    def get_enabled_state_str(self):
        """
        Returns string describing authentication method
        known state
        :return:
            "Unknown"
            "Enabled"
            "Disabled"
        """
        if self.enabled_state is None:
            return "Unknown"
        elif self.enabled_state:
            return "Enabled"
        else:
            return "Disabled"

    def add_extra(self, key, value):
        """
```

```
        Sets extra information for the authentication method
        :param key:     name of the information, i.e. user_name
        :param value:   extra information about the method
        """
        self.extra[key]=value

    def get_extra_as_string(self):
        """
        Returns extra information string
        :return:
            String containing extra information, each line formatted as 'key:
value'
        """
        return     "\n".join(["{0}:     {1}".format(x,y)     for     (x,y)     in
self.extra.items()])

    def __str__(self):
        raise NotImplementedError


class LCPAuthMethod(AuthMethod):
    """ Base class for holding PPP Authentication option state"""

    def get_lcp_option(self):
        """
        Get LCP option for the method
        :return:
            LCPOption: LCPOption instance containing request for the method
        """
        return self.pkt

    def is_lcp_option(self, option):
        """
        Check if option is requesting the method
        :param option: LCPOption instance
        :return:
            bool:
        """
        if option.auth_protocol == self.pkt.auth_protocol:
            if option.auth_protocol == 0xc223:
                return option.algorithm == self.pkt.algorithm
            else:
                return True
        else:
            return False

    def __str__(self):
        raise NotImplementedError


class PAP(LCPAuthMethod):
    """Password authentication protocol"""
    def __init__(self):
        LCPAuthMethod.__init__(self)
        self.pkt = PPP_LCP_Auth_Protocol_Option()

    @classmethod
    def __str__(cls):
        return "PAP"

    def is_lcp_option(self, option):
        return option.auth_protocol == 0xc023
```

```
class CHAP_MD5(LCPAuthMethod):
    """Challenge-Handshake authentication protocol + MD5"""
    def __init__(self):
        LCPAuthMethod.__init__(self)
        self.pkt       =       PPP_LCP_Auth_Protocol_Option(auth_protocol=0xc223,
algorithm='MD5')

    @classmethod
    def __str__(cls):
        return "CHAP+MD5"


class CHAP_SHA1(LCPAuthMethod):
    """Challenge-Handshake authentication protocol + SHA1"""
    def __init__(self):
        LCPAuthMethod.__init__(self)
        self.pkt       =       PPP_LCP_Auth_Protocol_Option(auth_protocol=0xc223,
algorithm='SHA1')

    @classmethod
    def __str__(self):
        return "CHAP+SHA1"


class MSCHAP(LCPAuthMethod):
    """Microsoft Challenge-Handshake authentication protocol"""
    def __init__(self):
        LCPAuthMethod.__init__(self)
        self.pkt       =       PPP_LCP_Auth_Protocol_Option(auth_protocol=0xc223,
algorithm='MS-CHAP')

    @classmethod
    def __str__(cls):
        return "MS-CHAP"


class MSCHAPv2(LCPAuthMethod):
    """Microsoft Challenge-Handshake authentication protocol v2"""
    def __init__(self):
        LCPAuthMethod.__init__(self)
        self.pkt       =       PPP_LCP_Auth_Protocol_Option(auth_protocol=0xc223,
algorithm="MS-CHAP-v2")

    @classmethod
    def __str__(cls):
        return "MS-CHAP-v2"


class EAP(LCPAuthMethod):
    """Extensible authentication protocol"""
    def __init__(self):
        LCPAuthMethod.__init__(self)
        self.pkt = PPP_LCP_Auth_Protocol_Option(auth_protocol=0xc227)

    @classmethod
    def __str__(cls):
        return "EAP"


class AuthMethodSet:
    """
    Helper class to hold state of all check authentication methods
```

```
    """
    MAX_TRIES = 5

    def __init__(self, methods=[PAP(), CHAP_MD5(), CHAP_SHA1(), MSCHAP(),
MSCHAPv2(), EAP()]):
        """Initialize authentication method set with given method instances"""
        self.methods = {method:0 for method in methods}

    def get_next_to_try(self):
        """

        :return:
            LCPAuthMethod: LCPAuthMethod instance of next authentication method
state of which is not known,
                           if state of all methods is known, it returns None
        """
        unknown_methods = sorted([method for method in self.methods.keys() if
not method.is_state_known()
                                  and self.methods[method] < self.MAX_TRIES],
                                 key=lambda x: self.methods[x])
        if len(unknown_methods) <= 0:
            return None
        method_to_try = unknown_methods[0]
        self.methods[method_to_try] += 1
        return unknown_methods[0]

    def get_method(self, cls):
        """
        Returns method state instance for authentication method specified by its
class
        :param cls: LCPAuthMethod subclass
        :return:
            LCPAuthMethod: Instance of LCPAuthMethod from the set, None if there
is no such method
        """
        for method in self.methods.keys():
            if isinstance(method, cls):
                return method
        return None

    def set_method_state_from_option(self, option, state):
        """
        Sets state of method requested by LCPOption to provided state
        :param option: (LCPOption) LCPOption instance
        :param state: (bool) true if method should be enabled, false otherwise
        """
        for method in self.methods.keys():
            if method.is_lcp_option(option):
                if state:
                    method.set_enabled()
                else:
                    method.set_disabled()

    def get_method_for_option(self, option):
        """
        Returns method state instance for authentication method from provided
LCPOption
        :param option: (LCPOption) LCPOption instance with authentication method
        :return:
            LCPAuthMethod instance from set, or None if there is no such method
        """
        for method in self.methods.keys():
            if method.is_lcp_option(option):
```

```
                return method
        return None

    def enable_method_from_option(self, option):
        """
        Mark method in provided LCPOption as enabled
        :param option: (LCPOption) LCPOption instance with authentication method
        """
        self.set_method_state_from_option(option, True)

    def disable_method_from_option(self, option):
        """
        Mark method in provided LCPOption as disabled
        :param option: (LCPOption) LCPOption instance with authentication method
        """
        self.set_method_state_from_option(option, False)

    def get_methods(self):
        """
        Get list of all method state instances from set
        :return:
            list of LCPAuthMethod instances
        """
        return [method for method in self.methods.keys()]

    def get_method_enabled_state(self, method):
        """
        Get state of method specified by LCPAuthMethod subclass
        :param method: LCPAuthMethod subclass
        :return:
            True if method is enabled
            False is method is disabled
            None if state of the method is not known
        """
        for m in self.methods.keys():
            if isinstance(m, method):
                return m.get_enabled_state()
        return None

    def get_number_of_known_methods(self):
        """
        Returns number of methods from the state state of whose is already known
        """
        known_methods_nr = 0
        for m in self.methods.keys():
            if m.is_state_known():
                known_methods_nr += 1
        return known_methods_nr

    def is_state_of_all_methods_known(self):
        """
        Whether state of all methods in set is known
        :return:
            bool:
        """
        return self.get_number_of_known_methods() == len(self.methods.keys())

    def __str__(self):
        """
        Get string representation of state of all methods
        """
        return                ','.join(['{0}:                {1}'.format(method,
method.get_enabled_state_str()) for method in self.methods.keys()])
```

```
class EAPAuthMethod(AuthMethod):
    """ Base class for EAP authentication method state """
    def __init__(self, eap_type=None):
        """
        Initialize EAPAuthMethod for provided eap type number
        :param eap_type: (int) eap type number according to IANA
        """
        AuthMethod.__init__(self)
        self.eap_type = eap_type

    def is_eap_option(self, eap_type):
        """Compare eap type to eap type of the method"""
        return self.eap_type == eap_type

    def get_eap_nak_response(self, id):
        """
        Get EAP-Legacy-Nak response packet requesting the method
        :param id: (int) response id
        :return:
            Instance of Scapy EAP-Legacy-Nak packet
        """
        return     EAP_pkt(code='Response',     id=id,     type='Legacy     Nak',
desired_auth_type=self.eap_type)

    def is_eap_request(self, request):
        """
        Check whether packet contains EAP request for the method
        :param request: Scapy packet to check
        :return:
            bool:
        """
        return EAP_pkt in request and request[EAP_pkt].type == self.eap_type

    def __str__(self):
        """
        Returns name of the EAP method according to eap type number
        """
        if self.eap_type in eap_types.keys():
            return eap_types[self.eap_type]
        else:
            return 'Unknown EAP auth method({0})'.format(self.eap_type)


def get_all_eap_authmethods():
    """
    Get method state objects for all EAP method types defined in Scapy
    :return:
        list of EAPAuthMethod instances
    """
    all_authmethods = []
    for eap_type in eap_types.keys():
        if eap_type == 4:
            all_authmethods.append(EAPCHAP())
        elif eap_type == 13:
            all_authmethods.append(EAPTLS())
        elif eap_type == 25:
            all_authmethods.append(EAPPEAP())
        elif eap_type == 29:
            all_authmethods.append(EAPMSCHAPv2())
        elif eap_type >= 4:
            all_authmethods.append(EAPAuthMethod(eap_type))
```

```
    return all_authmethods


class EAPCHAP(EAPAuthMethod):
    """EAP-MD5"""
    def __init__(self):
        EAPAuthMethod.__init__(self, 4)


class EAPTLS(EAPAuthMethod):
    """EAP-TLS"""
    def __init__(self):
        EAPAuthMethod.__init__(self, 13)


class EAPPEAP(EAPAuthMethod):
    """PEAP"""
    def __init__(self):
        EAPAuthMethod.__init__(self, 25)


class EAPMSCHAPv2(EAPAuthMethod):
    """MSCHAPv2"""
    def __init__(self):
        EAPAuthMethod.__init__(self, 29)


class EAPMSEAP(EAPAuthMethod):
    """MSCHAP"""
    def __init__(self):
        EAPAuthMethod.__init__(self, 26)
```