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
“____” _____ 2024 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки створення з’єднань OpenVPN з використанням бібліотеки OpenSSL”

Виконав здобувач вищої освіти
IV курсу, групи КБ-21-3СК
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Парталога В.О.
«____» _____ 2024 р.

Керівник проекту
кандидат технічних наук
_____ Смірнова Т.В.
«____» _____ 2024 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 2024 року

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

Парталозі Владиславу Олександровичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки створення з’єднань OpenVPN з використанням бібліотеки OpenSSL*

2. Керівник роботи *Смірнова Тетяна Віталіївна, канд. техн. наук*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 136-02 від 01.04.2024 року

3. Строк подання студентом роботи до захисту *23.05.2024 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи кібербезпеки створення з’єднань OpenVPN з використанням бібліотеки OpenSSL*

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 2024 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	10.03.2024 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2024 р.	
3.	Розробка моделі компонента	20.03.2024 р.	
4.	Розробка структур даних	25.03.2024 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2024 р.	
6.	Програмування алгоритмів	10.04.2024 р.	
7.	Оформлення ПЗ	17.04.2024 р.	
8.	Попередній захист роботи	23.05.2024 р.	

Дата видачі завдання
« 17 » січня 2024 р.

Підпис керівника
_____ Смірнова Т.В.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2024 р.

Підпис здобувача
_____ Парталога В.О.
(прізвище та ініціали)

АНОТАЦІЯ

Парталога В.О. Програмне забезпечення системи кібербезпеки створення з'єднань OpenVPN з використанням бібліотеки OpenSSL. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2024.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки створення з'єднань OpenVPN з використанням бібліотеки OpenSSL.

Метою розробки є програмне забезпечення системи кібербезпеки створення з'єднань OpenVPN з використанням бібліотеки OpenSSL.

Результат роботи – програмна реалізація системи кібербезпеки створення з'єднань OpenVPN з використанням бібліотеки OpenSSL.

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

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

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

Програму розроблено в середовищі Delphi 10.4.

Ключові слова: кібербезпека, OpenVPN, OpenSSL

ABSTRACT

Partaloha V.O. Cybersecurity software for creating OpenVPN connections using the OpenSSL library. 125 Cyber security. Central Ukrainian National Technical University. Kropyvnytskyi. 2024.

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 OpenVPN connections using the OpenSSL library.

The goal of the development is the software of the cyber security system to create OpenVPN connections using the OpenSSL library.

The result of the work is a software implementation of the cybersecurity system for creating OpenVPN connections using the OpenSSL library.

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 a PC with Windows 10/11 OS.

The program was developed in the Delphi 10.4 environment.

Keywords: cyber security, OpenVPN, OpenSSL

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 2
ВСТУП 3
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 5
1.1 Призначення системи 5
1.2 Область застосування 6
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 9
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти 9
2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки та мови програмування 15
2.3 Розгорнута постановка завдання 21
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 23
3.1 Опис функціонування системи 23
3.2 Розробка структурної схеми 30
3.3 Розробка функціональної схеми 40
3.4 Розробка діаграми процесів 53
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ 55
4.1 Розробка блок-схем та опис алгоритмів функціонування системи 55
4.2 Захист розробленого програмного забезпечення 72
5 ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ 73
6 ОСНОВНІ ВИСНОВКИ 75
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 77

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. Віртуальна приватна мережа (VPN) – це система, яка надає орієнтовані на підприємство послуги зв'язку в спільній загальнодоступній мережевій інфраструктурі та забезпечує налаштовані робочі характеристики рівномірно та універсально для всього підприємства. Термін використовується узагальнено для позначення голосових VPN.

Щоб уникнути плутанини, служби передачі даних на основі IP називають VPN даних. Постачальники послуг визначають VPN як WAN постійних віртуальних каналів, які зазвичай використовують режим асинхронної передачі (ATM) або Frame Relay для передачі IP.

Постачальники технологій визначають VPN як використання програмного або апаратного забезпечення шифрування для забезпечення конфіденційності зв'язку через загальнодоступну або ненадійну мережу передачі даних.

До 2025 року мережева архітектура безпеки допоможе організаціям зменшити фінансовий вплив окремих атак на 90%.

Лідери безпеки, які переосмислюють функцію кібербезпеки та технологічну архітектуру, позиціонують свій бізнес так, щоб підтримувати та збільшувати вартість.

Керівники безпеки та управління ризиками, які бажають краще протидіяти новим ризикам, повинні зробити наступні кроки:

– Децентралізація прийняття рішень.

– Надайте пріоритет інструментам, які можуть взаємодіяти.

– Передбачте безперервне розширення поверхні атаки підприємства.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки створення з'єднань OpenVPN з використанням бібліотеки OpenSSL.

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

– Огляд існуючих систем створення з’єднань OpenVPN з використанням бібліотеки OpenSSL.

– Дослідження системи кібербезпеки створення з’єднань OpenVPN з використанням бібліотеки OpenSSL.

– Програмна реалізація системи кібербезпеки створення з’єднань OpenVPN з використанням бібліотеки OpenSSL.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі створення з’єднань OpenVPN з використанням бібліотеки OpenSSL.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки створення з’єднань OpenVPN з використанням бібліотеки OpenSSL, 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

Система призначена для реалізації програмного забезпечення системи створення з’єднань OpenVPN з використанням бібліотеки OpenSSL.

VPN перетворилися з незрозумілої мережевої утиліти на великий бізнес. Ви, напевно, бачили рекламу від свого улюбленого користувача YouTube, у подкастах і навіть під час Superbowl із заявами про те, як VPN може зробити вас анонімним або надати доступ до безкоштовного потокового відео. Чи виправдовують продукти ажіотаж? Хоча VPN можуть бути корисними інструментами для захисту вашої конфіденційності, важливо розуміти, як ці інструменти працюють, щоб ви могли вирішити, чи допоможуть вони вам. Ми розбираємо, що VPN роблять і чого вони не роблять, щоб допомогти вам зрозуміти, чому вам потрібна така мережа та як вибрати найкращу для вас.

VPN означає віртуальну приватну мережу. Коли ми говоримо про VPN, ми зазвичай говоримо про комерційну VPN, яка продається безпосередньо споживачам для використання в повсякденному житті, але ідея VPN має набагато ширше застосування. Корпорації вже давно використовують технологію VPN, щоб надати працівникам доступ до цифрових ресурсів, де б вони не були, задовго до того, як COVID-19 зробив роботу вдома нормою.

Коли ви вмикаєте VPN, створюється зашифроване з’єднання (іноді його називають «тунелем») між вашим пристроєм і віддаленим сервером, керованим службою VPN. Весь ваш інтернет-трафік направляється через цей тунель на сервер, який потім направляє трафік у загальнодоступний Інтернет, як зазвичай. Дані, що повертаються на ваш пристрій, здійснюють ту саму подорож: з Інтернету на сервер VPN, через зашифроване з’єднання та назад на вашу машину.

1.2 Область застосування

Областю застосування є побудова захищених каналів зв'язку у мережах різного рівня.

Як працює VPN

Майте на увазі, що для налаштування VPN вам не потрібна інша компанія. Є кілька варіантів, як-от Outline, щоб налаштувати власний. Зробити це досить просто, але вам потрібно буде підтримувати сервер або орендувати його, що не так просто. Хоча є певні зусилля, щоб зробити власні VPN доступнішими, краще залишити це майстрам, які прагнуть забруднити руки (цифровими).

Чи VPN роблять вас анонімними в Інтернеті?

Шифруючи ваш трафік і направляючи його через сервер VPN, спостерігачам складніше, але не неможливо ідентифікувати вас і відстежувати ваші переміщення в Інтернеті. Жодна мережа VPN не забезпечує повної анонімності, але може допомогти покращити вашу конфіденційність.

Наприклад, ваш постачальник послуг Інтернету (ISP), мабуть, є єдиною організацією, яка найкраще розуміє, що ви робите в Інтернеті. У 2021 році FTC опублікувала звіт, у якому вказано, скільки саме ваш провайдер знає про те, що ви робите в Інтернеті, і це багато. Гірше того, завдяки Конгресу ваш провайдер може продавати анонімні дані клієнтів. Якщо вам не подобається, що компанія, якій ви вже платите, отримує прибуток від ваших даних, або якщо ви стурбовані тим, що інтернет-провайдери накопичують детальну інформацію про вашу діяльність, VPN допоможе. Навіть ваш інтернет-провайдер не може бачити ваш веб-трафік, коли ви використовуєте VPN.

VPN також ускладнюють відстеження вас в Інтернеті для рекламодавців та інших користувачів. Зазвичай дані передаються з Інтернету на ваш пристрій за допомогою його IP-адреси. Коли VPN активна, ваша справжня IP-адреса прихована, і будь-хто, хто спостерігає за вами, може бачити лише IP-адресу сервера VPN. Приховуючи вашу справжню IP-адресу, мережі VPN забороняють

стежити за одним із інструментів, який використовується для ідентифікації та відстеження вас в Інтернеті.

Незважаючи на це, VPN не роблять вас повністю анонімними в Інтернеті. Рекламодавці, наприклад, мають численні способи ідентифікувати та відслідковувати вас, коли ви пересуваєтеся в Інтернеті. Трекери та файли cookie на веб-сайтах намагаються однозначно ідентифікувати вас, а потім спостерігають, де ви з'явитьсяте далі.

Сайти та рекламодавці також можуть ідентифікувати вас за кількома унікальними характеристиками, такими як версія браузера, розмір екрана тощо. Ця інформація сама по собі нешкідлива, але коли компанії збирають достатню кількість цих ідентифікаторів, вони утворюють унікальний підпис – настільки, що цей процес називається відбитком пальця браузера.

Це не кажучи вже про конфіденційність, від якої ми відмовляємося в обмін на послуги. Amazon, Google і Meta (раніше Facebook) стали опорами сучасної інтернет-інфраструктури, і їх неможливо повністю уникнути. Навіть якби ви видалили всі свої облікові записи й більше ніколи ними не користувалися, вони ймовірно зможуть зібрати дані про вас.

Ці загрози конфіденційності потребують інших інструментів, крім VPN. Блокувальники реклами та трекерів, як ті, що є в деяких браузерах, або як окремі інструменти, як-от Privacy Badger від EFF, вирішують деякі з цих проблем.

Використання Tor може захистити вашу конфіденційність навіть краще, ніж VPN, і надати вам доступ до Dark Web. На відміну від VPN, Tor пропускає ваш трафік через кілька добровольчих серверних вузлів, що значно ускладнює його відстеження. Ним також керує некомерційна організація та поширюється безкоштовно. Деякі служби VPN навіть підключаються до Tor через VPN, що полегшує доступ до цієї таємничої системи. Однак вартість підключення до Інтернету висока, оскільки використання Tor погіршить ваше з'єднання набагато більше, ніж VPN. Tor також не ідеальний, і він також має багато недоліків, які слід враховувати.

Майте на увазі, що правоохоронні та державні органи мають доступ до більш передових та інвазивних методів. Маючи достатньо часу, рішучий, добре фінансований супротивник зазвичай може отримати те, що хоче.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки створення з'єднань OpenVPN з використанням бібліотеки OpenSSL, 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

Проведемо огляд сучасних сервісів VPN, які є з однієї сторони надійним засобом шифрування всього трафіку, а з іншого боку – є абсолютно універсальним рішенням, що дозволяє одержати анонімність для широкого діапазону завдань, які кожний визначає для себе сам.

HideME

HideME – сервіс для анонімного й захищеного доступу до веб-ресурсів. Він забезпечує:

– роботу із заблокованими сайтами (якщо блокування до них застосована на рівні провайдеру/країни або адміністратора локальної мережі);

– шифрування всіх адрес і видалення шкідливих і рекламних скриптів з “заблокованих” сайтів;

– постійно обновлювані проксі-списки зі зручним фільтром по безлічі параметрів і експортом в csv, txt, xml;

– власний проксі-чекер із системою розпізнання IP:Port проксі;

– IP з різних країн на вибір з повною підтримкою розблокування YouTube, анонімністю для всього ПК, окремим плагіном для Firefox.

Базові функції анонімайзера надаються безкоштовно. У пакет платних послуг включені додаткові можливості.

При використанні VPN доступні сервери з декількох десятків країн і міст: користувач може вибрати один з 120+ IP-адрес для конфіденційного й захищеного доступу до сайтів, які з якоїсь причини потрапили “під фільтр” у

вашій країні або у вашій локальній мережі. Команда постійно працює над розширенням бази проксі-серверів і IP-адрес.

Cryptocloud/Torrentfreedom

Це найбільший VPN-провайдер, бізнес якого повністю орієнтований на активних користувачів P2P-мереж, зокрема BitTorrent. Використовується технологія OpenVPN, що дозволяє дуже просто підключатися як до публічних, так і приватних торрент-трекерів по усьому світі. Підтримує роботу з Windows, Mac OS X і Linux/UNIX/BSD. Для підключення до сервісу використовуються ключі 1024-bit RSA, при передачі всіх даних через сервера трафік шифрується ключем на 2048 біт. Даний ключ шифрування трафіку динамічно міняється кожні 20 мінут. Ніякі особисті дані не зберігаються. Вартість підключення – $17 доларів на місяць, швидкість обмежена лише фізичною шириною вашого каналу. Буквально зовсім недавно відбулося злиття сервісу з cryptocloud.com – що надає більш широкий набір інструментів по мережній безпеці на базі хмарних технологій, у тому числі – і вищеописаний VPN-сервіс.

Рисунок 2.1 – Інтерфейс користувача Cryptocloud/Torrentfreedom

Perfect Privacy

У правилах використання сервісу написано: “Нам однаково, хто ви й чим займаєтеся в мережі; платите – і користуйтеся”. У доповненні до основної послуги – VPN, сервіс також пропонує цілий набір інструментів “на всі випадки життя” для анонімизації не тільки веб-серфингу. От лише частина із пропонованого набору: 4096-bit OpenVPN encryption, 4096 bit SSH-2 (Secure Shell 2), 4096 bit SSL/TLS, Squid proxy, CGI proxies і багато чого іншого. Підключення коштує 10 євро на місяць, швидкість не обмежена. Також потрібно заплатити одноразово 10 євро за налаштування й перше підключення до сервісу. Даний сервіс ідеально підходить для просунутих користувачів, які с допомогою широкого набору інструментів зуміють будь-яку свою програму, від банального браузера до покер-бота, зробити повністю анонімними для зовнішнього інтернет-спостерігача. На даний момент у сервісу більше 30 своїх серверів по усьому світі.

Рисунок 2.2 – Інтерфейс користувача Perfect Privacy

StrongVPN

Дуже відомий і великий сервіс, що розташовує свої сервера в більш ніж 30 країнах миру. Надає OpenVPN за $20 на місяць і $15 за звичайний VPN. Використовує ключ 2048-біт і шифрування через протоколи SSL/TLS. Протестований для роботи з мобільними пристроями, такими як смартфони, iPhone, VoIP-пристрої й т.п. Є можливість вибору територіальної приналежності робітника VPN-сервера, а також тестування швидкості роботи через кожний конкретний VPN-сервер через ваше підключення. Цей сервіс виділяється своєю повною легальністю, наявністю офісу й міжнародної служби підтримки доступної як за телефоном, так і по Skype і e-mail, оформленням належних договорів і юридичних гарантій.

Рисунок 2.3 – Інтерфейс користувача StrongVPN

HideMyAss VPN

Дуже популярна західна компанія, головний козир якої – орієнтація на людей, що слабко розбираються в технічній стороні питання, що робить всі підключення й налаштування максимально простими й зрозумілими для будь-якого середньостатистичного користувача інтернету. Проте, у розпорядженні цієї компанії перебуває значний парк із 75 серверів по усьому світі, а також пул з 7000 IP-адрес. Місяць використання сервісу коштує майже $12, ніяких обмежень по трафіку або швидкості підключення тут немає. При оплаті відразу за рік – знижка в 60% від вартості. До речі кажучи, варто помітити, що hidemyass.com пропонує безліч додаткових аналогічних сервісів, таких як анонімна пошта, анонімізатор посилань, web-проксі і т.п.

Рисунок 2.4 – Інтерфейс користувача HideMyAss VPN

Kebrum

Kebrum – це сейшельська компанія, створена російськими розроблювачами. Хотілося б зупинитися небагато детальніше на плюсах і мінусах цього одного з лідерів російського ринку. Головні плюси – це дуже вигідні ціни (базовий тариф починається з оцінки в $3.90 за 50Gb на місяць, цей же тариф можна купити відразу на рік за $40, що є одним із самих вигідних пропозицій на ринку). Друга особливість – сервіс заснований на прогресивному OpenVPN, хоча додатково підтримується, як допоміжний, застарілий PPTP-протокол для роботи будь-яких мобільних пристроїв (цей протокол підтримується нативно в iPhone, iPad, Android і Windows Mobile). І, нарешті, у цього сервісу дуже серйозна юридична платформа – для розміщення серверів обрана відома офшорна зона Сейшелли, тому, домогтися потрібного судового рішення у відомому своєю ліберальністю сейшельському суді третій стороні буде надзвичайно складно.

Навіть якщо хтось зможе це зробити, сервіс заздалегідь попереджає: ніяких логів він не веде й максимум що може трапитися – це блокування логіну-паролю.

Insorg VPN

Insorg – це інша відома російська команда, що займається безпекою з 2003 року. Головні переваги пропонованих рішень: доступ до 11-ти серверів розкиданих по усьому світі й підключення, що пропонують, OpenVPN, DoubleVPN, TripleVPN, PPTP VPN.

Для обходу будь-яких обмежень вашого провайдеру/організації до OpenVPN серверів можна підключитися по UDP і також до TCP-80, 110 і 443 портів. Доступна портативна версія PPTP VPN підключення, тобто немає необхідності створювати VPN підключення або інсталювати наш диспетчер підключень на чужому комп'ютері або в інтернет-кафі. У цьому випадку досить скористатися портативною версією PPTP VPN підключення, скопіювавши її на переносний накопичувач, наприклад на флеш-диск. На всіх сервісах

використовується дуже сильний 2048-бітний ключ шифрування. Ціни середні для ринку, починаючи з $20 на місяць за базовий пакет. Тут можна взяти безкоштовний тест на одна година перед тим, як купити VPN, звернувшись у техпідтримку.

Рисунок 2.5 – Інтерфейс користувача Insorg

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 з'єднань OpenVPN з використанням бібліотеки OpenSSL.

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

Кілька VPN кажуть, що вони включають певний захист від шкідливих файлів. Іноді це базовий захист від відомих шкідливих сайтів і файлів. Деякі служби VPN також включають спеціальні антивірусні інструменти, а деякі антивірусні компанії тепер пропонують VPN.

Зазвичай ми не перевіряємо здатність VPN виявляти зловмисне програмне забезпечення, оскільки розглядаємо VPN переважно як службу конфіденційності. Щоб усунути загрозу зловмисного програмного забезпечення, ми вважаємо, що автономне програмне забезпечення для захисту від зловмисного програмного забезпечення – куплене вами чи те, що постачається з вашим комп'ютером – справляється краще. Ми вважаємо, що VPN повинні приділяти якомога менше уваги вашому веб-трафіку.

Чи забезпечують VPN вашу безпеку в Інтернеті?

VPN приховає вміст вашого веб-трафіку від деяких спостерігачів і може ускладнити відстеження за вами в Інтернеті. Але VPN може в найкращому випадку забезпечити лише обмежений захист від загроз, з якими ви, швидше за все, зіткнетеся в Інтернеті: зловмисне програмне забезпечення, шахрайство соціальної інженерії та фішингові сайти.

Є кращі способи боротьби з цими загрозами. Ваш браузер має вбудовані інструменти для виявлення фішингових сайтів, як і більшість антивірусних програм, тому зверніть увагу, коли бачите попередження. Дотримуйтесь здорового глузду, якщо ви бачите підозріле спливаюче вікно або отримуєте незвичний електронний лист із проханням вжити певних дій. Багато людей повторно використовують паролі та використовують слабкі паролі, тому знайдіть менеджер паролів, щоб створювати та зберігати унікальні та складні паролі для

кожного сайту та служби, які ви використовуєте. Нарешті, захистіть свої онлайн-акаунти та ввімкніть багатофакторну автентифікацію скрізь, де вона доступна.

Коли VPN активна, увесь ваш трафік зашифровано. Це означає, що ваш провайдер не може бачити сайти, які ви відвідуєте, або файли, які ви переміщуєте.

Але хоча ваш інтернет-провайдер, можливо, не бачить, що ви завантажуєте весь пакет Great British Bake Off через торрент, він може припустити, що ви використовуєте велику пропускну здатність. Лише це може бути порушенням ваших умов. Піратський вміст також може порушувати умови використання VPN, тому уважно перевіряйте.

За допомогою VPN можна підключитися до сервера VPN в іншій країні та переглядати веб-сторінки так, ніби ви фізично перебуваєте там, де знаходиться сервер VPN. У деяких випадках це може обійти локальні обмеження вмісту та інші види цензури. Це найшляхетніше використання VPN, і компанії VPN часто відіграють свою роль у захисті свободи Інтернету.

Хоча це має працювати, важливо знати, що VPN не робить ваш трафік невидимим. Спостерігачі можуть бачити зашифрований трафік, але вони не повинні бачити вміст трафіку. Однак лише зашифрований трафік може привернути небажану увагу. Деякі мережі VPN включають режими, спрямовані на маскування трафіку VPN під звичайний трафік HTTPS.

Ми не перевіряємо здатність VPN обходити цензуру, і маємо серйозні занепокоєння щодо схвалення служби VPN, оскільки ця здатність може поставити під загрозу життя людей, якщо ми помилимося. Залежно від того, де ви перебуваєте, просте використання VPN може призвести до легальних проблем, тому знайте про ризики, перш ніж спробувати. Пам’ятайте, що жоден інструмент не може забезпечити повний захист, особливо від добре фінансованого та дієздатного супротивника, наприклад, національної держави.

За допомогою VPN ви можете підключитися до сервера в іншій країні та підробити своє місцезнаходження. Один із способів визначити, де знаходиться

комп’ютер, підключений до Інтернету, – подивитися на його IP-адресу. Ці адреси розподіляються географічно і іноді можуть бути досить близькими до вашого справжнього місцезнаходження. Якщо приховати свою справжню IP-адресу за IP-адресою сервера VPN, ваше справжнє місцезнаходження може бути прихованим.

Але пам’ятайте, що сайти та служби іноді мають інші засоби визначення вашого місцезнаходження. Крім того, багато сайтів чутливі до змін очікуваної поведінки. Якщо ваш банк побачить, що хтось, видаючи себе за вас, підключається з Латвії, він може вимагати від нього виконати додаткові перевірки безпеки, перш ніж надавати доступ. Загалом це добре, але може бути страшно, коли VPN використовуєте ви, а не шахрай.

Служби потокового передавання іноді пропонують різний вміст для різних країн. Донедавна жителі Великобританії могли дивитися «Зоряний шлях: Відкриття» на Netflix, тоді як жителі США мали використовувати Paramount+. Не виходячи з дому, ви можете зайти на віддалений сервер VPN, можливо, отримати доступ до потокового відео, недоступного в США.

Подібно до урядової цензури, потокові служби знають, що багато людей використовують VPN для доступу до свого вмісту та активно працюють над запобіганням цьому. Отже, хоча ви можете використовувати VPN для потокової передачі відео в Інтернеті, і ми впевнені, що більшість із вас, хто читає це, це так, це може працювати, але також може припинити працювати завтра.

Найбільша проблема з VPN – це не проблема технології, а проблема довіри. Оскільки весь ваш трафік проходить через її системи, компанія VPN перебуває в тому ж становищі, що й провайдер. За бажання він міг би бачити все, що ви робите в Інтернеті, і продавати ці дані. Це може вставляти рекламу на веб-сайти, які ви переглядаєте. Він міг зберігати непотрібні обсяги даних, які потім був змушений передати правоохоронним органам.

VPN прагнуть отримати таку довіру, але довести, що вони заслуговують такої довіри, важко. Коли ми перевіряємо VPN, ми вивчаємо її політику конфіденційності та надсилаємо анкету, щоб зрозуміти, яких зусиль кожна

компанія докладає для захисту конфіденційності клієнтів. Ми знаємо, що вони можуть нам збрехати, але наша мета – зафіксувати їх.

Ми хочемо, щоб мережі VPN вживали всіх можливих заходів для захисту своїх клієнтів, але нам також потрібна прозорість. Навіть якщо ми не згодні з усіма їхніми виборами, ми віддаємо перевагу компаніям, які відверто розповідають про свою діяльність. VPN також має опублікувати звіт про прозорість із зазначенням запитів, які компанія отримувала від правоохоронних органів, і як компанія реагувала.

Ми також хотіли б бачити сторонні аудити служб VPN, які перевіряють політики та безпеку інфраструктури компанії. Треба визнати, що аудити є недосконалими інструментами. Аудит замовляє компанія VPN, і компанія також визначає обсяг аудиту. Тим не менш, це цінний спосіб продемонструвати прихильність компанії до прозорості.

Кілька років тому мережі VPN мали чіткіше визначене місце в наборі інструментів конфіденційності та безпеки. Тоді більшість трафіку проходила через HTTP, іноді без будь-якого шифрування. Зараз більшість веб-трафіку надсилається через HTTPS, який шифрує ваше з'єднання. Переглядаючи трафік HTTPS, Інтернет-провайдер або хтось, хто шпигує за вашою мережею, може побачити лише найвищий рівень призначення вашого трафіку. Це як побачити PCMag.com, а не PCMag.com/max-is-great.

Рекламодавці також стали більш досконалими у своїх зусиллях відстеження. Відбитки пальців браузера та інші методи означають, що анонімні можливості VPN дещо обмежені. Навіть розхвалена здатність VPN підробляти місцезнаходження, обходити цензуру та розблокувати потокове передавання є менш певною, оскільки компанії та уряди стають дедалі агресивнішими у виявленні та блокуванні трафіку VPN.

Зростання складних методів відстеження та HTTPS часто називають причиною того, що VPN не варті грошей. Але це залежить від того, навіщо вам потрібен VPN. Якщо з будь-якої причини ви хочете, щоб ваш трафік надходив з

іншої країни, це зробить VPN. Якщо ви хочете, щоб рекламодавцям та іншим було трохи складніше відстежувати вас під час переміщення в мережі, VPN також може допомогти зробити це. І якщо ви хочете переконатися, що ваш інтернет-провайдер знає якомога менше про вашу онлайн-діяльність, VPN також може допомогти.

VPN не зробить вас непереможними в Інтернеті, але може допомогти захистити вашу конфіденційність. Це цінна частина вашого інструментарію безпеки та конфіденційності, і, як і будь-який інструмент, VPN працює найкраще, якщо ви використовуєте його для правильної роботи.

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.

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.

ІТ-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.

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, ДСТ28147-89 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 ДСТ28147-89 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.

3.2 Розробка структурної схеми

Всі продукти для створення VPN можна умовно розділити на дві категорії: програмні й апаратні. Програмне рішення для VPN – це, як правило, готовий додаток, що встановлюється на підключеному до мережі окремому комп'ютері. Ряд виробників, такі як компанії Axent Technologies, Check Point Software Technologies і NetGuard, поставляють VPN-пакети, які легко інтегруються із програмними міжмережними екранами.

На відміну від них апаратні VPN-рішення містять у собі все, що необхідно для з'єднання, – комп'ютер, приватну (як правило) операційну систему й спеціальне програмне забезпечення. Розгортати апаратні рішення, безумовно, легше. Вони містять у собі все, що необхідно для конкретних умов, тому час, за яке їх можна запустити, обчислюється мінутами або годинниками. Ще однією серйозною перевагою апаратних VPN-рішень є набагато більше висока продуктивність. До мінусів апаратних VPN-рішень можна віднести їхню високу вартість. Ще один недолік таких рішень полягає в тому, що управляються вони окремо від інших рішень по безпеці, що ускладнює завдання адміністрування інфраструктури безпеки, особливо за умови недостачі співробітників відділу захисту інформації.

Існують також інтегровані рішення, у яких функції побудови VPN реалізуються поряд з функцією фільтрації мережного трафіка, забезпечення якості обслуговування або розподіли смуги пропущення. Основна перевага такого рішення – централізоване керування всіма компонентами з єдиної консолі.

Друга перевага – більш низька вартість розраховуючи на кожний компонент у порівнянні із ситуацією, коли такі компоненти здобуваються окремо. Прикладом такого інтегрованого рішення може служити VPN-1 від компанії Check Point Software, що включає в себе крім VPN-модуля, модуль, що реалізує функції міжмережного екрана, модуль, відповідальний за балансування навантаження, розподіл смуги пропущення й т.д.

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

Який би шлях не був обраний, однаково прийде зштовхнутися із проблемою керування VPN-пристроями й підтримки погоджених правил безпеки для VPN і міжмережних екранів у масштабах всієї організації. Якщо співробітники не мають достатні навички в цій області, можна довірити створення віртуальної приватної мережі незалежної компанії, що робить відповідні послуги.

Слід також зазначити, що використання VPN не є приводом для відмови від спеціалізованих засобів безпеки. По статистиці, до 80% всіх інцидентів, пов'язаних з інформаційною безпекою, відбувається з вини авторизованих користувачів, що мають санкціонований доступ у корпоративну мережу, а це значить, що атака або вірус від такого користувача будуть зашифровані й передані нарівні з необразливим трафіком.

І необхідно згадати ще одну особливість VPN – використання цієї технології знижує продуктивність мережі, що обумовлено затримками встановлення захищеного з'єднання між VPN-пристроями, затримками шифрування даних, затримками контролю їхньої цілісності й збільшеним трафіком через використання більше довгих заголовків пакетів.

Сучасні OpenVPN надають ті ж послуги, що й раніше, але з деякими доповненнями. Структурна схема побудови такої мережі показана на рис. 3.1.

Така мережа має наступні додаткові властивості:

– Користувач не знає, якими засобами організується з'єднання (наприклад, через Інтернет).

– Користувачеві надаються засоби безпеки, включаючи забезпечення конфіденційності й вірогідності інформації.

– Співробітники й клієнти організації можуть з'єднатися з несучою мережею й одержати всі необхідні налаштування автоматично.

– Звичайно користувачі додзвонюються до місцевого сервера й одержують через Інтернет доступ до віддалених серверів (наприклад, OpenVPN-сервер на рис. 3.1). Використовуючи Інтернет, користувач може скоротити витрати на міжміські дзвінки.

– PPP (Point-to-Point Protocol – протокол двоточкового з'єднання), віддалені сервери RADIUS (Remote Authentication Dial-In Service – система віддаленої авторизації користувачів по лініям, що комутируються) і IPSec (Internet Security Protocol – протокол безпеки Інтернету) стають важливими інструментами підтримки такої структури.

Тут треба ще раз підкреслити, що користуватися мережею Інтернет для безпечного зв'язку між пристроями користувачів значно дешевше, ніж у випадку використання виділених телефонних ліній або інших способів.

Інтернет виступає в ролі несучої мережі. Користувач підключається до інтернет-провайдеру, що передає дані на маршрутизатори Інтернету (і навпаки). Ці пристрої іноді використовуються як брандмауери доступу (access firewall), у цьому випадку їхня основна функція полягає в перевірці IP-адрес вхідних повідомлень на дійсність. Ці ж комп'ютери можуть використовуватися і як сервери OpenVPN. У цьому випадку вони виконують розшифровку даних, зашифрованих відправником.

Сервер OpenVPN може також бути розташований за брандмауером доступу (у цьому випадку – маршрутизатором Інтернету), при цьому він буде частиною локальної мережі, як показано на рис. 3.1. При такій реалізації маршрутизатор або сервер повинен передати дані на сервер OpenVPN для їхньої наступної розшифровки. Якщо ці дані – запит на з'єднання із системою, то вони передаються на відповідний сервер для перевірки даних користувача. Часто такий сервер виконує й функції автоматичного налаштування з'єднання.

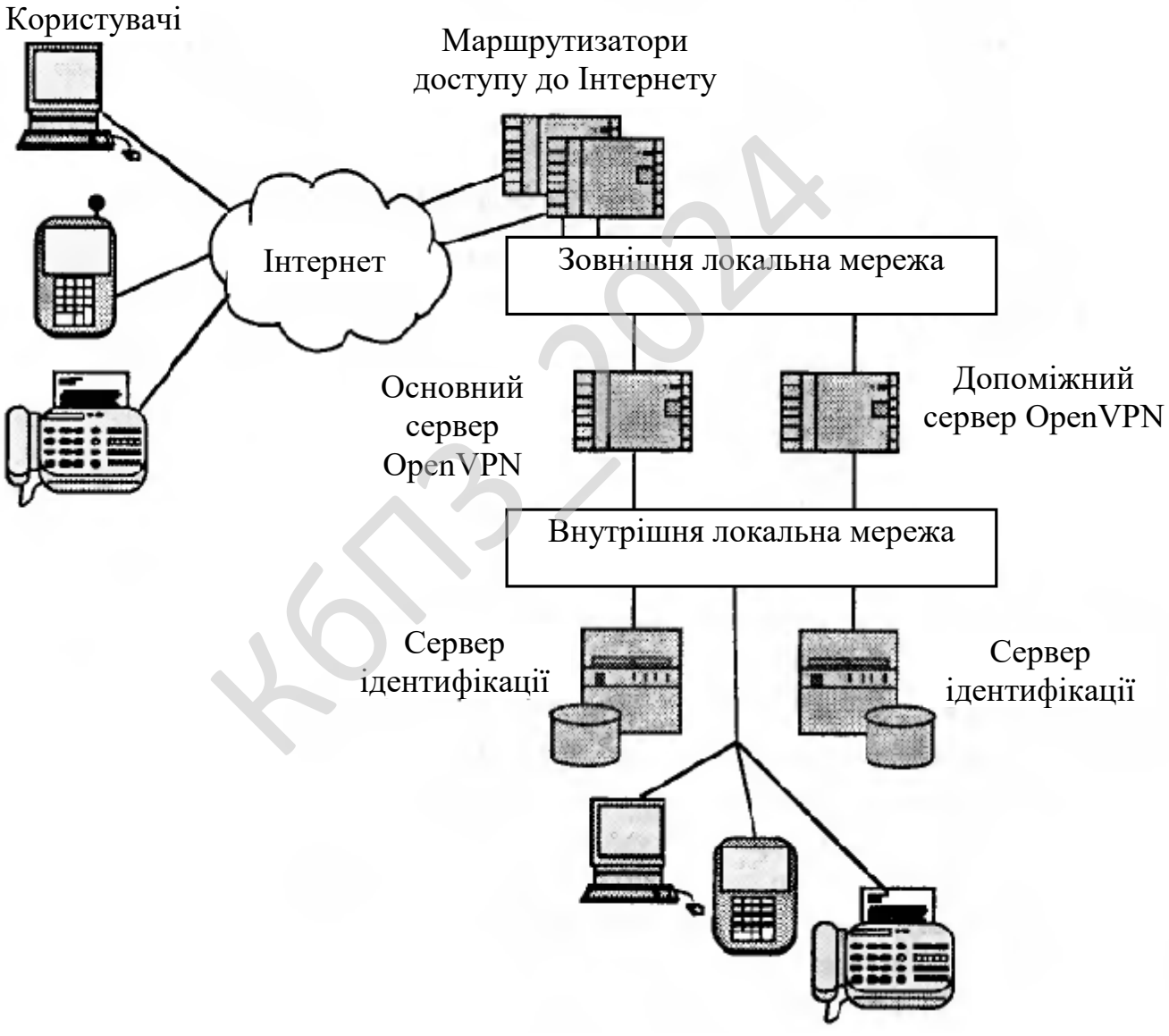

Рисунок 3.1 – Структурна схема системи

На рис. 3.1 показані кілька вузлів мережі, деякі з яких беруть участь у роботі OpenVPN, а інші – немає. Може виявитися більше зручним і ефективним (з погляду витрат) сполучити функції сервера перевірки прав користувача й OpenVPN-сервера на одному комп'ютері, тому що функції забезпечення конфіденційності й ідентифікації можуть бути найбільше ефективно реалізовані в рамках однієї функції (і отже, одного комп'ютера). Таким чином, немає нічого незвичайного в розміщенні функцій ідентифікації, кодування потоку даних і налаштування на вхідному шлюзі OpenVPN-мережі. З рис. 3.1 також видно, що найбільш важливі вузли системи дублюються допоміжними комп'ютерами.

Де б не був розташований OpenVPN-сервер, його основним завданням буде забезпечення безпеки логічних з'єднань (так званих IPSec-Тунелів) через Інтернет.

На рис. 3.1 показано, що користувачі можуть бути підключені до системи двома способами. При першому способі тунелі безпеки встановлюються один раз і залишаються без змін. Вони організуються між сайтами, які завжди перебувають на тому самому місці (наприклад, між відділенням компанії і її штаб-квартирою). Інший спосіб застосовується при роботі з окремими користувачами, які можуть дзвонити з різних місць: з мобільних телефонів, з номерів у готелі й т.п.

Налаштування OpenVPN під Windows

Для початку, звичайно, встановлюємо програму. Далі встановлюємо її в директорію «c:\openvpn», щоб потім не виникало зайвих проблем зі шляхами. Також відразу треба створити «c:\openvpn\ssl», після помістимо сюди всі наші «ключі», «c:\OpenVPN\log\openvpn.log» і «c:\OpenVPN\log\ openvpn-status.log» – для запису логів.

Приступаємо до редагування всіх конфігураційних файлів. Щоб фаєрвол не заважав майбутньому з'єднанню налаштуємо серверну частину на одному робочому місці, а клієнтську на іншому.

Налаштування сервера

Створюємо:

c:\openvpn\ easy-rsa\vars.bat

```
echo off
set path=%path%;c:\OpenVPN\bin
set HOME=c:\OpenVPN\ easy-rsa
set KEY_CONFIG=openssl.cnf
set KEY_DIR=c:\OpenVPN\ssl
set KEY_SIZE=1024
set KEY_COUNTRY=RU
set KEY_PROVINCE=mycity
set KEY_CITY= mycity
set KEY_ORG=Comp
set KEY_EMAIL=admin@local
```

c:\openvpn\ easy-rsa\openssl.cnf

```
HOME =.
RANDFILE = $ENV::HOME/.rnd
oid_section = new_oids

[ new_oids ]
[ ca ]
default_ca = CA_default

[ CA_default ]
dir = $ENV::KEY_DIR
certs = $dir
crl_dir = $dir
database = $dir/index.txt
new_certs_dir = $dir
certificate = $dir/ca.crt
serial = $dir/serial
crl = $dir/crl.pem
private_key = $dir/ca.key
RANDFILE = $dir/.rand
x509_extensions = usr_cert
default_days = 3650
default_crl_days= 30
default_md = md5
preserve = no
```

```
policy = policy_match

[ policy_match ]
countryName = match
stateOrProvinceName = match
organizationName = match
organizationalUnitName = optional
commonName = supplied
emailAddress = optional

[ policy_anything ]
countryName = optional
stateOrProvinceName = optional
localityName = optional
organizationName = optional
organizationalUnitName = optional
commonName = supplied
emailAddress = optional

[ req ]
default_bits = $ENV::KEY_SIZE
default_keyfile = privkey.pem
distinguished_name = req_distinguished_name
attributes = req_attributes
x509_extensions = v3_ca
string_mask = nombstr

[ req_distinguished_name ]
countryName = Country Name (2 letter code)
countryName_default = $ENV::KEY_COUNTRY
countryName_min = 2
countryName_max = 2
stateOrProvinceName = State or Province Name (full name)
stateOrProvinceName_default = $ENV::KEY_PROVINCE
localityName = Locality Name (eg, city)
localityName_default = $ENV::KEY_CITY
0.organizationName = Organization Name (eg, company)
0.organizationName_default = $ENV::KEY_ORG
organizationalUnitName = Organizational Unit Name (eg, section)
commonName = Common Name (eg, your name or your server\'s hostname)
commonName_max = 64
emailAddress = Email Address
```

```
emailAddress_default = $ENV::KEY_EMAIL
emailAddress_max = 40

[ req_attributes ]
challengePassword = A challenge password
challengePassword_min = 4
challengePassword_max = 20
unstructuredName = An optional company name

[ usr_cert ]
basicConstraints=CA:FALSE
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer:always

[ server ]
basicConstraints=CA:FALSE
nsCertType = server
nsComment = «OpenSSL Generated Server Certificate»
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer:always

[ v3_req ]
basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment

[ v3_ca ]
subjectKeyIdentifier=hash

authorityKeyIdentifier=keyid:always,issuer:always
basicConstraints = CA:true

[ crl_ext ]
authorityKeyIdentifier=keyid:always,issuer:always
```

Копіюємо index.txt.start в index.txt, a serial.start в serial у папку ssl.

Створюємо сертифікати

Відкриваємо командний рядок від імені адміністратора й виконуємо послідовно:

vars

clean-all

build-ca

#(приймаємо всі значення за замовчуванням натисканням клавіші Enter)

build-dh

build-key-server SERVER_NAME(на ваш вибір)

#при запиті на введення Common name необхідно ввести наше SERVER_NAME

Далі щоб уникнути проблем зі створенням сертифіката клієнта очищаємо index.txt папці ssl

buid-key KLIENT(на ваш вибір)

openvpn -igenkey -isecret %KEY_DIR%\ta.key

Створюємо server.ovpn у папці config і редагуємо його.

```
server.ovpn
dev tu
proto tcp-server
port 5190
tls-server
server 192.168.0.0 255.255.255.0
comp-lzo
dh C:\\OpenVPN\\ssl\\dh1024.pem
ca C:\\OpenVPN\\ssl\\ca.crt
cert C:\\OpenVPN\\ssl\\Server.crt
key C:\\OpenVPN\\ssl\\Server.key
tls-auth C:\\OpenVPN\\ssl\\ta.key 0
tun-mtu 1500
tun-mtu-extra 32
mssfix 1450
keepalive 10 120
status C:\\OpenVPN\\log\\ openvupn-status.log
log C:\\OpenVPN\\log\\openvpn.log
verb 3
```

Відправляємо CA.crt, klient.crt, klient.key, ta.key з «c:\openvpn\ssl» нашим клієнтам (поміщаємо їх у таку ж директорію « c:\openvpn\ssl»).

Налаштування клієнта

На комп'ютері клієнта необхідно встановити розроблений нами додаток по такому ж шляху c:\openvpn. Створюємо папку ssl і файли openvpn.log, openvpn-status.log

Створюємо clientVPN.ovpn у папці c:\openvpn\config і редагуємо його.

clientVPN.ovpn

```
dev tun
proto tcp
remote x.x.x.x 7777 (адреса сервера по мережі ip/dyndns)
route-delay 3
client
tls-client
ns-cert-type server
ca C:\\OpenVPN\\ssl\\ca.crt
cert C:\\OpenVPN\\ssl\\client.crt
key C:\\OpenVPN\\ssl\\client.key
tls-auth C:\\OpenVPN\\ssl\\ta.key 1
comp-lzo
tun-mtu 1500
tun-mtu-extra 32
mssfix 1450
ping-restart 60
ping 10
status C:\\OpenVPN\\log\\ openvpn-status.log
log C:\\OpenVPN\\log\\openvpn.log
verb 3
```

На сервері запускаємо файл server.ovpn(кнопка «StartOpenvpn...» у контекстному меню), на клієнті clientVPN.ovpn. При необхідності змінюємо тип запуску нашої служби(OpenVPN Service) на «Автоматично». Тунель піднятий, можете сміло заходити на роботу, допустимо по RDP. Адреса сервера в нашій віртуальній мережі буде 192.168.0.1.

3.3 Розробка функціональної схеми

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

Рисунок 3.2 – Функціональна схема системи

Розглянемо деякі з алгоритмів та складових реалізації OpenVPN з використанням бібліотеки OpenSSL

Криптографічні алгоритми

Blowfish

Blowfish є симетричним алгоритмом шифрування, тобто таким, у якому ключ шифрування дорівнює ключу дешифрування. Він є мережею Фейштеля, у якій кількість ітерацій дорівнює 16. Довжина блоку дорівнює 64 бітам, ключ може мати будь-яку довжину в межах 448 біт. Хоча перед початком будь-якого шифрування виконується складна фаза ініціалізації, саме шифрування даних виконується досить швидко.

Алгоритм призначений в основному для додатків, у яких ключ міняється нечасто, до того ж існує фаза початкового рукостискання, під час якої відбувається автентифікація сторін і узгодження загальних параметрів і секретів. При реалізації на 32-бітних мікропроцесорах з більшим кешем даних Blowfish значно швидше DES.

Алгоритм складається із двох частин: розширення ключа й шифрування даних. Розширення ключа перетворює ключ довжиною, принаймні, 448 біт у кілька масивів підключів загальною довжиною 4168 байт.

В основі алгоритму лежить мережа Фейштеля з 16 ітераціями. Кожна ітерація складається з перестановки, що залежить від ключа, і підстановки, що залежить від ключа й даних. Операціями є `XOR` і додавання 32-бітних слів.

Blowfish використовує велику кількість підключів. Ці ключі повинні бути обчислені заздалегідь, до початку будь-якого шифрування або дешифрування даних. Елементи алгоритму:

1. P – масив, що складається з вісімнадцяти 32-бітних підключів:

$$P_1, P_2, ..., P_{18}.$$

2. Чотири 32-бітних *S*-boxes с 256 входами кожний. Перший індекс означає номер S-box, другий індекс – номер входу.

$$S_{1,0}, S_{1,1}, \ldots S_{1,255};$$

$S_{2,0}, S_{2,1}, \ldots S_{2,255};$

$S_{3,0}, S_{3,1}, \ldots S_{3,255};$

$S_{4,0}, S_{4,1}, \ldots S_{4,255};$

Шифрування

Входом є 64-бітний елемент даних X, що ділиться на дві 32-бітні половини, X_l і X_r.

$X_l = X_l \text{ XOR } P_i$

$X_r = F(X_l) \text{ XOR } X_r$

Swap X_l and X_r

Функція F

Розділити X_l на чотири 8-бітних елементи A, B, C, D.

$$F(X_l) = ((S_{1,A} + S_{2,B} \bmod 2^{32}) \text{ XOR } S_{3,C}) + S_{4,D} \bmod 2^{32}$$

Дешифрування відрізняється від шифрування тим, що P_i використовуються у зворотному порядку.

Генерація підключів

Підключи обчислюються з використанням самого алгоритму Blowfish.

1. Ініціалізувати перший P-масив і чотири S-boxes фіксовані рядки.
2. Виконати операцію XOR P_1 з першими 32 бітами ключа, операцію XOR P_2 із другими 32 бітами ключа й т.д. Повторювати цикл доти, поки весь P-масив не буде побітово складний з усіма бітами ключа. Для коротких ключів виконується конкатенація ключа із самим собою.
3. Зашифрувати нульовий рядок алгоритмом Blowfish, використовуючи підключи, описані в пунктах (1) і (2).
4. Замінити P_1 і P_2 виходом, отриманим на кроці (3).
5. Зашифрувати вихід кроку (3), використовуючи алгоритм Blowfish з модифікованими підключами.
6. Замінити P_3 і P_4 виходом, отриманим на кроці (5).

7. Продовжити процес, заміняючи всі елементи *P*-масиву, а потім всі чотири *S*-boxes, виходами відповідним чином модифікованого алгоритму Blowfish.

Для створення всіх підключів потрібна 521 ітерація.

Md5

Md5отримує на вході повідомлення довільної довжини і створює на виході дайджест повідомлення довжиною 128 біт. Алгоритм складається з наступних кроків:

1. Додавання недостаючих біт. Повідомлення доповнюється так, щоб його довжина стала рівна 448 по модулю 512 (довжина 448 mod 512). Це означає, що довжина доданого повідомлення на 64 біта менше, ніж число, кратне 512. Додавання проводиться завжди, навіть якщо повідомлення має потрібну довжину. Наприклад, якщо довжина повідомлення 448 біт, воно доповнюється 512 бітами до 960 біт. Таким чином, число біт, що додаються, знаходиться в діапазоні від 1 до 512.

Додавання складається з одиниці, за якою слідує необхідна кількість нулів.

2. Додавання довжини. 64-бітове представлення довжини початкового (до додавання) повідомлення в бітах приєднується до результату першого кроку. Якщо первинна довжина більша, ніж 264, то використовуються тільки останні 64 біта. Таким чином, поле містить довжину початкового повідомлення по модулю 264.

В результаті перших двох кроків створюється повідомлення, довжина якого кратна 512 бітам. Це розширене повідомлення представляється як послідовність 512-бітових блоків Y_0, Y_1,..., Y_{l-1}, при цьому загальна довжина розширеного повідомлення рівна L*512 бітам. Таким чином, довжина отриманого розширеного повідомлення кратна шістнадцяти 32-бітовим словам.

3. Ініціалізація MD-буфера. У алгоритмі Md5 використовується 128-бітовий буфер для зберігання проміжних і остаточних результатів геш-функції.

Буфер може бути представлений як чотири 32-бітові регістри (*A*, *B*, *C*, *D*). Ці регістри ініціалізувалися наступними шістнадцятковими числами:

$$A = 01234567$$

$$B = 89abcdef$$

$$C = Fedcba98$$

$$D = 76543210$$

4. Обробка послідовності 512-бітових (16-словних) блоків. Основою алгоритму Md5 є модуль, що складається з чотирьох циклічних обробок, позначений як Hmd5. Чотири цикли мають схожу структуру, але кожен цикл використовує свою елементарну логічну функцію, *ff*, що позначається, *fg*, *fh* і *fi* відповідно.

Кожен цикл приймає на вхід поточний 512-бітовий блок Y_q, що обробляється в даний момент, і 128-бітове значення буфера ABCD, яке є проміжним значенням дайджесту, і змінює вміст цього буфера. Кожен цикл також використовує четвертую частина 64-елементної таблиці *T*[1 . 64], побудованої на основі функції sin. *i*-ий елемент *T*, *T*[i], що позначається, має значення, рівне цілій частині від 232 * abs (sin (*i*)), і задане в радіанах. Оскільки abs (sin (*i*)) є числом між 0 і 1, кожен елемент *T* є цілим, яке може бути представлене 32 бітами. Таблиця забезпечує “випадковий” набір 32-бітових значень, які повинні ліквідовувати будь-яку регулярність у вхідних даних.

Для отримання *Mdq*+1 вихід чотирьох циклів складається по модулю 232 з *Mdq*. Складання виконується незалежно для кожного з чотирьох слів в буфері.

5) Вихід Md5. Після обробки всіх *L* 512-бітових блоків виходом *L*-ої стадії є 128-бітовий дайджест повідомлення.

Детальніше логіку кожного з чотирьох циклів виконання одного 512-бітового блоку розглянуто нижче. Кожен цикл складається з 16 кроків, що оперують з буфером *ABCD*.

$$A \quad B + \mathrm{Clss}\,(A + f\,(B, C, D) + X\,[k] + T\,[i]),$$

A, *B*, *C*, *D* – чотири слова буфера; після виконання кожного окремого кроку відбувається циклічне зрушення вліво на одне слово.

f – одна з елементарних функцій *ff*, *fg*, *fh*, *fi*.

CLSs – циклічне зрушення вліво на s біт 32-бітового аргументу.

X [*k*] – *M* [*q* * 16 + *k*] – кодує 32-бітове слово в *q*-ому 512 блоці повідомлення.

T [*i*] – *i*-е 32-бітове слово в матриці T.

+ – складання по модулю 232.

На кожному з чотирьох циклів алгоритму використовується одна з чотирьох елементарних логічних функцій. Кожна елементарна функція отримує три 32-бітові слова на вході і на виході створює одне 32-бітове слово. Кожна функція є безліччю побітових логічних операцій, тобто *n*-ий біт виходу є функцією від n-ого біта трьох входів. Елементарні функції наступні:

$$ff = (B \ \& \ C) \ \ (\text{not } B \ \& \ D)$$

$$fg = (B \ \& \ D) \ \text{V} \ (C \ \& \ \text{not } D)$$

$$fh = B \ \ C \ \ D$$

$$fi = C \ \ (B \ \& \ \text{not } D)$$

Масив з 32-бітових слів *X* [0..15] містить значення поточного 512-бітового вхідного блоку, який обробляється зараз. Кожен цикл виконується 16 разів, а оскільки кожен блок вхідного повідомлення обробляється в чотирьох циклах, то кожен блок вхідного повідомлення обробляється по схемі 64 рази. Якщо представити вхідний 512-бітовий блок у вигляді шістнадцяти 32-бітових слів, то кожне вхідне 32-бітове слово використовується чотири рази, по одному разу в кожному циклі, і кожен елемент таблиці T, що складається з 64 32-бітових слів, використовується тільки один раз. Після кожного кроку циклу відбувається циклічне зрушення вліво чотирьох слів *A*, *B*, *C* і *D*. На кожному кроці змінюється тільки одне з чотирьох слів буфера *ABCD*. Отже, кожне слово буфера змінюється 16 разів, і потім 17-й раз в кінці для отримання остаточного виходу даного блоку.

RSA

Спочатку необхідно обчислити пару ключів (секретний ключ і відкритий ключ). Для цього відправник електронних документів обчислює два більших простих числа P и Q, потім знаходить їхній добуток $N = P*Q$ і значення функції $\varphi(N) = (P\text{-}1)(Q\text{-}1)$. Далі відправник обчислює число E з умов $E < \varphi(N)$, НЗД $(E, \varphi(N)) = 1$ і число D з умов $D < N$, $E*D \equiv 1 (\text{mod } \varphi(N))$.

Пари чисел (E, N) є відкритим ключем. Цю пару чисел автор передає партнерам по переписці для перевірки його цифрових підписів. Число D зберігається автором як секретний ключ для підписування.

Допустимо, що відправник хоче підписати повідомлення M перед його відправленням. Спочатку повідомлення M (блок інформації, файл, таблиця) стискають за допомогою геш-функції h(-) у ціле число m: $m = h(M)$.

Потім обчислюють цифровий підпис S під електронним документом M, використовуючи геш-значення m і секретний ключ D: $S = m \,(\text{mod } N)$.

Пари (M, S) передається партнерові-одержувачеві як електронний документ M, підписаний цифровим підписом S, причому підпис S сформований власником секретного ключа D.

Після прийому пари (M, S) одержувач обчислює геш-значення повідомлення M двома різними способами. Насамперед, він відновлює геш-значення m', застосовуючи криптографічне перетворення підпису S з використанням відкритого ключа E: $m' = S^E \,(\text{mod } N)$.

Крім того, він знаходить результат гешування прийнятого повідомлення M с допомогою такої ж геш-функції h(-): $m = h(M)$.

Якщо дотримується рівність обчислених значень, тобто $S^E(\text{mod}) = h(M)$, то одержувач визнає пару (M, S) справжньою. Доведено, що тільки власник секретного ключа D може сформувати цифровий підпис S по документі M, а визначити секретне число D по відкритому числу E не легше, ніж розкласти модуль N на множники. Крім того, можна строго математично довести, що результат перевірки цифрового підпису S буде позитивним тільки в тому

випадку, якщо при обчисленні *S* був використаний секретний ключ *D*, що відповідає відкритому ключу *E*. Тому відкритий ключ E іноді називають "ідентифікатором" того, хто підписав.

Функціональний блок реалізації OpenVPN з використанням бібліотеки OpenSSL

Протокол ISAKMP/Oakley

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

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

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

Обмін ключами в основному режимі здійснюється по методу Діффі-Хелмана (DH), що вимагає інтенсивного використання обчислювальних ресурсів. Цей метод є механізмом розподілу відкритих ключів для безпечного обміну секретною інформацією без застосування якої-небудь інформації, заздалегідь відомим обом сторонам. Тому ним активно користуються для встановлення безпечних сеансів зв'язку в тих випадках, коли необхідний динамічний захист і

коли кіцеві системи не належать однієй й тій же системі адміністративного керування. Наприклад, метод DH можна використовувати в електронній комерції при встановленні з'єднання для передачі транзакцій між двома компаніями.

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

В основному режимі обоє учасника обміну встановлюють SA-угоди для безпечного спілкування один з одним по протоколу Oakley. У швидкому режимі SA-угоди встановлюються вже "від імені" протоколу IPsec або будь-якої іншої служби, який необхідні дані для генерації ключів або узгодження параметрів. Протокол Oakley розроблений таким чином, що він ніяк не пов'язаний з IPsec. Наприклад, для підвищення безпеки процесу встановлення сеансів його цілком можна використовувати разом із протоколом SSL (Secure Sockets Layer) версії 4.0 замість механізму обміну ключами SSL 3.0.

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, гешування, 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 ESP – інкапсуляція зашифрованих даних

У випадку використання інкапсуляції зашифрованих даних заголовок ESP є останнім у ряді опціональних заголовків, "видимих" у пакеті. Оскільки основною метою ESP є забезпечення конфіденційності даних, різні види інформації можуть вимагати застосування істотно різних алгоритмів шифрування. Отже, формат ESP може перетерплювати значні зміни залежно від використовуваних криптографічних алгоритмів. Проте, можна виділити наступні обов'язкові поля: SPI (SPI – Secutity Parameter Index – індекс параметра безпеки), що вказує на контекст безпеки, поле порядкового номера, що містить послідовний номер пакета, і контрольна сума, призначена для захисту від атак на цілісність зашифрованих даних. Крім цього, як правило, у тілі ESP присутні

параметри (наприклад, режим використання) і дані (наприклад, вектор ініціалізації) застосовуваного алгоритму шифрування. Частина ESP заголовка може бути зашифрована на відкритому ключі одержувача або на спільному ключі пари відправник-одержувач. Одержувач пакета ESP розшифровує ESP заголовок і використовує параметри й дані застосовуваного алгоритму шифрування для декодування інформації транспортного рівня.

Розрізняють два режими застосування ESP – транспортний і тунельний.

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

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

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.

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 лініями зв'язку, що комутуються або по каналах локальних мереж і орієнтованих на виправлення випадкових помилок середовища передачі, механізми забезпечення цілісності даних у відкритих телекомунікаційних мережах повинні мати засоби захисту від внесення цілеспрямованих змін.

Одним з таких механізмів є спеціальне застосування алгоритму MD5: у процесі формування AH послідовно обчислюється геш-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. Розглянувши всі 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 3.3.

Рисунок 3.3 – Діаграма взаємодії процесів

З діаграми видно що після початку роботи програми проводиться виведення на екран інтерфейсу ПЗ тобто головного вікна ПЗ.

Після чого можна здійснювати створення VPN з'єднання, перегляд розширеної довідкової інформації та при необхідності проводити роботи з вже існуючими з'єднаннями – Обрання існуючого VPN з'єднання.

Якщо обирається існуюче VPN з'єднання проводиться підключення до VPN з'єднання з подальшим обміном даних з захистом алгоритмом OpenSSL чи модифікація VPN з'єднання з зміною назви VPN з'єднання, зміною логіну та паролю VPN з'єднання, зміна адреси VPN з'єднання.

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

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

Спершу відбувається виведення основного вікна програми. Після цього проводиться читання налаштувань та виведення списку існуючих з'єднань, з перевіркою створення VPN з'єднання чи ні. Якщо воно створене проводиться введення адреси VPN-сервера, перевірка прав доступу.

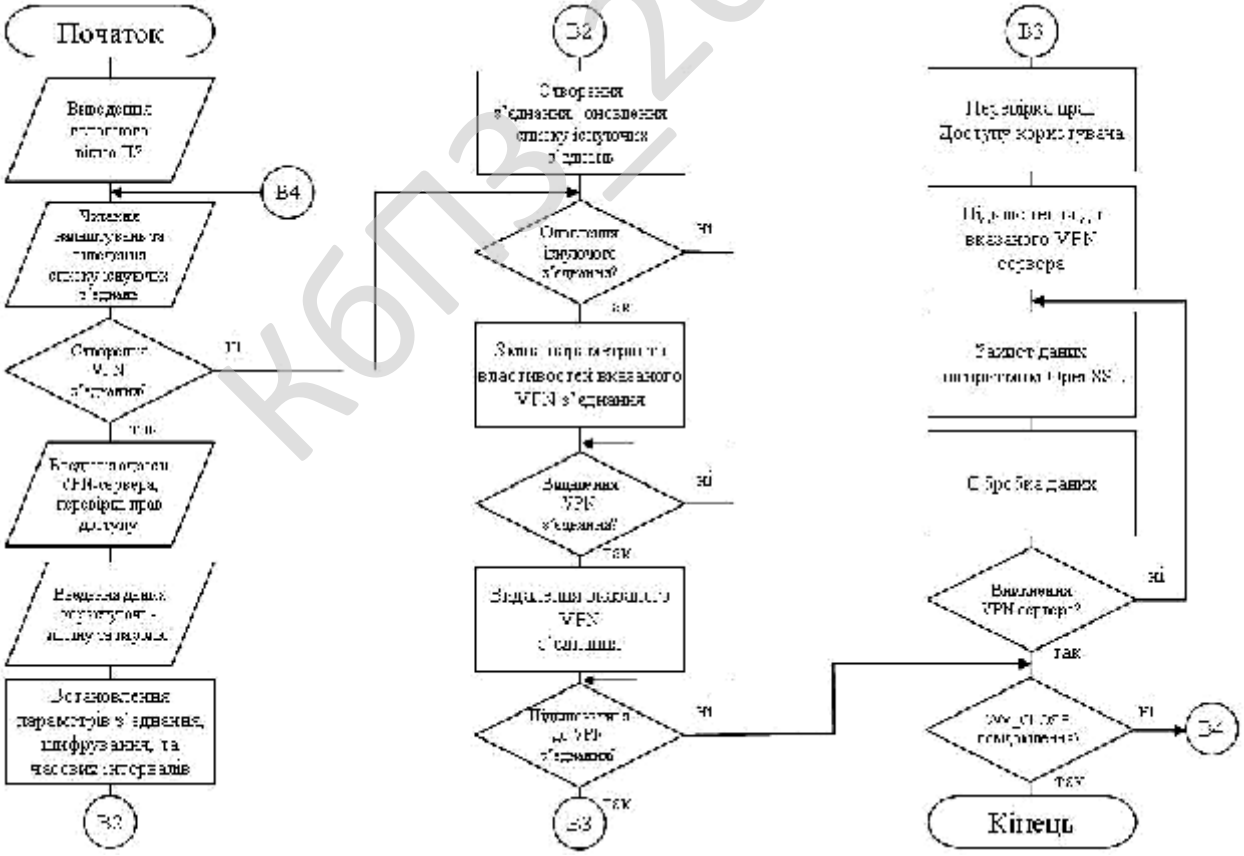

Рисунок 4.1 – Блок-схема основної програми

Далі проходе введення даних користувача – таких як логін та пароль. Встановлюються параметри з'єднання, шифрування, та часові інтервали. Проводиться створення з'єднання, оновлення списку існуючих з'єднань та запит на оновлення існуючого з'єднання. Якщо запит підтверджено проходе зміна параметрів та властивостей вказаного VPN з'єднання. Якщо ні то проходе запит видалення VPN з'єднання з подальшим видаленням вказаного VPN з'єднання. Після чого проводиться запит підключитися до VPN з'єднання з перевіркою прав доступу користувача, підключення до вказаного VPN сервера, захист даних алгоритмом OpenSSL та обробка даних. При надходженні сигналу вимкнення VPN сервера та повідомлення WM_CLOSE розроблене ПЗ завершується.

На рисунку 4.2 зображено роботу підпрограми захисту інформації з використанням бібліотеки OpenSSL.

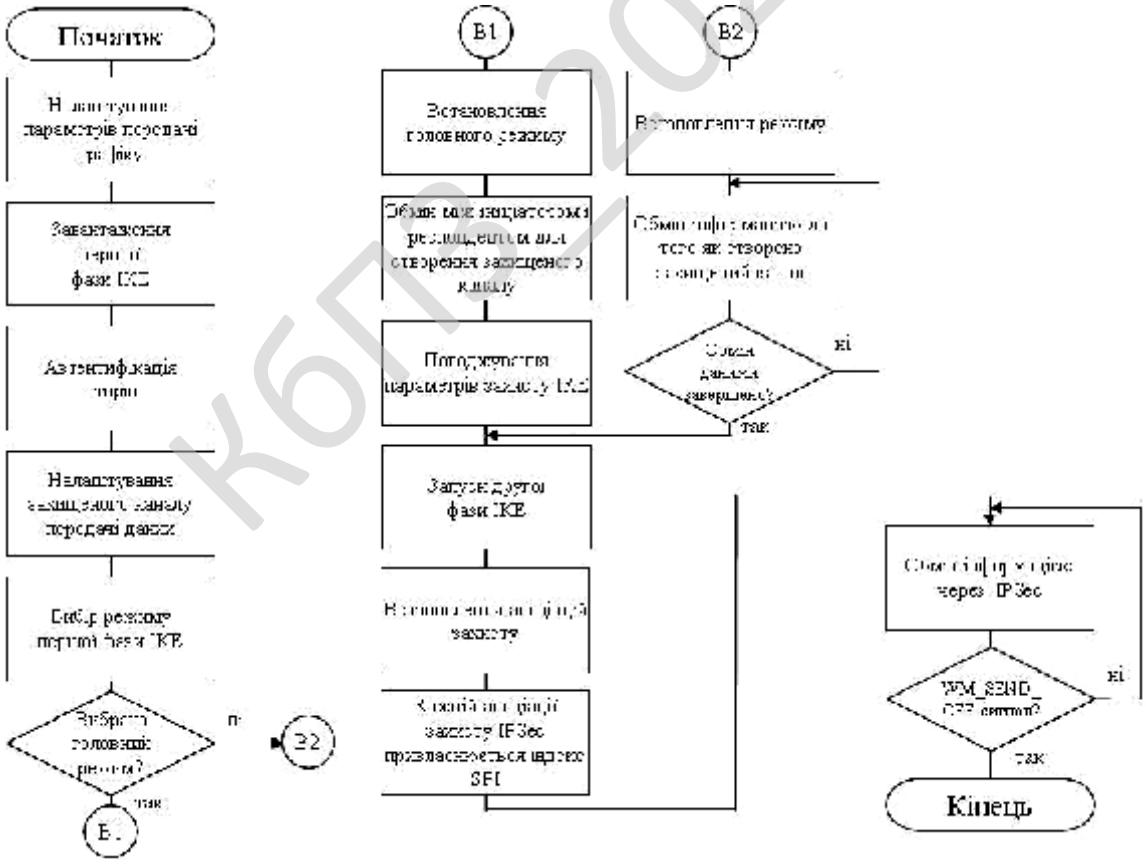

Рисунок 4.2 – Блок-схема підпрограми захисту інформації з використанням бібліотеки OpenSSL

Де після початку роботи проводиться налаштування параметрів передачі трафіку, завантаження першої фази, автентифікація сторін, налаштування захищеного каналу передачі даних, вибір режиму першої фази.

Далі слідує перевірка обрання головного режиму, якщо його обрано проводиться встановлення головного режиму, проводиться обмін між ініціатором і респондентом для створення захищеного каналу, погоджування параметрів захисту, запуск другої фази, встановлення асоціацій захисту, кожній асоціації захисту IPSec привласнюється індекс SPI. Якщо було обрано не головний режим проводиться встановлення режиму з послідуючим обміном інформацією до того як створено захищений канал далі якщо обмін даними завершено проходе запуск другої фази, встановлення асоціацій захисту, кожній асоціації захисту IPSec привласнюється індекс SPI. Далі проходе обмін інформацією через IPSec до того часу коли не з'явиться сигнал WM_SEND_OFF.

Для вирішення поставленого завдання необхідно реалізувати завдання обміну між windows-додатком і web-сервером.

Необхідно просто передати на сервер деякі дані і отримати відповідь, виключивши при цьому можливість підміни сервера (шляхом правки файлу hosts) і відповідно уникнути атаки підміною даних відповіді від помилкового сервера.

Клієнтський додаток розроблявся на Delphi, а в якості сервера – Apache+PHP.

Інтуїтивно зрозуміло, що дана задача вирішується при використанні інфраструктури відкритих/закритих ключів для шифрування трафіку, і вибір природно припав на алгоритм RSA .

Говорячи про сам алгоритмі RSA, хочу відзначити кілька неочевидних фактів, підтверджених експериментами:

1. Які б загадкові терміни (ключ, сертифікат, цифровий підпис і т.д.) не використовували при описі його роботи, заснований він усього лише на 3-х числах: модулі, приватній і публічної експоненті.

Модуль і приватна експонента, у відмінності від експоненти публічної, це дуже великі числа, розрядність яких визначається в алгоритмі і може варіюватися до 1024 біт (стандартне значення – 512).

Публічна ж експонента, як правило, дорівнює 65537. Цих 3-х чисел достатньо для реалізації асиметричного шифрування RSA.

2. Вважається, що шифрування має здійснюватися відкритим ключем, а дешифрування – закритим. Це не так. Насправді навіть якщо зашифрувати дані приватним ключем – вони прекрасно розшифрує публічним.

Реалізація RSA на PHP лише одна – php-OpenSSL, тут вибирати не доводиться. А ось в Delphi, вибір класів і компонент для шифрування значно ширше:

– TurboPower LockBox. Вже не підтримуються, вихідні коди викладені в публічний доступ , є порт tbLockBox для Delphi 2009. Інтуїтивно зрозуміла, добре документована і достатньо проста бібліотека.

– SecureBlackbox. Комерційна бібліотека без вихідних кодів. Досить громіздка і не зовсім зручна.

– OpenSSL. Можна використовувати і на клієнті. Для цього необхідно буде включити в проект бібліотеку libeay32.dll. Заголовний файл libeay32.pas на Delphi.

– Windows CryptoAPI. Рідна Windows криптосистема, відповідно, програма не стає важкою, і автоматично знімаються проблеми сумісності бібліотеки з майбутніми версіями Delphi. Заголовний файл Wcrypt2.pas добре працює на Delphi.

Всі бібліотеки дозволяють генерувати пари ключів для роботи. У OpenSSL для цього використовується однойменна утиліта openssl (необхідно встановити OpenSSL for Windows).

Генерація приватного 1024-бітного ключа:

```
openssl genrsa – out private. pem 1024
```

Створення парного публічного ключа:

```
openssl rsa-pubout-in private.pem-out public.pem
```

Відображення змісту ключа:

```
openssl rsa-text-in private.pem
```

У tbLockBox для генерації пари RSA-ключів використовується функція TtbRSA.GenerateKeyPair (). А в CryptoAPI функція CryptGenKey () .

Однак при спробі використовувати пару ключів у різних системах виникає проблема формату ключів:

– openssl-функції PHP працюють лише з ключами у форматі PEM, який містить base64 кодовану ASN.1 структуру, що містить всі дані ключа.

– tbLockBox зберігає і читає ключі в двійковому ASN.1 форматі, однак структура цих даних не відповідає структурі, використовуваної в PEM-файлах. Крім того, для приватних ключів використовується лише модуль і приватна експонента, в той час як PEM-файли приватних ключів містять і інші дані, необхідні для роботи openssl.

– CryptoAPI для експорту та імпорту ключів використовує з структури PRIVATEKEYBLOB і PUBLICKEYBLOB.

У процесі вирішення проблеми конвертації і вивчення структури файлів ключів, була знайдена чудова утиліта ASN.1Editor, яка сильно допомогла зрозуміти структуру RSA-ключів.

В результаті довгих пошуків був знайдений єдиний робочий спосіб – конвертації CryptoAPI структури PRIVATEKEYBLOB в PEM-файл придатний для використання в OpenSSL.

Спосіб цей полягає у використанні утиліти OpenSSL і вказівки формату вихідного файлу "MS PRIVATEKEYBLOB":

```
openssl rsa-inform MS\PRIVATEKEYBLOB-in private.dat-outform PEM-out private.pem
```

Зверніть увагу на зворотну косу риску, вона екранує символ пробілу в командному рядку.

І тут же слід одна дуже важливе застереження: формат "MS PRIVATEKEYBLOB" підтримується openssl лише починаючи з версії 1.0.0 beta.

Виходячи з усього вище сказаного випливає, що на клієнті повинне використовуватися CryptoAPI. У ньому ж необхідно згенерувати ключі і зберегти їх у форматі PRIVATEKEYBLOB, а потім конвертуватись у PEM-формат, придатний для використання в PHP.

Невелике пояснення: PRIVATEKEYBLOB так само як і PEM-файл, містять дані не тільки приватного, а й публічного ключа. І конвертування лише PRIVATEKEYBLOB можна отримати потім з PEM і відкритий ключ (openssl rsa-pubout-in private.pem-out public.pem).

Ось частина програми, що демонструє роботу з читання публічного ключа і шифровки тексту (рядки в php ANSI, тому перед шифруванням перетворимо Unicode в ANSI):

```
var
Stream: TMemoryStream;
res: boolean;
RSA: HCRYPTPROV;
PublicKey: HCRYPTKEY;
str: ANSIString;
strlen: DWORD;
begin
// Читаємо ключ з файлу
Stream: = TMemoryStream.Create;
Stream.LoadFromFile ('public.key');
// Ініціалізаіруем CryptoAPI
CryptAcquireContext (@ RSA, nil, nil, PROV_RSA_FULL, CRYPT_VERIFYCONTEXT);
// Імпортуємо ключ
CryptImportKey (RSA, PByte (Stream.Memory), Stream.Size, 0, 0, @ PublicKey);
// Беремо шіфруемий текст, перетворюючи його з Unicode в ANSI
str: = Edit1.Text;
// Обчислюємо розмір зашифрованих даних
strlen: = Length (str);
CryptEncrypt (PublicKey, 0, true, 0, nil, @ strlen, 0);
// Підготовляємо буфер потрібного розміру
Stream.SetSize (strlen);
strlen: = Length (str);
CopyMemory (Stream.Memory, pointer (str), strlen);
```

```
// Кріптуем
CryptEncrypt (PublicKey, 0, true, 0, PByte (Stream.Memory), @ strlen,
Stream.Size);
// Зберігаємо результат в файл
Stream.SaveToFile ('encrypted.dat');
// Звільняємо зайняті ресурси
Stream.Free;
CryptDestroyKey (PublicKey);
CryptReleaseContext (RSA, 0);
```

Для передачі зашифрованих даних серверу, зручніше використовувати кодування base64, в якій бінарні дані мають текстове представлення. Для цієї мети відмінно підійшов модуль DCPbase64.pas з біблотекі DCPcrypt :

```
SetLength (str, ((Stream.Size + 2) div 3) * 4);
Base64Encode (pointer (Stream.Memory), pointer (str), Stream.Size);
Edit1.Text: = str;
...
```

Переходячи до розгляду серверної частини, відразу зазначу, що є одна неврахована мною особливість зв'язки CryptoAPI і OpenSSL, через яку відразу все не запрацювало. А саме – порядок проходження байтів в зашифрованому повідомленні. Справа в тому, що в CryptoAPI використовується little-endian порядок, а в OpenSSL – big-endian. Тому потрібно перестановка (простіше зробити в php).

Ну і власне серверний приклад:

```
<?Php
	// Читаємо приватний ключ
$ PrivateKey = openssl_pkey_get_private (array ("file://private.pem", ""));
	if ($privateKey) print "\nPrivate Key OK";
else
print "\nPrivate key NOT OK";
	// Зашифроване повідомлення
	// Декодуємо
$Str = base64_decode ($str);
	// Міняємо порядок байт little-endian на big-endian
$Str = strrev ($str);
	// Дешифруемо
if (openssl_private_decrypt ($str, $res, $privateKey))
print "Result = $res";
```

```
else
print "Decrypting Error";?>
```

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

Пояснимо зміст полів, специфічних для AH:

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

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

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

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

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

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

Забезпечення конфіденційності мережного трафіку. Протокол інкапсулюючий захисту вмісту (Encapsulating Security Payload, ESP) надає три види сервісів безпеки:

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

– забезпечення цілісності IP-пакетів і автентифікації джерела даних;

– забезпечення захисту від відтворення IP-пакетів.

Можна бачити, що функціональність ESP ширше, ніж в AH (додається шифрування); взаємодія цих протоколів ми докладніше розглянемо пізніше. Тут же відзначимо, що ESP не обов'язково надає всі сервіси, але або конфіденційність, або автентифікація повинні бути задіяні. Формат заголовка ESP виглядає трохи незвичайно.

Причина в тім, що це не стільки заголовок, скільки обгортка (інкапсулююча оболонка) для зашифрованого вмісту. Наприклад, посилання на наступний заголовок не можна виносити в початок, у незашифровану частину, тому що вона втратиться конфіденційності.

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

Застосування протоколу ESP до вихідних пакетів можна уявляти собі в такий спосіб. Назвемо залишком пакета ту його частину, що міститься після передбачуваного місця вставки заголовка ESP.

При цьому не важливо, який режим використовується – транспортний або тунельний.

Кроки протоколу такі:

– залишок пакета копіюється в буфер;

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

– поточний уміст буфера шифрується;

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

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

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

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

Два захисних протоколи – AH і ESP – можуть комбінуватися різними способами. При виборі транспортного режиму AH повинен використовуватися після ESP (аналогічно тому, як у рамках ESP автентифікація йде слідом за шифруванням).

У тунельному режимі AH і ESP застосовуються, строго говорячи, до різного (вкладеним) пакетам, число припустимих комбінацій тут більше (хоча б тому, що можливо багаторазову вкладеність тунелів з різними початковими й/або кінцевими крапками).

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

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

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

Стандартизованими механізмами IP безпеки можуть (і повинні)користуватися протоколи більше високих рівнів і, зокрема, що управляють протоколи, протоколи конфігурування й маршрутизації.

Засоби безпеки для IP описуються сімейством специфікацій IPsec, розроблених робочою групою IP Security.

Протоколи IPsec забезпечують керування доступом , цілісність поза з'єднанням, автентифікацію джерела даних, захист від відтворення, конфіденційності, частковий захист від аналізу трафіку.

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

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

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

Протоколи забезпечення автентичності й конфіденційності в IPsec не залежать від конкретних криптографічних алгоритмів. (Більше того, саме розподіл на автентичність і конфіденційність надає й розроблювачам, і користувачам додатковий ступінь волі в ситуації, коли до криптографічного відносять тільки шифрувальні засоби.) У кожній країні можуть застосовуватися свої алгоритми, що відповідають національним стандартам, але для цього, як мінімум, потрібно подбати про їхню реєстрацію в домені інтерпретації.

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

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

У першому випадку захищається тільки вміст пакетів і, бути може, деякі поля заголовків. Як правило, транспортний режим використовується хостами. У тунельному режимі захищається весь пакет – він інкапсулюється в інший IP-пакет. Тунельний режим звичайно реалізують на спеціально виділених захисних шлюзах.

У наступних розділах ми докладно розглянемо основні елементи IPsec.

Протокольні контексти й політика безпеки. Системи, що реалізують IPsec, повинні підтримувати дві бази даних:

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

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

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

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

– пакет може бути ліквідований;

– пакет може бути оброблений без участі засобів IPsec;

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

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

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

Протокольний контекст безпеки в IPsec – це односпрямоване "з'єднання" (від джерела до одержувача), що надає обслуговуються потокам, що, даних набір захисних сервісів у рамках якогось одного протоколу (AH або ESP). У випадку симетричної взаємодії партнерам прийде організувати два контексти (по одному в кожному напрямку). Якщо використовуються й AH, і ESP, буде потрібно чотири контексти.

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

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

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

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

– час життя контексту;

– режим роботи IPsec: транспортний або тунельний;

– максимальний розмір пакетів;

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

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

Керуючі контексти – двосторонні, тобто кожний з партнерів може ініціювати новий ключовий обмін. Пара вузлів може одночасно підтримувати кілька активних керуючих контекстів, якщо є додатки з істотно різними криптографічними вимогами. Наприклад, припустимо вироблення частини ключів на основі попередньо розподіленого матеріалу, у той час як інша частина породжується по алгоритму Діффі-Хелмана.

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

IPsec зобов'язує підтримувати ручне й автоматичне керування контекстами безпеки й криптографічних ключів. У першому випадку всі системи заздалегідь забезпечуються ключовим матеріалом і іншими даними, необхідними для захищеної взаємодії з іншими системами. У другому – матеріал і дані виробляються динамічно, на основі певного протоколу – IKE, підтримка якого обов'язкова.

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

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

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

– SKEYID_a – ключовий матеріал для автентифікації.

– SKEYID_e – ключовий матеріал для шифрування.

Всі перераховані види ключів задіяні в обміні. Ключем SKEYID_e шифруються повідомлення. Ключ SKEYID_a служить аргументом геш-функцій і тим самим автентифікує повідомлення.

Нарешті, протокольні ключі – результат застосування псевдовипадкової (геш) функції до SKEYID_d з додатковими параметрами, у число яких входять одноразові номери ініціатора й партнера. У результаті створення протокольного контексту виявляється автентифікованим, захищеним від несанкціонованого ознайомлення, від відтворення повідомлень і від перехоплення з'єднання.

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

Строго говорячи, протокольні контексти відіграють допоміжну роль, будучи лише засобом проведення в життя політики безпеки; вона повинна бути задана для кожного мережного інтерфейсу із задіяними засобами IPsec і для кожного напрямку потоків даних (вхідні/вихідні). Відповідно до специфікацій IPsec, політика розраховується на безконтекстну (незалежну) обробку IP-пакетів, у дусі сучасних фільтруючих маршрутизаторів. Зрозуміло, повинні існувати засобу адміністрування бази даних SP, так само, як і засобу адміністрування бази

правил міжмережного екрана, однак цей аспект не входить до числа стандартизуємих.

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

– сукупність селекторів;

– сукупність протокольних контекстів безпеки.

Селектори служать для відбору пакетів, контексти задають необхідну обробку. Якщо правило посилається на неіснуючий контекст, воно повинне містити достатню інформацію для його (контексту) динамічного створення. Очевидно, у цьому випадку потрібна підтримка автоматичного керування контекстами й ключами. У принципі, функціонування системи може починатися із завдання бази SP при порожній базі контекстів (SAD); остання буде наповнюватися в міру необхідності.

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

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

Всі реалізації IPsec повинні підтримувати селекцію наступних елементів:

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

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

– транспортний протокол;

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

Обробка вихідні й вхідного трафіку не є симетричною. Для вихідних пакетів проглядається база SP, перебуває підходяще правило, витягають асоційовані з ним протокольні контексти й застосовуються відповідні механізми безпеки. У вхідних пакетах для кожного захисного протоколу вже проставлене значення SP, однозначно визначальний контекст. Перегляд бази SP у такому випадку не потрібно; можна вважати, що політика безпеки враховувалася при формуванні відповідного контексту. (Практично це означає, що ISAKMP-пакети мають потребу в особливому трактуванню, а правила з відповідними селекторами повинні бути включені в SP.)

Відзначена асиметрія, на наш погляд, відбиває певну незавершеність архітектури IPsec. У більше ранньому документі RFC 1825 поняття бази даних політики безпеки й селекторів були відсутні. У новій редакції специфікований перегляд бази SP як обов'язковий для кожного вихідного пакета, але не змінена обробка вхідних пакетів.

Звичайно, добування контексту по індексі SP ефективніше, ніж перегляд набору правил, але при такому підході, щонайменше, утрудняється оперативна зміна політики безпеки. Що стосується ефективності перегляду правил, те її можна підвищити методами кешування, широко використовуваними при реалізації IP.

Можливо, ще більш серйозним недоліком є неможливість узагальнення запропонованих процедур формування контекстів (керуючих і протокольних) на багатоадресний випадок. У поточних специфікаціях IPsec змішуються дві різні речі – область дії контексту (зараз це однобічний або двосторонній потік даних) і спосіб його ідентифікації (по індексі SP або парі ідентифікуючих ланцюжків). Виходить, що спосіб ідентифікації (іменування) нав'язує трактування області дії, що представляється невірним. На наш погляд, питання іменування можуть вирішуватися локально, а область дії контексту потенційно повинна поширюватися на довільне число партнерів.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм SEED – у криптографії симетричний блоковий криптоалгоритм на основі Мережі Фейстеля, розроблений Корейським агентством інформаційної безпеки (Korean Information Security Agency, KISA) в 1998 році. В алгоритмі використовується 128-бітний блок і ключ довжиною 128 біт. Алгоритм одержав широке поширення й використовується фінансовими й банківськими структурами, виробничими підприємствами й бюджетними установами Південної Кореї, оскільки 40-бітний SSL не забезпечує на даний момент мінімально необхідного рівня безпеки. Агентством по захисту інформації специфіковане використання шифру SEED у протоколах TLS і S/MIME. У той же час, алгоритм SEED не реалізований у більшості сучасних браузерів і інтернет-додатків, що утрудняє його використання в даній сфері поза межами Південної Кореї. SEED являє собою мережу Фейстеля з 16 раундами, 128-бітовими блоками й 128-бітовим ключем. Алгоритм використовує дві 8×8 таблиці підстановки, які, як такі з Safer, виведені з дискретного зведення в ступінь (у цьому випадку, x^{247} і x^{251} – плюс деякі «несумісні операції»). Це є деякою подібністю с MISTY1 у рекурсивності його структури: 128-бітовий повний шифр – мережа Фейстеля з F-функцією, що впливає на 64-бітові половини, у той час як сама F-функція – Мережа Фейстеля, складена з G-функції, що впливає на 32-розрядні половини. Однак рекурсія не простягнеться далі, тому що G-функція – не Мережа Фейстеля. В G-функції 32-розрядне слово розглядають як чотири 8-бітових байта, кожний з яких проходить через одну або іншу таблицю підстановки, потім поєднується в помірковано комплексному наборі булевих функцій таким чином, що кожний біт виводу залежить від 3 з 4 вхідних байтів. SEED має складний ключовий розклад, генеруючи тридцять два 32-розрядних додаткових символу, використовуючи G-функції на серіях обертань вихідного неопрацьованого ключа, комбінованого зі спеціальними раундовими константами (як в TEA) від «Золотого співвідношення» (англ. Golden ratio). Згідно з дослідженнями KISA, алгоритм SEED «надійно протистоїть відомим атакам».

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ яке зображено на рисунку 5.1. Після початку роботи програма проводе перевірку внутрішніх структур та починає проводити моніторинг мережі з одночасним чеканням можливих додаткових дій.

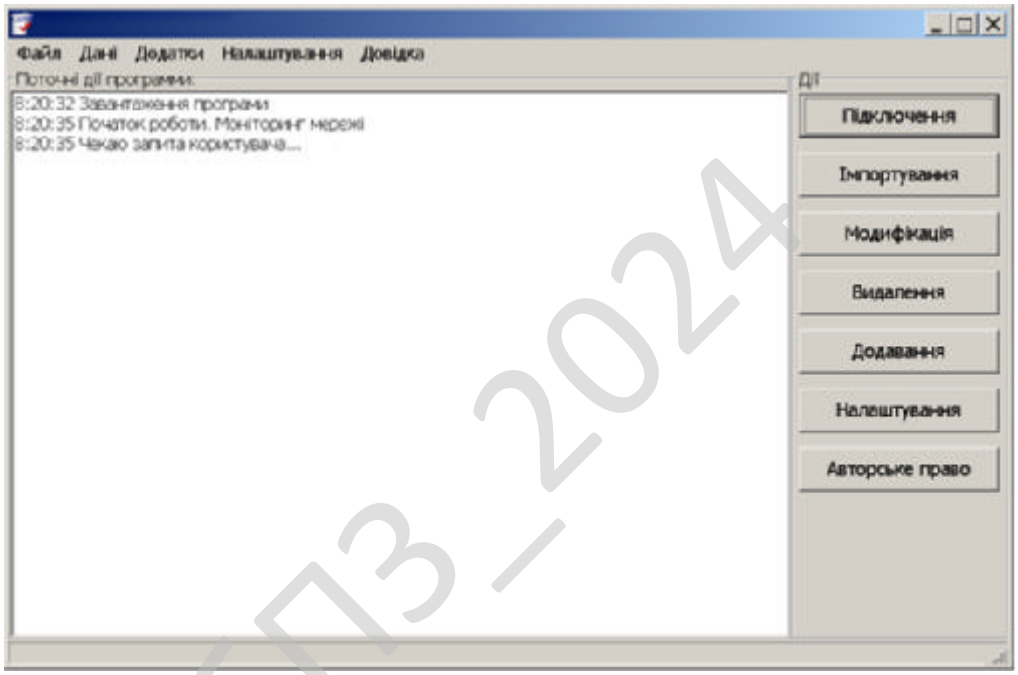

Рисунок 5.1 – Головне вікно програми

Основний функціонал ПЗ зображено у блоці дії. Він складається з наступних елементів:

– Підключення.

– Імпортування.

– Модифікація.

– Видалення.

– Додавання.

– Налаштування.

– Авторське право.

У вікні авторського права зображуються дані розробника, поточна версія програми, та назва кафедри.

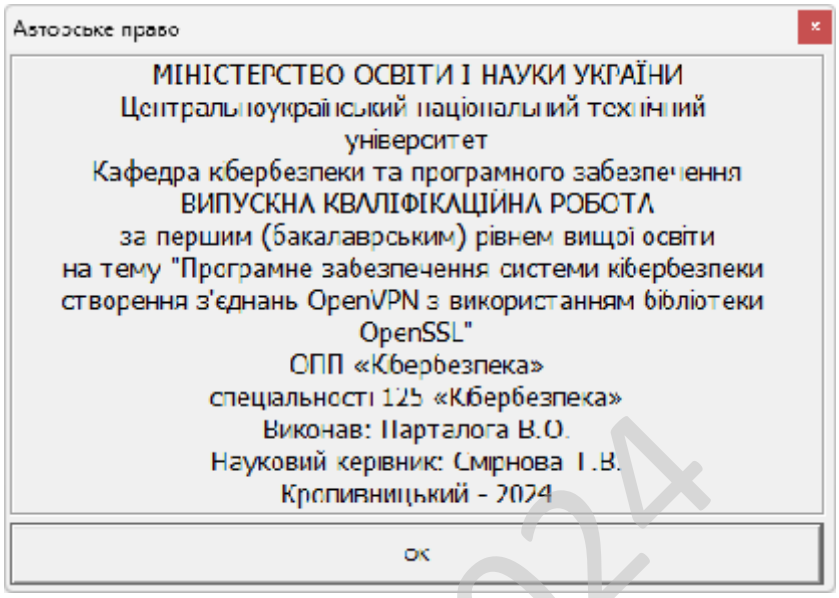

Рисунок 5.2 – Довідка ПЗ

Авторське право є ключовою галуззю права інтелектуальної власності; воно призначене захищати лише зовнішню форму вираження об'єкта, тобто їхнє матеріальне втілення. Авторське право не може використовуватись для захисту абстрактних ідей, концепцій, фактів, стилів та технік, що можуть бути використані у творі. Захист авторського права — одна з важливих категорій теорії цивільного та цивільно-процесуального права. Під захистом авторських прав слід розуміти передбачені законом заходи із їхнього визнання, припинення їхнього порушення, застосування до правопорушників заходів юридичної відповідальності. Захист особистих немайнових і майнових прав суб'єктів авторського права здійснюється в порядку, встановленому адміністративним, цивільним і кримінальним законодавством.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки створення з'єднань OpenVPN з використанням бібліотеки OpenSSL.

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 з'єднань OpenVPN з використанням бібліотеки OpenSSL.

– Досліджена система створення з'єднань OpenVPN з використанням бібліотеки OpenSSL.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки створення з'єднань OpenVPN з використанням бібліотеки OpenSSL.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання створення з'єднань OpenVPN з використанням бібліотеки OpenSSL.

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 Delphi 10.4. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи кібербезпеки створення з'єднань OpenVPN з використанням бібліотеки

OpenSSL. 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 SEED.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

15. Smirnov O., Kuznetsov A., Onikiychuk A., Makushenko T., Anisimova O., Arischenko A. «Adaptive pseudo-random sequence generation for spread spectrum

image steganography». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Ukraine, Kyiv, May 14-18. 2020. P. 161-165.

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

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

18. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

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

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

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

22. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019,

Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

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

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

25. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». *CEUR Workshop Proceedings*, Vol 2588, P. 90-106, 2019.

26. Smirnov, O., Kuznetsov, A., Kiian, A., Gorbenko, Y., Cherep, O., Bexhter L. «Code-based Pseudorandom Generator for the Post-Quantum Period», *2019 IEEE International Conference on Advanced Trends in Information Theory (IEEE ATIT 2019).* 18.12.19-20.12.19 Kyiv Ukraine. P. 204 – 209.

27. Smirnov, O., Kuznetsov, A., Nariezhnii, O., Stelnyk, S., Kokhanovska, T., Kuznetsova T., «Side Channel Attack on a Quantum Random Number Generator», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18 - 21 September 2019. P.713-718.

28. Kuznetsova, T., «Code-Based Schemes for Post-Quantum Digital Signatures», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019;* Metz; France; 18-21 September 2019. P. 707-712.

29. Smirnov, O., Kuznetsov, A., Stefanovych, O., Gorbenko, Y., Krasnobaev, V., Kuznetsova K. «Information Hiding Using 3D-Printing Technology»,

10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019; Metz; France; 18-21 September 2019. P.701-706.

30. Smirnov, O., Hu, Z., Vasiliu, Y., Sydorenko, V., Polishchuk, Y., «Abstract Model of Eavesdropper and Overview on Attacks in Quantum Cryptography Systems», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P.399-405.

31. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019*/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

32. Smirnov, O., Kuznetsov, A., Kiian, A., Babenko, B., Zhosan, H., Prokopovych-Tkachenko, D., «Soft Decoding Method for Turbo-Productive Codes», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT 2019*, Lviv, Ukraine, 2-6 July, 2019, P. 129-134.

33. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 353-358.

34. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 347-352.

35. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 618-629.

36. Smirnov, O., Kuznetsov, A., Kiian, A., Kuznetsova, K., Ivko, T., Prokopovych-Tkachenko, D., «Soft Decoding Based on Ordered Subsets of Verification Equations of Turbo-Productive Codes», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 873-884.

37. Smirnov A.A., Kuznetsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», *Telecommunications and Radio Engineering*. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

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

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

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

41. Козлов Я.О., Козірова Н.Л., Смірнов О.А. «Дослідження структури та принципу роботи SIEM-системи». *VII міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології" до 30-ти річчя кафедри кібербезпеки та програмного забезпечення,* м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 59.

42. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А., Коваленко А.С. «Дослідження нормативних документів та галузевих стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС,

важливих для безпеки». *Системи управління, навігації та зв'язку*, 2023, вип. 2(72), С. 170-178.

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

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

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

46. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New Technique for Hiding Data in Cover Images Using Adaptively Generated Pseudorandom Sequences». *CEUR Workshop Proceedings* Volume 2732, 2020, Pages 214-227.

47. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Книшук А.В. «Вступ до кібербезпеки»: навчальний посібник – Кропивницький: ЦНТУ – 2022. – 968 с.

48. Теорія та практика сучасного інформаційно-психологічного протиборства: навчальний посібник / [В.М. Петрик, С.О. Гнатюк, М.М. Присяжнюк та ін.]; за заг. ред. С.О. Гнатюка, В.М. Петрика та О.А Смірнова. – Полтава, 2022. – 334 с.

49. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». *International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019*; Odessa; Ukraine; 9-13 September 2019. P.22-28.

50. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Поліщук Л.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2020. – 294 с.

51. О.А. Смірнов, П.С. Усік, «Дослідження перспектив використання технологічних рішень в мережах 5G» у *Кібербезпека та інформаційні технології: монографія.* – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. 122-135.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування.....2
2 Підстава для розробки.....2
3 Мета та призначення розробки.....2
4 Джерела розробки.....2
5 Технічні вимоги.....2
5.1 Вміст проекту.....2
5.2 Показники призначення.....3
5.3 Вимоги до функціональних характеристик.....3
5.4 Вимоги до архітектури.....3
5.5 Вимоги до надійності.....3
5.6 Умови експлуатації.....4
5.7 Вимоги до складу та параметрів технічних засобів.....4
5.8 Вимоги до інформаційної і програмної сумісності.....4
5.8.1 Обладнання.....4
5.8.2 Мова програмування.....4
5.8.3 Вхідні дані.....5
5.8.4 Вихідні дані.....5
6 Вимоги до програмної документації.....5
7 Перелік документів, що розробляються.....5
8 Етапи розробки.....6
9 Порядок контролю та приймання.....6

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки створення з’єднань OpenVPN з використанням бібліотеки OpenSSL.

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. № 136-02 від 01.04.2024 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи кібербезпеки створення з’єднань OpenVPN з використанням бібліотеки OpenSSL.

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 з’єднань OpenVPN з використанням бібліотеки OpenSSL;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним 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 Delphi 10.4.

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 – 84 аркуші.

8 Етапи розробки

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

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

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2024 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 8.06.2024 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

_____ Смірнова Т.В.

Програмне забезпечення системи кібербезпеки створення з’єднань OpenVPN з використанням бібліотеки OpenSSL

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 54

Літера: РП

Кропивницький – 2024 року

OpenVPN_OpenSSL_WinMain.pas - основна програма

```
unit OpenVPN_OpenSSL_WinMain;

interface

uses
  Windows, Messages, CommCtrl, OpenVPN_OpenSSL_CommCtrl,
OpenVPN_OpenSSL_SysUtils, OpenVPN_OpenSSL_FileInfo, OpenVPN_OpenSSL_Resources,
  OpenVPN_OpenSSL_WelcWind, OpenVPN_OpenSSL_SettWind, OpenVPN_OpenSSL_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_STANDAROpenVPN_OpenSSL_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.
```

OpenVPN_OpenSSL_FinsWind.pas - VPN-з'єднання

```
unit OpenVPN_OpenSSL_FinsWind;

interface

uses
  Windows, Messages, CommCtrl, OpenVPN_OpenSSL_Windows,
OpenVPN_OpenSSL_Controls, OpenVPN_OpenSSL_FileInfo, OpenVPN_OpenSSL_SysUtils,
  OpenVPN_OpenSSL_MyMsgBox, OpenVPN_OpenSSL_Ole2, OpenVPN_OpenSSL_Active,
OpenVPN_OpenSSL_ShlObj, OpenVPN_OpenSSL_RasApi, OpenVPN_OpenSSL_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,
IIOpenVPN_OpenSSL_IShellFolder, Network);
              Network.EnumObjects(0, SHCONTOpenVPN_OpenSSL_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(CLSIOpenVPN_OpenSSL_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 OpenVPN_OpenSSL 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
(OpenVPN_OpenSSL)
    rEntry.dwType                   := RASET_Vpn;

    // Значення списку, що випадає, 'Тип OpenVPN_OpenSSL' = '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 OpenVPN_OpenSSL 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);
          OpenVPN_OpenSSL_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_OpenVPN_OpenSSLINFO),
        [
        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_OpenVPN_OpenSSLINFO),
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.
```

OpenVPN_OpenSSL_WinSvc.pas - сервіси

```
unit OpenVPN_OpenSSL_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         = (STANDAROpenVPN_OpenSSL_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.
```

OpenVPN_OpenSSL_SysUtils.pas - утиліти

```
unit OpenVPN_OpenSSL_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.
```

OpenVPN_OpenSSL_StatAnim.pas – анімація при підключенні OpenVPN_OpenSSL

```
unit OpenVPN_OpenSSL_StatAnim;

interface

uses
  Windows, Messages, CommCtrl, OpenVPN_OpenSSL_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,
PROpenVPN_OpenSSL_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,
      ILOpenVPN_OpenSSL_NORMAL or ILOpenVPN_OpenSSL_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.
```

OpenVPN_OpenSSL_ShlObj.pas - інтерпретатор команд операційної системи

```
unit OpenVPN_OpenSSL_ShlObj;

interface

uses
  Windows, OpenVPN_OpenSSL_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
  IIOpenVPN_OpenSSL_IShellFolder: TGUID = (D1: $000214E6; D2: $0000; D3: $0000;
D4: ($C0, $00, $00, $00, $00, $00, $00, $46));
  CLSIOpenVPN_OpenSSL_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
  SHCONTOpenVPN_OpenSSL_NONFOLDERS = 64;

{ Рядкові константи для інтерфейсу IDs }

const
  SIOpenVPN_OpenSSL_IShellLink  = '{000214 0000-0000-C 000-000000000046}';
  SIOpenVPN_OpenSSL_IShellLink  = '{000214F 0000-0000-C 000-000000000046}';
  SIOpenVPN_OpenSSL_IShellFolder = '{000214E 0000-0000-C 000-000000000046}';
  SIOpenVPN_OpenSSL_IEnumIDList  = '{000214F 0000-0000-C 000-000000000046}';

{ IShellLink інтерфейс }

type
  IShellLink = interface(IUnknown)
    [SIOpenVPN_OpenSSL_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)
    [SIOpenVPN_OpenSSL_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)
    [SIOpenVPN_OpenSSL_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)
    [SIOpenVPN_OpenSSL_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.
```

OpenVPN_OpenSSL_Resources.pas – ресурси OpenVPN_OpenSSL

```
unit OpenVPN_OpenSSL_Resources;

interface

uses
  Windows, CommCtrl, OpenVPN_OpenSSL_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_OpenVPN_OpenSSLINFO  = 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_OpenVPN_OpenSSLINFO   = 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.
```

OpenVPN_OpenSSL_RasApi.pas - з'єднання OpenVPN_OpenSSL

```
  unit OpenVPN_OpenSSL_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; // OpenVPN_OpenSSL
// 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.
```

OpenVPN_OpenSSL_MyMsgBox.pas – повідомлення OpenVPN_OpenSSL

```
  unit OpenVPN_OpenSSL_MyMsgBox;

interface

uses
  Windows, Messages, OpenVPN_OpenSSL_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.
```

OpenVPN_OpenSSL_LinkStat.pas - інтерфейс користувача

```
unit OpenVPN_OpenSSL_LinkStat;

interface

uses
  Windows, Messages, CommCtrl, OpenVPN_OpenSSL_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   := TTOpenVPN_OpenSSL_SUBCLASS or
TTOpenVPN_OpenSSL_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.
```

OpenVPN_OpenSSL_SettWind.pas - параметри програми роботи з OpenVPN_OpenSSL

```
unit OpenVPN_OpenSSL_SettWind;

interface

uses
  Windows, Messages, CommCtrl, OpenVPN_OpenSSL_FileInfo,
OpenVPN_OpenSSL_LinkStat, OpenVPN_OpenSSL_SysUtils, OpenVPN_OpenSSL_MyMsgBox,
  OpenVPN_OpenSSL_Controls, OpenVPN_OpenSSL_Resources, OpenVPN_OpenSSL_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.
```

OpenVPN_OpenSSL_ScanProc.pas - пошук підключень OpenVPN_OpenSSL

```
unit OpenVPN_OpenSSL_ScanProc;

interface

uses
  Windows, Messages, CommCtrl, WinSock, OpenVPN_OpenSSL_SysUtils,
OpenVPN_OpenSSL_StatAnim, OpenVPN_OpenSSL_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, THREAOpenVPN_OpenSSL_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.
```