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
“____” _____ 2023 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки створення VPN підключень”

Виконав здобувач вищої освіти
IV курсу, групи КБ-19
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Москальов А.В.
«____» _____ 2023 р.

Керівник проекту
кандидат фізико-математичних наук, доцент
_____ Якименко Н.М.
«____» _____ 2023 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 *125 “*Кібербезпека*”*

Освітньо-професійна (освітньо-наукова) програма _*“*Кібербезпека*”*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2023 року

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

Москальову Антону Віталійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки створення VPN підключень*

2. Керівник роботи *Якименко Наталія Миколаївна, канд. фіз.-мат. наук, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 12-02 від 5.01.2023 року

3. Строк подання студентом роботи до захисту *23.05.2023 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи кібербезпеки створення VPN підключень*

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 кібербезпеки в промислову експлуатацію.

6. Висновки

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

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

7. Дата видачі завдання « 17 » січня 2023 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	10.03.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2023 р.	
3.	Розробка моделі компонента	20.03.2023 р.	
4.	Розробка структур даних	25.03.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2023 р.	
6.	Програмування алгоритмів	10.04.2023 р.	
7.	Оформлення ПЗ	17.04.2023 р.	
8.	Попередній захист роботи	23.05.2023 р.	

Дата видачі завдання
« 17 » січня 2023 р.

Підпис керівника
_____ Якименко Н.М.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2023 р.

Підпис здобувача
_____ Москальов А.В.
(прізвище та ініціали)

АНОТАЦІЯ

Москальов А.В. Програмне забезпечення системи кібербезпеки створення VPN підключень. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки створення VPN підключень.

Метою розробки є програмне забезпечення системи кібербезпеки створення VPN підключень.

Результат роботи – програмна реалізація системи кібербезпеки створення VPN підключень.

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 архітектури IBM PC з ОС Windows 10/11.

Програму розроблено в середовищі Embarcadero RAD Studio.

Ключові слова: кібербезпека, VPN

ABSTRACT

Moskalov A.V. Cyber security system software for creating VPN connections. 125 Cyber security. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this graduation thesis for the first (bachelor) level of higher education, software is developed, which is intended for the cyber security system of creating VPN connections.

The purpose of the development is the software of the cyber security system for creating VPN connections.

The result of the work is the software implementation of the cyber security system for creating VPN connections.

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 tools are provided.

The program can be used on PCs of IBM PC architecture with Windows 10/11 OS.

The program was developed in the Embarcadero RAD Studio environment.

Keywords: cyber security, VPN

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. Все частіше й частіше при підключенні до інтернету нам пропонують скористатися технологією VPN. При об'єднанні віддалених офісів однієї компанії в єдину комп'ютерну мережу також застосуються VPN. Якщо ви прийдете в Wi-Fi кафе, то теж можете зштовхнутися з тим, що необхідно настроїти з'єднання з VPN.

VPN розшифровується як Virtual Private Network, що означає «Віртуальна Приватна Мережа». VPN створюється поверх уже існуючої мережі, наприклад звичайної локальної мережі або Інтернет, і може поєднувати комп'ютери в різних куточках світу в одну логічну мережу. При цьому всі передані по такій мережі дані звичайно зашифровуються для захисту від несанкціонованого прослуховування й перехоплення. Таким чином, шифрування трафіку – одна з головних переваг використання технології віртуальних приватних мереж.

Якщо фізично комп'ютери поєднуються між собою кабелем або радіохвилями (Wi-Fi), то логічне їхнє об'єднання за допомогою VPN можливо тільки з використанням спеціального устаткування, називаного VPN-сервером. Це може бути просто комп'ютер, що має спеціальне програмне забезпечення. VPN-сервер управляє підключенням інших (звичайних) комп'ютерів до віртуальної мережі.

На комп'ютері, що підключається до віртуальної приватної мережі, настроюється спеціальне VPN з'єднання, у конфігурації якого вказується ім'я VPN-сервера й інші, потрібні для успішного підключення, параметри. У кожному конкретному випадку ці параметри можуть розрізнятися, але послідовність дій при створенні VPN підключення та сама.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки створення VPN підключень.

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

– Огляд існуючих систем створення VPN підключень.

– Дослідження системи кібербезпеки створення VPN підключень.

– Програмна реалізація системи кібербезпеки створення VPN підключень.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі створення VPN підключень.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки створення VPN підключень, 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

VPN (Virtual Private Network – віртуальна приватна мережа) – узагальнена назва технологій, що дозволяють забезпечити одне або кілька мережних з'єднань (логічну мережу) поверх іншої мережі (наприклад, Інтернет). Незважаючи на те, що комунікації здійснюються по мережах з меншим невідомим рівнем довіри (наприклад, по публічних мережах), рівень довіри до побудованої логічної мережі не залежить від рівня довіри до базових мереж завдяки використанню засобів криптографії (шифруванню, автентифікації, інфраструктури публічних ключів, засобам для захисту від повторів і зміни переданих по логічній мережі повідомлень).

Залежно від застосовуваних протоколів і призначення, VPN може забезпечувати з'єднання трьох видів: вузол-вузол, вузол-мережа й мережа-мережа.

Рівні реалізації

Звичайно VPN розгортають на рівнях не вище мережного, тому що застосування криптографії на цих рівнях дозволяє використовувати в незмінному виді транспортні протоколи (такі як TCP, UDP).

Користувачі Microsoft Windows позначають терміном VPN одну з реалізацій віртуальної мережі – PPTP, причому використовувану найчастіше не для створення приватних мереж.

Найчастіше для створення віртуальної мережі використовується інкапсуляція протоколу PPP у який-небудь інший протокол – IP (такий спосіб використовує реалізація PPTP – Point-to-Point Tunneling Protocol) або Ethernet (PPPoE) (хоча й вони мають розходження). Технологія VPN останнім часом використовується не тільки для створення властиво приватних мереж, але й деякими провайдерами «останньої милі» для надання виходу в Інтернет.

При належному рівні реалізації й використанні спеціального програмного забезпечення мережа VPN може забезпечити високий рівень шифрування переданої інформації. При правильному налаштуванні всіх компонентів технологія VPN забезпечує анонімність у Мережі.

Структура VPN

VPN складається із двох частин: «внутрішня» (підконтрольна) мережа, якої може бути трохи, і «зовнішня» мережа, по якій проходить інкапсульоване з'єднання (звичайно використовується Інтернет). Можливо також підключення до віртуальної мережі окремого комп'ютера. Підключення віддаленого користувача до VPN виробляється за допомогою сервера доступу, що підключений як до внутрішньої, так і до зовнішньої (загальнодоступної) мережі. При підключенні віддаленого користувача (або при установці з'єднання з іншою захищеною мережею) сервер доступу вимагає проходження процесу ідентифікації, а потім процесу автентифікації. Після успішного проходження обох процесів, віддалений користувач (віддалена мережа) наділяється повноваженнями для роботи в мережі, тобто відбувається процес авторизації.

Класифікація VPN

Класифікувати VPN рішення можна по декількох основних параметрах:

1. За ступенем захищеності використовуваного середовища:

– Захищені. Найпоширеніший варіант віртуальних приватних мереж. З його допомогою можливо створити надійну й захищену на основі ненадійної мережі, як правило, Інтернету. Прикладом захищених VPN є: IPSec, OpenVPN і PPTP.

– Довірчі. Використовуються у випадках, коли передавальне середовище можна вважати надійним й необхідно вирішити лише завдання створення віртуальної підмережі в рамках більшої мережі. Проблеми безпеки стають неактуальними. Прикладами подібних VPN рішень є: Multi-protocol label switching (MPLS) і L2TP (Layer 2 Tunnelling Protocol) (точніше сказати, що ці протоколи перекладають завдання забезпечення безпеки на інші, наприклад L2TP, як правило, використовується в парі з IPSec).

2. За способом реалізації:

– У вигляді спеціального програмно-апаратного забезпечення. Реалізація VPN мережі здійснюється за допомогою спеціального комплексу програмно-апаратних засобів. Така реалізація забезпечує високу продуктивність і, як правило, високий ступінь захищеності.

– У вигляді програмного рішення. Використовують персональний комп'ютер зі спеціальним програмним забезпеченням, що забезпечує функціональність VPN.

– Інтегроване рішення. Функціональність VPN забезпечує комплекс, що вирішує також завдання фільтрації мережного трафіку, організації мережного екрана й забезпечення якості обслуговування.

3. За призначенням:

– Intranet VPN. Використовують для об'єднання в єдину захищену мережу декількох розподілених філій однієї організації, що обмінюються даними по відкритим каналам зв'язку.

– Remote Access VPN. Використовують для створення захищеного каналу між сегментом корпоративної мережі (центральним офісом або філією) і одиночним користувачем, що, працюючи дома, підключається до корпоративних ресурсів з домашнього комп'ютера, корпоративного ноутбука, смартфона або інтернет-кіоску.

– Extranet VPN. Використовують для мереж, до яких підключаються «зовнішні» користувачі (наприклад, замовники або клієнти). Рівень довіри до них набагато нижче, ніж до співробітників компанії, тому потрібне забезпечення спеціальних «рубежів» захисту, що запобігають або обмежують доступ останніх до особливо коштовній, конфіденційній інформації.

– Internet VPN. Використовується для надання доступу до інтернету провайдерами, звичайно у випадку якщо по одному фізичному каналі підключаються декілька користувачів.

– Client/Server VPN. Він забезпечує захист переданих даних між двома

вузлами (не мережами) корпоративної мережі. Особливість даного варіанта у тому, що VPN будується між вузлами, що перебувають, як правило, в одному сегменті мережі, наприклад, між робочою станцією й сервером. Така необхідність дуже часто виникає в тих випадках, коли в одній фізичній мережі необхідно створити кілька логічних мереж. Наприклад, коли треба розділити трафік між фінансовим департаментом і відділом кадрів, що звертаються до серверів, які перебувають в одному фізичному сегменті. Цей варіант схожий на технологію VLAN, але замість поділу трафіку, використовується його шифрування.

4. За типом протоколу. Існують реалізації віртуальних приватних мереж під TCP/IP, IPX і AppleTalk. Але на сьогоднішній день спостерігається тенденція до загального переходу на протокол TCP/IP, і абсолютна більшість VPN рішень підтримує саме його. Адресація в ньому найчастіше вибирається у відповідності зі стандартом RFC5735, з діапазону Приватних мереж TCP/IP.

5. За рівнем мережного протоколу. За рівнем мережного протоколу на основі зіставлення з рівнями еталонної мережної моделі ISO/OSI.

Приклади VPN:

– IPSec (IP security) – часто використовується поверх IPv4.

– PPTP (point-to-point tunneling protocol) – розроблявся спільними зусиллями декількох компаній, включаючи Microsoft.

– PPPoE (PPP (Point-to-Point Protocol) over Ethernet).

– L2TP (Layer 2 Tunnelling Protocol) – використовується в продуктах компаній Microsoft і Cisco.

– L2TPv3 (Layer 2 Tunnelling Protocol version 3).

– OpenVPN SSL VPN з відкритим вихідним кодом, підтримує режими PPP, bridge, point-to-point, multi-client server.

Багато великих провайдерів пропонують свої послуги з організації VPN-мереж для бізнесів-клієнтів.

1.2 Область застосування

Щодо застосування, можна виділити чотири основних варіанти побудови мережі VPN, які використовуються в усьому світі.

– Варіант «Intrenet VPN», що дозволяє об'єднати в єдину захищену мережу кілька розподілених філій однієї організації, взаємодіючих по відкритих каналах зв'язку. Саме цей варіант одержав широке поширення в усьому світі, і саме його в першу чергу реалізують компанії-розроблювачі.

– Варіант «Client/Server VPN», що забезпечує захист переданих даних між двома вузлами (не мережами) корпоративної мережі. Особливість даного варіанта у тому, що VPN будується між вузлами, що перебувають, як правило, в одному сегменті мережі, наприклад між робочою станцією й сервером. Така необхідність дуже часто виникає в тих випадках, коли необхідно створити в одній фізичній, кілька логічних мереж. Наприклад, коли потрібно розділити трафік між фінансовим департаментом і відділом кадрів, які звертаються до серверів, що перебуває в одному фізичному сегменті. Цей варіант схожий на технологію VLAN, що діє на рівні вище канального.

– Варіант «Extranet VPN» призначений для тих мереж, куди підключаються так звані користувачі з боку, рівень довіри до яких набагато нижче, ніж до своїх співробітників.

– Варіант «Remote Access VPN», що дозволяє реалізувати захищену взаємодію між сегментом корпоративної мережі (центральним офісом або філією) і одиночним користувачем, що підключається до корпоративних ресурсів з будинку (домашній користувач) або через notebook (мобільний користувач). Даний варіант відрізняється тим, що віддалений користувач не має «статичної» адреси й підключається до ресурсу, який захищається, не через виділений пристрій VPN, а прямо із власного комп'ютера, де й установлюється програмне забезпечення, що реалізує функції VPN. Цим варіантом ми й скористаємося.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки створення VPN підключень, 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

Послуги VPN можна розділити на три типи. Перший – VPN доступ – забезпечення захищених з'єднань для віддаленого доступу одиночних користувачів (мобільні й надомні працівники) до корпоративних мереж через інфраструктуру оператора зв'язку, з дотриманням тої ж політики, що й у приватній мережі.

Другий – VPN-інтранет – побудова внутрішніх корпоративних мереж через мережі загального доступу, з дотриманням політики приватної мережі.

Третій – VPN-екстранет – побудова корпоративних мереж через мережі загального доступу з наданням додаткового доступу для постачальників, замовників і ділових партнерів при дотриманні політики приватної мережі.

Для побудови VPN пропонується широкий вибір архітектур – віртуальні канали Frame Relay/ATM, IP-тунелі з використанням технологій GRE або IPSec, а також MPLS на маршрутизуємих магістралях, IP+ATM магістралях або Frame Relay/ATM магістралях.

Таблиця 2.1 – Послуги й архітектура VPN

Послуги	Архітектура	Технології
VPN доступ	Від клієнта, від вузла доступу	L2F/L2TP, IPSec, Dial, ISDN, DSL, Mobile IP, Cable
VPN інтранет і екстранет	IP тунелі, Віртуальні канали, MPLS	GRE, IPSec, Mobile IP, GRE, IPSec, Mobile IP, IP, IP+ATM

Всі варіанти VPN, наведені в даній таблиці, забезпечують функції поділу доступу. А от функції забезпечення цілісності й конфіденційності переданої інформації VPN забезпечують тільки при використанні технологій шифрування даних. Саме про цей варіант побудови VPN буде йти мова далі.

Розглянемо існуючі програмні продукти по VPN підключеннях.

VpnProxy

VpnProxy – це VPN проксі сервер, що дозволяє переміщати з'єднання між VPN клієнтом і сервером.

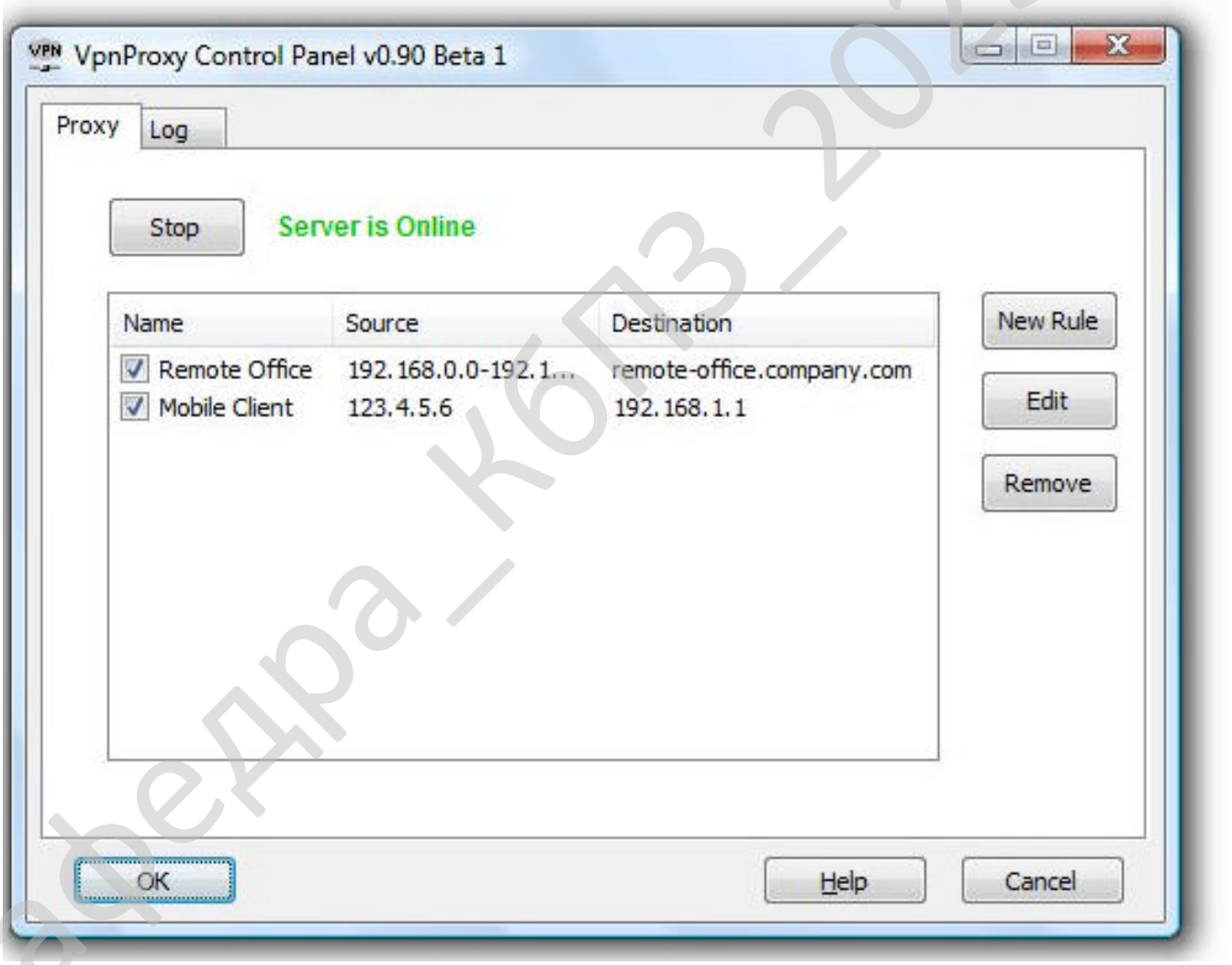

Рисунок 2.1 – Інтерфейс користувача VpnProxy

З VpnProxy можна:

– Установлювати VPN підключення в мережі через брандмауер або NAT, не міняючи мережну конфігурацію й не зменшуючи рівень безпеки.

– Створювати й управляти правилами для контролю VPN доступу до мережі.

Технічні деталі:

– Підтримує всі VPN рішення, що працюють по PPTP протоколах, включаючи: стандартні Windows, Unix і Mac VPN.

– Може функціонувати в операційних системах Microsoft Windows (це не обмежує можливості доступу до VpnProxy з інших платформ, таких як: Linux і Mac).

– Програма розширювана до рівня підприємства – підтримує сотні одночасних підключень.

ViPNet OFFICE

ViPNet OFFICE – програмне рішення для організації віртуальних приватних мереж (VPN) типових конфігурацій – захищених мереж ViPNet™.

ViPNet OFFICE призначений для використання в невеликих локальні й розподілених IP-мережах, забезпечує захищену роботу віддалених користувачів VPN з будь-якими типами підключення до мережі Інтернет.

ViPNet OFFICE – це пакет програмного забезпечення, що містить три компоненти технології ViPNet: ViPNet Manager, ViPNet Координатор і ViPNet Клієнт. На відміну від рішення ViPNet CUSTOM створення й модифікація захищеної мережі виробляється за допомогою ViPNet Manager.

ViPNet Manager – це полегшена версія програмного забезпечення ViPNet Адміністратор, що дозволяє простим образом, на інтуїтивному рівні, задавати й змінювати структуру захищеної мережі ViPNet™. Вхідний до складу ViPNet Координатора й Клієнта ViPNet Драйвер забезпечує надійне шифрування IP-трафіка, а також функції міжмережного й персонального мережного екранів відповідно. Як криптографічне ядро в програмному забезпеченні рішення ViPNet

OFFICE використовується СКЗІ Домен-к. Якщо потрібно швидко, якісно й за невеликі гроші побудувати VPN і забезпечити захищену передачу інформації через Інтернет, то ViPNet OFFICE – найбільш оптимальне рішення завдань.

NCP Secure Entry Linux Client

NCP Secure Entry Client – це програмне забезпечення для комунікацій з використанням IPSec гейтвея. Даний клієнт підтримує всі мережні передачі, такі як ISDN, аналогова мережа, xDSL, Internet, GSM, GPRS, IMTS, LAN, WLAN, а також сполучимо з усіма версіями операційної системи Windows. Інтегрований персональний брандмауер захищає персональний комп'ютер від атак у всіх комунікаційних середовищах.

Рисунок 2.2 – Інтерфейс користувача NCP Secure Entry Client

Всі механізми безпеки починають працювати із запуском системи, а також залишаються активними навіть при завершенні роботи Entry Client. Виявлення IP-NAT, певних правил фільтрації, а також визначення гарячих точок (автоматичне визначення середовища поточної мережі й активація відповідних умов фільтрації). Ідентифікація, що ставиться до місця розташування гейтвея, може бути запущена за допомогою OTP сертифікатів. Вся інформація зашифровується під час передачі.

Підтримка: Triple DES 128, 192-біт, Blowfish 128-біт, AES 128, 192, 256-біт і RSA 1024, 2048-біт. Цей клієнт також може бути включений в IT середовище без постійних IP адрес. Для підключення до центрального VPN гейтвею з мінливими публічними IP адресами використовується DynDNS (Динамічний DNS). Як альтернатива до Microsoft RAS, даний клієнт має свій власний додаток для дозвону, що незалежно від операційної системи. Надаються наступні можливості: економія грошей на телефонні розмови, передача й оптимізація центральних ресурсів VPN за допомогою інтелігентної функції короткої затримки, віддаленої адміністрації й захисту від сторонніх додатків для дозвона.

AdventNet ManageEngine Asset Explorer

ManageEngine AssetExplorer – це розширена система керування майном, що пропонує відмінний огляд майна по всьому підприємству й контроль керування всім IT і не IT майном. Дана програма пропонує швидкий вид для спостереження й керування володінням за всіма м майном. ManageEngine AssetExplorer сканує й проводить аудит по всім робочих станціях вашого підприємства підключеним по LAN, WAN і VPN. Дана програма сканує вашу мережу й автоматично відкриває всі наявні програми на кожному комп'ютері.

ManageEngine AssetExplorer допоможуть просканувати всі робочі станції Windows з Active Directory. Виявляє Linux комп'ютери й інше IT майно, таке як: принтери, маршрутизатори й світчи, використовуючи мережне сканування. Детальне сканування надає деталі по встаткуванню й програмам, установленим

на всіх комп'ютерах. Менеджери по майну можуть проводити аудита робочих станцій і одержувати звіти по встаткуванню й програмам для кращого контролю.

ManageEngine AssetExplorer допомагає створити сфальцьовані групи по майну для кращого керування й контролю. ManageEngine AssetExplorer допомагає призначити майно користувачам і стежити за ними по всьому циклі життя майна. Детальна історія володіння майном допомагає відстежити за всіма попередніми власниками й записами всіх змін у майні. Одержуйте детальні звіти по інвентарі встаткування про робочі станції в мережі, такі як: деталі операційної системи, деталі ЦП, мережна інформація, деталі жорсткого диска, RAM слоти. Звіт по програмному інвентарі, ви одержите інформацію із всіх установлених програм на всіх комп'ютерах у мережі. Дана програма допомагає згрупувати ліцензовані програми, дозволяючи стежити за загальною кількістю реально встановлених напроти куплених програмних ліцензій.

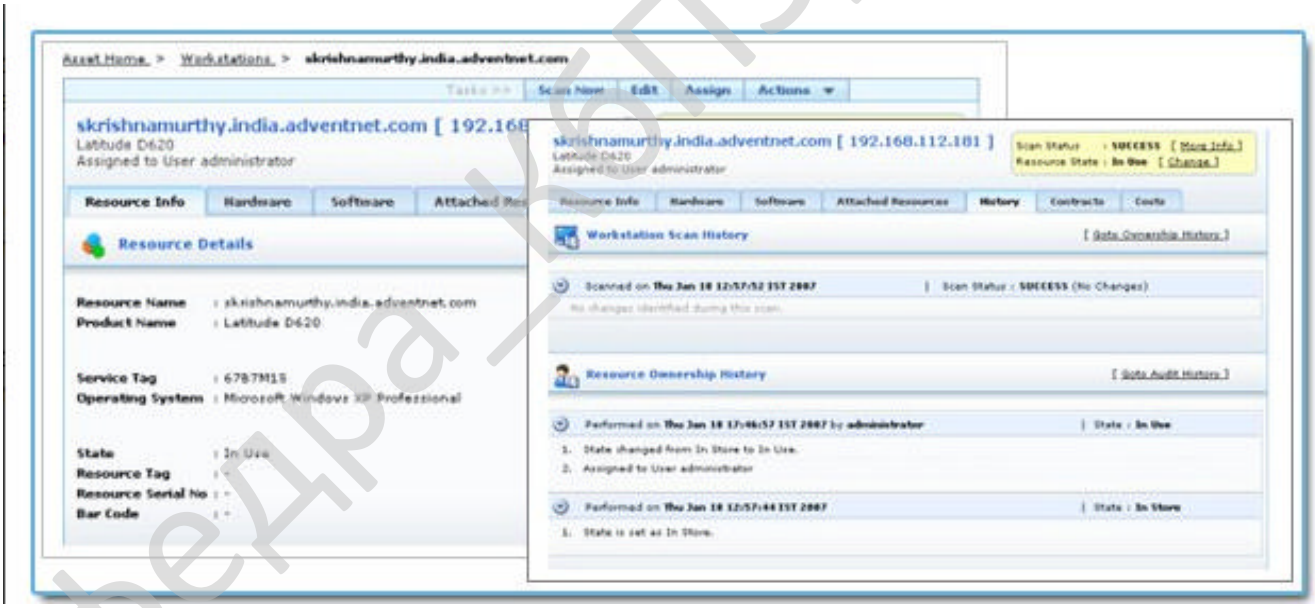

Рисунок 2.3 – Інтерфейс користувача ManageEngine AssetExplorer

ManageEngine AssetExplorer допомагає стежити за всіма важливими змінами, такими як керовані програми й забуті програмні інсталяції, зміни в устаткуванні, які трапилися за останні 7 днів.

Менеджери по майну зможуть із легкістю стежити за сумісністю встаткування, ліцензованими програмами й програмами з минулою ліцензією. Дана програма надає систему покупки, що допомагає управляти м програмним забезпеченням з підтвердженням.

iPIG Secure Access VPN Server

Установить VPN сервер менше ніж за одну мінуту. Безкоштовна програма клієнт. Використовуючи потужну 256 бітну технологію кодування AES, iOpus Private Internet Gateway (iPIG) створює безпечний «тунель», що захищає вхідну й вихідну комунікацію (електронну пошту, веб, службу обміну швидкими повідомленнями, VOIP, дзвінки, FTP і т.д.) на будь-якому Wi-Fi модулі або бездротовій мережі.

iPIG захищає дані від найбільш витончених методів онлайн шпигунства й підглядання, таких як «Атак злих близнюків». На додаток, ваша інформація не тільки захищена між м комп'ютером і бездротовою точкою доступу. Але й по шляху до безпечного з'єднання iOpus' в Інтернеті. Що гарантує безпеку передачі даних через повітря до бездротового Ethernet з'єднанню.

На відміну від інших технологій, iPig працює з кожним Інтернет з'єднанням (Wifi, WLAN і бездротовою Ethernet) і не вимагає конфігурації будь-якого виду. Просто запускаєте браузер, поштовий клієнт або чат програму й перемикаєтеся на iPig кодування: iPig захоплює весь Інтернет трафік перед тим як він залишає комп'ютер і кодує його для більшої безпеки. iPig також працює з більшістю популярних брандмауерів, таких як Zonealarm або Norton Інтернет Безпека. Дана програма підтримує Windows 2000, XP, 7, сервер 2003, 2008 (на жаль, Windows 98, ME не підтримуються).

Якщо подивитися правді в очі, то за рахунок того, що ринок засобів шифрування за рубежем був менш «зарегульованим» і більше конкурентним, розвиток подібних технологій там ішов швидше. Ми зі смутком змушені констатувати, що також як і в інших областях інфокомунікацій, продукти закордонних виробників перевершують вітчизняні. Причому ця різниця лежить не в стійкості криптографічних алгоритмів, а в зручності впровадження,

керування якістю обслуговування для різних типів потоків даних, масштабованості, взаємній сумісності й тому подібних речах. Інтеграція технологій VPN з іншими технологіями, такими як міжмережне екранування й/або маршрутизація також є великою перевагою західних рішень. Світовими лідерами, що ділять між собою ринок VPN, є такі компанії як Cisco Systems, Checkpoint Software, Juniper Networks і ряд інших. При цьому приємно відзначити, що українські розроблювачі VPN в останні роки активно трудилися, щоб зменшити відставання. І, що особливо викликає повагу, не тільки в технологічній області. Розробка й опублікування в IETF інформаційного RFC 4357 «Additional Cryptographic Algorithms for Use with GOST 28147-89, GOST R 34.10-94, GOST R 34.10-2001, and GOST R 34.11-94 Algorithms» з'явилося досить значним кроком, у тому числі в області сумісності між собою й самими українськими рішеннями. Активне додавання функцій підтримки якості обслуговування й відказостійкості дозволяє забезпечити стабільне функціонування подібних систем у великих розподілених мережах. Підтвердженням цього є дуже розповсюджена ситуація, коли сама по собі мережна інфраструктура будуватися на встаткуванні західних виробників, а вже на цю інфраструктуру накладаються мережі VPN побудовані на базі сертифікованих рішень. Подібний підхід використовується в багатьох державних установах України.

До основних українських виробників засобів VPN можна віднести наступні компанії: «Амікон» – із продуктами ФПСУ-IP і ФПСУ-IP-міні, «Інформзахист» – із продуктом «Континет», «Інфотекс» – із серією продуктів VipNet, «Фактор ТС» – із серією продуктів Діоніс, «Інфосистеми Джет» – із продуктом «Тропа», «Елвіс Плюс» – із продуктом «Застава», «З-терра» – із продуктом CSP VPN, «Голлард» – с продуктом «Заслін». Короткий порівняльний аналіз продуктів можна побачити в таблиці 3.2. На закінчення, хотілося б ще раз підкреслити основні тенденції українського ринку VPN, а саме: високий рівень державного регулювання, що приводить до повільного росту цього сегмента ринку ІБ; технологічна перевага західних продуктів за критеріями

масштабованості, простоти обслуговування, і інтеграції додаткових функцій; широкий вибір і активний технологічний розвиток українських засобів побудови VPN. Два останніх аспекти в результаті дають нам саме головне – наявність на ринку й можливість вибору саме того рішення, що щонайкраще буде відповідати потребам кожного конкретного підприємства, залежно від його масштабу, виду власності, категорій оброблюваної інформації й т.д.

Таблиця 2.2 – Короткий порівняльний аналіз популярних українських VPN-продуктів

Рішення	Виробник	Алгоритми шифрування	Протоколи передачі даних	Мережне середовище	Швидкість шифрування	Підтримувані операційні системи
Програмно-апаратний комплекс «ФПСУ-IP»	Амікон	ДСТУ 28147:2009	IP v. 4, Протокол VPN – © AMICON.	Ethernet, TCP-IP	До 270 Мбіт/с.	Власна, функціонально замкнута, 32-розрядна.
Програмно-апаратний VPN-агент «КРИПТОН IP Mobile»	Анкад	ДСТУ 28147:2009	Ethernet/Fast Ethernet, TCP/IP	Ethernet/Fast Ethernet	До 80 Мбіт/с	Windows, Linux
Широкополосний апаратний IP-шифратор «Заслін»	Голлард	ДСТУ 28147:2009	IP v. 4	Заснована на протоколі IP	180 Мбіт/с	Будь-які

Продовження таблиці 2.2

Рішення	Виробник	Алгоритми шифрування	Протоколи передачі даних	Мережне середовище	Швидкість шифрування	Підтримувані операційні системи
Апаратно-програмний комплекс шифрування «Континент»	Інформзахист	ДСТУ 28147:2009	Власний, на основі ДСТУ 28147:2009		До 80 Мбіт/с (на процесорі Pentium IV, 2,4 МГц)	Апаратно-програмна реалізація. Власна ОС на базі FreeBSD Windows 98/NT/2000/XP
VPN-шлюз Тропа-джет	Інфо-системи Джет	ДСТУ 28147:2009 Хеш ДСТР 34.11-94	IPsec, UDP	TCP/IP	До 100 Мбіт/с	Solaris SPARC і x86
Програмний комплекс ViPNet	Інфотекс	ДСТУ 28147:2009	TCP/IP	Ethernet	До 100 Мбіт/с	Windows , Linux, Solaris 8
Багаторівневий крипто-маршрутизатор DioNIS FW/KM	Фактор-тс	ДСТУ 28147:2009	TCP/IP, TCP/IP over X.25	TCP/IP, X.25	100 Мбіт/с	Власна, DOS

Продовження таблиці 2.2

Рішення	Виробник	Алгоритми шифру-вання	Протоколи передачі даних	Мережне середовище	Швидкість шифрування	Підтри-мувані операційні системи
VPN-клієнт CSP VPN Client (Шлюз безпеки CSP VPN Gate)	З-терра CSP	ДСТУ 28147:2009, DES 3DES AES RSA MD5 SHA-1	IPsec (RFC 2401) AH (RFC 2402) ESP (RFC 2406) IKE (RFC 2409) OAKLEY (RFC 2412) ISAKMP (RFC 2408) ISAKMPDOI (RFC 2407) XAUTH	Ethernet 10/100/1000 Base V.35, RS530, X.21, G.703, G.704, G823 (у шлюзах CSP VPN Gate 3000, 7000 Cronyx)	До 230 Мбіт/с (IPsec ESP ДСТ28147) на одному процесорі Intel Xeon 3.06 Ггц.	Windows (Solaris 8, 9/Linux)
Застава	Елвіс-плюс	ДСТУ 28147:2009, ВЕСТУ-2М, AES, DES, 3DES, IDEA, RC 4-40, RC2, NULL	IPSec	Протоколи IP	160 Мбіт/с на однопроцесор-ній системі. 450 Мбіт/с на Х-процессор-ній системі.	Solaris 8 (64 bit) MCBC 3.0, Windows NT/ 2000/XP

2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки та мови програмування

Embarcadero Delphi, раніше Borland Delphi і Codegear Delphi, – інтегроване середовище розробки ПЗ для Microsoft Windows, Mac OS, iOS і Android мовою Delphi (що раніше носила назву Object Pascal), створена спочатку фірмою Borland і на даний момент приналежна й розроблювальна Embarcadero Technologies. Embarcadero Delphi є частиною пакета Embarcadero RAD Studio і поставляється в чотирьох редакціях: Community (поширюється безкоштовно й має обмежену ліцензію на використання в комерційних цілях), Professional, Enterprise і Architect.

Delphi 10.4 Sydney

Випущено 26 травня 2020 року. RAD Studio Delphi 10.4 забезпечує значно поліпшену високопродуктивну нативну підтримку Windows, кращу продуктивність розробки, миттєві підказки code completion, прискорення виконання коду із синтаксисом керованих записів, поліпшення виконання паралельних завдань на сучасних багатоядерних CPU, а також містить більш 1000 виправлень багів, поліпшення продуктивності середовища й бібліотек і багато чого крім того.

Основні можливості Delphi 10.4.1:

− Істотні розширення для Windows: поліпшення для застосунків на моніторах 4K High DPI, інтеграція з новим WebView2 на базі Chromium, використання розширених title bars, таких же, як в Office, Explorer, Google Chrome.

− Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

− Істотне поліпшення Delphi Code Insight (без можливого блокування IDE – в окремому процесі), що допоможе при роботі з великими проектами.

− Тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання.

− Розширена підтримка бібліотек C++: ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode.

− Відладник Win 64 (на LLDB) і збирач для C++.

− Поліпшення для C++: Включена велика кількість поліпшень STL з Dinkumware.

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

− Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API. Реалізація компонента Media Player для macOS тепер використовує Avfoundation. Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

− Численні поліпшення швидкості й стабільності роботи нашої бібліотеки The Parallel Programming Library (PPL).

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

− Клієнтські бібліотеки HTTP і REST Client розширені застосунковими можливостями роботи з HTTPS. Також були розширені можливості підтримки Amazon AWS services

− У технологію Visual LiveBindings внесена безліч поліпшень, у тому числі швидкодії, що стосуться, застосунків на VCL і FireMonkey

RAD Studio 10.4 Короткий огляд:

– Істотні розширення для Windows. Створення застосунків, що чудово виглядають, із чіткими елементами інтерфейсу на 4k моніторах High DPI за допомогою нової гнучкої підтримки стилів елементів керування на екрані. Інтеграція із сучасними, безпечними web-технологіями від Microsoft – новим WebView2 на базі Chromium. Використання сучасних розширених title bars, таких же, як в Office, Explorer, Google Chrome, у своїх проектах. Істотні поліпшення надійності налагодження в новому відладнику для C++ Windows 64-bit.

– Зросла продуктивність розробки. Ріст продуктивності за рахунок миттєвої реакції підказок code completion у середовищі IDE. Краща сумісність із уже наявною кодовою базою, і спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю. Швидке зв'язування даних і візуальних елементів за допомогою розширеної технології Visual LiveBindings з підвищеною швидкодією. Просте використання розповсюджених бібліотек C++, наприклад, ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode. Оновлена підтримка Amazon AWS cloud.

– Поліпшення швидкодії і якості. Більш 1000 поліпшень швидкодії і якості. Краща ефективність коду за допомогою нового синтаксису custom managed records. Більш швидке виконання паралельних завдань на сучасних багатоядерних CPU. Переконаєтеся в прискоренні відображення на екрані з підтримкою Metal API на macOS і iOS. Краща сумісність із уже наявною кодовою базою й спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю.

Істотне поліпшення Delphi Code Insight

Як найбільше й головне поліпшення інструментів програмування Delphi за багато років, в 10.4 Delphi Code Insight реалізований через Language Server Protocol (LSP). LSP – це технологія генерації результатів для code completion, навігації й інших сервісів в окремому процесі. Це значить, що code completion і Code Insight одержать більш точні результати без блокування IDE. 10.4 забезпечує набагато більш високу продуктивність розроблювачів, які працюють із більшими проектами, що містять мільйони рядків коду.

Delphi Custom Managed Records

Ключове розширення мови Delphi: тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання. Управляйте тем, як ці структури створюються, копіюються й звільняються з допомогу вашого коду, який буде виконуватися у відповідний момент.

Це розширює потужність конструкцій records в Delphi, які використовуться щоб одержати більшу ефективність у порівнянні із класами.

Єдине керування пам'яттю

Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

У порівнянні з Automatic Reference Counting (ARC), це дає кращу сумісність із існучим кодом і спрощує написання компонентів, бібліотек і застосунків.

ARC модель керування пам'яттю model залишилася для керування рядками й посиланнями на тип інтерфейсу на всіх платформах. Для C++ це означає, що при створенні й звільненні Delphi-style класів в C++ використовується звичайне керування пам'яттю, як у будь-якого heap-allocated класу C++, що значно знижує складність коду.

Розширена підтримка бібліотек C++

В 10.4 ми портували багато популярних бібліотек C++ у C++Builder.

Забезпечивши оптимізовану підтримку бібліотек ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode, поряд із уже підтримуваними Boost і Eigen, які можуть бути додані за допомогою менеджера пакетів Getit.

Win 64-відладник і збирач для C++

В 10.4 з'явився новий відладник C++ для Windows 64-bit. Відладник заснований на LLDB і показує значне збільшення стабільності при налагодженні 64-bit застосунків поряд з новими відладочними можливостями, такими як перегляд і інспекція типів начебто рядків C++ і Delphi, а також колекцій STL, включаючи std::vector, std::map і інших. Крім того, згенерована для застосунку відладочна інформація має інший внутрішній формат, сприяючи більш стабільному й багатому на можливості процесу налагодження, більш докладним перегляду й інспекції в debug-time.

Підвищення якості й швидкодії інструментів

– Велика кількість поліпшень STL від Dinkumware.

– Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.

– Поліпшена підтримка Cmake.

– Велика кількість виправлень для підвищення стабільності і якості.

– Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.

– Загальні вдосконалення в бібліотеці доступу до БД FireDAC, включаючи оновлені драйвера для FireBird, PostgreSQL і SQLite. Вибір статичного або динамічного підключення SQLite до застосунку.

Змінені стилі VCL для High DPI

В 10.4, архітектура стилізації VCL була суттєво розширена для підтримки High DPI і 4K моніторів. Тепер усі елементи UI на формі VCL автоматично масштабуться під відповідне до монітора дозвіл для показу форми. Був оновлений API стилізації для підтримки стилів high DPI.

Кожний графічний елемент UI може бути обраний з наборів різних масштабів і масштабований до потрібного DPI, що дає чітке зображення елементів UI на всіх моніторах.

Нові High DPI стилі й стилізація окремих VCL компонент

Обновлено велике число вбудованих і преміальних VCL стилів для підтримки нового режиму стилізації High-dpi. Це дозволяє вам створювати застосунку з відмінним дизайном для всіх моніторів.

Розроблювачі VCL застосунків тепер можуть використовувати трохи VCL стилів на різних формах в одному застосунку або в різних компонентів на одній формі. Це також включає стилізацію компонентів загальною темою для платформи. Крім застосункової гнучкості використання стилів, це дозволяє використовувати нестилізуємі компоненти із зовнішніх бібліотек в VCL застосунках, що використовуть стиль.

Поліпшена кроссплатформеність

– Додана підтримка Metal Driver GPU для macOS і iOS.

– Крім підтримки останнього iOS SDK, в RAD Studio 10.4 розроблювачі можуть задовольнити нові вимоги Apple до набору стартових екранів.

– Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

– Користувачам редакцій Enterprise або Architect доступна повна інтеграція Fmxlinux з IDE для створення клієнтських застосунків Linux з GUI.

– Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.

– Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

Дати випуску релізів пакетів тепер видні, і можливе сортування списку по цих датах; відбір тільки встановлених пакетів, контенту, доступного тільки при наявності підписки, багато чого іншого.

Універсальний інсталятор для установки Online і Offline

В 10.4 включений новий універсальний інсталятор, який використовує технологію на базі Getit. Цей інсталятор підтримує як online, так і offline (з ISO) варіанти установки.

Тепер обоє варіанта установки дозволяють вам указати початковий набір можливостей RAD Studio для установки, наприклад, свою комбінацію мов програмування й цільових платформ, мов інтерфейсу, і додавати до нього або видаляти непотрібне в будь-який момент.

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 кібербезпеки створення VPN підключень.

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 кібербезпеки 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 кібербезпеки 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

По своїй суті VPN має багато властивостей виділеної лінії, однак розгортається вона в межах загальнодоступної мережі, наприклад Інтернету. За допомогою методики туннелювання пакети даних транслюються через загальнодоступну мережу як по звичайному двоточечному з'єднанню. Між кожною парою « відправник-одержувач даних» установлюється своєрідний тунель – безпечне логічне з'єднання, що дозволяє інкапсулювати дані одного протоколу в пакети іншого. Дуже важливою властивістю тунелів є можливість диференціації різних типів трафіку й призначення їм необхідних пріоритетів обслуговування.

Основними компонентами тунелю є:

– ініціатор;

– маршрутизуєма мережа;

– тунельний комутатор;

– один або кілька тунельних термінаторів.

Ініціювати й розривати тунель можуть всілякі мережні пристрої й програмне забезпечення. Наприклад, тунель може бути ініційований ноутбуком мобільного користувача, обладнаним модемом і відповідним програмним забезпеченням для встановлення з'єднань віддаленого доступу. Як ініціатор може виступити також маршрутизатор екстрамережі (локальної мережі), наділений відповідними функціональними можливостями. Тунель звичайно завершується комутатором екстрамережі або шлюзом провайдеру послуг.

Сам по собі принцип роботи VPN не суперечить основним мережним технологіям і протоколам. Наприклад, при встановленні з'єднання віддаленого доступу клієнт посилає серверу потік пакетів стандартного протоколу PPP. У

випадку організації віртуальних виділених ліній між локальними мережами їхні маршрутизатори також обмінюються пакетами PPP. Проте, принципово новим моментом є пересилання пакетів через безпечний тунель, організований у межах загальнодоступної мережі.

Туннелювання дозволяє організувати передачу пакетів одного протоколу в логічному середовищі, що використовує інший протокол. У результаті з'являється можливість вирішити проблеми взаємодії декількох різнотипних мереж, починаючи з необхідності забезпечення цілісності й конфіденційності переданих даних і закінчуючи подоланням невідповідностей зовнішніх протоколів або схем адресації.

Існуюча мережна інфраструктура корпорації може бути підготовлена до використання VPN як за допомогою програмного, так і за допомогою апаратного забезпечення. Організацію віртуальної приватної мережі можна зрівняти із прокладкою кабелю через глобальну мережу. Як правило, безпосереднє з'єднання між віддаленим користувачем і окінцевим пристроєм тунелю встановлюється за протоколом PPP.

Найпоширеніший метод створення тунелів VPN – інкапсуляція мережних протоколів (IP, IPX, AppleTalk і т.д.) в PPP і наступна інкапсуляція утворених пакетів до протоколу туннелювання. Звичайно в якості останнього виступає IP або (набагато рідше) ATM і Frame Relay. Такий підхід називається туннелюванням другого рівня, оскільки «пасажиром» тут є протокол саме другого рівня.

Альтернативний підхід – інкапсуляція пакетів мережного протоколу безпосередньо до протоколу туннелювання (наприклад, VTP) називається туннелюванням третього рівня.

Незалежно від того, які протоколи використовуються або які цілі переслідуються при організації тунелю, основна методика залишається практично незмінною. Звичайно один протокол використовується для встановлення з'єднання з віддаленим вузлом, а інший – для інкапсуляції даних і службової

інформації з метою передачі через тунель. Як приклад використання тунелю для усунення невідповідностей між протоколами й схемами адресації можна привести технологію Simple Internet Transition (SIT), що повинна з'явитися разом із протоколом IPv6. Це ретельно розроблена групою інженерів (IETF) методологія туннелювання, покликана полегшити перехід від четвертої версії міжмережного протоколу (IPv4) до шостого (IPv6). Ці версії досить відрізняються, щоб говорити про безпосередню сумісність мереж. Інкапсуляція ж пакетів протоколу IPv6 у пакети IPv4 дозволяє досягти необхідного рівня функціональної сумісності.

Історія появи VPN тісно пов'язана з послугою CENTREX у телефонних мережах. Поняття Centrex з'явилося на рубежі 60-х років у США як загальна назва способу надання послуг ділового зв'язку абонентам декількох компаній на основі спільно використовуваного встаткування однієї станціє для установ PBX (Private Branch Exchange). З початком впровадження в США й Канаді станцій із програмним керуванням термін придбав інший зміст і став означати спосіб надання діловим абонентам додаткових послуг телефонного зв'язку, еквівалентних послугам PBX, на базі модифікованих станцій мережі загального користування. Основна перевага Centrex полягало у тому, що фірми й компанії при створенні виділених корпоративних мереж заощаджували значні засоби, необхідні на покупку, монтаж і експлуатацію власних станцій. Хоча для зв'язку між собою абоненти Centrex використовують ресурси й устаткування мережі загального користування, самі вони утворять так звані замкнуті групи користувачів CUG (Closed Users Group) з обмеженим доступом ззовні, для яких у станціях мережі реалізуються віртуальні PBX.

У прагненні перебороти властиві Centrex обмеження була висунута ідея віртуальної приватної мережі VPN – як об'єднання CUG, що становлять одну корпоративну мережу й перебувають на віддаленні друг від друга. Ресурси VPN (кожна зі своїм планом нумерації) можуть бути розподілені по декількох станціях місцевої мережі, оснащеним функціями Centrex і, що має в зоні свого

обслуговування одну або трохи CUG. При цьому в станцію можуть бути включені як PBX, безпосередньо приналежному власникові VPN, так і лінії звичайних індивідуальних абонентів.

Безпека

Природно, ніяка компанія не хотіла б відкрито передавати в Інтернет фінансову або іншу конфіденційну інформацію. Канали VPN захищені потужними алгоритмами шифрування, закладеними в стандарти протоколу безпеки IPsec. IPsec (Internet Protocol Security – стандарт, обраний міжнародним співтовариством, групою IETF – Internet Engineering Task Force) створює основи безпеки для Інтернет-протоколу (IP), незахищеність якого довгий час була проблемою. Протокол IPsec забезпечує захист на мережному рівні й вимагає підтримки стандарту IPsec тільки від пристроїв, що спілкуються між собою, по обох сторони з'єднання. Всі інші пристрої, розташовані між ними, просто забезпечують трафік IP-пакетів.

Спосіб взаємодії осіб, що використовують технологію IPsec, прийнято визначати терміном "захищена асоціація" – Security Association (SA). Захищена асоціація функціонує на основі угоди, укладеного сторонами, які користуються засобами IPsec для захисту переданої один одному інформації. Ця угода регулює кілька параметрів: IP-адреси відправника й одержувача, криптографічний алгоритм, порядок обміну ключами, розміри ключів, термін служби ключів, алгоритм автентифікації.

Інші стандарти включають протокол PPTP (Point to Point Tunneling Protocol), що розвивається Microsoft, L2F (Layer 2 Forwarding), що розвивається Cisco, – обоє для віддаленого доступу. Microsoft і Cisco працюють разом з IETF, щоб з'єднати ці протоколи в єдиний стандарт L2P2 (Layer 2 Tunneling Protocol) з метою використання IPsec для тунельної автентифікації, захисту приватної власності й перевірки цілісності.

Проблема полягає в тому, щоб забезпечити прийнятну швидкодію мережі при обміні шифрованою інформацією. Алгоритми кодування вимагають значних

обчислювальних ресурсів процесора, іноді в 100 разів більших, ніж при звичайній IP-маршрутизації. Щоб домогтися необхідної продуктивності, треба подбати про адекватне підвищення швидкодії, як серверів, так і клієнтських ПК. Крім того, є спеціальні шлюзи з особливими схемами, які помітно прискорюють шифрування.

IT-менеджер може вибирати конфігурацію віртуальної приватної мережі залежно від конкретних потреб. Наприклад співробітникові, що працює вдома може бути наданий обмежений доступ до мережі, а менеджерові віддаленого офісу або керівникові компанії – широкі права доступу. Один проект може обмежуватися лише мінімальним (56-розрядним) шифруванням при роботі через віртуальну мережу, а фінансова й планова інформація компанії вимагає могутніших засобів шифрування – 168-розрядних.

Захист від зовнішніх і внутрішніх атак

На жаль, доводиться відзначити, що засоби побудови VPN не є повноцінними засобами виявлення й блокування атак. Вони можуть запобігти ряду несанкціонованих дій, але далеко не всі можливості, які можуть використовувати хакери для проникнення в корпоративну мережу. Вони не можуть виявити віруси й атаки типу "відмова в обслуговуванні" (це роблять антивірусні системи й засоби виявлення атак), вони не можуть фільтрувати дані по різних ознаках (це роблять міжмережні екрани) і т.д. На це можна заперечити, що ці небезпеки не страшні, тому що VPN не прийме незашифрований трафік і відкине його. Однак на практиці це не так. По-перше, у більшості випадків засіб побудови VPN використовується для захисту лише частини трафіку, наприклад, спрямованого у віддалену філію. Інший трафік (наприклад, до публічних Web-серверів) проходить через VPN-пристрій без обробки. А по-друге, статистика стверджує, що до 80% всіх інцидентів, пов'язаних з інформаційною безпекою, відбувається з вини авторизованих користувачів, що мають санкціонований доступ у корпоративну мережу. Із чого слідує вивід, що атака або вірус будуть зашифровані нарівні з безпечним трафіком.

Продуктивність

Продуктивність мережі – це досить важливий параметр, і на будь-які засоби, що сприяють його зниженню, у будь-якій організації дивляться з підозрою. Не є виключенням і засоби побудови VPN, які створюють додаткові затримки, пов'язані з обробкою трафіку, що проходить через VPN-пристрій. Всі затримки, що виникають при криптографічній обробці трафіку, можна розділити на три типи:

– Затримки при встановленні захищеного з'єднання між VPN-пристроями.

– Затримки, пов'язані із зашифровуванням і розшифровуванням захищаємих даних, а також з перетвореннями, необхідними для контролю їхньої цілісності.

– Затримки, пов'язані з додаванням нового заголовка до переданих пакетів.

Реалізація першого, другого й четвертого варіантів побудови VPN передбачає встановлення захищених з'єднань не між абонентами мережі, а тільки між VPN-пристроями. З урахуванням криптографічної стійкості використовуваних алгоритмів зміна ключа можлива через досить тривалий інтервал часу. Тому при використанні засобів побудови VPN затримки першого типу практично не впливають на швидкість обміну даними. Зрозуміло, це положення стосується стійких алгоритмів шифрування, що використовують ключі не менш 128 біт (Triple DES, ДСТУ 28147:2009 і т.ін.). Пристрої, що використовують колишній стандарт DES, здатні вносити певні затримки в роботу мережі.

Затримки другого типу починають позначатися тільки при передачі даних по високошвидкісних каналах (від 10 Мбіт/с). У всіх інших випадках швидкодія програмної або апаратної реалізації обраних алгоритмів шифрування й контролю цілісності звичайно досить велика й у ланцюжку операцій «зашифровування пакета – передача пакета в мережу» і «прийом пакетів з мережі – розшифровування пакета» час зашифровування (розшифровування) значно менше часу, необхідного для передачі даного пакета в мережу.

Основна проблема тут пов'язана з додаванням додаткового заголовка до кожного пакета, що пропускається через VPN-пристрій. Як приклад розглянемо систему диспетчерського керування, що у реальному масштабі часу здійснює обмін даними між віддаленими станціями й центральним пунктом. Розмір переданих даних не великий – не більше 25 байтів. Дані порівнянного розміру передаються в банківській сфері (платіжні доручення) і в IP-телефонії. Інтенсивність переданих даних – 50-100 змінних у секунду. Взаємодія між вузлами здійснюється по каналах із пропускною здатністю в 64 Кбіт/с.

Пакет зі значенням однієї змінної процесу має довжину 25 байтів (ім'я змінної – 16 байтів, значення змінної – 8 байт, службовий заголовок – 1 байт). IP-протокол додає до довжини пакета ще 24 байта (заголовок IP-пакета). При використанні як середовище передачі каналів Frame Relay LMI додається ще 10 байтів FR-заголовка. Усього – 59 байтів (472 біта). Таким чином, для передачі 750 значень змінних процесу за 10 секунд (75 пакетів у секунду) необхідна смуга пропущення 75×472 = 34,5 Кбіт/с, що добре вписується в наявні обмеження пропускної здатності в 64 Кбіт/с. Тепер подивимося, як поводиться мережа при включенні в неї засобу побудови VPN. Перший приклад – засобу на основі протоколу SKIP. До 59 байтів даних додається 112 байт додаткового заголовка (для ДСТ28148-89), що складе 171 байт (1368 біт). 75x1368 = 102,6 Кбіт/с, що на 60% перевищує максимальну пропускну здатність наявного каналу зв'язку.

Для протоколу IPsec і вищевказаних параметрів пропускна здатність буде перевищена на 6% (67,8 Кбіт/с). Це за умови, що додатковий заголовок для алгоритму ДСТУ 28147:2009 складе 54 байта. Для протоколу, використовуваного в українському програмно-апаратному комплексі «Континент-К», додатковий заголовок, що додається до кожного пакета, становить усього 36 байтів (або 26 – залежно від режиму роботи), що не викликає ніякого зниження пропускної здатності (57 і 51 Кбіт/с відповідно). Справедливості заради необхідно відзначити, що всі ці викладення вірні лише за умови, що, крім зазначених змінних, у мережі більше нічого не передається.

Протоколи віртуальних приватних мереж

У цей час найпоширенішим протоколом VPN є протокол двоточечного тунельного зв'язку (Point-to-Point Tunnelling Protocol – PPTP). Розроблений він компаніями 3Com і Microsoft з метою надання безпечного віддаленого доступу до корпоративних мереж через Інтернет. PPTP використовує існуючі відкриті стандарти TCP/IP і багато в чому покладається на застарілий протокол двоточечного зв'язку PPP. На практиці PPP так і залишається комунікаційним протоколом сеансу з'єднання PPTP.

PPTP створює тунель через мережу до NT-сервера одержувача й передає по ньому PPP-пакети віддаленого користувача. Сервер і робоча станція використовують віртуальну приватну мережу й не звертають уваги на те, наскільки безпечною або доступною є глобальна мережа між ними. Завершення сеансу з'єднання з ініціативи сервера (на відміну від спеціалізованих серверів віддаленого доступу) дозволяє адміністраторам локальної мережі не пропускати віддалених користувачів за межі системи безпеки Windows NT Server. У результаті користувач використовує віртуальну приватну мережу, не наносячи при цьому збитку функціональним можливостям загальнодоступної мережі. Всі служби домену NT, включаючи DHCP, WINS і доступ до Network Neighborhood, без усяких застережень надаються віддаленому користувачеві.

Хоча компетенція протоколу PPTP поширюється тільки на пристрої, що працюють під керуванням Windows, він надає компаніям можливість взаємодіяти з існуючими мережними інфраструктурами й не завдавати шкоди власній системі безпеки. Таким чином, віддалений користувач може підключитися до Інтернету за допомогою місцевого провайдеру по аналоговій телефонній лінії або каналу ISDN і встановити з'єднання із сервером NT. При цьому компанії не доводиться витрачати великі суми на організацію й обслуговування пула модемів, що надає послуги віддаленого доступу.

У найближчому майбутньому очікується ріст кількості віртуальних приватних мереж, розгорнутих на базі нового протоколу туннелювання другого

рівня (Layer 2 Tunneling Protocol – L2TP). Цей протокол дозволяє об'єднати функціонуючі на другому рівні PPTP і L2F (Layer 2 Forwarding – протокол пересилання другого рівня) і розширити їхньої можливості. Однієм з них є багатоточечне туннелювання, що дозволяє користувачам ініціювати створення декількох мереж VPN, наприклад, для одночасного доступу до Інтернету й корпоративної мережі.

Протоколи L2TP і PPTP відрізняються від протоколів туннелювання третього рівня рядом особливостей:

1. Надання корпораціям можливості самостійно вибирати спосіб автентифікації користувачів і перевірки їхніх повноважень – на власній «території» або в провайдеру Інтернет-послуг. Обробляючи туннельовані пакети PPP, сервери корпоративної мережі одержують всю інформацію, необхідну для ідентифікації користувачів.

2. Підтримка комутації тунелів – завершення одного тунелю й ініціювання іншого до одному з безлічі потенційних термінаторів. Комутація тунелів дозволяє як би продовжити PPP-з'єднання до необхідної кінцевої точки.

3. Надання системним адміністраторам корпоративної мережі можливості реалізації стратегій призначення користувачам прав доступу безпосередньо на брандмауері й внутрішніх серверах. Оскільки термінатори тунелю одержують пакети PPP з відомостями про користувачів, вони в стані застосовувати сформульовані адміністраторами стратегії безпеки до трафіку окремих користувачів. (Туннелювання третього рівня не дозволяє розрізняти вступників від провайдеру пакети, тому фільтри стратегії безпеки доводиться застосовувати на кінцевих робочих станціях і мережних пристроях.) Крім того, у випадку використання тунельного комутатора з'являється можливість організувати «продовження» тунелю другого рівня для безпосередньої трансляції трафіку окремих користувачів до відповідних внутрішніх серверів. На такі сервери може бути покладене завдання додаткової фільтрації пакетів.

Переваги VPN

Переваги технології VPN настільки переконливі, що багато компаній починають будувати свою стратегію з урахуванням використання Інтернету як головного засобу передачі інформації, навіть тієї, котра є уразливою. Переваги VPN уже оцінені по достоїнству багатьма підприємствами.

При правильному виборі VPN:

– ми одержуємо захищені канали зв'язку за ціною доступу в Інтернет, що в кілька разів дешевше виділених ліній;

– при установці VPN не потрібно змінювати топологію мереж, переписувати додатки, навчати користувачів – все це значна економія;

– забезпечується масштабування, оскільки VPN не створює проблем росту й зберігає зроблені інвестиції;

– ви незалежні від криптографії й можете використовувати модулі криптографії будь-яких виробників відповідно до національних стандартів тої або іншої країни;

– відкриті інтерфейси дозволяють інтегрувати вашу мережу з іншими програмними продуктами й бізнес-додатками.

Недоліки VPN

До них можна віднести порівняно низьку надійність. У порівнянні з виділеними лініями й мережами на основі Frame relay віртуальні частки мережі менш надійні, однак в 5-10, а іноді й в 20 разів дешевше. На думку західних аналітиків, це не зупинить продаж VPN, оскільки лише п'яти відсоткам користувачів, що торгують, наприклад, на ринку цінних паперів, потрібні такі високі стандарти. Інші 95% не настільки серйозно відносяться до проблем зі зв'язком, а витрати більшої кількості часу на одержання інформації не приводять до колосальних збитків.

У силу того, що послуга VPN надається й підтримується зовнішнім оператором, можуть виникати проблеми зі швидкістю внесення змін у бази доступу, у налаштування firewall, а також з відновленням устаткування, що

вийшло з ладу. У цей час проблема вирішується вказівкою в договорах максимального часу на усунення неполадок і внесення змін. Звичайно цей час становить кілька годин, але зустрічаються провайдери, що гарантують усунення неполадок протягом доби.

Ще один істотний недолік – у споживачів немає зручних засобів керування VPN. Хоча останнім часом розробляється устаткування, що дозволяє автоматизувати керування VPN. Серед лідерів цього процесу – компанія Indus River Networks Inc., дочірня компанія MCI WorldCom і Novell. Як говорять аналітики Forester Research, VPN повинні контролюватися користувачами, управлятися компаніями-операторами, а завдання розроблювачів програмного забезпечення – вирішити цю проблему.

Перспективи VPN

У міру свого розвитку VPN перетворяться в системи взаємозалежних мереж, які будуть з'єднувати мобільних користувачів, торговельних партнерів і постачальників із критично важливими корпоративними додатками, що працюють у протоколі IP. VPN стануть фундаментом для нових комерційних операцій і послуг, які будуть стимулювати ринок і допомагати модернізувати виробництво.

Імовірно, першим з основних компонентів завтрашніх VPN буде сервер каталогів, що містить профілі кінцевих користувачів і дані про конфігурацію мережі. Це буде окрема комп'ютерна система в корпоративній і частково в загальнодоступній мережі, якою буде управляти провайдер VPN. При наявності мережних каталогів, а також забезпечення безпеки інформації і якості обслуговування кінцеві користувачі зможуть практично миттєво встановлювати з'єднання по VPN.

Цілком можливо, що буде використовуватися протокол IPv6, роботи над яким активно тривають. Даний протокол має всі можливості взаємодії з VPN, яких тільки можуть побажати собі мережні розроблювачі, включаючи керування смугою пропущення. Також можна буде визначати приналежність IPv6-пакетів

до певного потоку, наприклад, вищий пріоритет будуть одержувати пакети мультимедійних даних для передачі в реальному часі.

Головні гравці мережного ринку, такі, як Cisco Systems, Cabletron Systems, 3Com, Bay Networks, HCL Comnet, уже активно готуються до прийдешнього буму VPN. Нинішні вендори програмного забезпечення й устаткування пропонують набори пристроїв для того, щоб створити й експлуатувати VPN.

Вигоду від розгортання VPN наступного покоління одержать не тільки мережні розроблювачі – не менш зацікавлені в них і оператори. Фірми AT&T Level 3 Communications, MCI Worldcom і Sprint створюють високошвидкісні IP-канали в ATM-мережах для передачі відео, голосу й даних. VPN у цей час чи роблять не вирішальний вплив на розробку стратегії глобальних операторів, таких, як Unisource (AT&T, Telia, PTT Suisse і PTT Netherlands), Concert (BT/MCI) і Global One (Deutsche Telekom, France Telekom). Чим більше компаній будуть пропонувати VPN-послуги, тим помітніше буде рости їхня якість і падати ціни, що, у свою чергу, вплине на число клієнтів.

Кожна революція в бізнесі починалася з винаходу, що різко збільшував приватну ініціативу. Наприклад, поділ перевізників і компаній, що експлуатують державну залізницю, привело до різкого росту комерційних перевезень. Те ж саме відбувається при створенні VPN поверх національних і міжнародних телекомунікаційних інфраструктур. Найближчий час покаже, до яких змін це приведе.

3.2 Розробка структурної схеми

Технологія VPN (Virtual Private Network – віртуальна приватна мережа) – не єдиний спосіб захисту мереж і переданих по них даних. Але я вважаю, що вона досить ефективна, і її повсюдне впровадження – це не тільки данина моді, досить прихильної до VPN в останні пару років.

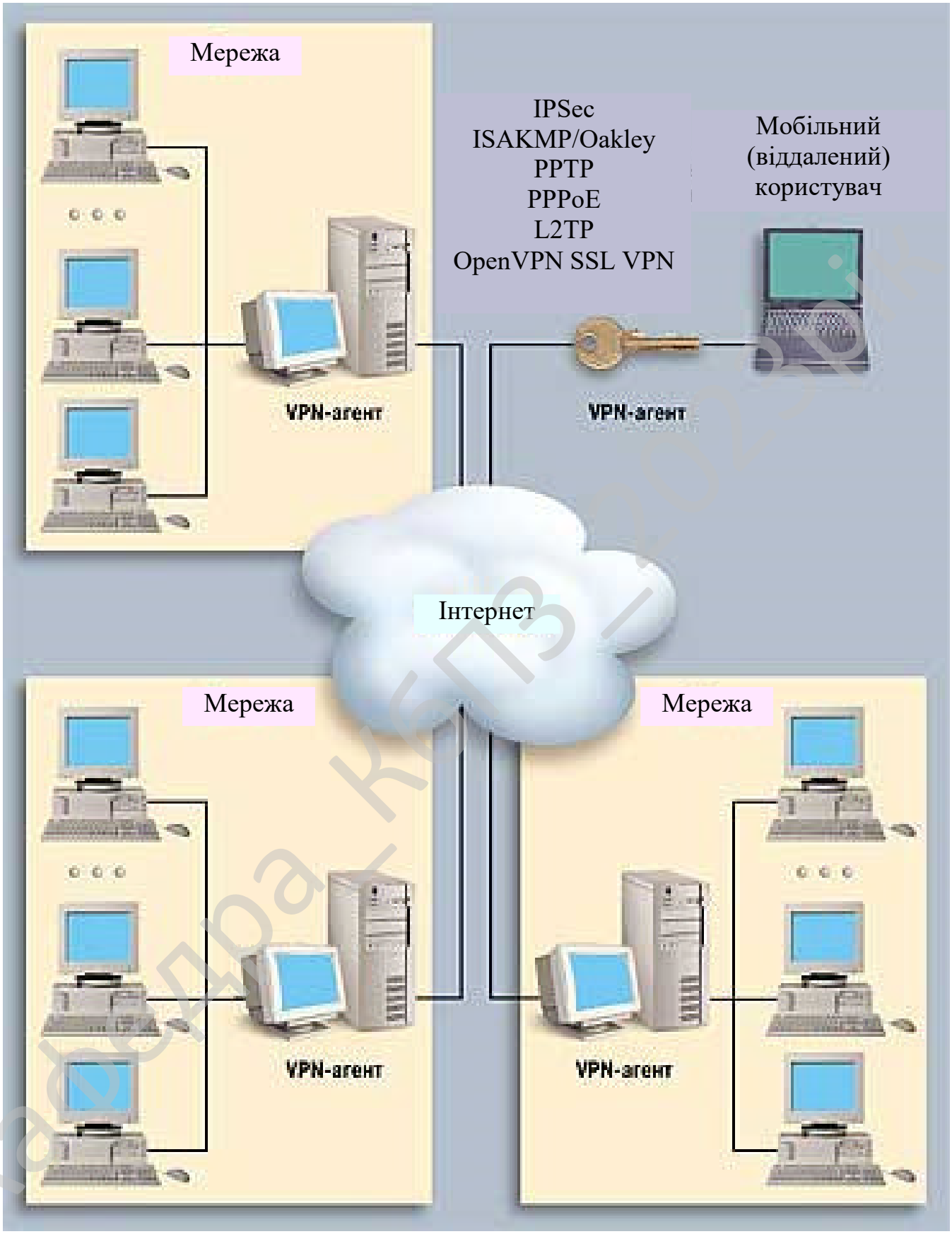

Рисунок 3.1 – Структурна схема системи з використанням VPN

У системі використовуються наступні протоколи VPN:

– IPSec (IP security) – часто використовується поверх IPv4.

– PPTP (point-to-point tunneling protocol) – розроблявся спільними зусиллями декількох компаній, включаючи Microsoft.

– PPPoE (PPP (Point-to-Point Protocol) over Ethernet).

– L2TP (Layer 2 Tunnelling Protocol) – використовується в продуктах компаній Microsoft і Cisco.

– L2TPv3 (Layer 2 Tunnelling Protocol version 3).

– OpenVPN SSL VPN з відкритим вихідним кодом, підтримує режими PPP, bridge, point-to-point, multi-client server.

Суть VPN полягає в наступному:

– На всі комп'ютери, що мають вихід в Інтернет, установлюється засіб, що реалізує VPN (VPN-агент). Не повинно залишитися жодного незахищеного.

– VPN-агенти автоматично шифрують всю вихідну інформацію (і відповідно розшифровують всю вхідну). Вони також стежать за її цілісністю за допомогою ЕЦП або імітовставок (криптографічна контрольна сума, розрахована з використанням ключа шифрування).

Оскільки інформація, що циркулює в Інтернеті, являє собою безліч пакетів протоколу IP, VPN-агенти працюють саме з ними.

Перед відправленням IP-пакета VPN-агент діє в такий спосіб:

– З декількох підтримуваних їм алгоритмів шифрування й ЕЦП по IP-адресі одержувача вибирається потрібний для захисту даного пакета, а також ключі. Якщо ж у його налаштуваннях такого одержувача ні, то інформація не відправляється.

– Визначає й додає в пакет ЕЦП відправника або імітовставку.

– Шифрує пакет (цілком, включаючи заголовок).

– Проводить інкапсуляцію, тобто формує новий заголовок, де вказується адреса зовсім не одержувача, а його VPN-агента. Ця корисна додаткова функція дозволяє представити обмін між двома мережами як обмін всього-на-всього між

двома комп'ютерами, на яких установлені VPN-агенти. Усяка корисна для темних цілей зловмисника інформація, наприклад внутрішні IP-адреси, йому вже недоступна.

При одержанні IP-пакета виконуються зворотні дії:

– Заголовок містить відомості про VPN-агента відправника. Якщо такий не входить у список дозволених у налаштуваннях, то інформація просто відкидається. Те ж саме відбувається при прийманні пакета з навмисно або випадково ушкодженим заголовком.

– Відповідно до налаштувань вибираються алгоритми шифрування й ЕЦП, а також необхідні криптографічні ключі.

– Пакет розшифровується, потім перевіряється його цілісність. Якщо ЕЦП невірна, то він викидається.

– І, нарешті, пакет у його вихідному виді відправляється справжньому адресатові по внутрішній мережі.

Всі операції виконуються автоматично. Складним в технології VPN є тільки налаштування VPN-агентів, що, втім, цілком під силу досвідченому користувачеві.

VPN-агент може перебувати безпосередньо на ПК, який захищається, що корисно для мобільних користувачів, що підключаються до Інтернет. У цьому випадку він забезпечить обмін даними тільки того комп'ютера, на якому встановлений.

Можливе сполучення VPN-агента з маршрутизатором (у цьому випадку його називають криптографічним) IP-пакетів. До речі, що ведуть світові виробники останнім часом випускають маршрутизатори з вбудованою підтримкою VPN, наприклад Express VPN від Intel, що шифрує всі пакети, які проходять, за алгоритмом Triple DES.

Як видно з опису, VPN-агенти створюють канали між мережами, що захищаються, які звичайно називають “тунелями”. І дійсно, вони “прориті” через

Інтернет від однієї мережі до іншої, циркулююча усередині інформація захована від чужих очей.

Крім того, всі пакети “фільтруються” відповідно до налаштувань. Таким чином, всі дії VPN-агентів можна звести до двох механізмів: створення тунелів і фільтрація минаючих пакетів.

Сукупність правил створення тунелів, що називається “політикою безпеки”, записується в налаштуваннях VPN-агентів. IP-пакети направляються в той або інший тунель або відкидаються після того, як будуть перевірені:

– IP-адреса джерела (для вихідного пакета – адреса конкретного комп'ютера мережі, що захищається);

– IP-адреса призначення;

– протокол більш високого рівня, якому належить даний пакет (наприклад, TCP або UDP);

– номер порту, з якого або на який відправлена інформація (наприклад, 1080).

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 системи кібербезпеки створення VPN підключень покладені протоколи забезпечення безпеки інформації IPsec та SSL.

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.

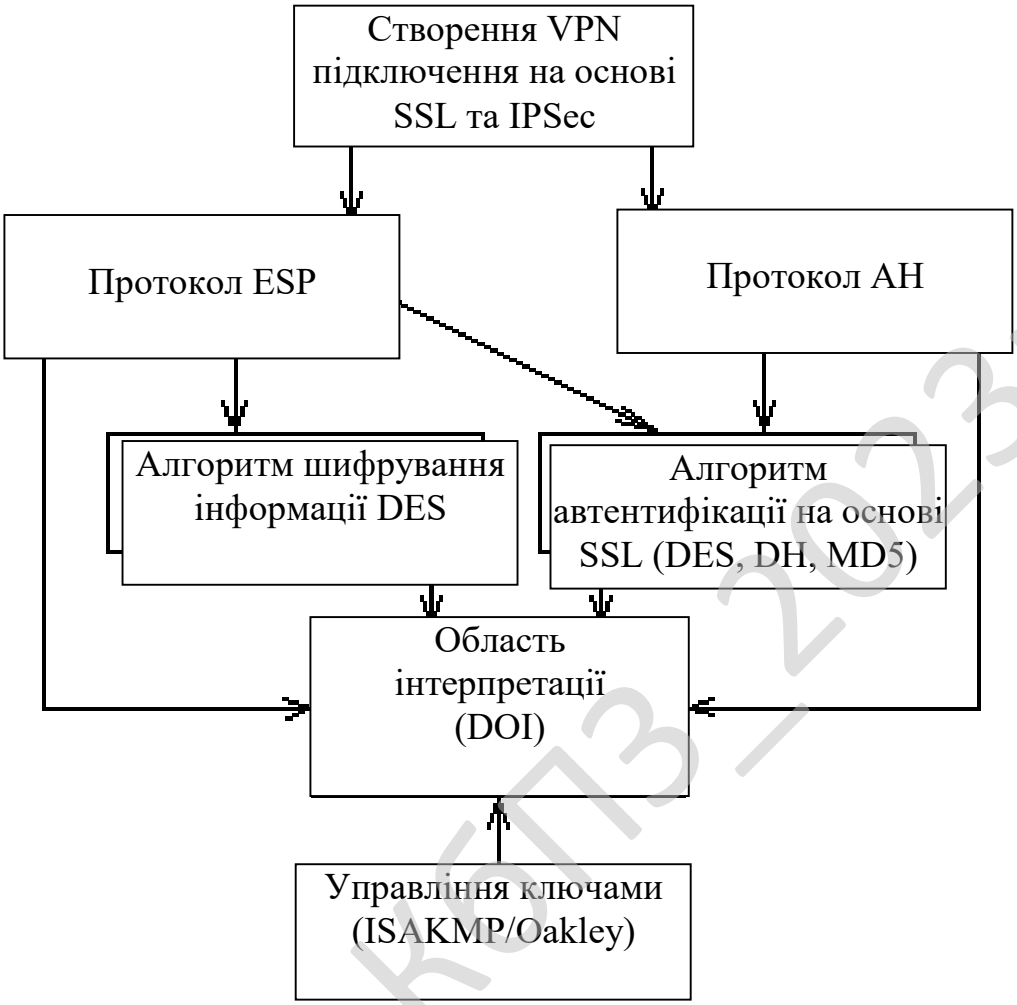

Рисунок 3.2 – Функціональна схема системи

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.

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 АН

Автентифікуючий заголовок (АН) є звичайним опціональним заголовком і, як правило, розташовується між основним заголовком пакета IP і полем даних. Наявність АН ніяк не впливає на процес передачі інформації транспортного й більш високого рівнів. Основним і єдиним призначенням АН 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 АН 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.

Наступний заголовок	Довжина навантаження	Зарезервовано
Індекс параметрів безпеки (SPI)		
Поле послідовного номеру		
Дані аутентифікації (перемінної довжини)		

Рисунок 3.3 – Формат заголовка АН

Послідовний номер пакета був введений в АН 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 линиям 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.

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.

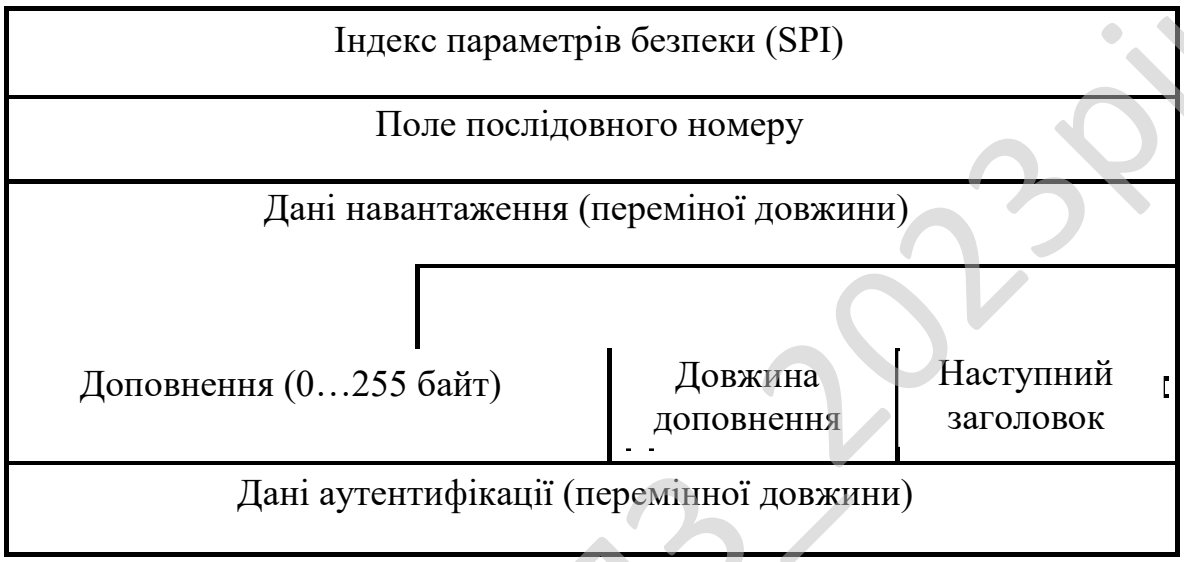

Рисунок 3.4 – Формат заголовка ESP

Розрізняють два режими застосування ESP – транспортний і тонельний.

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.

Тонельний режим – припускає шифрування всього пакета, включаючи заголовок мережного рівня. Тонельний режим застосовується якщо буде потреба приховання інформаційного обміну організації із зовнішнім миром. При цьому, адресні поля заголовка мережного рівня пакета, що використовує тонельний 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.

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

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання бакалаврського проектування, наведена на рисунку 3.5.

Спершу запускається процес завантаження вікна створення VPN-з’єднання. Цей процес взаємодіє з процесом виведення параметрів з’єднання.

Процес виведення параметрів з’єднання взаємодіє з наступними процесами:

– Процес введення назви з’єднання.

– Процес введення адреси VPN-сервера.

– Процес введення логіну та паролю.

– Процес створення з’єднання.

Останній процес взаємодіє з процесом закриття вікна створення VPN-з’єднання.

Після цього запускається процес створення іконки з’єднання у папці «Мережні підключення».

Рисунок 3.5 – Діаграма взаємодії процесів

Процес створення іконки з'єднання у папці «Мережні підключення» взаємодіє з процесом налаштування з'єднання.

Цей процес, у свою чергу, взаємодіє з процесом підключення до VPN-сервера.

Після цього запускається процес обміну даними, який взаємодіє з наступними процесами:

– Процес захисту даних алгоритмами SSL та IPSec.

– Процес відключення від VPN-сервера, який є завершальним у системі.

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 РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рисунку 4.1 наведено блок-схему основної програми. Її робота складається з виконання наступних кроків.

Спершу відбувається виведення вікна створення VPN-з’єднання. Після цього вводиться назва VPN-з’єднання.

Наступним кроком є введення адреси VPN-сервера.

Після цього користувач вводить логін та пароль.

Відбувається створення з’єднання й закривається вікно створення VPN-з’єднання.

Наступним кроком є створення іконки з’єднання у папці «Мережні підключення».

Встановлюються параметри VPN-з’єднання та шифрування.

Якщо потрібно здійснити VPN-з’єднання, тоді відбувається виконання наступних ітерацій:

– Користувач проходить процедуру авторизації.

– Відбувається підключення до VPN-сервера.

– Відбувається захист даних алгоритмами SSL та IPSec.

– Відбувається сеанс обміну даними.

Після цього користувач обирає відключатися йому від VPN-сервера або ні.

Якщо він приймає рішення відключитися, то програма завершує свою роботу.

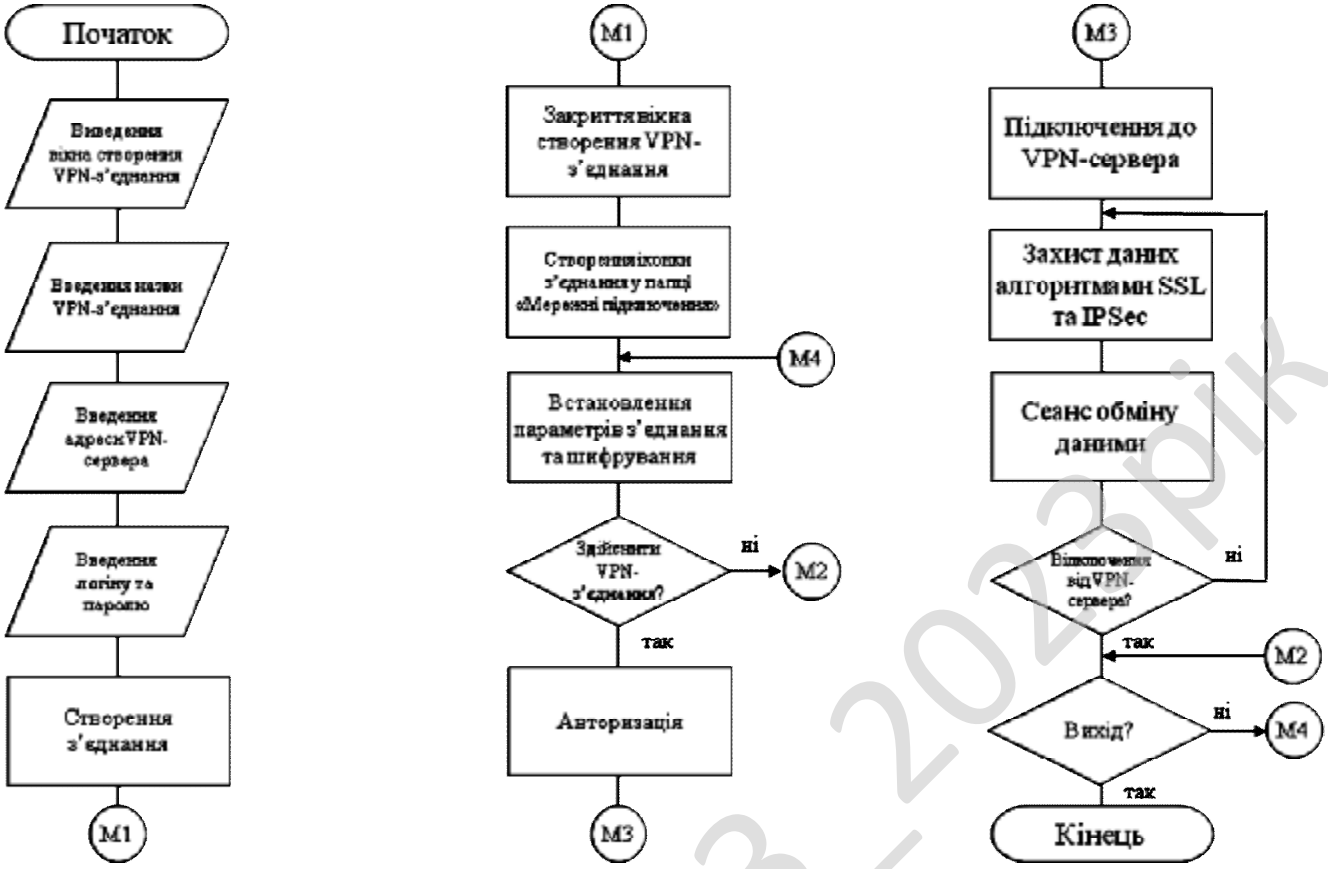

Рисунок 4.1 – Блок-схема основної програми

На рисунку 4.2 зображена блок схема підпрограми, що реалізує метод захисту інформації IPsec. Підпрограма працює наступним чином.

Спершу відбувається налаштування політики безпеки для різних видів трафіку.

Після цього запускається перша фаза IKE.

Наступним етапом є автентифікація сторін IPsec.

Після автентифікації відбувається встановлення захищеного тунелю, для обміну даними.

Після цього обирається режим першої фази IKE.

Якщо обрано основний режим, тоді виконуються наступні дії:

– Встановлюється основний режим.

– Відбуваються три двосторонні обміни між ініціатором й респондентом для створення захищеного каналу.

– Узгоджуються параметри асоціацій захисту IKE.

У противному випадку, відбуваються наступні дії:

– Встановлюється енергійний режим.

– Обидві сторони обмінюються інформацією до того, як створено захищений канал.

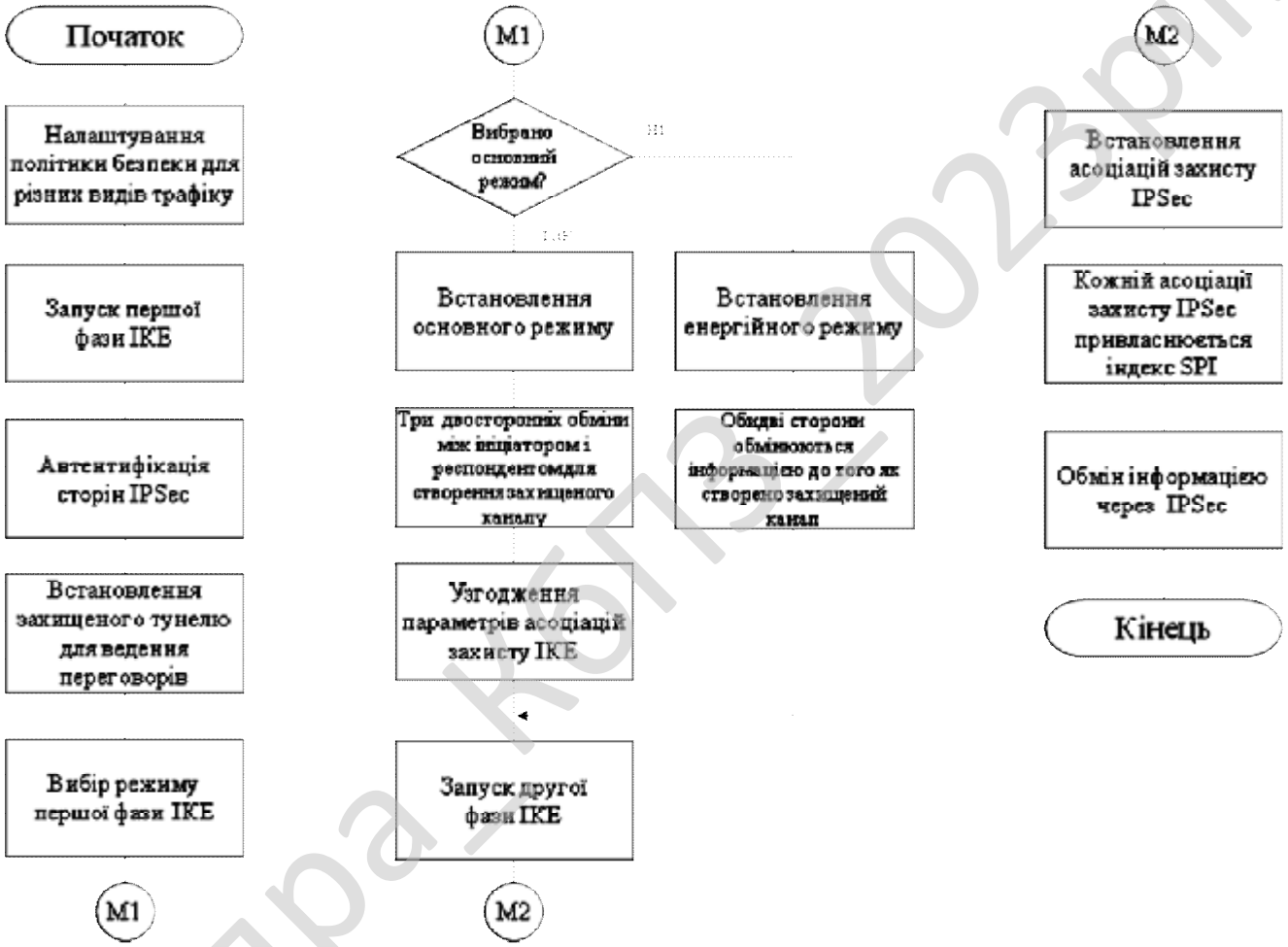

Рисунок 4.2 – Блок-схема підпрограми, що реалізує метод захисту інформації IPsec

Після цього запускається друга фаза IKE.

Наступним кроком є встановлення асоціацій захисту IPsec.

Кожній асоціації захисту IPsec привласнюється індекс SPI.

Відбувається обмін інформацією через IPsec. Після цього підпрограма

завершує свою роботу.

IPsec опирається на ряд технологічних рішень і методів шифрування, але дію IPsec у загальному можна представити у вигляді наступних головних кроків:

Крок 1. Початок процесу IPsec. Трафік, якому потрібне шифрування відповідно до політики захисту IPsec, погодженої сторонами IPsec, починає IKE-процес.

Крок 2. Перша фаза IKE. IKE-процес виконує автентифікацію сторін IPsec і веде переговори про параметри асоціацій захисту IKE, у результаті чого створюється захищений канал для ведення переговорів про параметри асоціацій захисту IPsec у ході другої фази IKE.

Крок 3. Друга фаза IKE. IKE-процес веде переговори про параметри асоціації захисту IPsec і встановлює відповідні асоціації захисту IPsec для пристроїв сторін, що обмінюються інформацією.

Крок 4. Передача даних. Відбувається обмін даними між сторонами IPsec, що ґрунтується на параметрах IPsec і ключах, збережених у базі даних асоціацій захисту.

Крок 5. Завершення роботи тунелю IPsec. Асоціації захисту IPsec завершують свою роботу або в результаті їхнього видалення, або через перевищення граничного часу їхнього існування.

Розглянемо зазначені кроки докладніше.

Крок 1. Початок процесу IPsec

Тип трафіку, що повинен захищатися засобами IPsec, визначається в рамках політики захисту для VPN. Потім ця політика реалізується у вигляді команд конфігурації інтерфейсів пристроїв кожної сторони IPsec. Наприклад, у маршрутизаторах Cisco і брандмауерах PIX Firewall для визначення трафіку, що підлягає шифруванню, використовують списки доступу. Списки доступу реалізують політику шифрування, наприклад, за допомогою операторів permit, що вказують, що відповідний трафік повинен шифруватися, та операторів deny, що забороняють шифрування відповідного трафіку. У випадку клієнта Cisco VPN

використовуються вікна меню, де вказуються з'єднання, яким повинна забезпечуватися захист IPsec. Коли підлягаючий шифруванню трафік генерується клієнтом IPsec або проходить через нього, клієнт ініціює наступний крок процесу, починаючи першу фазу IKE.

Крок 2. Перша фаза IKE

Головною метою обміну даними, що відбуваються в першій фазі IKE, є автентифікація сторін IPsec і створення захищеного каналу між сторонами, що дозволяє почати обмін IKE. У ході першої фази IKE виконуються наступні дії.

– Ведуться переговори про узгодження політики асоціацій захисту IKE між сторонами, щоб забезпечити захист обміну IKE. Асоціація захисту IKE одержує погоджені параметри IKE і є двосторонньою.

– Виконується обмін Діффі-Хеллмана, у результаті якого вибирається загальний секретний ключ для використання в алгоритмах шифрування IPsec.

– Виконується автентифікація й забезпечується захист сторін IPsec.

– Встановлюється захищений тунель для ведення переговорів про параметри другої фази IKE.

Для першої фази IKE припустимі два режими: основний і енергійний.

Основний режим першої фази IKE (Main Mode)

У цьому режимі виконуються три двосторонніх обміни між ініціатором і респондентом:

1. У ході першого обміну алгоритми, використовувані для захисту зв'язку IKE, погоджуються доти, поки не буде досягнута відповідність для всіх асоціацій захисту IKE сполучених сторін.

2. У процесі другого обміну виконується алгоритм Діффі-Хеллмана, щоб погодити загальний секретний матеріал, на основі якого створюються спільні секретні ключі, передати так звані "оказії" (випадкові значення, що посилаються іншій стороні), підписати їх і повернути назад, щоб довести "свою особистість".

3. У ході третього обміну виконується автентифікація сторони-партнера. Ідентифікаційним значенням у цьому випадку виступає IP-адреса сторони IPsec у

шифрованому вигляді.

Основним результатом цього режиму є узгодження параметрів асоціацій захисту IKE між сторонами з метою створення захищеного каналу для наступних обмінів IKE.

Асоціація захисту IKE визначає параметри обміну IKE: використовуваний метод автентифікації, алгоритми шифрування й хешування, використовувана група Діффі-Хеллмана (одна із двох доступних), максимальний час існування асоціації захисту IKE у секундах або кілобайтах і спільно використовувані секретні значення ключів для алгоритмів шифрування. Асоціації захисту IKE у пристроях кожної зі сторін є двосторонніми.

Енергійний режим першої фази IKE (Aggressive mode)

У даному режимі менше й число обмінів, і число пакетів, що пересилаються при цьому, у результаті чого потрібно менше часу для установки сеансу IPsec. У цьому випадку виконуються наступні дії:

1. Під час першого обміну майже все необхідне включається в пропоновані значення для асоціацій захисту IKE, відкритий ключ Діффі-Хеллмана, оказія, що підписується другою стороною, і пакет ідентифікації, який можна використовувати для того, щоб автентифікувати другу сторону за допомогою третьої сторони.

2. Одержувач відправляє назад усе, що потрібно, щоб завершити обмін. Ініціаторові залишається тільки підтвердити обмін.

Недоліком використання енергійного режиму є те, що обидві сторони обмінюються інформацією до того, як створений захищений канал. Таким чином, можна підключитися до лінії й з'ясувати, хто формує нову асоціацію захисту. З іншого боку, обмін відбувається швидше, ніж в основному режимі. Енергійний режим для обміну IKE звичайно не ініціюється продуктами Cisco, але маршрутизатори Cisco і брандмауери PIX Firewall можуть відповідним чином відповісти стороні IPsec, що ініціювала обмін в енергійному режимі.

Крок 3. Друга фаза IKE (Quick Mode)

Завданням другої фази IKE є узгодження параметрів асоціації захисту IPsec з метою створення тунелю IPsec. У цій фазі виконуються наступні дії.

– Ведуться переговори про параметри асоціації захисту IPsec, що захищаються існуючою асоціацією захисту IKE.

– Встановлюються асоціації захисту IPsec.

– Періодично відновлюються переговори про асоціації захисту IPsec, щоб гарантувати захист.

– У необов'язковому порядку може виконуватися додатковий обмін Діффі-Хеллмана.

Друга фаза IKE виконується тільки у швидкому режимі, після того як у результаті першої фази IKE створюється захищений тунель. Потім ведуться переговори про погоджену політику IPsec, витягає загальний секретний матеріал для роботи алгоритмами захисту IPsec і створюються асоціації захисту IPsec. У швидкому режимі виконується обмін оказіями, які забезпечують захист від відтворення повідомлень. Оказії використовуються для того, щоб гарантувати створення нових секретних ключів і не допустити проведення атак відтворення, у результаті яких супротивник міг би створити "фальшиві" асоціації захисту.

Швидкий режим використовується також для того, щоб домовитися про нові асоціації захисту IPsec, коли виявляється перевищеною межа часу існування старої асоціації захисту IPsec. Базовий варіант швидкого режиму використовується для того, щоб обновити секретний матеріал, призначений для створення спільного секретного ключа на основі значень, отриманих при обміні Діффі-Хеллмана в ході першої фази.

В IPsec є опція PFS (Perfect Forward Secrecy – досконала пряма таємність), що підсилює захист ключів. Якщо політикою IPsec запропоноване використання опції PFS, то для кожного обміну у швидкому режимі потрібен новий обмін Діффі-Хеллмана, що забезпечує нові дані для ключів, у результаті чого дані для ключів будуть мати більшу ентропію ("нерегулярність") і тому більшу стійкість

відносно криптографічних атак. Кожний обмін Діффі-Хеллмана вимагає великої кількості піднесень у ступінь, що збільшує завантаження процесора й знижує загальну продуктивність системи. Асоціації захисту, узгоджувані у швидкому режимі, ідентифікуються IP-адресами IKE-сторін.

Узгодження перетворень IPsec

У ході другої фази в рамках протоколу IKE ведуться переговори про перетворення IPsec (алгоритмах захисту IPsec). IPsec складається із двох головних протоколів захисту й безлічі протоколів підтримки.

Перетворення IPsec і пов'язані з ними алгоритми шифрування є наступними.

– Протокол AH (Authentication Header – заголовок автентифікації). Протокол захисту, що забезпечує автентифікацію й (як опція) сервіс виявлення відтворення. Протокол AH діє як цифровий підпис і гарантує, що дані в пакеті IP не будуть несанкціонованно змінені. Протокол AH не забезпечує сервіс шифрування та дешифрування даних. Даний протокол може використовуватися або самостійно, або разом із протоколом ESP.

– Протокол ESP (Encapsulating Security Payload – включаючий захист корисний вантаж). Протокол захисту, що забезпечує конфіденційність і захист даних, а також (як опція) сервіс автентифікації й виявлення відтворення. Підтримуючі IPsec продукти Cisco використовують ESP для шифрування корисного вантажу IP-пакетів. Протокол ESP може використовуватися самостійно або разом з AH.

– Стандарт DES (Data Encription Standard – стандарт шифрування даних). Алгоритм шифрування й дешифрування даних пакетів. Алгоритм DES використовується як у рамках IPsec, так і IKE. Для алгоритму DES використовується 56-бітовий ключ, що означає не тільки більш високе споживання обчислювальних ресурсів, але й більш надійне шифрування. Алгоритм DES є симетричним алгоритмом шифрування, для якого потрібні ідентичні секретні ключі шифрування в пристроях кожної зі сполучених сторін

IPsec. Для створення симетричних ключів застосовується алгоритм Діффі-Хеллмана. IKE і IPsec використовують алгоритм DES для шифрування повідомлень.

– "Потрійний" DES (3DES). Варіант DES, заснований на використанні трьох ітерацій стандартного DES із трьома різними ключами, що практично потроює стійкість DES. Алгоритм 3DES використовується в рамках IPsec для шифрування й дешифрування потоку даних. Даний алгоритм використовує 168-бітовий ключ, що гарантує високу надійність шифрування. IKE і IPsec використовують алгоритм 3DES для шифрування повідомлень.

– При перетворенні IPsec використовується також два стандартних алгоритми хешування, що забезпечують автентифікацію даних.

– Алгоритм MD5 (Message Digest 5). Алгоритм хешування, застосовуваний для автентифікації пакетів даних. У продуктах Cisco використовується обчислюється з допомогою MD5 код HMAC (Hashed Message Authentication Code – хеш-код автентичності повідомлення) – варіант коду автентичності повідомлення, якому забезпечується додатковий захист за допомогою хешування. Хешування являє собою процес однобічного (тобто необоротного) шифрування, у результаті якого для поступаючого на вхід повідомлення довільної довжини виходить код фіксованої довжини. IKE, AH і ESP використовують MD5 для автентифікації даних.

– Алгоритм SHA-1 (Secure Hash Algorithm-1 – захищений алгоритм хешування 1). Алгоритм хешування, використовуваний для автентифікації пакетів даних. В продуктах Cisco застосовується варіант коду HMAC, що обчислюється за допомогою SHA-1. IKE, AH і ESP використовують SHA-1 для автентифікації даних.

– В рамках протоколу IKE симетричні ключі створюються за допомогою алгоритму Діффі-Хеллмана, що використовує DES, 3DES, MD5 і SHA. Протокол Діффі-Хеллмана є криптографічним протоколом, заснованим на застосуванні відкритих ключів. Він дозволяє двом сторонам погодити спільній секретний

ключ, не маючи досить надійного каналу зв'язку. Спільні секретні ключі потрібні для алгоритмів DES і HMAC. Алгоритм Діффі-Хеллмана використовується в рамках IKE для створення сеансових ключів. У продуктах Cisco підтримуються 768– і 1024-бітові групи Діффі-Хеллмана. 1024-бітна група забезпечує більш надійний захист.

Кожній асоціації захисту IPsec привласнюється індекс SPI (Security Parameter Index – індекс параметрів захисту) – число, використовуване для ідентифікації асоціації захисту IPsec. Асоціація захисту IPsec визначає використовуване перетворення IPsec (ESP і/або AH і відповідні алгоритми шифрування й хешування), межу часу існування асоціації захисту IPsec у секундах або кілобайтах, вказує необхідність застосування опції PFS, IP-Адреси сторін, а також спільні значення секретних ключів для алгоритмів шифрування й інші параметри. Всі асоціації захисту IPsec є однобічними.

Один цикл узгодження асоціації захисту IPsec завершується створенням двох асоціацій захисту: однієї вхідної й однієї вихідної.

Протоколи AH і ESP IPsec можуть діяти або в тунельному, або в транспортному режимах. Тунельний режим використовується для зв'язку між шлюзами IPsec, і в цьому випадку засобам IPsec доводиться створювати зовсім новий заголовок IPsec. Транспортний режим звичайно застосовується між клієнтом і сервером VPN, і при цьому використовується існуючий заголовок IP.

Крок 4. Передача даних

Після завершення другої фази IKE і створення асоціацій захисту IPsec у швидкому режимі, починається обмін інформацією через тунель IPsec, що зв'язує сторони IPsec. Пакети шифруються й дешифруються за допомогою алгоритмів шифрування й ключів, зазначених асоціацією захисту IPsec. Асоціація захисту IPsec задає також ліміт часу свого існування в кілобайтах переданих даних або в секундах. Асоціація захисту має спеціальний лічильник, значення якого зменшується на одиницю за кожну секунду або після передачі кожного кілобайта даних.

Крок 5. Завершення роботи тунелю IPsec

Асоціації захисту IPsec завершують свою роботу або через їхнє видалення, або тому, що виявляється перевищена межа часу їхнього існування. Коли асоціації захисту закінчують роботу то ключі, що відповідають їм теж стають недійсними. Якщо для потоку даних потрібні нові асоціації захисту IPsec, у рамках протоколу IKE знову виконується обмін другої фази, а якщо необхідно, то й першої. У результаті успішного їхнього завершення створюються нові асоціації захисту й нові ключі. Нові асоціації захисту можуть створюватися й до витікання часу існування попередніх, щоб потік даних міг рухатися безупинно. Звичайно переговори другої фази виконуються частіше, ніж переговори першої фази.

Мережі на основі IPsec можуть бути побудовані за допомогою самих різних пристроїв Cisco:

– маршрутизаторів Cisco;

– брандмауерів CiscoSecure PIX Firewall;

– програмного забезпечення клієнта CiscoSecure VPN;

– концентраторів Cisco VPN серій 3000 і 5000.

Маршрутизатори Cisco мають вбудовану підтримку VPN з відповідними багатими можливостями програмного забезпечення Cisco IOS, що зменшує складність мережних рішень і знижує загальну вартість VPN при можливості побудови багаторівневого захисту надаваних сервісів.

Брандмауер PIX Firewall є високопродуктивним мережним пристроєм, що може обслуговувати кінцеві точки тунелів, забезпечуючи їм високу пропускну здатність і прекрасні функціональні можливості брандмауера. Програмне забезпечення клієнта CiscoSecure VPN підтримує самі строгі вимоги VPN вилученого доступу для операцій електронної комерції, а також додатків мобільного доступу, пропонуючи закінчену реалізацію стандартів IPsec і забезпечуючи надійну взаємодію маршрутизаторів Cisco і брандмауерів PIX Firewall.

Розповсюджені помилки VPN підключення

Якщо при запуску VPN-з'єднання виникла помилка, то необхідно запам'ятати її тризначний код. Як правило помилки мають код 6XX (помилки, пов'язані з некоректним введенням пароля, ім'я або вибору типу протоколу зв'язку й говорять про те, що мережа працює, але необхідно перевірити дані, що вводяться), 7XX (помилки, пов'язані з некоректними налаштуваннями з'єднання й говорять про те, що мережа працює, але необхідно перевірити налаштування) або 8XX (помилки, пов'язані із проблемами в налаштуванні мережі або відсутністю підключення до мережі. У випадку виникнення помилок цієї серії зв'яжіться з нашою службою підтримки). Нижче наведені розшифровки деяких кодів:

А тепер докладні пояснення по основних видах помилок:

400 Bad Request. Означає, що сервер виявив у запиті клієнта синтаксичну помилку.

1. Відключити міжмережний екран (брандмауер) або програми, що впливають на мережну активність і безпеку.

2. Спробувати обновити поточний браузер або використовувати іншої.

3. Поставити налаштування браузера по-умовчанню й обнулити всі збережені файли, сертифікати та інше, тобто видалити всі що пише браузер на диск.

Помилка 624, проблема при підключенні до VPN:

1. Відсутній файл RASPHONE.PBK в {папка з Windows}/system32/RAS або ... Documents and Settings/{ваш юзер або всі юзери}/Application Data/Microsoft/Network/Connections/Pbk, а якщо навіть і є, то перейменувати його в щось інше, наприклад RASPHONE.OLD, ще можна запустити rasphone.exe, може допомогти.

2. Є заборона на створення підключення. – Дозволити користувачам створювати нові підключення. Пуск-виконати-gpedit.msc і там розбиратися.

3. Заборонено доступ до папки Documents and Settings/All Users/Application Data/Microsoft/Network/Connections/Pbk – Відкрити доступ до

папки. Але швидше за все проблему вдасться вирішити в такий спосіб: зняти галку з пункту "Тільки для читання" у властивостях файлу Rasphone.pbk.

Помилка 691, проблема при підключенні до VPN:

1. Немає грошей.

2. Неправильний логін або пароль.

3. Неправильно зазначений у налаштуваннях сервер VPN.

4. Є користувальницьке або адміністративне блокування в системі статистики.

Помилка 800, проблема при підключенні до VPN:

Можливі причини:

1. Брандмауер блокує вихідні запити на VPN з'єднання, що в принципі трапляється, але не так часто.

2. Запит з якої-небудь причини не доходить до сервера, тобто можливо шлюз сегмента не пропускає запит у силу виниклого навантаження або збою. Що теж може траплятися, але дуже-дуже рідко.

3. Сервер відправляє відповідь про неможливість підключитися тому що в цей момент спостерігається велика кількість одночасних спроб з'єднання.

Можливі виправлення:

1. Перевірити чи працює локальна мережа в цей момент часу.

2. Перевірити проходження сигналу командою ping до шлюзу, а потім до сервера авторизації.

Натисніть кнопку "Пуск"->"Виконати". У рядку введіть команду.

Приклад:

ping 172.22.0.1

ping 172.22.0.254

де в цьому випадку 172.22.0.254 – сервер VPN доступу, а 172.22.0.1 – адреса локального сервера. Свій сервер доступу й адреса шлюзу Ви можете довідатися в "анкеті абонента" або через службу тех. підтримки.

Якщо в DOS-вікні що відкрилося побачите щось начебто "Заданий вузол недоступний" або "Перевищений інтервал очікування для запиту", то можливо проблеми на комп'ютері, або на лінії. Якщо піде відповідь від вузла із зазначеною швидкістю (звичайно це відбувається швидко й віконце швидко зникає), то швидше за вся проблема у комп’ютері. У кожному разі перевірте перший пункт (1) і виконаєте дії наступного пункту (3).

3. Перезавантажити свій комп'ютер.

Помилка 650. "Сервер віддаленого доступу не відповідає". Недоступний сервер доступу в Інтернет. Перевірте, чи включене "Підключення по локальній мережі", чи справна мережна карта, чи справний мережний кабель, чи не виставлений у налаштуваннях IP-з'єднання якась певна IP-адреса.

Помилка 735. "Запитана адреса була відкинута сервером". Ви неправильно настроїли VPN-з'єднання для доступу в мережу Інтернет. Швидше за все в налаштуваннях TCP/IP з'єднання VPN прописали якусь IP-адресу, а він повинен виділяться автоматично.

Помилка 789, "Обраний невірний тип VPN з'єднання". Зайдіть у налаштування VPN з'єднання й на вкладці "Мережа" зі списку "Тип VPN" виберіть "Автоматично". Спробуйте повторно підключитися.

Помилки 741 – 743, "Невірно настроєні параметри шифрування". Зайдіть у налаштування VPN з'єднання, і у вкладці "Безпеку" відключите пункт "шифрування даних".

Помилки 600, 601, 603, 606, 607, 610, 613, 614, 616, 618, 632, 635, 637, 638, 645. Перезавантажте комп'ютер. Якщо помилка не зникає, спробуйте переустановити VPN-з'єднання. Перевірте комп'ютер на предмет наявності вірусів. Видалите недавно встановлені програми, або скасуєте недавні зміни в налаштуваннях Windows, якщо такі були.

Помилки 604, 605, 608, 609, 615, 620, "Файл телефонної книги підсистеми віддаленого доступу Windows і поточна конфігурація Віддаленого Доступу до

Мережі несумісні один з одним". Перезавантажте комп'ютер. Якщо помилка не зникає, то видаліть й заново створіть VPN з'єднання.

Помилки 611, 612, "Внутрішня конфігурація мережі Windows некоректно настроєна". Перезавантажте комп'ютер і переконайтесь, що локальна мережа нормально функціонує на комп'ютері. Якщо помилка не зникає, то зверніться в службу технічної підтримки. Крім того, помилка може бути викликана недостачею ресурсів (пам'яті) на комп'ютері. Спробуйте закрити запущені програми.

Помилка 617, "Windows перебуває в процесі підключення до Інтернету, або відбулася внутрішня помилка Windows". Почекайте кілька мінут. Якщо підключення не встановилося, і при повторному підключенні помилка повторюється, то перезавантажте комп'ютер.

Помилка 619, "Неправильно настроєні параметри безпеки VPN з'єднання, VPN-трафік блокується на шляху до шлюзу, або налаштування VPN не вступили в дію". У властивостях VPN з'єднання відкрийте вкладку "Безпека" – повинне бути обране "Звичайні (параметри, що рекомендуються)" і повинна бути знята галочка "Потрібно шифрування даних (інакше відключатися)". Перезавантажте комп'ютер і спробуйте підключитися ще раз. Перевірте налаштування брандмауера, і, якщо не впевнені в їхній правильності, відключіть його.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм PRESENT – окремий випадок SP-мережі й складається з 31 раунду. Довжина блоку становить 64 біта, а ключі підтримуються в 2 варіантах, 80- і 128-бітні. Такого рівня захисту повинно цілком вистачати для низькозахищених додатків, звичайно використовуваних для розгортання на основі тегів, а крім того, що важливіше, PRESENT багато в чому збігається своїми конструктивними особливостями з потоковими шифрами проекту estream,

заточеними на ефективну реалізацію в залозі, що дозволяє нам адекватно порівнювати їх.

Вимоги з безпеки й експлуатаційні властивості 128-бітних версій надані в додатку до оригінальної статті.

Кожний з 31 раундів складається з операції XOR, щоб увести ключ K_i для $1 \leq i \leq 32$, де K_{32} використовується для «відбілювання» ключа, лінійної побітової перестановки й нелінійного шару заміщення (або, попросту говорячи, збільшення стійкості шифрування . Нелінійний шар використовує роздільні 4-бітні S-блоки, які застосовуються паралельно 16 раз на кожному раунді. Шифр, описаний псевдо-кодом представлено на рисунку 4.3.

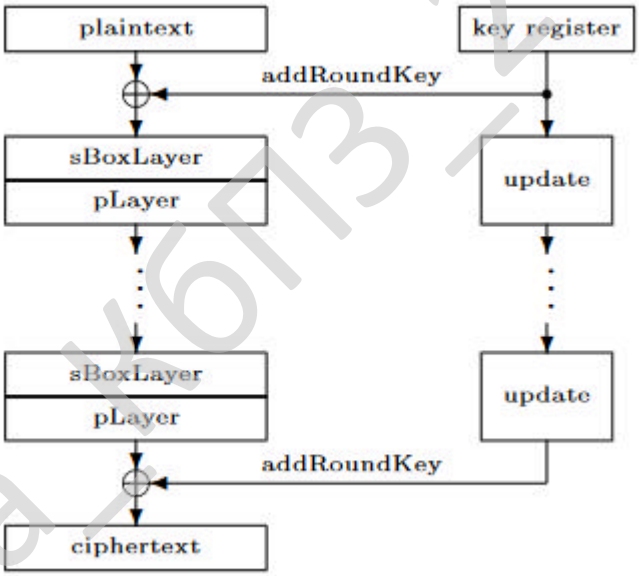

Рисунок 4.3 – Шифр PRESENT

Тепер кожна стадія визначається по черзі. Обґрунтування конструкції наведені нижче, а біти всюди нумеруються з нуля, починаючи із правого в блоці або слові.

Додавання раундового ключа (addRoundKey)

Заданий раундовий ключ $K_i = k^i_{63} \ldots k^i_0$, де $1 \le i \le 32$, а так само поточний стан $b_{63} \ldots b_0$. Додавання раундового ключа до поточного стану відбувається за модулем 2 ($b_j = b_j \oplus k^i_j$, де $0 \le j \le 63$).

Шар S-блоків (sBoxlayer)

Використовувані в PRESENT S-блоки відображають 4-бітні блоки в 4-бітні блоки. Дія цього блоку в шістнадцятковій системі числення наведене в наступній таблиці.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S[x]$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

Рисунок 4.4 – Дія блоку в шістнадцятковій системі числення

Для шару S-блоків поточний стан $b_{63} \ldots b_0$ представляє із себе 16 4-бітних слів $w_{15} \ldots w_0$, де $w_i = b_{4*i+3} \,||\, b_{4*i+2} \,||\, b_{4*i+1} \,||\, b_{4*i}$ для $0 \le i \le 15$. Вихід рамки $S[w_i]$ видає оновлені значення станів очевидним образом.

Шар перестановки (player)

Побітова перестановка, використовувана в PRESENT задається наступною таблицею (біт і стану зміщається на позицію P(i).

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P(i)$	0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P(i)$	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$P(i)$	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$P(i)$	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

Рисунок 4.5 – Побітова перестановка, використовувана в PRESENT

Перетворення ключа (The key schedule)

PRESENT може використовувати 80- і 128-бітні ключі, однак зосередимося на 80-бітній версії. Надаваний користувачем ключ зберігається в регістрі ключів K, представлений у вигляді $k_{79}k_{78}\ldots k_0$. На i-ом раунді 64-бітний раундовий ключ $Ki = k_{63}k_{62}\ldots k_0$, що полягає з 64 лівих битов поточного вмісту регістру K. Таким чином, на i-ом раунді маємо:

$K_i = k_{63}k_{62}\ldots k_0 = k_{79}k_{78}\ldots k_{16}$.

Після розпакування раундового ключа K_i регістр ключа $K = k_{79}k_{78}\ldots k_0$ обновляється в такий спосіб:

1. $[k_{79}k_{78}\ldots k_1k_0] = [k_{18}k_{17}\ldots k_{20}k_{19}]$
2. $[k_{79}k_{78}k_{77}k_{76}] = S[k_{79}k_{78}k_{77}k_{76}]$
3. $[k_{19}k_{18}k_{17}k_{16}k_{15}] = [k_{19}k_{18}k_{17}k_{16}k_{15}] \oplus$ round_counter

Отже, регістр ключа зрушується на 61 позицію вліво, 4 крайніх лівих біта, що пройшли через S-блок і round_counter значення і складається за модулем 2 з бітами $k_{19}k_{18}k_{17}k_{16}k_{15}$ з K з найменшим значущим бітом з round_counter праворуч.

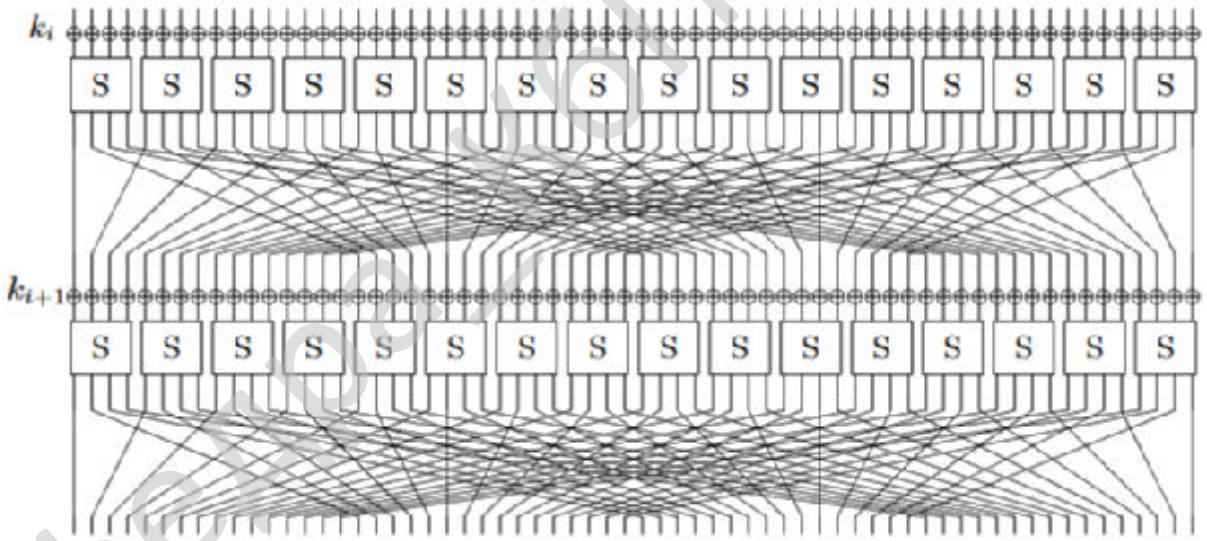

Рисунок 4.6 – Перетворення ключа (The key schedule)

Перетворення ключа для 128-бітного алгоритму можна знайти в додатку до оригінальної статті.

Конструктивні особливості PRESENT

Крім безпеки й ефективної реалізації, основне досягнення PRESENT – його простота. по цьому не дивно, що схожі проекти були прийняті в інших обставинах, і навіть були використані як навчальний посібник для студентів. У даній секції ми обґрунтуємо рішення, прийняті нами при проектуванні PRESENT. Однак, у першу чергу, опишемо очікувані прикладні вимоги.

Мети й середовище застосування

При проектуванні блокового шифру, застосовного в жорстко обмежених оточеннях, важливо зрозуміти, що ми не створюємо блоковий шифр, який, неодмінно, буде застосовний у багатьох ситуаціях – для цього існує AES. Навпаки, ми націлені на досить специфічне застосування, для якого AES не підходить. Вищесказане визначає нам наступні характеристики:

– Шифр буде реалізований «у залізі»

– Додатки будуть вимагатися лише для регулювання рівня безпеки. Отже, 80-бітний ключ буде здоровим розв'язком. Відзначимо, що такої ж позиції дотримуються розроблювачі потокових шифрів проекту eSTREAM.

– Додатки не припускають шифровки великої кількості даних. Таким чином, реалізація може бути оптимізована для продуктивності або простору без внесення занадто великих змін.

– У деяких застосуваннях можлива ситуація, що ключ буде зафіксований при виробництві. У такому випадку, не треба буде змінювати ключ пристрою (що може вилитися в атаки з маніпуляцією ключем .

– Фізичний обсяг пристрою буде першим пріоритетом, після безпеки, що спричинить обмеження на пікові й середні споживання енергії, і, отже, зрушить швидкодія в область низькопріоритетних параметрів.

– У пристроях, що вимагають найбільш ефективного використання фізичного простору, блоковий шифр найчастіше зможе лише шифрувати дані (encryption-only mode). Таким чином, він зможе бути використаний у запит-відповідь (challenge-response) протоколах авторизації, і, при дотриманні

контролю стану, може бути використаний для шифровки й дешифрування переговорів із пристроєм, використовуючи режим лічильника.

Виходячи з таких міркувань, розв'язали створити PRESENT як 64-бітний блоковий шифр із 80-бітним ключем. Шифровка й дешифрування, у цьому випадку, мають приблизно схожі фізичні вимоги. Маючи можливість підтримувати як шифрацію, так і дешифрацію, PRESENT буде компактніше, чим підтримуючий лише шифрацію AES. А у випадку encryption-only виконання, наш шифр виявиться й зовсім понад-легко. Суб-ключі що шифрують будуть обчислюватися на ходу.

У літературі є безліч прикладів атак компромісу між часом, датою й пам'яттю, або атак з використанням парадокса днів народження при шифровці великих обсягів даних. Однак, дані атаки залежать тільки від параметрів шифру й не використовують внутрішню структуру. Наша мета полягає в тому, щоб ці атаки були кращим, що можуть застосувати проти нас. Атаки стороннього каналу й атаки з безпосереднім зломом чипа загрожують PRESENT тією самою мірою, як і іншим криптографічним примітивам. Однак для ймовірних застосувань, помірні вимоги безпеки роблять вигоду, одержувану зловмисником на практиці, досить обмеженої. В оцінці ризиків, подібні погрози не сприймаються як істотний фактор.

Перестановочний шар

При виборі шару змішування ключа, наша увага до апаратної ефективності вимагає наявності лінійного шару, який може бути реалізований з мінімальною кількістю керуючих елементів (наприклад, транзисторів. це приводить до побітової перестановки. Приділяючи увагу простоті, ми вибрали регулярну бітову перестановку, що допомагає провести прозорий аналіз безпеки.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунках 5.1-5.6 наведено скріншоти розробленого, у результаті виконання бакалаврського проектування, програмного забезпечення.

На рисунку 5.1 зображено скриншот вікна створення VPN-з'єднання.

На рисунку 5.2 зображено скриншот вікна створення VPN-з'єднання (введення параметрів з'єднання).

На рисунку 5.3 зображено скриншот вікна створення VPN-з'єднання (завершення роботи).

На рисунку 5.4 зображено скриншот ярлика створеного VPN-з'єднання на робочому столі.

На рисунку 5.5 зображено скриншот вікна довідки.

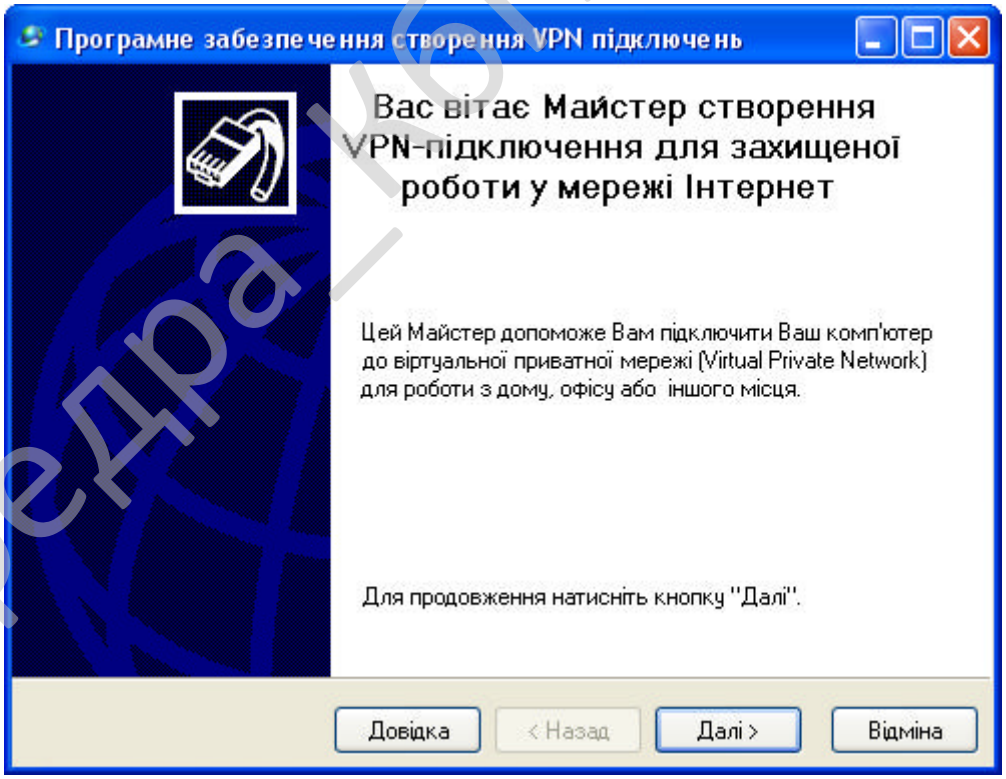

Рисунок 5.1 – Вікно створення VPN-з'єднання

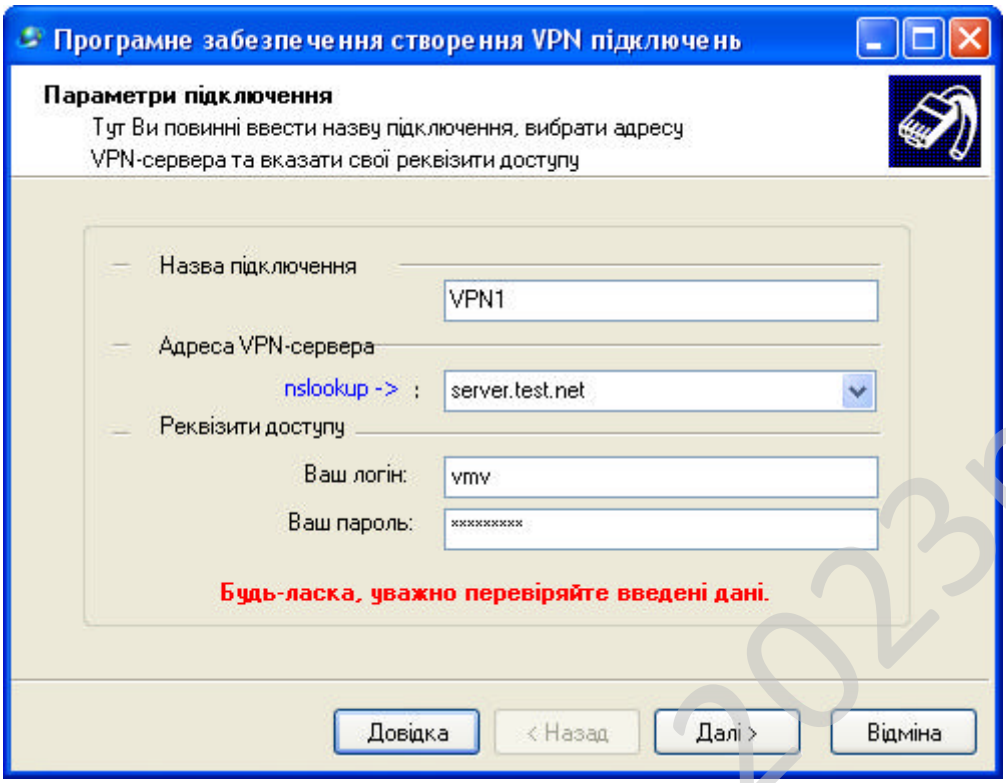

Рисунок 5.2 – Вікно створення VPN-з’єднання (введення параметрів з’єднання)

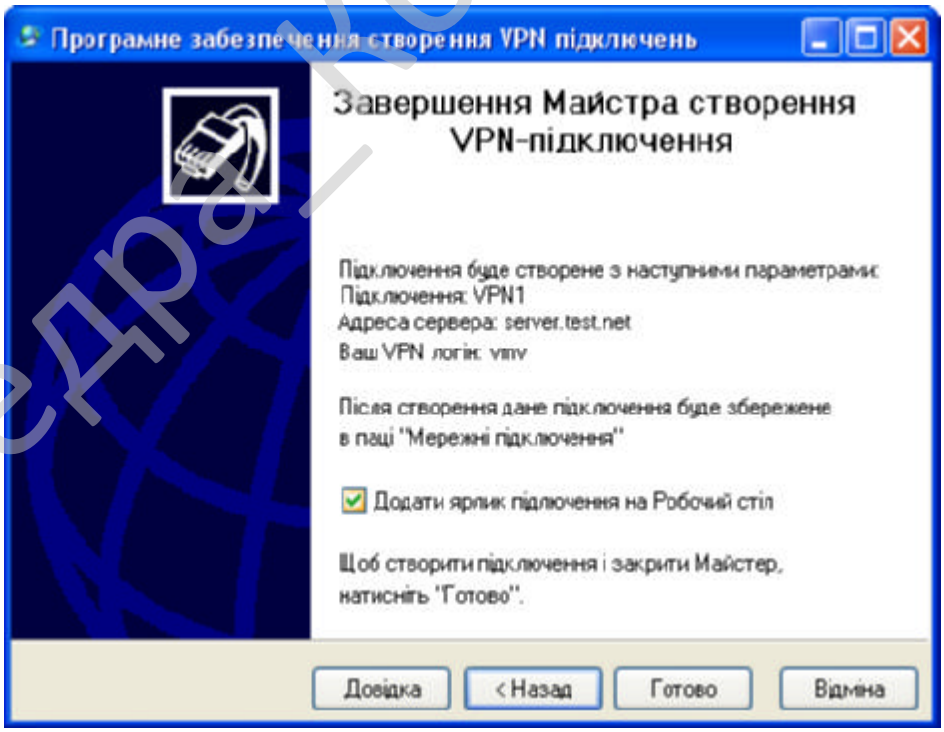

Рисунок 5.3 – Вікно створення VPN-з’єднання (завершення роботи)

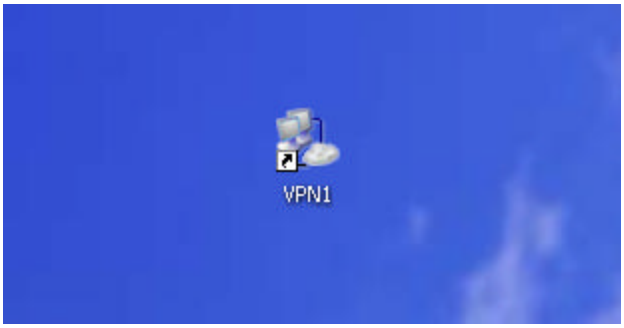

Рисунок 5.4 – Ярлик створеного VPN-з’єднання на робочому столі

Рисунок 5.5 – Довідка

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки створення VPN підключень.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем створення VPN підключень.

– Досліджена система створення VPN підключень.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки створення VPN підключень.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання створення VPN підключень.

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 Embarcadero RAD Studio. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи кібербезпеки створення VPN підключень. 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 кібербезпеки й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм PRESENT.

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. Смирнов А.А. Анализ и сравнительное исследование перспективных направлений развития цифровых телекоммуникационных систем и сетей / А.А.Смирнов, В.В.Босько, Е.В.Мелешко // Системи обробки інформації. – Х.: ХУ ПС, 2008. – Вип.7(74). – С.120-123.

2. Смирнов А.А. Усовершенствование метода управления очередями в многопротокольных узлах телекоммуникационной сети / А.А.Смирнов, Е.В.Мелешко // Збірник тез та доповідей другої всеукраїнської науково-практичної конференції «Системний аналіз. Інформатика. Управління». Запоріжжя. Тези доповідей. Запоріжжя: КПУ, 2011.

3. Смирнов С.А. Метод безопасной маршрутизации метаданных в облачные антивирусные системы / А.К. Дидык, С.А. Смирнов // Информационные технологии в управлении, образовании, науке и промышленности: монография / Под редакцией профессора В.С. Пономаренко. – Х.: Видавець Рожко С.Г., 2016. – 566 с.

4. Смирнов С. А. Сравнительные исследования математических моделей технологии распространения компьютерных вирусов в информационно-телекоммуникационных сетях / Мохамад Абу Таам Гани, А. А. Смирнов, А. В. Коваленко, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2014. – Вип. 9(125). – 105-110.

5. Смирнов С. А. Математическая модель интеллектуального узла коммутации с обслуживанием информационных пакетов различного приоритета / Мохамад Абу Таам Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов // Збірник наукових праць Харківського університету Повітряних Сил. – Харків: ХУПС, 2014. – Вип. 4 (41). – С. 48-52.

6. Смирнов С. А. Исследование показателей качества функционирования интеллектуальных узлов коммутации в телекоммуникационных системах и сетях

/ Мохамад Абу Таам Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов // Наука і техніка Повітряних Сил Збройних Сил України: наук. журн. – Х.: ХУПС, 2014. – № 4(17). – С. 90-95.

7. Смирнов С. А. Усовершенствованный алгоритм управления доступом к «облачным» телекоммуникационным ресурсам / Мохамад Абу Таам Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2015. –Вип. 1(126). – С. 150-153.

8. Smirnov S.A. Method of controlling access to intellectual switching nodes of telecommunication networks and systems / A.A. Smirnov, Mohamad Abou Taam, S.A. Smirnov // International Journal of Computational Engineering Research (IJCER). – Volume 5, Issue 5. – India. Delhi. – 2015. – P. 1-7.

9. Смирнов С. А. Анализ и исследование методов управления сетевыми ресурсами для обеспечения антивирусной защиты даннях / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Системи озброєння і військова техніка: наук. журн. – Х.: ХУПС, 2015. – № 3(43). – С. 100-107.

10. Смирнов С. А. Исследование эффективности метода управления доступом к облачным антивирусным телекоммуникационным ресурсам / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Наука і техніка Повітряних Сил Збройних Сил України: наук. журн. –Х.: ХУПС, 2015. – № 3(20). – С. 134-141.

11. Смирнов С. А. Комплекс gert-моделей технологии облачной антивирусной защиты телекоммуникационной системы / А. А. Смирнов, А. К. Дидык, А. Н. Дреев, С. А. Смирнов // Безпека інформації: наук. - практ. журн. – К.: НАУ, 2015. – Т. 21, № 3. – С. 251-262.

12. Смирнов С. А. Метод безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, А. К. Дидык, С. А. Смирнов // Системи озброєння і військова техніка: наук. журн. – Х.: ХУПС, 2016. – № 2 (46). – С. 146-149.

13. Смирнов С. А. Модели системы нейросетевых экспертов безопасной маршрутизации в облачных антивирусных системах / А. А. Смирнов, А. К. Дидык, А. Н. Дреев, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2016. – Вип. 3 (140). – С. 36-39.

14. Смирнов С. А. Метод безопасной маршрутизации на базовом множестве путей передачи метаданных в облачные антивирусные системы / В. Л. Бурячок, С. А. Смирнов // Системи управління, навігації та зв'язку. – Полтава, 2016. – Вип. 4(40). – С. 57-62.

15. Смирнов С. А. Способ контроля линий связи телекоммуникационной системы облачного антивируса / А. А. Смирнов, А. К. Дидык, А. Н. Дреев, С. А. Смирнов // Збірник наукових праць Харківського університету Повітряних Сил. – Харків: ХУПС, 2016. – № 2 (47). – С. 148-152.

16. Смирнов С. А. Дослідження та реалізація GERT-моделі технології розповсюдження комп'ютерних вірусів для захисту телекомунікаційних систем / В. Л. Бурячок, Мохамад Абу Таам Гани, С. А. Смирнов // Інформаційні технології та комп'ютерна інженерія: зб. тез доп. наук.-практ. конф., м. Кіровоград, 4 грудня 2014 р. – Кіровоград: КНТУ, 2014. – С. 168.

17. Смирнов С. А. Исследование математических моделей технологии распространения компьютерных виру сов / А. А. Смирнов, Мохамад Абу Таам Гани, С. А. Смирнов // Актуальні питання забезпечення кібернетичної безпеки та захисту інформації: зб. наук. праць міжнар. наук.-практ. конф., м. Київ, 25-28 лютого 2015 р. – К.: Європейський університет, 2015. – С. 90-91.

18. Смирнов С. А. Метод управления доступом к «облачным» ресурсам для защиты телекоммуникационных систем / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Всеукраїнська науково-практична конференція «Інформаційна безпека держави, суспільства та особистості», м. Кіровоград, 16 квітня 2015 р.: зб. тез доп. – Кіровоград: КНТУ, 2015. – С. 50-52.

19. Смирнов С. А. Разработка метода управления доступом в интеллектуальных узлах коммутации / А. А. Смирнов, Мохамад Абу Таам Гани,

С. А. Смирнов // Проблеми і перспективи розвитку ІТ-індустрії: зб. тез VII міжнар. наук.-практ. конф., м. Харків, 17-18 квітня 2015 р. – Х.: ХНЕУ, 2015. – С. 14.

20. Смирнов С.А. Реализация метода управления доступом в интеллектуальных узлах коммутации / А.А. Смирнов, Мохамад Абу Таам Гани, С.А. Смирнов // Збірник тез XVII міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування». м. Кіровоград. 17-18 квітня 2015 р. – Кіровоград: КНТУ. – 2015. – С. 91-92.

21. Смирнов С. А. технология передачи сигнатур в облачные антивирусные системы для обеспечения защищенности телекоммуникационных сетей / А. А. Смирнов, С. А. Смирнов // Збірник тез V міжнародної науково-технічної конференції «ITSEC», Київ, 19-22 травня 2015 р. – К.: НАУ 2015. – С. 12-13.

22. Смирнов С. А. Реализация математической модели интеллектуального узла коммутации для обеспечения защищенности телекоммуникационной сети / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Інформаційна та економічна безпека (INFECO-2015): зб. тез II Міжнар. наук.-практ. Інтернет-конф., м. Харків, 21-22 травня 2015 р. – Х.: ХІБС УБС НБУ, 2015. – С. 20-24.

23. Смирнов С. А. Разработка математической модели технологии распространения компьютерных вирусов в информационно-телекоммуникационных сетях / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Сборник тезисов XI международной конференции «Стратегия качества в промышленности и образовании», г. Варна, Болгария, 01-06 июня 2015 г. – Варна: ТУВ, 2015. – С. 488-491.

24. Смирнов С. А. Метод управления доступом к облачным телекоммуни-кационным ресурсам для обеспечения защиты даннях / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Комп'ютерні технології та

інформаційна безпека: зб. тез доп. міжнар. наук.-практ. конф., м. Кіровоград, 2-3 липня 2015 р. – Кіровоград: КНТУ, 2015. – С. 4-5.

25. Смирнов С. А. Имитационная модель системы управления доступом к облачным антивирусным телекоммуникационным ресурсам / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Збірник тез першої всеукраїнської науково-практичної конференції «Перспективні напрями захисту інформації» (м. Затока, 7-9 вересня 2015 р.). – Одеса: ОНАЗ, 2015. – С. 90-94.

26. Смирнов С. А. Разработка комплекса gert-моделей технологии облачной антивирусной защиты телекоммуникационной системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформаційні технології та взаємодії» (IT & I): зб. тез II міжнар. наук.-практ. конф., м. Київ, 3-5 листопада 2015 р. – К.: КНУ ім. Тараса Шевченка, 2015. – С. 65-67.

27. Смирнов С. А. Разработка моделей телекоммуникационной системы формирования и обработки метаданных в облачных антивирусных системах / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Информационные и телекоммуникационные технологии: образование, наука, практика: сб. тезисов II междунар. научно-практ. конф., г. Алматы, Казахстан, 3-4 декабря 2015 г. – Алматы: КазНИТУ им. К.И. Сатпаева, 2015. – С. 309-313.

28. Смирнов С. А. gert-модели технологии облачной антивирусной защиты / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Безпека українського суспільства в концепції вступу в постіндустріальне суспільство ЄС: зб. тез Круглого столу, м. Київ, 16 грудня 2015 р. – К.: Європейський університет, 2015. – С.41-43.

29. Смирнов С. А. Алгоритмы формирования множества маршрутов передачи метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Актуальні питання забезпечення кібернетичної безпеки та захисту інформації: зб. наук. праць II Міжнар. наук.-практ. конф., м. Київ, 24-27 лютого 2016 р. – К.: Європейський університет, 2016. – С. 140-142.

30. Смирнов С. А. Разработка и реализация метода безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Securitea informationala 2015-2016: Conferenta internationala (editia a XII-a), Chisinau, Moldova, 3 martie 2016. – Chisinau: ADSEM, 2016. – С. 90-96.

31. Смирнов С. А. Алгоритм формирования базового множества маршрутов передачи метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформатика та системні науки (ІСН-2016): зб. тез VII всеукр. наук.-практ. конф., м. Полтава, 10-12 березня 2016 р. – Полтава: ПУЕТ, 2016. – С. 261-263.

32. Смирнов С. А. Система обработки и формирования начального состояния маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Проблеми кібербезпеки інформаційно-телекомунікаційних систем: зб. тез наук.-практ. конф., м. Київ, 10-11 березня 2016 р. – К.: КНУ ім. Тараса Шевченка, 2016. – С. 81-82.

33. Смирнов С. А. Алгоритм безопасной маршрутизации на базовом множестве путей передачи метаданных в программный сервер облачной антивирусной системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформаційна безпека та комп'ютерні технології (IS&CT): зб. тез міжнар. наук.-практ. конф., м. Кіровоград, 24-25 березня 2016 р. – Кіровоград: КНТУ, 2016. – С. 73.

34. Смирнов С. А. Исследование способа контроля линий связи телекоммуникационной системы для облачных антивирусов / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Збірник тез першої міжнародної науково-практичної конференції «Проблеми науково-технічного та правового забезпечення кібербезпеки у сучасному світі» (ПНПЗК-2016), м. Харків, 30 березня - 1 квітня 2016 р. – Х.: НТУ «ХПІ», 2016. – С. 14.

35. Смирнов С. А. Разработка способа контроля линий связи телекоммуникационной системы для облачных антивирусов / А. А. Смирнов,

С. А. Смирнов, А. К. Дидык // Матеріали XVIII міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування» (м. Кіровоград, 15-16 квітня 2016 р.). –Кіровоград: КНТУ, 2016. – С. 182-186.

36. Смирнов С. А. Разработка и исследование способа контроля линий связи телекоммуникационных сетей для облачных антивирусных систем / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Проблеми і перспективи розвитку ІТ-індустрії: VIII міжнар. наук.-практ. конф., м. Харків, 28-29 квітня 2016 р.: зб. тез. – Х.: ХНЕУ, 2016. – С. 48.

37. Смирнов С. А. Модель системы нейросетевых экспертов безопасной маршрутизации для облачных антивирусных систем / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформаційна та економічна безпека (INFECO-2016): зб. тез III міжнар. наук.-практ. конф., м. Харків, 28-30 кві. 2016 р. – Х.: ХННІ ДВНЗ «УБС», 2016. – С. 178-182.

38. Смирнов С. А. Метод безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Сборник тезисов XII международной конференции «Стратегия качества в промышленности и образовании» (г. Варна, Болгария, 30 мая - 02 июня 2016 г.). – Варна: ТУВ, 2016. – С. 581-585.

39. Смирнов С. А. Оценка эффективности метода безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. С. Коваленко // РадіоЕлектроніка та ІнфоКомунікації: зб. тез першої наук. - техн. конф., м. Київ, 11-16 вересня 2016 р. – К.: НТУУ «КПІ», 2016. – С. 17.

40. Современные телекоммуникации. Технологии и экономика / [В.Л. Банкет, О.В. Бондаренко, П.П. Воробьенко и др.]; под ред. С.А. Довгого. – М.: Эко-Трендз, 2003. – 320 с.

41. Столлингс В. Современные компьютерные сети / Вильям Столлингс. –СПб.: Питер, 2003. – 778 с.

42. Таненбаум Э. Компьютерные сети / Эндрю Таненбаум; пер. с англ. А. Леонтьев. – СПб.: Питер, 2002. – 848 с.

43. Телекоммуникационные системы и сети: учебное пособие. В 3 томах / [В.В. Величко, Е.А. Субботин, В.П. Шувалов, А.Ф. Ярославцев]; под ред. В.П. Шувалова. – М.: Горячая линия-Телеком, 2005, т. 3 – 592 с.

44. Хайкин С. Нейронные сети: полный курс / С. Хайкин. – М.:Вильямс, 2006. – 1103 с.

45. Шелухин О.И. Фрактальные процессы в телекоммуникациях: моногр. / О.И. Шелухин, А.М. Тенякшев, А.В. Осин – М.: Радиотехника, 2003. – 480 с.

46. Elwalid, D. Mitra, I. Saniee, and I. Widjaja. Routing and Protection in GMPLS Networks: From Shortest Paths to Optimized Designs // Journal of lightwave technology. – 2003. – №21(11), P. 2828-28-38.

47. A.B. Bagula, M. Botha, and A.E Krzesinski. Online Traffic Engineering: The Least Interference Optimization Algorithm // IEEE Communications Society – 2004, P. 1232-1236.

48. Anees Shaikh, Jennifer Rexford, and Kang G. Shin. Evaluating the Impact of Stale Link State on Quality-of-Service Routing // IEEE/ACM Transactions on Networking. – 2001. – №9(2), P. 162-176.

49. Basabi Chakraborty. Simultaneous Search for Multiple Routes using Genetic Algorithm / IEEE lntemational Conference on Computational oltelligencs for Measurement System and Applications Boston. MA, USA, 14-16, July 2004, P. 77-80/

50. Barakat, E. Altman, and W. Dabbous. On TCP performance in a heterogeneous network: a survey // IEEE Communications Magazine. – 2000. – №38(1). – P. 40 - 46.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

					ВКРБ-125.23.0014.00.00.ТЗ	
Вим.	Арк.	№ документа	Підпис	Дата		
Розробив		Москальов А.В.			Програмне забезпечення системи кібербезпеки створення VPN підключень	Літ. Б; Аркуш 1; Аркушів 6
Перевірив		Якименко Н.М.				
Н. Контр.		Гермак В.С.				ЦНТУ КБ-19
Затв.		Смірнов О.А.				

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки створення VPN підключень.

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. № 12-02 від 5.01.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи кібербезпеки створення VPN підключень.

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 кібербезпеки з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи кібербезпеки створення VPN підключень;

– 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 архітектури IBM PC, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

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 Embarcadero RAD Studio.

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 – 85 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.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2023 р.

11.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 2.06.2023 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

_____ Якименко Н.М.

Програмне забезпечення системи кібербезпеки створення VPN підключень

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 54

Літера: РП

Кропивницький – 2023 року

VPN_WinMain.pas - основна програма

```
unit VPN_WinMain;

interface

uses
  Windows, Messages, CommCtrl, VPN_CommCtrl, VPN_SysUtils, VPN_FileInfo,
VPN_Resources,
  VPN_WelcWind, VPN_SettWind, VPN_FinsWind;

function WinMain(hInstance: HINST; hPrevInstance: HINST; lpCmdLine: LPSTR;
nCmdShow: Integer): Integer; stdcall;

implementation

function WinMain(hInstance: HINST; hPrevInstance: HINST; lpCmdLine: LPSTR;
nCmdShow: Integer): Integer; stdcall;
var
  hMutex : THandle;
  pszText: WideString;
  iccex  : TInitCommonControlsEx;
  psh    : TPropSheetHeader;
begin

  // витягаємо інформацію з ресурсу версії й заповнюємо їй підготовлену
  // структуру, що у подальшому будемо використовувати для читання/запису
  // настроювань програми й виводу тексту в заголовку повідомлень.

  ZeroMemory(@exeInfo, SizeOf(TStringFileInfo));
  GetFileInfo(AnsiStringToWide(ParamStr(0), CP_ACP), exeInfo);

  // створюємо Mutex для перевірки запуску копій додатка.

  hMutex := CreateMutex(nil, FALSE, MAKEINTRESOURCEW(exeInfo.pszProductName));
  if (GetLastError = ERROR_ALREADY_EXISTS) then
  begin
    pszText := LoadStrInst(hInstance, RC_STRING_COPYRUN);
    MessageBox(
      GetActiveWindow,
      @pszText[1],
      MAKEINTRESOURCEW(exeInfo.pszProductName),
      MB_OK or MB_ICONEXCLAMATION or MB_SYSTEMMODAL
    );
    Halt;
  end;

  // ініціалізуємо бібліотеку стандартних органів управління.

  iccex.dwSize := SizeOf(TInitCommonControlsEx);
  iccex.dwICC  := ICC_ANIMATE_CLASS or ICC_PROGRESS_CLASS or ICC_TAB_CLASSES or
   ICC_STANDARVPN_CLASSES or ICC_WIN95_CLASSES;
  InitCommonControlsEx(iccex);

  //

  pszText := Format(LoadStrInst(hInstance, RC_STRING_CWINDOW),
    [exeInfo.pszProductName, exeInfo.pszFileVersion]);

  // створюємо й відображаємо сторінки майстра.

  ZeroMemory(@psp, SizeOf(TPropSheetPage));

  psp.dwSize            := SizeOf(TPropSheetPage);
  psp.dwFlags           := PSP_USETITLE or PSP_HIDEHEADER;
  psp.pszTitle          := @pszText[1];
```

```
  psp.pfnDlgProc        := @WelcDlgProc;
  psp.pszTemplate       := MAKEINTRESOURCEW(RC_DIALOG_WELCOME);
  ahpsp[0]              := CreatePropertySheetPage(psp);

  ZeroMemory(@psp, SizeOf(TPropSheetPage));

  psp.dwSize            := SizeOf(TPropSheetPage);
  psp.dwFlags           := PSP_USETITLE or PSP_USEHEADERTITLE or
PSP_USEHEADERSUBTITLE;
  psp.pszTitle          := @pszText[1];
  psp.pszHeaderTitle    := MAKEINTRESOURCEW(LoadStrInst(hInstance,
RC_STRING_THEADER));
  psp.pszHeaderSubTitle := MAKEINTRESOURCEW(LoadStrInst(hInstance,
RC_STRING_SHEADER));
  psp.pszTemplate       := MAKEINTRESOURCEW(RC_DIALOG_SETTINGS);
  psp.pfnDlgProc        := @SettDlgProc;
  ahpsp[1]              := CreatePropertySheetPage(psp);

  ZeroMemory(@psp, SizeOf(TPropSheetPage));

  psp.dwSize            := SizeOf(TPropSheetPage);
  psp.dwFlags           := PSP_USETITLE or PSP_HIDEHEADER;
  psp.pszTitle          := @pszText[1];
  psp.pszTemplate       := MAKEINTRESOURCEW(RC_DIALOG_FINISH);
  psp.pfnDlgProc        := @FinsDlgProc;
  ahpsp[2]              := CreatePropertySheetPage(psp);

  ZeroMemory(@psh, SizeOf(TPropSheetHeader));

  psh.dwSize          := SizeOf(TPropSheetHeader);
  psh.hInstance       := hInstance;
  psh.hwndParent      := 0;
  psh.phpage          := @ahpsp[0];
  psh.nStartPage      := 0;
  psh.nPages          := Length(ahpsp);
  psh.pszbmWatermark := MAKEINTRESOURCEW(RC_BITMAP_WATERMARK);
  psh.pszbmHeader     := MAKEINTRESOURCEW(RC_BITMAP_HEADER);
  psh.dwFlags         := PSH_WIZARD97 or PSH_WATERMARK or PSH_HEADER or
PSH_USEICONID;
  psh.pszIcon         := MAKEINTRESOURCEW(RC_ICONEX_CAPTION);

  PropertySheet(psh);

  // видаляємо іменований об'єкт.

  if (hMutex <> 0) then
  begin
    ReleaseMutex(hMutex);
    CloseHandle(hMutex);
  end;

  //

  Result := 0;

end;

end.
```

VPN_WinSvc.pas - сервіси

```
unit VPN_WinSvc;

interface

uses
  Windows;

type
  LPSERVICE_STATUS = ^SERVICE_STATUS;
  _SERVICE_STATUS = record
    dwServiceType: DWORD;
    dwCurrentState: DWORD;
    dwControlsAccepted: DWORD;
    dwWin32ExitCode: DWORD;
    dwServiceSpecificExitCode: DWORD;
    dwCheckPoint: DWORD;
    dwWaitHint: DWORD;
  end;
  SERVICE_STATUS = _SERVICE_STATUS;
  TServiceStatus = SERVICE_STATUS;
  PServiceStatus = LPSERVICE_STATUS;
  LPSERVICE_STATUS_PROCESS = ^SERVICE_STATUS_PROCESS;
  _SERVICE_STATUS_PROCESS = record
    dwServiceType: DWORD;
    dwCurrentState: DWORD;
    dwControlsAccepted: DWORD;
    dwWin32ExitCode: DWORD;
    dwServiceSpecificExitCode: DWORD;
    dwCheckPoint: DWORD;
    dwWaitHint: DWORD;
    dwProcessId: DWORD;
    dwServiceFlags: DWORD;
  end;
  SERVICE_STATUS_PROCESS = _SERVICE_STATUS_PROCESS;
  TServiceStatusProcess = SERVICE_STATUS_PROCESS;
  PServiceStatusProcess = LPSERVICE_STATUS_PROCESS;


//
// Структура перерахунку статуса послуги
//
  LPENUM_SERVICE_STATUSA = ^ENUM_SERVICE_STATUSA;
  {$EXTERNALSYM LPENUM_SERVICE_STATUSA}
  _ENUM_SERVICE_STATUSA = record
    lpServiceName: LPSTR;
    lpDisplayName: LPSTR;
    ServiceStatus: SERVICE_STATUS;
  end;
  {$EXTERNALSYM _ENUM_SERVICE_STATUSA}
  ENUM_SERVICE_STATUSA = _ENUM_SERVICE_STATUSA;
  {$EXTERNALSYM ENUM_SERVICE_STATUSA}
  TEnumServiceStatus = ENUM_SERVICE_STATUSA;
  PEnumServiceStatus = LPENUM_SERVICE_STATUSA;
  LPENUM_SERVICE_STATUSW = ^ENUM_SERVICE_STATUSW;
  {$EXTERNALSYM LPENUM_SERVICE_STATUSW}
  _ENUM_SERVICE_STATUSW = record
    lpServiceName: LPWSTR;
    lpDisplayName: LPWSTR;
    ServiceStatus: SERVICE_STATUS;
  end;
  {$EXTERNALSYM _ENUM_SERVICE_STATUSW}
  ENUM_SERVICE_STATUSW = _ENUM_SERVICE_STATUSW;
  {$EXTERNALSYM ENUM_SERVICE_STATUSW}
  TEnumServiceStatus = ENUM_SERVICE_STATUSW;
  PEnumServiceStatus = LPENUM_SERVICE_STATUSW;
```

```
  PEnumServiceStatus = PEnumServiceStatus;

  _SC_STATUS_TYPE = (SC_STATUS_PROCESS_INFO);
  SC_STATUS_TYPE = _SC_STATUS_TYPE;

//
// Типи заголовків
//

  {$EXTERNALSYM SC_HANDLE}
  SC_HANDLE = THandle;
  {$EXTERNALSYM LPSC_HANDLE}
  LPSC_HANDLE = ^SC_HANDLE;

const  //
// Стан сервісу – для перерахуємих послуг (бітова маска)
//
  {$EXTERNALSYM SERVICE_ACTIVE}
  SERVICE_ACTIVE                = $00000001;
  {$EXTERNALSYM SERVICE_INACTIVE}
  SERVICE_INACTIVE              = $00000002;
  {$EXTERNALSYM SERVICE_STATE_ALL}
  SERVICE_STATE_ALL             = (SERVICE_ACTIVE   or
                                   SERVICE_INACTIVE);


//
// Менеджер сервісу управління об’єкту типу специфічного доступа
//
  {$EXTERNALSYM SC_MANAGER_CONNECT}
  SC_MANAGER_CONNECT            = $0001;
  {$EXTERNALSYM SC_MANAGER_CREATE_SERVICE}
  SC_MANAGER_CREATE_SERVICE     = $0002;
  {$EXTERNALSYM SC_MANAGER_ENUMERATE_SERVICE}
  SC_MANAGER_ENUMERATE_SERVICE  = $0004;
  {$EXTERNALSYM SC_MANAGER_LOCK}
  SC_MANAGER_LOCK               = $0008;
  {$EXTERNALSYM SC_MANAGER_QUERY_LOCK_STATUS}
  SC_MANAGER_QUERY_LOCK_STATUS  = $0010;
  {$EXTERNALSYM SC_MANAGER_MODIFY_BOOT_CONFIG}
  SC_MANAGER_MODIFY_BOOT_CONFIG = $0020;

  {$EXTERNALSYM SC_MANAGER_ALL_ACCESS}
  SC_MANAGER_ALL_ACCESS         = (STANDARVPN_RIGHTS_REQUIRED or
                                   SC_MANAGER_CONNECT or
                                   SC_MANAGER_CREATE_SERVICE or
                                   SC_MANAGER_ENUMERATE_SERVICE or
                                   SC_MANAGER_LOCK or
                                   SC_MANAGER_QUERY_LOCK_STATUS or
                                   SC_MANAGER_MODIFY_BOOT_CONFIG);

//
// Сервіс  об’єкту типу специфічного доступа
//
  SERVICE_STOP                  = $0020;
  SERVICE_QUERY_STATUS          = $0004;
  SERVICE_ENUMERATE_DEPENDENTS  = $0008;
//
// Стан сервісу – для поточного стану
//
  {$EXTERNALSYM SERVICE_STOPPED}
  SERVICE_STOPPED               = $00000001;
  {$EXTERNALSYM SERVICE_START_PENDING}
  SERVICE_START_PENDING         = $00000002;
  {$EXTERNALSYM SERVICE_STOP_PENDING}
  SERVICE_STOP_PENDING          = $00000003;
  {$EXTERNALSYM SERVICE_RUNNING}
  SERVICE_RUNNING               = $00000004;
  {$EXTERNALSYM SERVICE_CONTINUE_PENDING}
```

```
  SERVICE_CONTINUE_PENDING       = $00000005;
  {$EXTERNALSYM SERVICE_PAUSE_PENDING}
  SERVICE_PAUSE_PENDING          = $00000006;
  {$EXTERNALSYM SERVICE_PAUSED}
  SERVICE_PAUSED                 = $00000007;
//
// Управління
//
  {$EXTERNALSYM SERVICE_CONTROL_STOP}
  SERVICE_CONTROL_STOP           = $00000001;
  {$EXTERNALSYM SERVICE_CONTROL_PAUSE}
  SERVICE_CONTROL_PAUSE          = $00000002;
  {$EXTERNALSYM SERVICE_CONTROL_CONTINUE}
  SERVICE_CONTROL_CONTINUE       = $00000003;
  {$EXTERNALSYM SERVICE_CONTROL_INTERROGATE}
  SERVICE_CONTROL_INTERROGATE    = $00000004;
  {$EXTERNALSYM SERVICE_CONTROL_SHUTDOWN}
  SERVICE_CONTROL_SHUTDOWN       = $00000005;

function OpenSCManager(lpMachineName: LPCWSTR; lpDatabaseName: LPCSTR;
dwDesiredAccess: DWORD): SC_HANDLE; stdcall;
function OpenSCManager(lpMachineName: LPCWSTR; lpDatabaseName: LPCWSTR;
dwDesiredAccess: DWORD): SC_HANDLE; stdcall;
function OpenSCManager(lpMachineName: LPCWSTR; lpDatabaseName: LPCSTR;
dwDesiredAccess: DWORD): SC_HANDLE; stdcall;

function OpenService(hSCManager: SC_HANDLE; lpServiceName: LPCSTR;
dwDesiredAccess: DWORD): SC_HANDLE; stdcall;
function OpenService(hSCManager: SC_HANDLE; lpServiceName: LPCWSTR;
dwDesiredAccess: DWORD): SC_HANDLE; stdcall;
function OpenService(hSCManager: SC_HANDLE; lpServiceName: LPCSTR;
dwDesiredAccess: DWORD): SC_HANDLE; stdcall;

function CloseServiceHandle(hSCObject: SC_HANDLE): BOOL; stdcall;

function QueryServiceStatusEx(hService: SC_HANDLE; InfoLevel: SC_STATUS_TYPE;
lpBuffer: PByte; cbBufSize: DWORD; var pcbBytesNeeded: DWORD): BOOL; stdcall;

function ControlService(hService: SC_HANDLE; dwControl: DWORD; var
lpServiceStatus: TServiceStatusProcess): BOOL; stdcall;

function EnumDependentServices(hService: SC_HANDLE; dwServiceState: DWORD;
lpServices: LPENUM_SERVICE_STATUSA; cbBufSize: DWORD; var pcbBytesNeeded,
lpServicesReturned: DWORD): BOOL; stdcall;
function EnumDependentServices(hService: SC_HANDLE; dwServiceState: DWORD;
lpServices: LPENUM_SERVICE_STATUSW; cbBufSize: DWORD; var pcbBytesNeeded,
lpServicesReturned: DWORD): BOOL; stdcall;
function EnumDependentServices(hService: SC_HANDLE; dwServiceState: DWORD;
lpServices: LPENUM_SERVICE_STATUSA; cbBufSize: DWORD; var pcbBytesNeeded,
lpServicesReturned: DWORD): BOOL; stdcall;

implementation
function OpenSCManager; external advapi32 name 'OpenSCManager';
function OpenSCManager; external advapi32 name 'OpenSCManager';
function OpenSCManager; external advapi32 name 'OpenSCManager';
function OpenService; external advapi32 name 'OpenService';
function OpenService; external advapi32 name 'OpenService';
function OpenService; external advapi32 name 'OpenService';
function CloseServiceHandle; external advapi32 name 'CloseServiceHandle';
function QueryServiceStatusEx; external advapi32 name 'QueryServiceStatusEx';
function ControlService; external advapi32 name 'ControlService';
function EnumDependentServices; external advapi32 name 'EnumDependentServices';
function EnumDependentServices; external advapi32 name 'EnumDependentServices';
function EnumDependentServices; external advapi32 name 'EnumDependentServices';
end.
```

VPN_SysUtils.pas - утиліти

```
unit VPN_SysUtils;

interface

uses
  Windows, Messages;

function LoadStrInst(hInst: HMODULE; I: Integer): WideString;
function Format(szString: WideString; const Params: Array of const): WideString;
function WideStringToAnsi(pszText: WideString; CodePage: WORD): AnsiString;
function AnsiStringToWide(pszText: AnsiString; CodePage: WORD): WideString;
function ExtractFilePath(pszText: WideString): WideString;
function ExcludeTrailingPathDelimiter(szString: WideString): WideString;
function SetCenterDialogPos(hDialog, hParent: HWND; IsParent: Boolean): Boolean;
function GetWindowFontSize(hWnd: HWND; pSize: Integer): Integer;
function GetWindowBoldFont(hWnd: THandle; fntHeight: Integer): HFONT;

implementation

//

function LoadStrInst(hInst: HMODULE; I: Integer): WideString;
var
  lpBuffer: Array [0..MAX_ PATH-1] of WideChar;
begin
  LoadString(hInst, I, lpBuffer, Length(lpBuffer));
  Result := lpBuffer;
end;

//

function Format(szString: WideString; const Params: Array of const): WideString;
var
  lpChar: Array [0..1023] of WideChar;
  lpWord: Array [0..15] of LongWord;
  nIndex: Integer;
begin
  for nIndex := High(Params) downto 0 do
    lpWord[nIndex] := Params[nIndex].VInteger;
  wvsprintf(@lpChar, @szString[1], @lpWord);
  Result := lpChar;
end;

//

function WideStringToAnsi(pszText: WideString; CodePage: WORD): AnsiString;
var
  dwBytes: Integer;
  dwFlags: DWORD;
begin
  if (pszText <> '') then
    begin
      dwFlags := WC_COMPOSITECHECK or WC_DISCARDNS or WC_SEPCHARS or
WC_DEFAULTCHAR;
      dwBytes := WideCharToMultiByte(CodePage, dwFlags, @pszText[1], -1, nil, 0,
        nil, nil);
      SetLength(Result, dwBytes - 1);
      if (dwBytes > 1) then
        WideCharToMultiByte(CodePage, dwFlags, @pszText[1], -1, @Result[1],
          dwBytes - 1, nil, nil);
    end
  else
    Result := '';
end;

//
```

```
function AnsiStringToWide(pszText: AnsiString; CodePage: WORD): WideString;
var
  dwBytes: Integer;
begin
  if (pszText <> '') then
    begin
      dwBytes := MultiByteToWideChar(CodePage, MB_PRECOMPOSED, @pszText[1], -1,
        nil, 0);
      SetLength(Result, dwBytes - 1);
      if (dwBytes > 1) then
        MultiByteToWideChar(CodePage, MB_PRECOMPOSED, @pszText[1], -1,
@Result[1],
          dwBytes - 1);
    end
  else
    Result := '';
end;

//

function ExtractFilePath(pszText: WideString): WideString;
var
  L: Integer;
begin
  Result := '';
  L := Length(pszText);
  while (L > 0) do
    begin
      if (pszText[L] = ':') or (pszText[L] = '\') then
        begin
          Result := Copy(pszText, 1, L);
          Break;
        end;
      Dec(L);
    end;
end;

//

function ExcludeTrailingPathDelimiter(szString: WideString): WideString;
var
  I: Integer;
begin
  Result := szString;
  I := Length(Result);
  while (I > 0) and (Result[I] = '\') do
    Dec(I);
  SetLength(Result, I);
end;

//

function SetCenterDialogPos(hDialog, hParent: HWND; IsParent: Boolean): Boolean;
var
  wRect  : TRect;
  pRect  : TRect;
  wArea  : TRect;
  xLeft  : Integer;
  yTop   : Integer;
  iWidth : Integer;
  iHeight: Integer;
  dwFlags: DWORD;
begin
  case IsParent of
    FALSE:
      begin
        GetWindowRect(hDialog, wRect);
        iWidth  := wRect.Right - wRect.Left;
```

```
        iHeight := wRect.Bottom - wRect.Top;
        xLeft := (GetSystemMetrics(SM_CXSCREEN) - iWidth) div 2;
        yTop  := (GetSystemMetrics(SM_CYSCREEN) - iHeight) div 2;
      end;
    TRUE:
      begin
        GetWindowRect(hDialog, wRect);
        GetWindowRect(hParent, pRect);
        iWidth  := wRect.Right - wRect.Left;
        iHeight := wRect.Bottom - wRect.Top;
        SystemParametersInfo(SPI_GETWORKAREA, 0, @wArea, 0);
        xLeft := pRect.Left + ((pRect.Right - pRect.Left - iWidth) div 2);
        if (xLeft < 0) then
          xLeft := 0
        else
        if ((xLeft + iWidth) > (wArea.Right - wArea.Left)) then
          xLeft := wArea.Right - wArea.Left - iWidth;
        yTop := pRect.Top + ((pRect.Bottom - pRect.Top - iHeight) div 2);
        if (yTop < 0) then
          yTop := 0
        else
        if ((yTop + iHeight) > (wArea.Bottom - wArea.Top)) then
          yTop := wArea.Bottom - wArea.Top - iHeight;
      end;
  end;
  dwFlags := SWP_NOACTIVATE or SWP_NOSIZE or SWP_NOZORDER;
  Result := SetWindowPos(hDialog, 0, xLeft, yTop, 0, 0, dwFlags);
end;

//

function GetWindowFontSize(hWnd: HWND; pSize: Integer): Integer;
var
  dc: HDC;
begin
  dc := GetDC(hWnd);
  Result := -MulDiv(pSize, GetDeviceCaps(dc, LOGPIXELSY), 72);
  ReleaseDC(hWnd, dc);
end;

//

function GetWindowBoldFont(hWnd: THandle; fntHeight: Integer): HFONT;
var
  lf   : TLogFont;
  dwRes: Integer;
  hfnt : HFONT;
begin
  hfnt := HFONT(SendMessage(hWnd, WM_GETFONT, 0, 0));
  ZeroMemory(@lf, SizeOf(TLogFont));
  if (hfnt <> 0) then
    dwRes := GetObject(hfnt, SizeOf(TLogFont), @lf);
  if (dwRes <> 0) then
  begin
    lf.lfHeight := fntHeight;
    lf.lfWeight := FW_BOLD;
    hfnt := CreateFontIndirect(lf);
  end;
  Result := hfnt;
end;

end.
```

VPN_StatAnim.pas – анімація при підключенні VPN

```
unit VPN_StatAnim;

interface

uses
  Windows, Messages, CommCtrl, VPN_Windows;

procedure CreateAnimateStatic(hWnd: HWND);
procedure RemoveAnimateStatic(hWnd: HWND);

const
  SS_SETIMAGELIST   = WM_USER + 101;
  SS_SETELAPSEDTIME = WM_USER + 102;
  SS_GETIMAGELIST   = WM_USER + 111;
  SS_GETELAPSEDTIME = WM_USER + 112;

implementation

const
  IDC_ANIMATETIMER = 101;

type
  TStatWndProc = function(hWnd: HWND; uMsg: UINT; wParam: WPARAM; lParam:
LPARAM): LRESULT; stdcall;

  P_STAT_PRO = ^T_STAT_PRO;
  T_STAT_PRO = packed record
    StatProc  : TStatWndProc;
    rcClient  : TRect;
    //
    hdcMem    : HDC;
    hbmMem    : HBITMAP;
    hbmOld    : HBITMAP;
    //
    himl      : HIMAGELIST;
    //
    imgSize   : Integer;
    imgCount  : Integer;
    imgCurrent: Integer;
    //
    dwElapse  : Integer;
  end;

var
  psp: P_STAT_PRO;

//

function StatWndProc_OnWmSize(psp: P_STAT_PRO; hWnd: HWND; uMsg: UINT; wParam:
WPARAM; lParam: LPARAM): LRESULT;
var
  hdcIn: HDC;
begin

  //

  GetClientRect(hWnd, psp.rcClient);

  //

  if (psp.hdcMem <> 0) then
    begin
      SelectObject(psp.hdcMem, psp.hbmOld);
      DeleteObject(psp.hbmMem);
      DeleteDC(psp.hdcMem);
    end;
```

```
  //

  hdcIn := GetDC(hWnd);
  psp.hdcMem := CreateCompatibleDC(hdcIn);
  psp.hbmMem := CreateCompatibleBitmap(
    hdcIn,
    psp.rcClient.Right - psp.rcClient.Left,
    psp.rcClient.Bottom - psp.rcClient.Top
  );
  psp.hbmOld := SelectObject(psp.hdcMem, psp.hbmMem);
  ReleaseDC(hWnd, hdcIn);

  //

  Result := CallWindowProc(@psp.StatProc, hWnd, uMsg, wParam, lParam);

  //

  RedrawWindow(hWnd, nil, 0, RDW_INVALIDATE or RDW_UPDATENOW or RDW_NOERASE);

end;

//

function StatWndProc_OnWmPaint(psp: P_STAT_PRO; hWnd: HWND; uMsg: UINT; wParam:
WPARAM; lParam: LPARAM): LRESULT; stdcall;

var
  hdcIn: HDC;
  ps   : TPaintStruct;
begin

  //

  if (wParam = 0) then
    hdcIn := BeginPaint(hWnd, ps)
  else
    hdcIn := wParam;

  //

  CallWindowProc(@psp.StatProc, hWnd, WM_PRINTCLIENT, psp.hdcMem, PRVPN_CLIENT);

  {
  CallWindowProc(@psp.StatProc, hWnd, WM_ERASEBKGND, psp.hdcMem, 0);
  }

  if (psp.himl <> 0) then
    ImageList_DrawEx(
      psp.himl,
      psp.imgCurrent - 1,
      psp.hdcMem,
      psp.rcClient.Left + ((psp.rcClient.Right - psp.rcClient.Left) div 2) -
(psp.imgSize div 2),
      psp.rcClient.Top + ((psp.rcClient.Bottom - psp.rcClient.Top) div 2) -
(psp.imgSize div 2),
      psp.imgSize,
      psp.imgSize,
      CLR_DEFAULT,
      CLR_DEFAULT,
      ILVPN_NORMAL or ILVPN_TRANSPARENT
    );

  BitBlt(
    hdcIn,
    0,
    0,
    psp.rcClient.Right - psp.rcClient.Left,
```

```
    psp.rcClient.Bottom - psp.rcClient.Top,
    psp.hdcMem,
    0,
    0,
    SRCCOPY
  );

  //

  if (wParam = 0) then
    EndPaint(hWnd, ps);

  //

  Result := 0;

end;

//

function StatWndProc_OnWmTimer(psp: P_STAT_PRO; hWnd: HWND; uMsg: UINT; wParam:
WPARAM; lParam: LPARAM): LRESULT;
begin

  //

  Inc(psp.imgCurrent);
  if (psp.imgCurrent > psp.imgCount) then
    psp.imgCurrent := 1;

  //

  RedrawWindow(hWnd, nil, 0, RDW_INVALIDATE or RDW_UPDATENOW or RDW_NOERASE);

  //

  Result := 0;

end;

//

function StatWndProc_OnWmEraseBkgnd(psp: P_STAT_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin

  //

  Result := 1;

end;

//

function StatWndProc_OnSetImageList(psp: P_STAT_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin

  //

  KillTimer(hWnd, IDC_ANIMATETIMER);

  //

  psp.himl := HIMAGELIST(wParam);

  //
```

```
  if (psp.himl <> 0) then
    begin

      ImageList_GetIconSize(psp.himl, psp.imgSize, psp.imgSize);
      psp.imgCount := ImageList_GetImageCount(psp.himl);
      SetTimer(hWnd, IDC_ANIMATETIMER, psp.dwElapse, nil);

    end;

  //

  Result := 0;

end;

//

function StatWndProc_OnSetElapsedTime(psp: P_STAT_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin

  //

  KillTimer(hWnd, IDC_ANIMATETIMER);

  //

  psp.dwElapse := wParam;

  //

  SetTimer(hWnd, IDC_ANIMATETIMER, psp.dwElapse, nil);


  //

  Result := 0;

end;

//

function StatWndProc_OnGetImageList(psp: P_STAT_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin

  //

  Result := LRESULT(psp.himl);

end;

//

function StatWndProc_OnGetElapsedTime(psp: P_STAT_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin

  //

  Result := LRESULT(psp.dwElapse);

end;

//

function StatWndProc(hWnd: HWND; uMsg: UINT; wParam: WPARAM; lParam: LPARAM):
LRESULT; stdcall;
begin
```

```
psp := P_STAT_PRO(GetWindowLong(hWnd, GWL_USERDATA));

if (psp = nil) then
  begin
    Result := DefWindowProc(hWnd, uMsg, wParam, lParam);
    Exit;
  end;

case uMsg of

  //

  WM_DESTROY:
    begin
      RemoveAnimateStatic(hWnd);
    end;

  //

  WM_SIZE:
    begin
      Result := StatWndProc_OnWmSize(psp, hWnd, uMsg, wParam, lParam);
    end;

  //

  WM_PRINTCLIENT,
  WM_PAINT,
  WM_UPDATEUISTATE: // перемальовування вікна без виклику WM_PAINT.
    begin
      Result := StatWndProc_OnWmPaint(psp, hWnd, uMsg, wParam, lParam);
    end;

  //

  WM_TIMER:
    begin
      Result := StatWndProc_OnWmTimer(psp, hWnd, uMsg, wParam, lParam);
    end;

  //

  WM_ERASEBKGND:
    begin
      Result := StatWndProc_OnWmEraseBkgnd(psp, hWnd, uMsg, wParam, lParam);
    end;

  //

  SS_SETIMAGELIST:
    begin
      Result := StatWndProc_OnSetImageList(psp, hWnd, uMsg, wParam, lParam);
    end;

  //

  SS_SETELAPSEDTIME:
    begin
      Result := StatWndProc_OnSetElapsedTime(psp, hWnd, uMsg, wParam, lParam);
    end;

  //

  SS_GETIMAGELIST:
    begin
      Result := StatWndProc_OnGetImageList(psp, hWnd, uMsg, wParam, lParam);
    end;
```

```
    //

    SS_GETELAPSEDTIME:
      begin
        Result := StatWndProc_OnGetElapsedTime(psp, hWnd, uMsg, wParam, lParam);
      end;

    else
      Result := CallWindowProc(@psp.StatProc, hWnd, uMsg, wParam, lParam);
  end;

end;

//

procedure CreateAnimateStatic(hWnd: HWND);
begin

  RemoveAnimateStatic(hWnd);

  psp := P_STAT_PRO(HeapAlloc(GetProcessHeap, HEAP_ZERO_MEMORY,
SizeOf(T_STAT_PRO)));
  ZeroMemory(psp, SizeOf(T_STAT_PRO));

  psp.StatProc   := TStatWndProc(Pointer(GetWindowLong(hWnd, GWL_WNDPROC)));
  psp.himl       := 0;
  psp.imgSize    := 0;
  psp.imgCount   := 0;
  psp.imgCurrent := 0;
  psp.dwElapse   := 50;

  KillTimer(hWnd, IDC_ANIMATETIMER);

  SetWindowLong(hWnd, GWL_USERDATA, Longint(psp));

  SetWindowLong(hWnd, GWL_WNDPROC, Longint(@StatWndProc));

  SendMessage(hWnd, WM_SIZE, 0, 0);

end;

//

procedure RemoveAnimateStatic(hWnd: HWND);
begin

  psp := P_STAT_PRO(GetWindowLong(hWnd, GWL_USERDATA));
  if (psp <> nil) then
    begin

      //

      if (psp.hdcMem <> 0) then
        begin
          SelectObject(psp.hdcMem, psp.hbmOld);
          DeleteObject(psp.hbmMem);
          DeleteDC(psp.hdcMem);
        end;

      //
      KillTimer(hWnd, IDC_ANIMATETIMER);
      //
      SetWindowLong(hWnd, GWL_WNDPROC, Longint(@psp.StatProc));
      RedrawWindow(hWnd, @psp.rcClient, 0, RDW_INVALIDATE or RDW_ERASE);
      SetWindowLong(hWnd, GWL_USERDATA, 0);
      HeapFree(GetProcessHeap, 0, psp);
    end;
end;
end.
```

VPN_ShlObj.pas - інтерпретатор команд операційної системи

```
unit VPN_ShlObj;

interface

uses
  Windows, VPN_Active;

{ Ідентифікація об’єкту у просторі імен(ItemID and IDList) }

const
  // Мережні та Dial-up підключення
  CSIDL_CONNECTIONS = $0031;

{ SHGetSpecialFolderLocation constants }

const
  // Десктоп
  CSIDL_DESKTOP = $0000;

{ Інтерфейс IDs }

const
  IIVPN_IShellFolder: TGUID = (D1: $000214E6; D2: $0000; D3: $0000; D4: ($C0,
$00, $00, $00, $00, $00, $00, $46));
  CLSIVPN_ShellLink : TGUID = (D1: $00021401; D2: $0000; D3: $0000; D4: ($C0,
$00, $00, $00, $00, $00, $00, $46));

{ IShellFolder.GetDisplayNameOf/SetNameOf uFlags }

const
  // по замовчуванню (виключно для дисплею)
  SHGDN_NORMAL       = 0;
  // для перегляду по замовчуванню
  SHCONTVPN_NONFOLDERS = 64;

{ Рядкові константи для інтерфейсу IDs }

const
  SIVPN_IShellLink  = '{000214 0000-0000-C 000-000000000046}';
  SIVPN_IShellLink  = '{000214F 0000-0000-C 000-000000000046}';
  SIVPN_IShellFolder = '{000214E 0000-0000-C 000-000000000046}';
  SIVPN_IEnumIDList  = '{000214F 0000-0000-C 000-000000000046}';

{ IShellLink інтерфейс }

type
  IShellLink = interface(IUnknown)
    [SIVPN_IShellLink]
    function GetPath(pszFile: PAnsiChar; cchMaxPath: Integer; var pfd:
TWin32FindData; fFlags: DWORD): HResult; stdcall;
    function GetIDList(var ppidl: PItemIDList): HResult; stdcall;
    function SetIDList(pidl: PItemIDList): HResult; stdcall;
    function GetDescription(pszName: PAnsiChar; cchMaxName: Integer): HResult;
stdcall;
    function SetDescription(pszName: PAnsiChar): HResult; stdcall;
    function GetWorkingDirectory(pszDir: PAnsiChar; cchMaxPath: Integer):
HResult; stdcall;
    function SetWorkingDirectory(pszDir: PAnsiChar): HResult; stdcall;
    function GetArguments(pszArgs: PAnsiChar; cchMaxPath: Integer): HResult;
stdcall;
    function SetArguments(pszArgs: PAnsiChar): HResult; stdcall;
    function GetHotkey(var pwHotkey: Word): HResult; stdcall;
    function SetHotkey(wHotkey: Word): HResult; stdcall;
    function GetShowCmd(out piShowCmd: Integer): HResult; stdcall;
    function SetShowCmd(iShowCmd: Integer): HResult; stdcall;
```

```
    function GetIconLocation(pszIconPath: PAnsiChar; cchIconPath: Integer; out
piIcon: Integer): HResult; stdcall;
    function SetIconLocation(pszIconPath: PAnsiChar; iIcon: Integer): HResult;
stdcall;
    function SetRelativePath(pszPathRel: PAnsiChar; dwReserved: DWORD): HResult;
stdcall;
    function Resolve(Wnd: HWND; fFlags: DWORD): HResult; stdcall;
    function SetPath(pszFile: PAnsiChar): HResult; stdcall;
  end;

  IShellLink = interface(IUnknown)
    [SIVPN_IShellLink]
    function GetPath(pszFile: PWideChar; cchMaxPath: Integer; var pfd:
TWin32FindData; fFlags: DWORD): HResult; stdcall;
    function GetIDList(var ppidl: PItemIDList): HResult; stdcall;
    function SetIDList(pidl: PItemIDList): HResult; stdcall;
    function GetDescription(pszName: PWideChar; cchMaxName: Integer): HResult;
stdcall;
    function SetDescription(pszName: PWideChar): HResult; stdcall;
    function GetWorkingDirectory(pszDir: PWideChar; cchMaxPath: Integer):
HResult; stdcall;
    function SetWorkingDirectory(pszDir: PWideChar): HResult; stdcall;
    function GetArguments(pszArgs: PWideChar; cchMaxPath: Integer): HResult;
stdcall;
    function SetArguments(pszArgs: PWideChar): HResult; stdcall;
    function GetHotkey(var pwHotkey: Word): HResult; stdcall;
    function SetHotkey(wHotkey: Word): HResult; stdcall;
    function GetShowCmd(out piShowCmd: Integer): HResult; stdcall;
    function SetShowCmd(iShowCmd: Integer): HResult; stdcall;
    function GetIconLocation(pszIconPath: PWideChar; cchIconPath: Integer; out
piIcon: Integer): HResult; stdcall;
    function SetIconLocation(pszIconPath: PWideChar; iIcon: Integer): HResult;
stdcall;
    function SetRelativePath(pszPathRel: PWideChar; dwReserved: DWORD): HResult;
stdcall;
    function Resolve(Wnd: HWND; fFlags: DWORD): HResult; stdcall;
    function SetPath(pszFile: PWideChar): HResult; stdcall;
  end;
  IShellLink = IShellLink;

{ IEnumIDList інтерфейс }

type
  IEnumIDList = interface(IUnknown)
    [SIVPN_IEnumIDList]
    function Next(celt: ULONG; out rgelt: PItemIDList; var pceltFetched: ULONG):
HResult; stdcall;
    function Skip(celt: ULONG): HResult; stdcall;
    function Reset: HResult; stdcall;
    function Clone(out ppenum: IEnumIDList): HResult; stdcall;
  end;

{ record for returning strings from IShellFolder member functions }

type
  PSTRRet = ^TStrRet;
  _STRRET = record
     uType: UINT;                                { одне з значень STRRET_* }
     case Integer of
       0: (pOleStr: LPWSTR);                     { повинно бути вільно для
виклику GetDisplayNameOf }
       1: (pStr: LPSTR);                         { НЕ ВИКОРИСТОВУЄТЬСЯ }
       2: (uOffset: UINT);                       { Зсув у SHITEMID (ANSI) }
       3: (cStr: array[0..MAX_ PATH-1] of Char); { Буфер для заповнювання}
    end;
  TStrRet = _STRRET;
  STRRET = _STRRET;

{ structure STRRET for returning strings from IShellFolder member functions }
```

```
const
  STRRET_WSTR = $0000;
  STRRET_CSTR = $0002;

{ IShellFolder інтерфейс }

type
  IShellFolder = interface(IUnknown)
    [SIVPN_IShellFolder]
    function ParseDisplayName(hwndOwner: HWND; pbcReserved: Pointer;
lpszDisplayName: POLESTR; out pchEaten: ULONG; out ppidl: PItemIDList; var
dwAttributes: ULONG): HResult; stdcall;
    function EnumObjects(hwndOwner: HWND; grfFlags: DWORD; out EnumIDList:
IEnumIDList): HResult; stdcall;
    function BindToObject(pidl: PItemIDList; pbcReserved: Pointer; const riid:
TIID; out ppvOut): HResult; stdcall;
    function BindToStorage(pidl: PItemIDList; pbcReserved: Pointer; const riid:
TIID; out ppvObj): HResult; stdcall;
    function CompareIDs(lParam: LPARAM; pidl1, pidl2: PItemIDList): HResult;
stdcall;
    function CreateViewObject(hwndOwner: HWND; const riid: TIID; out ppvOut):
HResult; stdcall;
    function GetAttributesOf(cidl: UINT; var apidl: PItemIDList; var rgfInOut:
UINT): HResult; stdcall;
    function GetUIObjectOf(hwndOwner: HWND; cidl: UINT; var apidl: PItemIDList;
const riid: TIID; prgfInOut: Pointer; out ppvOut): HResult; stdcall;
    function GetDisplayNameOf(pidl: PItemIDList; uFlags: DWORD; var lpName:
STRRET): HResult; stdcall;
    function SetNameOf(hwndOwner: HWND; pidl: PItemIDList; lpszName: POLEStr;
uFlags: DWORD; var ppidlOut: PItemIDList): HResult; stdcall;
  end;

function SHGetMalloc(var ppMalloc: IMalloc): HResult; stdcall;
function SHGetFolderLocation(hWndOwner: HWND; csidl: Integer; hToken: THandle;
dwReserved: DWORD; var pidl: PItemIDList): HResult; stdcall;
function SHGetDesktopFolder(var ppshf: IShellFolder): HResult; stdcall;
function SHGetSpecialFolderLocation(hwndOwner: HWND; nFolder: Integer; var
ppidl: PItemIDList): HResult; stdcall;
function SHGetPathFromIDList(pidl: PItemIDList; pszPath: PChar): BOOL; stdcall;
function SHGetPathFromIDList(pidl: PItemIDList; pszPath: PAnsiChar): BOOL;
stdcall;
function SHGetPathFromIDList(pidl: PItemIDList; pszPath: PWideChar): BOOL;
stdcall;

implementation

const
  shell32 = 'shell32.dll';

function SHGetMalloc;                external shell32 name 'SHGetMalloc';
function SHGetFolderLocation;        external shell32 name
'SHGetFolderLocation';
function SHGetDesktopFolder;         external shell32 name 'SHGetDesktopFolder';
function SHGetSpecialFolderLocation; external shell32 name
'SHGetSpecialFolderLocation';
function SHGetPathFromIDList;        external shell32 name
'SHGetPathFromIDList';
function SHGetPathFromIDList;       external shell32 name 'SHGetPathFromIDList';
function SHGetPathFromIDList;       external shell32 name 'SHGetPathFromIDList';

end.
```

VPN_Resources.pas – ресурси VPN

```
unit VPN_Resources;

interface

uses
  Windows, CommCtrl, VPN_FileInfo;

const

  { ресурси id діалогу}

  RC_DIALOG_WELCOME   = 101;
  RC_DIALOG_SETTINGS  = 102;
  RC_DIALOG_FINISH    = 103;
  RC_DIALOG_UPDATE    = 104;

  { ресурси id вікна}

  RC_ICONEX_CAPTION   = 101;

  { ресурси id бітової площини}

  RC_BITMAP_WATERMARK = 101;
  RC_BITMAP_HEADER    = 102;
  RC_BITMAP_WAITING   = 103;

  { елементи управління діалога#101 }

  IDC_STATIC_WELCOME  = 10101;

  { елементи управління діалога#102 }

  IDC_STATIC_ENTRY    = 10201;
  IDC_STATIC_SERVER   = 10202;
  IDC_COMBO_SERVER    = 10203;
  IDC_STATIC_USER     = 10204;
  IDC_STATIC_PASSW    = 10205;
  IDC_STATIC_WARN     = 10206;

  { елементи управління діалога#103 }

  IDC_STATIC_FINISH   = 10301;
  IDC_STATIC_VPNINFO  = 10302;
  IDC_CHECK_SHORTCUT  = 10303;

  { елементи управління діалога#104 }

  IDC_STATIC_ANIMATE  = 10401;
  IDC_STATIC_ADDRESS  = 10402;

  { Ресурси рядків таблиць }

  RC_STRING_CWINDOW   = 1600;
  RC_STRING_COPYRUN   = 1601;
  RC_STRING_QCANCEL   = 1602;
  RC_STRING_RESMAN    = 1603;

  RC_STRING_THEADER   = 1616;
  RC_STRING_SHEADER   = 1617;

  RC_STRING_VPNINFO   = 1632;

  RC_STRING_IPSERVER  = 1648;
  RC_STRING_IPHOST    = 1649;
```

```
var
  psp    : TPropSheetPage;
  ahpsp  : Array [0..2] of HPropSheetPage;
  hApp   : Array [0..3] of HWND;
  exeInfo: TStringFileInfo;
  pszServ: WideString = 'server.avtograd.ru';
  hThread: DWORD;

implementation

end.
```

VPN_RasApi.pas - з'єднання VPN

```
  unit VPN_RasApi;

interface

uses
  Windows;

// RASIPADDR структура

type
  PRASIPADDR = ^RASIPADDR;
  RASIPADDR = record
    a: Byte;
    b: Byte;
    c: Byte;
    d: Byte;
  end;

const
  RAS_MaxAreaCode       = 10;
  RAS_MaxPhoneNumber    = 128;
  RAS_MaxDeviceType     = 16;
  RAS_MaxDeviceName     = 128;
  RAS_MaxPadType        = 32;
  RAS_Max25Address     = 200;
  RAS_MaxFacilities     = 200;
  RAS_MaxUserData       = 200;
  RAS_MaxDnsSuffix      = 255;
  RAS_MaxEntryName      = 256;
  RAS_MaxCallbackNumber = RAS_MaxPhoneNumber;
  UNLEN                 = 256; // Максимальна довжина імені користувача
  PWLEN                 = 256; // Максимальна довжина паролю
  CNLEN                 = 15; // Максимальна довжина імені комп'ютера
  DNLEN                 = CNLEN; // Максимальна довжина імені домену

// структура RASCREDENTIALS

type
  RASCREDENTIALSA = record
    dwSize    : DWORD;
    dwMask    : DWORD;
    szUserName: Array [0..UNLEN] of AnsiChar;
    szPassword: Array [0..PWLEN] of AnsiChar;
    szDomain  : Array [0..DNLEN] of AnsiChar;
  end;

  RASCREDENTIALSW = record
    dwSize    : DWORD;
    dwMask    : DWORD;
    szUserName: Array [0..UNLEN] of WideChar;
    szPassword: Array [0..PWLEN] of WideChar;
    szDomain  : Array [0..DNLEN] of WideChar;
  end;

  LPRASCREDENTIALSW = ^RASCREDENTIALSW;
  LPRASCREDENTIALSA = ^RASCREDENTIALSA;
  LPRASCREDENTIALS = ^RASCREDENTIALS;
  RASCREDENTIALS = RASCREDENTIALSA;

const
  // значення RASCREDENTIALS dwMask

  RASCM_UserName = $00000001;
  RASCM_Password = $00000002;
```

```
// структура RASDIALPARAMS

type
  tagRASDIALPARAMSA = record
    dwSize          : DWORD;
    szEntryName     : Array [0..RAS_MaxEntryName] of AnsiChar;
    szPhoneNumber   : Array [0..RAS_MaxPhoneNumber] of AnsiChar;
    szCallbackNumber: Array [0..RAS_MaxCallbackNumber] of AnsiChar;
    szUserName      : Array [0..UNLEN] of AnsiChar;
    szPassword      : Array [0..PWLEN] of AnsiChar;
    szDomain        : Array [0..DNLEN] of AnsiChar;
    // {$IFDEF WINVER_0x401_OR_GREATER}
    dwSubEntry      : DWORD;
    dwCallbackId    : DWORD;
  end;

  tagRASDIALPARAMSW = record
    dwSize          : DWORD;
    szEntryName     : Array [0..RAS_MaxEntryName] of WideChar;
    szPhoneNumber   : Array [0..RAS_MaxPhoneNumber] of WideChar;
    szCallbackNumber: Array [0..RAS_MaxCallbackNumber] of WideChar;
    szUserName      : Array [0..UNLEN] of WideChar;
    szPassword      : Array [0..PWLEN] of WideChar;
    szDomain        : Array [0..DNLEN] of WideChar;
    // {$IFDEF WINVER_0x401_OR_GREATER}
    dwSubEntry      : DWORD;
    dwCallbackId    : DWORD;
  end;

  PRASDIALPARAMSA = ^RASDIALPARAMSA;
  PRASDIALPARAMSW = ^RASDIALPARAMSW;
  PRASDIALPARAMS = PRASDIALPARAMSA;
  tagRASDIALPARAMS = tagRASDIALPARAMSA;
  RASDIALPARAMSA = tagRASDIALPARAMSA;
  RASDIALPARAMSW = tagRASDIALPARAMSW;
  RASDIALPARAMS = RASDIALPARAMSA;

// структура RASENTRY

type
  tagRASENTRYA = record
    dwSize                  : DWORD;
    dwfOptions              : DWORD;

    // Настроювання мережевого номера
    dwCountryID             : DWORD;
    dwCountryCode           : DWORD;
    szAreaCode              : Array [0..RAS_MaxAreaCode] of AnsiChar;
    szLocalPhoneNumber      : Array [0..RAS_MaxPhoneNumber] of AnsiChar;
    dwAlternateOffset       : DWORD;

    // PPP(Протокол Point-to-point)/Ip
    ipaddr                  : RASIPADDR;
    ipaddrDns               : RASIPADDR;
    ipaddrDnsAlt            : RASIPADDR;
    ipaddrWins              : RASIPADDR;
    ipaddrWinsAlt           : RASIPADDR;

    // Протокол
    dwFrameSize             : DWORD;
    dwfNetProtocols         : DWORD;
    dwFramingProtocol       : DWORD;

    // Сценарії
    szScript                : Array [0..MAX_ PATH-1] of AnsiChar;

    // Автодозвон
    szAutodialDll           : Array [0..MAX_ PATH-1] of AnsiChar;
    szAutodialFunc          : Array [0..MAX_ PATH-1] of AnsiChar;
```

```
    // Пристрій
    szDeviceType              : Array [0..RAS_MaxDeviceType] of AnsiChar;
    szDeviceName              : Array [0..RAS_MaxDeviceName]of AnsiChar;

    // X.25
    sz25PadType              : Array [0..RAS_MaxPadType] of AnsiChar;
    sz25Address              : Array [0..RAS_Max25Address] of AnsiChar;
    sz25Facilities           : Array [0..RAS_MaxFacilities] of AnsiChar;
    sz25UserData             : Array [0..RAS_MaxUserData] of AnsiChar;
    dwChannels                : DWORD;

    // Зарезервовано
    dwReserved1               : DWORD;
    dwReserved2               : DWORD;
    //  {$IFDEF WINVER_0x401_OR_GREATER}

    // Підключення з багатьох з'єднань
    dwSubEntries              : DWORD;
    dwDialMode                : DWORD;
    dwDialExtraPercent        : DWORD;
    dwDialExtraSampleSeconds  : DWORD;
    dwHangUpExtraPercent      : DWORD;
    dwHangUpExtraSampleSeconds: DWORD;

    // Час простою до роз'єднання
    dwIdleDisconnectSeconds   : DWORD;
    // {$IFDEF WINVER_0x500_OR_GREATER}
    dwType                    : DWORD;
    dwEncryptionType          : DWORD;
    dwCustomAuthKey           : DWORD;
    guidId                    : TGUID;
    szCustomDialDll           : Array [0..MAX_ PATH-1] of AnsiChar;
    dwVpnStrategy             : DWORD;
    //  {$IFDEF WINVER_0x501_OR_GREATER}
    dwfOptions2               : DWORD;
    dwfOptions3               : DWORD;
    szDnsSuffix               : Array [0..RAS_MaxDnsSuffix] of AnsiChar;
    dwTcpWindowSize           : DWORD;
    szPrerequisitePbk         : Array [0..MAX_ PATH-1] of AnsiChar;
    szPrerequisiteEntry       : Array [0..RAS_MaxEntryName] of AnsiChar;
    dwRedialCount             : DWORD;
    dwRedialPause             : DWORD;
    //  {$IFDEF WINVER_0x600_OR_GREATER}
    //    ipv6addrDns   : RASIPV6ADDR;
    //    ipv6addrDnsAlt: RASIPV6ADDR;
    //  {$ENDIF}
    //    dwIPv4InterfaceMetric: DWORD;
    //    dwIPv6InterfaceMetric: DWORD;
  end;

  tagRASENTRYW = record
    dwSize                    : DWORD;
    dwfOptions                : DWORD;

    // Настроювання мережного номера
    dwCountryID               : DWORD;
    dwCountryCode             : DWORD;
    szAreaCode                : Array [0..RAS_MaxAreaCode] of WideChar;
    szLocalPhoneNumber        : Array [0..RAS_MaxPhoneNumber] of WideChar;
    dwAlternateOffset         : DWORD;

    // PPP(Протокол Point-to-point)/Ip
    ipaddr                    : RASIPADDR;
    ipaddrDns                 : RASIPADDR;
    ipaddrDnsAlt              : RASIPADDR;
    ipaddrWins                : RASIPADDR;
    ipaddrWinsAlt             : RASIPADDR;
```

```
    // Протокол
    dwFrameSize               : DWORD;
    dwfNetProtocols           : DWORD;
    dwFramingProtocol         : DWORD;

    // Сценарії
    szScript                  : Array [0..MAX_ PATH-1] of WideChar;

    // Автодозвон
    szAutodialDll             : Array [0..MAX_ PATH-1] of WideChar;
    szAutodialFunc            : Array [0..MAX_ PATH-1] of WideChar;

    // Пристрій
    szDeviceType              : Array [0..RAS_MaxDeviceType] of WideChar;
    szDeviceName              : Array [0..RAS_MaxDeviceName]of WideChar;

    // X.25
    sz25PadType              : Array [0..RAS_MaxPadType] of WideChar;
    sz25Address              : Array [0..RAS_Max25Address] of WideChar;
    sz25Facilities           : Array [0..RAS_MaxFacilities] of WideChar;
    sz25UserData             : Array [0..RAS_MaxUserData] of WideChar;
    dwChannels                : DWORD;

    // Зарезервовано
    dwReserved1               : DWORD;
    dwReserved2               : DWORD;
    //  {$IFDEF WINVER_0x401_OR_GREATER}

    // Підключення з багатьох з'єднань
    dwSubEntries              : DWORD;
    dwDialMode                : DWORD;
    dwDialExtraPercent        : DWORD;
    dwDialExtraSampleSeconds  : DWORD;
    dwHangUpExtraPercent      : DWORD;
    dwHangUpExtraSampleSeconds: DWORD;

    // Час простою до роз'єднання
    dwIdleDisconnectSeconds   : DWORD;
    // {$IFDEF WINVER_0x500_OR_GREATER}
    dwType                    : DWORD;
    dwEncryptionType          : DWORD;
    dwCustomAuthKey           : DWORD;
    guidId                    : TGUID;
    szCustomDialDll           : Array [0..MAX_ PATH-1] of WideChar;
    dwVpnStrategy             : DWORD;
    //  {$IFDEF WINVER_0x501_OR_GREATER}
    dwfOptions2               : DWORD;
    dwfOptions3               : DWORD;
    szDnsSuffix               : Array [0..RAS_MaxDnsSuffix] of WideChar;
    dwTcpWindowSize           : DWORD;
    szPrerequisitePbk         : Array [0..MAX_ PATH-1] of WideChar;
    szPrerequisiteEntry       : Array [0..RAS_MaxEntryName] of WideChar;
    dwRedialCount             : DWORD;
    dwRedialPause             : DWORD;
    //  {$IFDEF WINVER_0x600_OR_GREATER}
    //    ipv6addrDns   : RASIPV6ADDR;
    //    ipv6addrDnsAlt: RASIPV6ADDR;
    //  {$ENDIF}
    //    dwIPv4InterfaceMetric: DWORD;
    //    dwIPv6InterfaceMetric: DWORD;
  end;

  tagRASENTRY = tagRASENTRYA;
  RASENTRYA = tagRASENTRYA;
  RASENTRYW = tagRASENTRYW;
  RASENTRY = RASENTRYA;

const
// RASENTRY dwfOptions bit flags
```

```
  RASEO_RemoteDefaultGateway      = $00000010;
  RASEO_ModemLights               = $00000100;
  RASEO_RequireEncryptedPw        = $00000400;
  RASEO_RequireMsEncryptedPw      = $00000800;
  RASEO_RequireDataEncryption     = $00001000;
  RASEO_PreviewUserPw             = $01000000;
  RASEO_ShowDialingProgress       = $04000000;

// Біти прапорів RASENTRY dwfOptions

  RASEO2_DontNegotiateMultilink   = $00000004;
  RASEO2_ReconnectIfDropped       = $00000100;

// Біти прапорів RASENTRY dwProtocols

  RASNP_Ip = $00000004;

// Біти прапорів RASENTRY dwFramingProtocols

  RASFP_Ppp = $00000001;

// константи RASENTRY dwIdleDisconnectSeconds

  RASIDS_Disabled = $FFFFFFFF;

// рядок по замовчуванню RASENTRY szDeviceType

  RASDT_Vpn = 'vpn';

// значення RASENTRY dwDialMode

  RASEDM_DialAll = 1;

// Тип входу використаний, для визначення того, які властивості UI повині бути
представлені споживачу

  RASET_Vpn = 2; // VPN
// Немає ніякої різниці між RASCTRYINFOA та RASCTRYINFOW.

  ET_None    = 0; // Без шифрування
  VS_Default = 0; // по замовчуванню (PPTP)

function RasSetEntryProperties(lpszPhonebook, szEntry: PAnsiChar; lpbEntry:
Pointer; dwEntrySize: Longint; lpbDeviceInfo: Pointer; dwDeviceInfoSize:
Longint): Longint; stdcall;
function RasSetEntryProperties(lpszPhonebook, szEntry: PWideChar; lpbEntry:
Pointer; dwEntrySize: Longint; lpbDeviceInfo: Pointer; dwDeviceInfoSize:
Longint): Longint; stdcall;
function RasSetEntryProperties(lpszPhonebook, szEntry: PAnsiChar; lpbEntry:
Pointer; dwEntrySize: Longint; lpbDeviceInfo: Pointer; dwDeviceInfoSize:
Longint): Longint; stdcall;

function RasGetEntryProperties(lpszPhonebook, szEntry: PAnsiChar; lpbEntry:
Pointer; var lpdwEntrySize: Longint; lpbDeviceInfo: Pointer; var
lpdwDeviceInfoSize: Longint): Longint; stdcall;
function RasGetEntryProperties(lpszPhonebook, szEntry: PWideChar; lpbEntry:
Pointer; var lpdwEntrySize: Longint; lpbDeviceInfo: Pointer; var
lpdwDeviceInfoSize: Longint): Longint; stdcall;
function RasGetEntryProperties(lpszPhonebook, szEntry: PAnsiChar; lpbEntry:
Pointer; var lpdwEntrySize: Longint; lpbDeviceInfo: Pointer; var
lpdwDeviceInfoSize: Longint): Longint; stdcall;

function RasSetEntryDialParams(lpszPhonebook: PAnsiChar; lprasdialparams:
PRASDIALPARAMSA; fRemovePassword: BOOL): DWORD; stdcall;
function RasSetEntryDialParams(lpszPhonebook: PWideChar; lprasdialparams:
PRASDIALPARAMSW; fRemovePassword: BOOL): DWORD; stdcall;
function RasSetEntryDialParams(lpszPhonebook: PAnsiChar; lprasdialparams:
PRASDIALPARAMS; fRemovePassword: BOOL): DWORD; stdcall;
```

```
function RasSetCredentials(lpszPhoneBook, lpszEntry: PAnsiChar; var
lpCredentials: RASCREDENTIALSA; fRemovePassword: LongBool): Longint; stdcall;
function RasSetCredentials(lpszPhoneBook, lpszEntry: PWideChar; var
lpCredentials: RASCREDENTIALSW; fRemovePassword: LongBool): Longint; stdcall;
function RasSetCredentials(lpszPhoneBook, lpszEntry: PAnsiChar; var
lpCredentials: RASCREDENTIALS; fRemovePassword: LongBool): Longint; stdcall;

implementation

const
  raslib = 'rasapi32.dll';

function RasSetEntryProperties; external raslib name 'RasSetEntryProperties';
function RasSetEntryProperties; external raslib name 'RasSetEntryProperties';
function RasSetEntryProperties;  external raslib name 'RasSetEntryProperties';

function RasGetEntryProperties; external raslib name 'RasGetEntryProperties';
function RasGetEntryProperties; external raslib name 'RasGetEntryProperties';
function RasGetEntryProperties;  external raslib name 'RasGetEntryProperties';

function RasSetEntryDialParams; external raslib name 'RasSetEntryDialParams';
function RasSetEntryDialParams; external raslib name 'RasSetEntryDialParams';
function RasSetEntryDialParams;  external raslib name 'RasSetEntryDialParams';

function RasSetCredentials;     external raslib name 'RasSetCredentials';
function RasSetCredentials;     external raslib name 'RasSetCredentials';
function RasSetCredentials;      external raslib name 'RasSetCredentials';
end.
```

VPN_MyMsgBox.pas – повідомлення VPN

```
  unit VPN_MyMsgBox;

interface

uses
  Windows, Messages, VPN_SysUtils;

function ExtMessageBox(hWnd: HWND; pszText, pszCaption: PWideChar; dwFlags:
DWORD): Integer;

implementation

var
  hhk: HHOOK;
  ico: HICON;

//

function SysMsgProc(nCode: UINT; wParam: WPARAM; lParam: LPARAM): Integer;
stdcall;
begin
  case nCode of
    HCBT_ACTIVATE:
      begin
        if (ico <> 0) then
          SendMessage(wParam, WM_SETICON, ICON_SMALL, ico);
        SetCenterDialogPos(wParam, GetParent(wParam), TRUE);
        UnhookWindowsHookEx(hhk);
        Result := 0;
      end;
    else
      Result := CallNextHookEx(hhk, nCode, wParam, lParam);
  end;
end;

//

function ExtMessageBox(hWnd: HWND; pszText, pszCaption: PWideChar; dwFlags:
DWORD): Integer;
begin
  ico := GetClassLong(hWnd, GCL_HICON);
  if (ico = 0) then
    ico := SendMessage(hWnd, WM_GETICON, ICON_SMALL, 0);
  hhk := SetWindowsHookEx(WH_CBT, @SysMsgProc, hInstance, 0);
  Result := MessageBox(hWnd, pszText, pszCaption, dwFlags);
end;

end.
```

VPN_LinkStat.pas - інтерфейс користувача

```
unit VPN_LinkStat;

interface

uses
  Windows, Messages, CommCtrl, VPN_Windows;

const
  //
  SCM_EX_SETHOVERCLR  = WM_USER + 101; // установити колір для наведеного стану.
  SCM_EX_SETNORMALCLR = WM_USER + 102; // установити колір для звичайного стану.
  SCM_EX_SETPRESSCLR  = WM_USER + 103; // установити колір для натиснутого
стану.
  SCM_EX_SETBCKGNDCLR = WM_USER + 104; // установити колір для тла тексту.
  SCM_EX_SETTIPTEXT   = WM_USER + 105; // установити текст спливаючої підказки.
  //
  SCM_EX_GETHOVERCLR  = WM_USER + 111; // одержати колір для наведеного стану.
  SCM_EX_GETNORMALCLR = WM_USER + 112; // одержати колір для звичайного стану.
  SCM_EX_GETPRESSCLR  = WM_USER + 113; // одержати колір для натиснутого стану.
  SCM_EX_GETBCKGNDCLR = WM_USER + 114; // одержати колір для тла тексту.
  SCM_EX_GETTIPTEXT   = WM_USER + 115; // одержати текст спливаючої підказки.

// створення елемента управління Hyperlink.

procedure CreateStaticHyperlink(hWnd: HWND);

// видалення елемента управління Hyperlink.

procedure RemoveStaticHyperlink(hWnd: HWND);

implementation

type
  TLinkWndProc = function(hWnd: HWND; uMsg: UINT; wParam: WPARAM; lParam:
LPARAM): LRESULT; stdcall;

  P_LINK_PRO = ^T_LINK_PRO;
  T_LINK_PRO = packed record
    LinkProc  : TLinkWndProc;
    hCursor   : HCURSOR;
    hFont     : HFONT;
    rcClient  : TRect;
    //
    clrHover  : TColorRef;
    clrNormal : TColorRef;
    clrPress  : TColorRef;
    clrBckgnd : TColorRef; // CLR_NONE
    pszText   : Array [0..MAX_ PATH-1] of WideChar;
    //
    bIsHover  : Boolean;
    bIsPress  : Boolean;
    bIsEnabled: Boolean;
    //
    hToolTip  : HWND;
    ti        : TToolInfo;
    pszToolTip: Array [0..MAX_ PATH-1] of WideChar;
    //
    dtStyle   : DWORD;
    //
    hdcMem    : HDC;
    hbmMem    : HBITMAP;
    hbmOld    : HBITMAP;
  end;

var
  plp: P_LINK_PRO;
```

```
//

function LinkWndProc_OnSetHoverClr(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin
  plp.clrHover := TColorRef(wParam);
  RedrawWindow(hWnd, nil, 0, RDW_INVALIDATE or RDW_UPDATENOW or RDW_NOERASE);
  //
  Result := 0;
end;

//

function LinkWndProc_OnGetHoverClr(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin
  //
  Result := LRESULT(plp.clrHover);
end;

//

function LinkWndProc_OnSetNormalClr(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin
  plp.clrNormal := TColorRef(wParam);
  RedrawWindow(hWnd, nil, 0, RDW_INVALIDATE or RDW_UPDATENOW or RDW_NOERASE);
  //
  Result := 0;
end;

//

function LinkWndProc_OnGetNormalClr(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin
  //
  Result := LRESULT(plp.clrNormal);
end;

//

function LinkWndProc_OnSetPressClr(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin
  plp.clrPress := TColorRef(wParam);
  RedrawWindow(hWnd, nil, 0, RDW_INVALIDATE or RDW_UPDATENOW or RDW_NOERASE);
  //
  Result := 0;
end;

//

function LinkWndProc_OnGetPressClr(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin
  //
  Result := LRESULT(plp.clrPress);
end;

//

function LinkWndProc_OnSetBckgdClr(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin
  plp.clrBckgnd := TColorRef(wParam);
  RedrawWindow(hWnd, nil, 0, RDW_INVALIDATE or RDW_UPDATENOW or RDW_NOERASE);
  //
```

```
  Result := 0;
end;

//

function LinkWndProc_OnGetBckgdClr(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin
  //
  Result := LRESULT(plp.clrBckgnd);
end;

//

function LinkWndProc_OnSetTipText(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin
  lstrcpyn(plp.pszToolTip, PWideChar(wParam), wParam);
  //
  Result := 0;
end;

//

function LinkWndProc_OnGetTipText(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin
  //
  lstrcpyn(PWideChar(lParam), plp.pszToolTip, lstrlen(plp.pszToolTip) + 1);
  //
  Result := 0;
end;

//

function LinkWndProc_OnWmSetFont(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin
  plp.hFont := HFONT(wParam);
  //
  Result := CallWindowProc(@plp.LinkProc, hWnd, uMsg, wParam, lParam);
  //
  RedrawWindow(hWnd, nil, 0, RDW_INVALIDATE or RDW_UPDATENOW or RDW_NOERASE);
end;

//

function LinkWndProc_OnWmSetText(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin
  ZeroMemory(@plp.pszText, SizeOf(plp.pszText));
  lstrcpyn(plp.pszText, PWideChar(lParam), lParam);
  RedrawWindow(hWnd, nil, 0, RDW_INVALIDATE or RDW_UPDATENOW or RDW_NOERASE);
  Result := DefWindowProc(hWnd, uMsg, wParam, lParam);
end;

//

function LinkWndProc_OnWmEnable(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT; wParam:
WPARAM; lParam: LPARAM): LRESULT;
begin
  plp.bIsEnabled := BOOL(wParam);
  RedrawWindow(hWnd, nil, 0, RDW_INVALIDATE or RDW_UPDATENOW or RDW_NOERASE);
  //
  Result := 0;
end;

//
```

```
function LinkWndProc_OnWmMouseLeave(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
var
  pt: TPoint;
begin
  if IsWindow(plp.hToolTip) then
    SendMessage(plp.hToolTip, TTM_TRACKACTIVATE, Integer(FALSE), 0);
  //
  GetCursorPos(pt);
  ScreenToClient(hWnd, pt);
  //
  plp.bIsHover := FALSE;
  plp.bIsPress := FALSE;
  //
  RedrawWindow(hWnd, nil, 0, RDW_INVALIDATE or RDW_UPDATENOW or RDW_NOERASE);
  //
  Result := 0;
end;

//

function LinkWndProc_OnWmMouseMove(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
var
  tme: Windows.TTrackMouseEvent;
  pt : TPoint;
begin
  //
  GetCursorPos(pt);
  ScreenToClient(hWnd, pt);
  //
  tme.cbSize      := SizeOf(Windows.TTrackMouseEvent);
  tme.dwFlags     := TME_LEAVE;
  tme.hwndTrack   := hWnd;
  tme.dwHoverTime := HOVER_DEFAULT;
  //
  plp.bIsHover := Windows.TrackMouseEvent(tme) and PtInRect(plp.rcClient, pt);
  plp.bIsPress := {(wParam = MK_LBUTTON) and} (GetCapture = hWnd) and
PtInRect(plp.rcClient, pt);
  //
  RedrawWindow(hWnd, nil, 0, RDW_INVALIDATE or RDW_UPDATENOW or RDW_NOERASE);
  //
  Result := 0;
end;

//

function LinkWndProc_OnWmCaptureChanged(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin
  plp.bIsPress := FALSE;
  //
  Result := 0;
end;

//

function LinkWndProc_OnWmNcHitTest(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin
  //
  Result := HTCLIENT;
end;

//

function LinkWndProc_OnWmlButtonDown(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin
```

```
  //
  if IsWindow(plp.hToolTip) then
    SendMessage(plp.hToolTip, TTM_TRACKACTIVATE, Integer(FALSE), 0);
  plp.bIsPress := TRUE;
  SetFocus(hWnd);
  SetCapture(hWnd);
  //
  RedrawWindow(hWnd, nil, 0, RDW_INVALIDATE or RDW_UPDATENOW or RDW_NOERASE);
  //
  Result := 0;
end;

//

function LinkWndProc_OnWmlButtonUp(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
var
  pt: TPoint;
begin
  //
  GetCursorPos(pt);
  ScreenToClient(hWnd, pt);
  if (PtInRect(plp.rcClient, pt) and (GetCapture = hWnd)) then
    SendMessage(GetParent(hWnd), WM_COMMAND, MakeLong(GetDlgCtrlID(hWnd),
STN_CLICKED), 0);
  // plp.bIsPress := FALSE;
  ReleaseCapture;
  //
  RedrawWindow(hWnd, nil, 0, RDW_INVALIDATE or RDW_UPDATENOW or RDW_NOERASE);
  //
  Result := 0;
end;

//

function LinkWndProc_OnWmSetCursor(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
var
  pt: TPoint;
begin
  //
  if IsWindow(plp.hToolTip) then
    begin
      SendMessage(plp.hToolTip, TTM_TRACKACTIVATE, Integer(TRUE),
Integer(@plp.ti));
      GetCursorPos(pt);
      SendMessage(plp.hToolTip, TTM_TRACKPOSITION, 0, MakeLong(pt.x, pt.y));
    end;
  //
  if (plp.hCursor <> 0) then
   SetCursor(plp.hCursor);
  //
  Result := 0;
end;

//

function LinkWndProc_OnWmSize(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT; wParam:
WPARAM; lParam: LPARAM): LRESULT;
var
  hdcIn: HDC;
begin
  GetClientRect(hWnd, plp.rcClient);
  //
  if (plp.hdcMem <> 0) then
    begin
      SelectObject(plp.hdcMem, plp.hbmOld);
      DeleteObject(plp.hbmMem);
      DeleteDC(plp.hdcMem);
```

```
    end;
  hdcIn := GetDC(hWnd);
  plp.hdcMem := CreateCompatibleDC(hdcIn);
  plp.hbmMem := CreateCompatibleBitmap(hdcIn, plp.rcClient.Right -
plp.rcClient.Left, plp.rcClient.Bottom - plp.rcClient.Top);
  plp.hbmOld := SelectObject(plp.hdcMem, plp.hbmMem);
  ReleaseDC(hWnd, hdcIn);
  //
  RedrawWindow(hWnd, nil, 0, RDW_INVALIDATE or RDW_UPDATENOW or RDW_NOERASE);
  //
  Result := CallWindowProc(@plp.LinkProc, hWnd, uMsg, wParam, lParam);
end;

//

function LinkWndProc_OnWmPaint(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT; wParam:
WPARAM; lParam: LPARAM): LRESULT; stdcall;
var
  hdcIn : HDC;
  ps    : TPaintStruct;
  hbrNew: HBRUSH;
begin
  if (wParam = 0) then
    hdcIn := BeginPaint(hWnd, ps)
  else
    hdcIn := wParam;

  if (plp.clrBckgnd = CLR_DEFAULT) then
    FillRect(plp.hdcMem, plp.rcClient, HBRUSH(COLOR_BTNFACE + 1))
  else
    begin
      hbrNew := CreateSolidBrush(plp.clrBckgnd);
      FillRect(plp.hdcMem, plp.rcClient, hbrNew);
      DeleteObject(hbrNew);
    end;

  if plp.bIsEnabled then
    begin
      if (plp.bIsHover and plp.bIsPress) then
        SetTextColor(plp.hdcMem, plp.clrPress)
      else
      if (plp.bIsHover and not plp.bIsPress) then
        SetTextColor(plp.hdcMem, plp.clrHover)
      else
        SetTextColor(plp.hdcMem, plp.clrNormal);
    end
  else
    SetTextColor(plp.hdcMem, GetSysColor(COLOR_GRAYTEXT));

  SetBkMode(plp.hdcMem, TRANSPARENT);
  SetBkColor(plp.hdcMem, TRANSPARENT);

  SelectObject(plp.hdcMem, plp.hFont);

  DrawText(plp.hdcMem, plp.pszText, {lstrlen(plp.pszText)}-1, plp.rcClient,
plp.dtStyle);

  BitBlt(hdcIn, 0, 0, plp.rcClient.Right - plp.rcClient.Left,
plp.rcClient.Bottom - plp.rcClient.Top, plp.hdcMem, 0, 0, SRCCOPY);

  if (wParam = 0) then
    EndPaint(hWnd, ps);

  Result := 0;
end;

//
```

```
function LinkWndProc_OnWmEraseBkgnd(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin
  if (plp.clrBckgnd <> CLR_DEFAULT) then
   begin
     FillRect(HDC(wParam), plp.rcClient, HBRUSH(COLOR_BTNFACE + 1));
     //
     Result := 1;
   end
  else
    Result := DefWindowProc(hWnd, uMsg, wParam, lParam);
end;

//

function LinkWndProc_OnWmSysColorChange(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin
  RedrawWindow(hWnd, nil, 0, RDW_INVALIDATE or RDW_UPDATENOW or RDW_NOERASE);
  //
  Result := 0;
end;

//

function LinkWndProc_OnWmNotify(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT; wParam:
WPARAM; lParam: LPARAM): LRESULT;
var
  pnmh: PNMHdr;
  ptit: PToolTipText;
begin
  //
  pnmh := PNMHdr(lParam);
  case pnmh.code of
    TTN_NEEDTEXTW:
      begin
        ptit := PToolTipText(lParam);
        ptit.lpszText := plp.pszToolTip;
      end;
  end;
  //
  Result := 0;
end;

//

function LinkWndProc(hWnd: HWND; uMsg: UINT; wParam: WPARAM; lParam: LPARAM):
LRESULT; stdcall;
begin

  plp := P_LINK_PRO(GetWindowLong(hWnd, GWL_USERDATA));

  if (plp = nil) then
    begin
      Result := DefWindowProc(hWnd, uMsg, wParam, lParam);
      Exit;
    end;

  case uMsg of

    //

    SCM_EX_SETHOVERCLR:
      begin
        Result := LinkWndProc_OnSetHoverClr(plp, hWnd, uMsg, wParam, lParam);
      end;

    //
```

```
SCM_EX_GETHOVERCLR:
  begin
    Result := LinkWndProc_OnGetHoverClr(plp, hWnd, uMsg, wParam, lParam);
  end;

//

SCM_EX_SETNORMALCLR:
  begin
    Result := LinkWndProc_OnSetNormalClr(plp, hWnd, uMsg, wParam, lParam);
  end;

//

SCM_EX_GETNORMALCLR:
  begin
    Result := LinkWndProc_OnGetNormalClr(plp, hWnd, uMsg, wParam, lParam);
  end;

//

SCM_EX_SETPRESSCLR:
  begin
    Result := LinkWndProc_OnSetPressClr(plp, hWnd, uMsg, wParam, lParam);
  end;

//

SCM_EX_GETPRESSCLR:
  begin
    Result := LinkWndProc_OnGetPressClr(plp, hWnd, uMsg, wParam, lParam);
  end;

//

SCM_EX_SETBCKGNDCLR:
  begin
    Result := LinkWndProc_OnSetBckgdClr(plp, hWnd, uMsg, wParam, lParam);
  end;

//

SCM_EX_GETBCKGNDCLR:
  begin
    Result := LinkWndProc_OnGetBckgdClr(plp, hWnd, uMsg, wParam, lParam);
  end;

//

SCM_EX_SETTIPTEXT:
  begin
    Result := LinkWndProc_OnSetTipText(plp, hWnd, uMsg, wParam, lParam);
  end;

//

SCM_EX_GETTIPTEXT:
  begin
    Result := LinkWndProc_OnGetTipText(plp, hWnd, uMsg, wParam, lParam);
  end;

//

WM_DESTROY:
  begin
    RemoveStaticHyperlink(hWnd);
  end;

//
```

```
    WM_SETFONT:
      begin
        Result := LinkWndProc_OnWmSetFont(plp, hWnd, uMsg, wParam, lParam);
      end;

    //

    WM_SETTEXT:
      begin
        Result := LinkWndProc_OnWmSetText(plp, hWnd, uMsg, wParam, lParam);
      end;

    //

    WM_ENABLE:
      begin
        Result := LinkWndProc_OnWmEnable(plp, hWnd, uMsg, wParam, lParam);
      end;

    //

    WM_MOUSELEAVE:
      begin
        Result := LinkWndProc_OnWmMouseLeave(plp, hWnd, uMsg, wParam, lParam);
      end;

    //

    WM_MOUSEMOVE:
      begin
        Result := LinkWndProc_OnWmMouseMove(plp, hWnd, uMsg, wParam, lParam);
      end;

    //

    WM_CAPTURECHANGED:
      begin
        Result := LinkWndProc_OnWmCaptureChanged(plp, hWnd, uMsg, wParam,
lParam);
      end;

    //

    WM_NCHITTEST:
      begin
        Result := LinkWndProc_OnWmNcHitTest(plp, hWnd, uMsg, wParam, lParam);
      end;

    //

    WM_LBUTTONDOWN:
      begin
        Result := LinkWndProc_OnWmlButtonDown(plp, hWnd, uMsg, wParam, lParam);
      end;

    //

    WM_LBUTTONUP:
      begin
        Result := LinkWndProc_OnWmlButtonUp(plp, hWnd, uMsg, wParam, lParam);
      end;

    //

    WM_SETCURSOR:
      begin
        Result := LinkWndProc_OnWmSetCursor(plp, hWnd, uMsg, wParam, lParam);
      end;
```

```
    //

    WM_SIZE:
      begin
        Result := LinkWndProc_OnWmSize(plp, hWnd, uMsg, wParam, lParam);
      end;

    //

    WM_PRINTCLIENT,
    WM_PAINT,
    WM_UPDATEUISTATE: // перемальовування вікна без виклику WM_PAINT.
      begin
        Result := LinkWndProc_OnWmPaint(plp, hWnd, uMsg, wParam, lParam);
      end;

    //

    WM_ERASEBKGND:
      begin
        Result := LinkWndProc_OnWmEraseBkgnd(plp, hWnd, uMsg, wParam, lParam);
      end;

    //

    WM_SYSCOLORCHANGE:
      begin
        Result := LinkWndProc_OnWmSysColorChange(plp, hWnd, uMsg, wParam,
lParam);
      end;

    //

    WM_NOTIFY:
      begin
        Result := LinkWndProc_OnWmNotify(plp, hWnd, uMsg, wParam, lParam);
      end;

    else
      Result := CallWindowProc(@plp.LinkProc, hWnd, uMsg, wParam, lParam);
  end;

end;

//

procedure CreateStaticHyperlink(hWnd: HWND);
var
  iccex  : TInitCommonControlsEx;
  dtStyle: DWORD;
  dwLen  : Integer;
begin

  InitCommonControls;
  iccex.dwSize := SizeOf(TInitCommonControlsEx);
  iccex.dwICC  := ICC_BAR_CLASSES;
  InitCommonControlsEx(iccex);

  RemoveStaticHyperlink(hWnd);

  plp := P_LINK_PRO(HeapAlloc(GetProcessHeap, HEAP_ZERO_MEMORY,
SizeOf(T_LINK_PRO)));

  ZeroMemory(plp, SizeOf(plp));
  plp.LinkProc   := TLinkWndProc(Pointer(GetWindowLong(hWnd, GWL_WNDPROC)));
  plp.hCursor    := LoadImage(0, MAKEINTRESOURCEW(IDC_HAND), IMAGE_CURSOR, 0, 0,
LR_SHARED or LR_DEFAULTSIZE);
```

```
  plp.hFont      := SendMessage(hWnd, WM_GETFONT, 0, 0);

  GetClientRect(hWnd, plp.rcClient);

  plp.clrHover   := RGB(255, 0, 0);
  plp.clrNormal  := RGB(0, 0, 255);
  plp.clrPress   := RGB(0, 0, 128);
  plp.clrBckgnd  := CLR_DEFAULT;

  dwLen := SendMessage(hWnd, WM_GETTEXTLENGTH, 0, 0);
  if (dwLen > 0) then
    begin
      ZeroMemory(@plp.pszText, SizeOf(plp.pszText));
      SendMessage(hWnd, WM_GETTEXT, SizeOf(plp.pszText), Integer(@plp.pszText));
    end;

  plp.bIsHover   := FALSE;
  plp.bIsPress   := FALSE;
  plp.bIsEnabled := IsWindowEnabled(hWnd);

  plp.hToolTip   := CreateWindowEx(WS_EX_TOPMOST, TOOLTIPS_CLASS, nil, WS_POPUP
or TTS_NOPREFIX or TTS_ALWAYSTIP, Integer(CW_USEDEFAULT),
Integer(CW_USEDEFAULT), Integer(CW_USEDEFAULT), Integer(CW_USEDEFAULT),
GetParent(hWnd), 0, hInstance, nil);
  if IsWindow(plp.hToolTip) then
    begin
      plp.ti.cbSize   := SizeOf(TToolInfo);
      plp.ti.uFlags   := TTVPN_SUBCLASS or TTVPN_IDISHWND;
      plp.ti.hwnd     := hWnd;
      plp.ti.uId      := hWnd;
      plp.ti.lpszText := LPSTR_TEXTCALLBACKW;
      SetRectEmpty(plp.ti.Rect);
      ZeroMemory(@plp.pszToolTip, SizeOf(plp.pszToolTip));
      SendMessage(plp.hToolTip, TTM_ADDTOOLW, 0, Integer(@plp.ti));
    end;

  dtStyle := GetWindowLong(hWnd, GWL_STYLE);

  case (dtStyle and SS_TYPEMASK) of
    SS_LEFT          : plp.dtStyle := DT_LEFT or DT_EXPANDTABS {or
DT_WORDBREAK};
    SS_CENTER        : plp.dtStyle := DT_CENTER or DT_EXPANDTABS {or
DT_WORDBREAK};
    SS_RIGHT         : plp.dtStyle := DT_RIGHT or DT_EXPANDTABS {or
DT_WORDBREAK};
    SS_SIMPLE        : plp.dtStyle := DT_LEFT or DT_SINGLELINE;
    SS_LEFTNOWORDWRAP: plp.dtStyle := DT_LEFT or DT_EXPANDTABS;
  end;
  if ((dtStyle and SS_CENTERIMAGE) = 0) then
    plp.dtStyle := plp.dtStyle or DT_VCENTER;
  if ((dtStyle and SS_NOTIFY) = 0) then
    SetWindowLong(hWnd, GWL_STYLE, dtStyle or SS_NOTIFY);

  SetWindowLong(hWnd, GWL_USERDATA, Longint(plp));

  SetWindowLong(hWnd, GWL_WNDPROC, Longint(@LinkWndProc));

  // так як ми створюємо hdcMem заново при зміні розмірів вікна елемента
  // управління, то не будемо тут створювати споконвічно контексти, а просто
  // повідомимо елемент управління повідомленням про зміну розмірів.

  SendMessage(hWnd, WM_SIZE, 0, 0); // RedrawWindow(hWnd, nil, 0, RDW_INVALIDATE
or RDW_UPDATENOW or RDW_NOERASE);

end;

//

procedure RemoveStaticHyperlink(hWnd: HWND);
```

```
begin

  plp := P_LINK_PRO(GetWindowLong(hWnd, GWL_USERDATA));
  if (plp <> nil) then
    begin

      if (plp.hCursor <> 0) then
        DestroyCursor(plp.hCursor);

      plp.ti.hwnd := hWnd;
      plp.ti.uId  := hWnd;
      if IsWindow(plp.hToolTip) then
        begin
          SendMessage(plp.hToolTip, TTM_DELTOOLW, 0, Integer(@plp.ti));
          DestroyWindow(plp.hToolTip);
        end;

      if (plp.hdcMem <> 0) then
        begin
          SelectObject(plp.hdcMem, plp.hbmOld);
          DeleteObject(plp.hbmMem);
          DeleteDC(plp.hdcMem);
        end;

      //
      SetWindowLong(hWnd, GWL_WNDPROC, Longint(@plp.LinkProc));
      RedrawWindow(hWnd, @plp.rcClient, 0, RDW_INVALIDATE or RDW_ERASE);

      SetWindowLong(hWnd, GWL_USERDATA, 0);
      HeapFree(GetProcessHeap, 0, plp);
    end;
end;
end.
```

VPN_SettWind.pas - параметри програми роботи з VPN

```
unit VPN_SettWind;

interface

uses
  Windows, Messages, CommCtrl, VPN_FileInfo, VPN_LinkStat, VPN_SysUtils,
VPN_MyMsgBox,
  VPN_Controls, VPN_Resources, VPN_ScanProc;

function SettDlgProc(hWnd: HWND; uMsg: UINT; wParam: WPARAM; lParam: LPARAM):
BOOL; stdcall;

implementation

//

function SettDlgProc_OnWmInitDialog(hWnd: HWND; uMsg: UINT; wParam: WPARAM;
lParam: LPARAM): LRESULT;
var
  bldfnt: HFONT;
begin

  //

  hApp[1] := hWnd;

  //

  CreateStaticHyperlink(GetDlgItem(hApp[1], IDC_STATIC_SERVER));

  //

  SendMessage(GetDlgItem(hApp[1], IDC_COMBO_SERVER), CB_ADDSTRING, 0,
    Integer(@pszServ[1]));
  SendMessage(GetDlgItem(hApp[1], IDC_COMBO_SERVER), CB_SETCURSEL, 0, 0);

  //

  bldfnt := GetWindowBoldFont(hApp[1], GetWindowFontSize(hApp[1], 8));
  if (bldfnt <> 0) then
    SendMessage(GetDlgItem(hApp[1], IDC_STATIC_WARN), WM_SETFONT,
      Integer(bldfnt), Integer(TRUE));

  //

  Result := 0;

end;

//

function SettDlgProc_OnWmCommand(hWnd: HWND; uMsg: UINT; wParam: WPARAM; lParam:
LPARAM): LRESULT;
const
  dwRes: Array [Boolean] of DWORD = (PSWIZB_BACK, PSWIZB_BACK or PSWIZB_NEXT);
var
  dwEntry: DWORD;
  dwUser : DWORD;
  dwPass : DWORD;
begin

  //

  case HiWord(wParam) of
```

```
      //

      BN_CLICKED:
        case LoWord(wParam) of

          //

          IDC_STATIC_SERVER:
            begin

              DialogBox(hInstance, MAKEINTRESOURCEW(RC_DIALOG_UPDATE), hApp[1],
                @ScanDlgProc);

            end;

        end;

      //

      EN_UPDATE:
        case LoWord(wParam) of
          IDC_STATIC_ENTRY,
          IDC_STATIC_USER,
          IDC_STATIC_PASSW:
          begin

            dwEntry := SendMessage(GetDlgItem(hApp[1], IDC_STATIC_ENTRY),
              WM_GETTEXTLENGTH, 0, 0);
            dwUser := SendMessage(GetDlgItem(hApp[1], IDC_STATIC_USER),
              WM_GETTEXTLENGTH, 0, 0);
            dwPass := SendMessage(GetDlgItem(hApp[1], IDC_STATIC_PASSW),
              WM_GETTEXTLENGTH, 0, 0);

            SendMessage(
              GetParent(hApp[1]),
              PSM_SETWIZBUTTONS,
              0,
              dwRes[(dwEntry > 0) and (dwUser > 0) and (dwPass > 0)]
            );

          end;
        end;

    end;

    //

    Result := 0;

  end;

  //

  function SettDlgProc_OnWmCtlColorStatic(hWnd: HWND; uMsg: UINT; wParam: WPARAM;
  lParam: LPARAM): LRESULT;
  begin

    //

    case GetDlgCtrlId(lParam) of
      IDC_STATIC_WARN:
      begin

        SetBkMode(wParam, TRANSPARENT);
        SetTextColor(wParam, RGB(255, 0, 0));
        Result := GetStockObject(NULL_BRUSH);

      end;
    else
```

```
    Result := 0;

  end;

end;

//

function SettDlgProc_OnWmNotify(hWnd: HWND; uMsg: UINT; wParam: WPARAM; lParam:
LPARAM): LRESULT;
var
  pnmh : PNMHdr;
  dwRes: DWORD;
begin

  //

  pnmh := PNMHdr(lParam);

  case pnmh.code of

    //

    PSN_WIZNEXT:
    begin

      SendMessage(GetParent(hApp[1]), PSM_SETWIZBUTTONS, 0,
        Integer(PSWIZB_NEXT));

    end;

    //

    PSN_SETACTIVE:
    begin

       SendMessage(hApp[1], WM_COMMAND, MAKELPARAM(IDC_STATIC_ENTRY, EN_UPDATE),
         0);
       SendMessage(hApp[1], WM_COMMAND, MAKELPARAM(IDC_STATIC_USER, EN_UPDATE),
         0);
       SendMessage(hApp[1], WM_COMMAND, MAKELPARAM(IDC_STATIC_PASSW, EN_UPDATE),
         0);

    end;

    //

    PSN_QUERYCANCEL:
    begin

      dwRes := ExtMessageBox(
        GetParent(hApp[1]),
        MAKEINTRESOURCEW(LoadStrInst(hInstance, RC_STRING_QCANCEL)),
        MAKEINTRESOURCEW(exeInfo.pszProductName),
        MB_YESNO or MB_ICONASTERISK
      );

      SetWindowLong(hApp[1], DWL_MSGRESULT, Integer(dwRes = IDNO));

    end;

    //

    PSN_WIZBACK:
    begin

      SendMessage(GetParent(hApp[1]), PSM_SETCURSEL, GetParent(hApp[1]),
        Integer(ahpsp[1]));
```

```
    end;

  end;

  //

  Result := 1;

end;


//

function SettDlgProc_OnWmDestroy(hWnd: HWND; uMsg: UINT; wParam: WPARAM; lParam:
LPARAM): LRESULT;
var
  bldfnt: HFONT;
begin

  //

  RemoveStaticHyperlink(GetDlgItem(hApp[1], IDC_STATIC_SERVER));

  //

  bldfnt := HFONT(SendMessage(GetDlgItem(hApp[1], IDC_STATIC_WARN),
    WM_GETFONT, 0, 0));
  if (bldfnt <> 0) then
    DeleteObject(bldfnt);

  //

  Result := 0;

end;

//

function SettDlgProc(hWnd: HWND; uMsg: UINT; wParam: WPARAM; lParam: LPARAM):
BOOL; stdcall;
begin

  case uMsg of

    //

    WM_INITDIALOG:
    begin

      Result := BOOL(SettDlgProc_OnWmInitDialog(hWnd, uMsg, wParam, lParam));

    end;

    //

    WM_COMMAND:
    begin

      Result := BOOL(SettDlgProc_OnWmCommand(hWnd, uMsg, wParam, lParam));

    end;

    //

    WM_CTLCOLORSTATIC:
    begin
```

```
      Result := BOOL(SettDlgProc_OnWmCtlColorStatic(hWnd, uMsg, wParam,
lParam));

    end;

    //

    WM_NOTIFY:
    begin

      Result := BOOL(SettDlgProc_OnWmNotify(hWnd, uMsg, wParam, lParam));

    end;

    //

    WM_DESTROY:
    begin

      Result := BOOL(SettDlgProc_OnWmDestroy(hWnd, uMsg, wParam, lParam));

    end;

  else
    Result := FALSE;
  end;

end;

end.
```

VPN_ScanProc.pas - пошук підключень VPN

```
unit VPN_ScanProc;

interface

uses
  Windows, Messages, CommCtrl, WinSock, VPN_SysUtils, VPN_StatAnim,
VPN_Resources;

function ScanDlgProc(hWnd: HWND; uMsg: UINT; wParam: WPARAM; lParam: LPARAM):
BOOL; stdcall;

implementation

//

function ThreadCallback(LpParameter: Pointer): DWORD; stdcall;
type
  TaPInAddr = Array [0..MAX_ PATH-1] of PInAddr;
  PaPInAddr = ^TaPInAddr;
var
  pszText: WideString;
  pszUTF8: AnsiString;
  dwErr  : DWORD;
  phe    : PHostEnt;
  addr   : PaPInAddr;
  ws     : TWSAData;
  i      : Integer;
begin

  //

  Result := 0;

  //

  SetThreadPriority(hThread, THREAVPN_PRIORITY_BELOW_NORMAL);

  //

  dwErr := WSAStartup(MAKEWORD(1, 0), ws);
  if (dwErr = NOERROR) then
  try
    pszUTF8 := WideStringToAnsi(pszServ, CP_ACP);
    phe := GetHostByName(@pszUTF8[1]);
    if (phe <> nil) then
    begin
      addr := PaPInAddr(phe^.h_addr_list);
      i := 0;
      SendMessage(GetDlgItem(hApp[1], IDC_COMBO_SERVER), CB_RESETCONTENT, 0, 0);
      SendMessage(GetDlgItem(hApp[1], IDC_COMBO_SERVER), CB_ADDSTRING, 0,
        Integer(@pszServ[1]));
      while (addr^[I] <> nil) do
      begin
        pszUTF8 := inet_ntoa(addr[I]^);
        pszText := AnsiStringToWide(pszUTF8, CP_ACP);
        SendMessage(GetDlgItem(hApp[1], IDC_COMBO_SERVER), CB_ADDSTRING, 0,
          Integer(@pszText[1]));
        pszText := Format(LoadStrInst(hInstance, RC_STRING_IPHOST), [pszText]);
        SendMessage(GetDlgItem(hApp[3], IDC_STATIC_ADDRESS), WM_SETTEXT, 0,
          Integer(@pszText[1]));
        Inc(i);
        Sleep(35);
      end;
      SendMessage(GetDlgItem(hApp[1], IDC_COMBO_SERVER), CB_SETCURSEL, 0, 0);
    end;
```

```
  finally
    WSACleanup;
  end;

  //

  SendMessage(hApp[3], WM_DESTROY, 0, 0);

end;

//

function ScanDlgProc_OnWmInitDialog(hWnd: HWND; uMsg: UINT; wParam: WPARAM;
lParam: LPARAM): LRESULT;
var
  pszText : WideString;
  ThreadID: LongWord;
  himl    : HIMAGELIST;
begin

  //

  hApp[3] := hWnd;

  //

  SetCenterDialogPos(hApp[3], hApp[1], TRUE);

  //

  pszText := Format(LoadStrInst(hInstance, RC_STRING_IPSERVER), [pszServ]);
  SendMessage(GetDlgItem(hApp[3], IDC_STATIC_ADDRESS), WM_SETTEXT, 0,
    Integer(@pszText[1]));

  //

  CreateAnimateStatic(GetDlgItem(hApp[3], IDC_STATIC_ANIMATE));
  himl := ImageList_LoadImage(hInstance, MAKEINTRESOURCEW(RC_BITMAP_WAITING),
    GetSystemMetrics(SM_CXSMICON), 0, CLR_DEFAULT, IMAGE_BITMAP, LR_DEFAULTCOLOR
    or LR_CREATEDIBSECTION);
  if (himl <> 0) then
    SendMessage(GetDlgItem(hApp[3], IDC_STATIC_ANIMATE), SS_SETIMAGELIST, himl,
      0);

  //

  hThread := CreateThread(nil, 0, @ThreadCallback, nil, 0, ThreadID);
  if (hThread <> 0) then
    begin
      CloseHandle(hThread);
      hThread := 0;
    end;

  //

  Result := 0;

end;

//

function ScanDlgProc_OnWmDestroy(hWnd: HWND; uMsg: UINT; wParam: WPARAM; lParam:
LPARAM): LRESULT;
var
  himl: HIMAGELIST;
begin

  //
```

```
  if (hThread <> 0) then
    begin
      CloseHandle(hThread);
      hThread := 0;
    end;

  //

  himl := SendMessage(GetDlgItem(hApp[3], IDC_STATIC_ANIMATE), SS_GETIMAGELIST,
    0, 0);
  if (himl <> 0) then
    ImageList_Destroy(himl);
  RemoveAnimateStatic(GetDlgItem(hApp[3], IDC_STATIC_ANIMATE));

  //

  EndDialog(hApp[3], wParam);

  //

  Result := 0;

end;

//

function ScanDlgProc(hWnd: HWND; uMsg: UINT; wParam: WPARAM; lParam: LPARAM):
BOOL; stdcall;
begin

  case uMsg of

    //

    WM_INITDIALOG:
      begin
        Result := BOOL(ScanDlgProc_OnWmInitDialog(hWnd, uMsg, wParam, lParam));
      end;

    //

    WM_DESTROY:
      begin
        Result := BOOL(ScanDlgProc_OnWmDestroy(hWnd, uMsg, wParam, lParam));
      end;

  else
    Result := FALSE;
  end;

end;

end.
```

VPN_FinsWind.pas - VPN-з'єднання

```
unit VPN_FinsWind;

interface

uses
  Windows, Messages, CommCtrl, VPN_Windows, VPN_Controls, VPN_FileInfo,
VPN_SysUtils,
  VPN_MyMsgBox, VPN_Ole2, VPN_Active, VPN_ShlObj, VPN_RasApi, VPN_Resources;

function FinsDlgProc(hWnd: HWND; uMsg: UINT; wParam: WPARAM; lParam: LPARAM):
BOOL; stdcall;

implementation

//

function FinsDlgProc_OnWmInitDialog(hWnd: HWND; uMsg: UINT; wParam: WPARAM;
lParam: LPARAM): LRESULT;
var
  bldfnt: HFONT;
begin

  //

  hApp[2] := hWnd;

  //

  bldfnt := HFONT(SendMessage(GetDlgItem(hApp[0], IDC_STATIC_WELCOME),
    WM_GETFONT, 0, 0));

  if (bldfnt <> 0) then
    SendMessage(GetDlgItem(hApp[2], IDC_STATIC_FINISH), WM_SETFONT,
      Integer(bldfnt), Integer(TRUE));

  //

  Result := 0;

end;

//

function FinsDlgProc_OnWmNotify(hWnd: HWND; uMsg: UINT; wParam: WPARAM; lParam:
LPARAM): LRESULT;
var
  pnmh  : PNMHdr;
  dwRes : DWORD;
  pszText: WideString;

  //

  function GetNextItemID(pidl: PItemIDList): PItemIDList;
  var
    cb: DWORD;
  begin
    Result := nil;
    if (pidl = nil) then
      Exit;
    cb := pidl.mkid.cb;
    if (cb = 0) then
      Exit;
    pidl := PItemIDList(Cardinal(pidl) + cb);
    if (pidl.mkid.cb <> 0) then
      Result := pidl;
```

```
end;

//

function GetPIDSize(pidl: PItemIDList): DWORD;
begin
  Result := 0;
  if (pidl <> nil) then
  begin
    Result := SizeOf(pidl.mkid.cb);
    while (pidl <> nil) do
    begin
      Inc(Result, pidl.mkid.cb);
      pidl := GetNextItemID(pidl);
    end;
  end;
end;

//

function IsDesktopFolder(pidl: PItemIDList): Boolean;
begin
  if Assigned(pidl) then
    Result := (pidl.mkid.cb = 0)
  else
    Result := FALSE;
end;

//

function ConcatPIDL(destpidl, srcpidl: PItemIDList): PItemIDList;
var
  cb1: DWORD;
  cb2: DWORD;
  pmc: IMalloc;
  hr : HRESULT;
begin
  Result := nil;
  hr := SHGetMalloc(pmc);
  if SUCCEEDED(hr) then
  begin
    cb1 := 0;
    cb2 := 0;
    if Assigned(destpidl) then
    begin
      if not IsDesktopFolder(destpidl) then
        cb1 := GetPIDSize(destpidl) - SizeOf(destpidl^.mkid.cb);
    end;
    if Assigned(srcpidl) then
      cb2 := GetPIDSize(srcpidl);
    Result := pmc.Alloc(cb1 + cb2);
    if Assigned(Result) then
    begin
      if Assigned(destpidl) then
        CopyMemory(Result, destpidl, cb1);
      if Assigned(srcpidl) then
        CopyMemory(Pointer(DWORD(Result) + cb1), srcpidl, cb2);
    end;
    pmc := nil;
  end;
end;

//

procedure CreateShellVpnLink(pszEntry: WideString);
var
  pMalloc    : IMalloc;
  Desktop    : IShellFolder;
  pidlDesktop: PItemIDList;
```

```
    pszPath    : Array [0..MAX_ PATH-1] of WideChar;
    pidlConnect: PItemIDList;
    Network    : IShellFolder;
    Items      : IEnumIDList;
    pidl2      : PItemIDList;
    dwFetched  : Cardinal;
    Connection : STRRET;
    ObjectName : WideString;
    pfLink     : IUnknown;
    isLink     : IShellLink;
    ipFile     : IPersistFile;
    pidl3      : PItemIDList;
    szFileName : WideString;
  begin
    CoInitialize(nil);
    try
      // визначається оболонка
      if (SHGetMalloc(pMalloc) = S_OK) then
      try
        // визначається простір імен корню директорій
        if (SHGetDesktopFolder(Desktop) = S_OK) then
        try
          if (SHGetSpecialFolderLocation(0, CSIDL_DESKTOP, pidlDesktop) = S_OK)
then
          try
            ZeroMemory(@pszPath, SizeOf(pszPath));
            SHGetPathFromIDList(pidlDesktop, @pszPath);
            if (SHGetSpecialFolderLocation(0, CSIDL_CONNECTIONS, pidlConnect) =
S_OK) then
            try
              Desktop.BindToObject(pidlConnect, nil, IIVPN_IShellFolder,
Network);
              Network.EnumObjects(0, SHCONTVPN_NONFOLDERS, Items);
              while (Items.Next(1, pidl2, dwFetched) = S_OK) do
              try
                if (dwFetched > 0) and Assigned(pidl2) then
                try
                  Network.GetDisplayNameOf(pidl2, SHGDN_NORMAL, Connection);
                  ObjectName := Connection.pOleStr;
                  if (lstrcmpi(@ObjectName[1], @pszEntry[1]) = 0) then
                  try
                    CoCreateInstance(CLSIVPN_ShellLink, nil,
CLSCTX_INPROC_SERVER, IUnknown, pfLink);
                    isLink := pfLink as IShellLink;
                    ipFile := pfLink as IPersistFile;
                    pidl3 := ConcatPIDL(pidlConnect, pidl2);
                    isLink.SetIDList(pidl3);
                    szFileName := Format('%s\%s.lnk',
[ExcludeTrailingPathDelimiter(pszPath), pszEntry]);
                    ipFile.Save(@szFileName[1], FALSE);
                    pMalloc.Free(pidl3);
                  finally
                    {
                    pfLink := nil;
                    isLink := nil;
                    ipFile := nil;
                    }
                  end;
                finally
                  pMalloc.Free(pidl2); // папка версій
                end;
              finally
              end;
            finally
              Network := nil;
              pMalloc.Free(pidlConnect); // папка версій
            end;
          finally
            pMalloc.Free(pidlDesktop); // папка версій
```

```
          end;
        finally
          Desktop := nil; // версії простору імен корню директорій
        end;
      finally
        pMalloc := nil; //
      end;
    finally
      CoUninitialize;
    end;
  end;

  //

  function CreateRasVpnConnection(szEntryName, szPhoneName, szUserName,
szPassword: WideString): LRESULT;
  var
    osvi     : TOSVersionInfo;
    rEntry   : RASENTRYW;
    rDial    : RASDIALPARAMSW;
    lpCred   : RASCREDENTIALSW;
    dwSize   : Integer;
    EntrySize: Integer;
    InfoSize : Integer;
    dwFlags  : DWORD;
    dwFlags2 : DWORD;
    dwRes    : DWORD;
  begin

    // заповнюємо структуру RASENTRY і довідаємося потрібний розмір для
коректного виклику
    // функції RasSetEntryProperties
    dwSize := SizeOf(RASENTRYW);
    RasGetEntryProperties(nil, nil, nil, EntrySize, nil, InfoSize);
    if (EntrySize < dwSize) then
      dwSize := EntrySize;

    // Задаємо прапорові параметри VPN з'єднання
    dwFlags :=

      // Вкладка 'Параметри', прапор 'Запитувати ім'я, пароль, сертифікат і
т.д.', вмик
      RASEO_PreviewUserPw or

      // Вкладка 'Загальні', прапор 'При підключенні вивести значок в області
повідомлень', вмик
      RASEO_ModemLights or

      // Вкладка 'Загальні', прапор 'Відображати хід підключення', вмик
      RASEO_ShowDialingProgress or

      // Використовувати основний шлюз
      RASEO_RemoteDefaultGateway or

      // Зашифрований пароль буде використовуватися при перевірці дійсності із
сервером
      RASEO_RequireEncryptedPw or

      // Використовувати автоматично логін, пароль і домен з Windows
      RASEO_RequireDataEncryption or

      // Пароль буде зашифрований за схемою Microsoft
      RASEO_RequireMsEncryptedPw;
    dwFlags2 :=

      // Вкладка 'Параметри', прапор 'Погоджувати багатоканальне підключення для
одноканальних', вимк

      RASEO2_DontNegotiateMultilink or
```

```
      // Вкладка 'Параметри', прапор 'Передзвонити при розриві зв'язку', вмик
      RASEO2_ReconnectIfDropped;

    // Заповнюємо структуру RASENTRY
    ZeroMemory(@rEntry, SizeOf(RASENTRYW));
    rEntry.dwSize                   := dwSize;
    rEntry.dwfOptions               := dwFlags;

    // Тип використовуваного протоколу = TCP/IP
    rEntry.dwfNetProtocols          := RASNP_Ip;

    // Тип використовуваного протоколу сервера віддаленого доступу = Point-to-
Point Protocol (PPP)
    rEntry.dwFramingProtocol        := RASFP_Ppp;

    // Тип створюваного підключення - Віртуальна приватна мережа (VPN)
    rEntry.dwType                   := RASET_Vpn;

    // Значення списку, що випадає, 'Тип VPN' = 'Автоматично'
    // Викликається спочатку тільки PPTP, якщо ж спроба закінчується невдачею,
то викликається L2TP
    rEntry.dwVpnStrategy            := VS_Default;
    rEntry.dwfOptions2              := dwFlags2;

    // Вкладка 'Безпека', прапор 'Потрібне шифрування даних', вмик
    // Діалог 'Додаткові параметри безпеки', список 'Шифрування даних' =
'обов'язкове'
    // Тип шифрування даних при підключенні = Шифрування не використовується
    rEntry.dwEncryptionType         := ET_None;

    // Використовуємо з'єднання пристроїв з безліччю «підвходів»
    rEntry.dwDialMode               := RASEDM_DialAll;

    // Вкладка 'Параметри', 'Число повторень набору номера' = 3
    rEntry.dwRedialCount            := 3;

    // Вкладка 'Параметри', 'Інтервал між повтореннями' = 60 секунд
    rEntry.dwRedialPause            := 60;
    lstrcpy(rEntry.szLocalPhoneNumber, @szPhoneName[1]);
    lstrcpy(rEntry.szDeviceType, RASDT_Vpn);

    // Створюємо нове підключення VPN з потрібними параметрами
    dwRes := RasSetEntryProperties(nil, @szEntryName[1], @rEntry, dwSize, nil,
0);
    case dwRes of
      ERROR_SUCCESS:
        Begin

          // виконуємо перевірку версії ОС. починаючи з ОС Win XP і старше,
логін і
          // пароль (фактично це нам і потрібно) можна змінити функцією
          // RasSetCredentials, а в попередніх ОС можна за допомогою функції
          // RasSetEntryDialParams. в Win XP ще можна змінити пароль функцією
          // RasSetEntryDialParams, а от уже в Win Vista і старше не вийде.

          ZeroMemory(@osvi, SizeOf(TOSVersionInfo));
          osvi.dwOSVersionInfoSize := SizeOf(TOSVersionInfo);
          VPN_Windows.GetVersionEx(osvi);

          if ((osvi.dwPlatformId = VER_PLATFORM_WIN32_NT) and
            (osvi.dwMajorVersion >= 5) and (osvi.dwMinorVersion >= 1)) then
            Begin

              // Заповнюємо структуру RASCREDENTIALS
              ZeroMemory(@lpCred, SizeOf(RASCREDENTIALSW));
              lpCred.dwMask := RASCM_UserName or RASCM_Password;
              lpCred.dwSize := SizeOf(RASCREDENTIALSW);
              lstrcpy(lpCred.szUserName, @szUserName[1]);
              lstrcpy(lpCred.szPassword, @szPassword[1]);
```

```
            // Змінюємо логін і пароль створеного підключення
            dwRes := RasSetCredentials(nil, @szEntryName[1], lpCred, FALSE);
          end
        else
          begin

            // Заповнюємо структуру RASDIALPARAMS
            ZeroMemory(@rDial, SizeOf(RASDIALPARAMSW));
            rDial.dwSize := SizeOf(RASDIALPARAMSW);
            lstrcpy(rDial.szEntryName, @szEntryName[1]);
            lstrcpy(rDial.szUserName, @szUserName[1]);
            lstrcpy(rDial.szPassword, @szPassword[1]);

            // Змінюємо логін і пароль створеного підключення
            dwRes := RasSetEntryDialParams(nil, @rDial, FALSE);
          end;
      end;
    end;
    Result := dwRes;
  end;

begin

  //

  pnmh := PNMHdr(lParam);

  case pnmh.code of

    //

    PSN_SETACTIVE:
    begin

      pszText := Format(
        LoadStrInst(hInstance, RC_STRING_VPNINFO),
        [
        Edit_GetText(GetDlgItem(hApp[1], IDC_STATIC_ENTRY)),
        Edit_GetText(GetDlgItem(hApp[1], IDC_COMBO_SERVER)),
        Edit_GetText(GetDlgItem(hApp[1], IDC_STATIC_USER))
        ]
      );

      SendMessage(GetDlgItem(hApp[2], IDC_STATIC_VPNINFO), WM_SETTEXT, 0,
        Integer(@pszText[1]));

      SendMessage(GetParent(hApp[2]), PSM_SETWIZBUTTONS, 0,
        Integer(PSWIZB_BACK or PSWIZB_FINISH));

    end;

    //

    PSN_QUERYCANCEL:
    begin

      dwRes := ExtMessageBox(
        GetParent(hApp[2]),
        MAKEINTRESOURCEW(LoadStrInst(hInstance, RC_STRING_QCANCEL)),
        MAKEINTRESOURCEW(exeInfo.pszProductName),
        MB_YESNO or MB_ICONASTERISK
      );

      SetWindowLong(hApp[2], DWL_MSGRESULT, Integer(dwRes = IDNO));

    end;

    //
```

```
    PSN_WIZFINISH:
    begin

      pszText := Edit_GetText(GetDlgItem(hApp[1], IDC_STATIC_ENTRY));

      dwRes := CreateRasVpnConnection(
        pszText,
        Edit_GetText(GetDlgItem(hApp[1], IDC_COMBO_SERVER)),
        Edit_GetText(GetDlgItem(hApp[1], IDC_STATIC_USER)),
        Edit_GetText(GetDlgItem(hApp[1], IDC_STATIC_PASSW))
      );

      if (dwRes = ERROR_SUCCESS) then
      begin
        dwRes := SendMessage(GetDlgItem(hApp[2], IDC_CHECK_SHORTCUT),
          BM_GETCHECK, 0, 0);
        if (dwRes = BST_CHECKED) then
          CreateShellVpnLink(pszText);
      end;

    end;

  end;

  //

  Result := 1;

end;

//

function FinsDlgProc(hWnd: HWND; uMsg: UINT; wParam: WPARAM; lParam: LPARAM):
BOOL; stdcall;
begin

  case uMsg of

    //

    WM_INITDIALOG:
    begin

      Result := BOOL(FinsDlgProc_OnWmInitDialog(hWnd, uMsg, wParam, lParam));

    end;

    //

    WM_NOTIFY:
    begin

      Result := BOOL(FinsDlgProc_OnWmNotify(hWnd, uMsg, wParam, lParam));

    end;

  else
    Result := FALSE;
  end;

end;

end.
```