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

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2023 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки передачі даних у мережі за протоколом SSL/TLS”

Виконав здобувач вищої освіти
IV курсу, групи КБ-21-2СК
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Черноліс Е.Р.
«____» _____ 2023 р.

Керівник проекту
кандидат технічних наук
_____ Смірнова Т.В.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

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

Спеціальність *125 “*Кібербезпека*”*

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

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

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

Чернолісу Едуарду Руслановичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки передачі даних у мережі за протоколом SSL/TLS*

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

затверджені наказом вищого навчального закладу № 14-02 від 5.01.2023 року

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

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи кібербезпеки передачі даних у мережі за протоколом SSL/TLS*

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

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи кібербезпеки в промислову експлуатацію.

6. Висновки

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

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

7. Дата видачі завдання « 17 » січня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2023 р.	
3.	Розробка моделі компонента	20.03.2023 р.	
4.	Розробка структур даних	25.03.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2023 р.	
6.	Програмування алгоритмів	10.04.2023 р.	
7.	Оформлення ПЗ	17.04.2023 р.	
8.	Попередній захист роботи	23.05.2023 р.	

Дата видачі завдання
« 17 » січня 2023 р.

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

Завдання прийнято до виконання
« 17 » січня 2023 р.

Підпис здобувача
_____ Черноліс Е.Р.
(прізвище та ініціали)

АНОТАЦІЯ

Чернoліс Е.Р. Програмне забезпечення системи кібербезпеки передачі даних у мережі за протоколом SSL/TLS. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки передачі даних у мережі за протоколом SSL/TLS.

Метою розробки є програмне забезпечення системи кібербезпеки передачі даних у мережі за протоколом SSL/TLS.

Результат роботи – програмна реалізація системи кібербезпеки передачі даних у мережі за протоколом SSL/TLS.

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

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

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

Програму розроблено в середовищі Builder C++.

Ключові слова: кібербезпека, SSL/TLS

ABSTRACT

Chernolis E.R. Software of the cyber security system of data transmission in the network using the SSL/TLS protocol. 125 Cyber security. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this graduation thesis for the first (bachelor) level of higher education, software is developed, which is intended for the cyber security system of data transmission in the network using the SSL/TLS protocol.

The purpose of the development is the software of the cyber security system of data transmission in the network using the SSL/TLS protocol.

The result of the work is the software implementation of the cyber security system of data transmission in the network using the SSL/TLS protocol.

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

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

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

The program was developed in the Builder C++ environment.

Keywords: cyber security, SSL/TLS

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..... 5
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 7
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти..... 7
2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки та мови програмування..... 13
2.3 Розгорнута постановка завдання 15
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 17
3.1 Опис функціонування системи 17
3.2 Розробка структурної схеми..... 48
3.3 Розробка функціональної схеми 53
3.4 Розробка діаграми процесів..... 61
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ..... 64
4.1 Розробка блок-схем та опис алгоритмів функціонування системи..... 64
4.2 Захист розробленого програмного забезпечення..... 73
5 ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ 75
6 ОСНОВНІ ВИСНОВКИ..... 81
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 83

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

ВСТУП

Актуальність теми. Сьогодні Інтернет стали використовувати не тільки для роботи, але й він для багатьох став зручною площадкою для здійснення всіляких угод, у тому числі й фінансових. Тут можна продавати, купувати або обмінюватися грошима, тобто просто робити транзакції, такі ж, як і в банку.

Але для того, щоб убезпечити себе й бути впевненим у тому, що після здійснення оплати гроші надійдуть саме про призначення, може підтвердити SSL сертифікат, що є гарантією того, що ви заходите на захищену сторінку й всі дані, які передаються, теж перебувають під захистом, а це сьогодні дуже важливо. SSL-сертифікат дозволяє переконатися будь-якому користувачеві у тому, що він належить конкретній організації, і передані дані на сайті захищені. Визначити такий сайт можна по піктограмі висячого замка, кликнувши по якому можна довідатися всі дані про організацію, який він належить. При вході на такий ресурс сервер і браузер клієнта обмінюються ключами, і захищене з'єднання завжди здійснюється за протоколом https і тому всі дані передаються в зашифрованому виді й розшифровується за допомогою спеціального ключа. Сьогодні купити SSL сертифікат можна в спеціалізованих компаніях, але видає його Центр, Що Засвідчує, після перевірки всіх документів організації, що його здобуває. При цьому можна придбати кожної з SSL сертифікатів, того рівня безпеки, що необхідний.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки передачі даних у мережі за протоколом SSL/TLS.

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

– Огляд існуючих систем передачі даних у мережі за протоколом SSL/TLS.

– Дослідження системи кібербезпеки передачі даних у мережі за протоколом SSL/TLS.

– Програмна реалізація системи кібербезпеки передачі даних у мережі за протоколом SSL/TLS.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі передачі даних у мережі за протоколом SSL/TLS.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки передачі даних у мережі за протоколом SSL/TLS, 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 передачі даних у мережі за протоколом SSL. SSL сертифікат – це цифровий підпис доменного імені. SSL сертифікати забезпечують безпечне з'єднання й передачу даних між сервером і клієнтом. Технологія передачі даних (SSL) гарантує захист, схоронність і цілісність переданих даних.

SSL сертифікат підписаний довіреним Центром Сертифікації – це престижно. Якщо на Вашому сайті використовується SSL сертифікат, що підписаний довіреним Центром Сертифікації – довіра відвідувачів до такого сайту буде значно вища. Це особливо актуально для інтернет-магазинів, порталів, поштових сервісів і інших ресурсів працюючих з персональними й платіжними даними клієнтів.

Довірений SSL сертифікат дозволяє підключити на сайті прийом оплат через пластикові карти або організувати захищений розділ, де користувачі можуть працювати з даними по безпечному з'єднанню не побоюючись, що їхні дані потраплять не по призначенню. SSL сертифікат сумісні з усім популярними інтернет-браузерами, період випуску сертифіката від 1 робочого дня.

1.2 Область застосування

Використання SSL сертифіката є обов'язковим для банків, Інтернет-Магазинів, платіжних систем, і всіх організацій, які працюють із персональними даними, а також для захисту всіх видів транзакцій і запобігання несанкціонованого доступу до інформації. Як правило, безпека каналу зв'язку забезпечують за допомогою автентифікації, тобто завжди сертифікат прив'язаний до певного домену й шифрування, а це означає, що інформація перетвориться

так, що її розшифрувати можна тільки за допомогою спеціального ключа. SSL сертифікат завжди містить таку інформацію, як:

– доменне ім'я;

– юридична особа-власник;

– офіційна адреса місцезнаходження;

– термін дії самого сертифіката;

– реквізити його компанії-постачальника.

Саме SSL-сертифікат підтверджує, що домен належить саме даній компанії й тому власник має повне право користуватися секретним ключем. При цьому SSL-сертифікат за рівнем захисту можна розділити на кілька груп, але найчастіше використовують стандартний, котрий захищає одне доменне ім'я.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки передачі даних у мережі за протоколом SSL/TLS, 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

Створення й використання сертифікатів SSL

В Winsows 10/11, коли для створення сертифікатів використовується діалогове вікно SSL & Certificates, те MDaemon генерує самопідписані сертифікати. Інакше кажучи, видавець сертифіката, або сертифікатний авторитет (CA) той же самий, що й власник сертифіката. Це повністю правомірно й дозволено, але тому що CA ще не входить у списки довірених CA ваших користувачів, то щораз, коли вони з'єднуються по посиланню з WorldClient або WebAdmin, їм буде задане питання, чи хочуть вони продовжити роботу із сайтом і/або встановити сертифікат. Після того, як вони встановлять сертифікат і виразять довіру вашому домену WorldClient як дійсному CA, вони більше ніколи, під час з'єднання з WorldClient або WebAdmin не побачать повідомлення з попередженням про безпеку.

Однак, коли з MDaemon з'єднуються через поштового клієнта, такого як Microsoft Outlook, у них не буде можливості встановити сертифікат. Їм буде дозволено вибрати, хочуть вони чи ні продовжувати тимчасово використовувати сертифікат, навіть якщо він не підтверджений. Щораз, коли вони запустять свої поштові клієнти й з'єднаються із сервером, їм буде необхідно підтвердити використання непідтвердженого сертифіката. Щоб цього уникнути, ви повинні експортувати ваш сертифікат і роздати його вашим користувачам поштою або іншими засобами. Потім, вони зможуть вручну встановити й виразити довіру вашому сертифікату, щоб уникнути майбутніх попереджувальних повідомлень.

Створення сертифіката

Щоб створити сертифікат в MDaemon, треба:

– Перейти на діалогове вікно SSL & Certificates в MDaemon (натисніть Ctrl+L або SecuritySSL/TLS/Certificates у рядку меню MDaemon).

– У текстовому полі з назвою «Host name» уведіть домен, якому буде належати сертифікат (наприклад, «mail.example.com»).

– У текстовому полі з назвою «Organization/company name» уведіть ім'я організації або компанії, що володіє сертифікатом.

– В «Alternative host names» укажіть всі інші імена доменів, які будуть використовувати ваші користувачі для доступу до вашого сервера (наприклад, «*.mydomain.com», «example.com», «wc.altn.com» і так далі).

– Зі списку, що випадає, виберіть довжину ключа шифрування.

– Виберіть країну/регіон, у якому перебуває ваш сервер.

– Натисніть Create certificate.

Використання сертифікатів, виданих стороннім CA

Якщо у вас є куплений або по-іншому згенерований сертифікат з відмінного від MDaemon джерела, ви можете продовжувати використовувати цей сертифікат і використовувати консоль керування Microsoft, щоб імпортувати його в сховище сертифікатів, використовуване MDaemon. Щоб це зробити, треба:

– На панелі Windows, виберіть Пуск Виконати, і потім напишіть «mmc /a» у поле «Відкрити:».

– Натисніть кнопку OK.

– У консолі керування Microsoft, виберіть Консоль Додати/Видалити оснащення (або натисніть на клавіатурі Ctrl+M).

– На закладці Ізольоване оснащення натисніть кнопку Додати.

– Натисніть Сертифікати, і потім натисніть кнопку Додати.

– Виберіть облікового запису комп'ютера й натисніть кнопку Далі.

– Виберіть локальним комп'ютером і натисніть Готово.

– Натисніть Закрити й потім OK.

– Під «Сертифікати (локальний комп'ютер), натисніть папку «Сертифікати» у папці «Довірені кореневі центри сертифікації», якщо імпортований вами сертифікат є самопідписаним. Якщо це не самопідписаний сертифікат, то виберіть папку «Особисті».

– Натисніть Дію Всі завдання Імпорт, і натисніть кнопку Далі>.

– Уведіть файловий шлях до сертифіката, що ви хочете імпортувати (використовуйте, якщо необхідно, кнопку Огляд), і натисніть кнопку Далі.

– Натисніть Далі, і натисніть на кнопку Фініш.

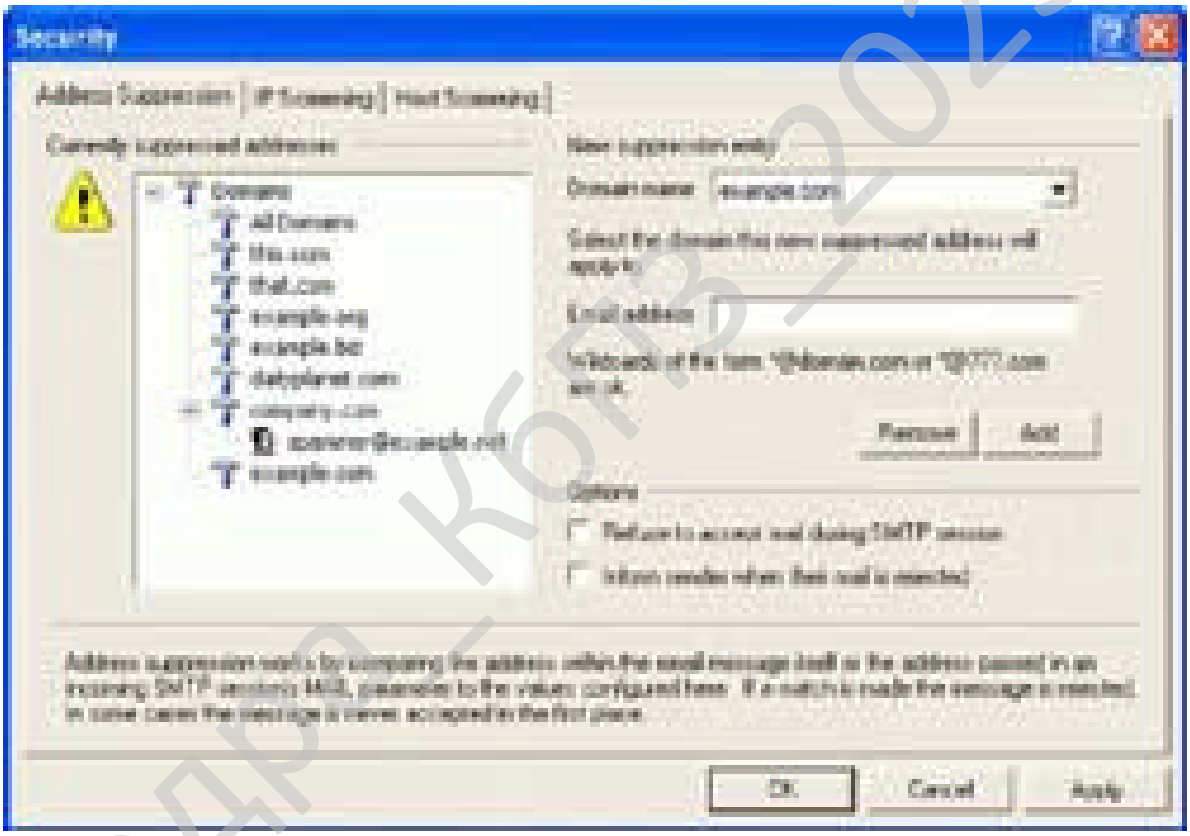

Рисунок 2.1 – Інтерфейс користувача SecurityAddress Suppression

Використовуйте SecurityAddress Suppression (або F4), щоб відредагувати адреси в списку придушення. Цей список містить адреси, якою буде відмовлено в пересиланні поштового потоку через ваш сервер. Якщо повідомлення прийшло з адреси, зазначеного в цьому списку, то воно, залежно від ваших установок, буде

або прийняте й перенесене в чергу поганої пошти, або відхилено під час сеансу SMTP, і, відповідно, ніколи не буде прийнято. Це корисно для контролю над проблемними користувачами. Адреси можуть бути подавлені в межах домену або глобально (застосовно для всіх доменів MDaemon).

Поточні придушувані адреси

Це вікно відображає всі адреси, що придушуються на поточний момент, перераховані під доменами, для яких вони придушуються.

Новий запис про придушення.

Domain name. Виберіть домен, до якого буде застосовуватися придушення адрес. Інакше кажучи, який домен ви хочете відгородити від одержання пошти з адреси, що придушується. Виберіть із цього списку «All Domains», щоб придушувати адреси глобально.

Email address. Уведіть адресу, активність якої ви хочете придушити. Можна використовувати групові символи, тому «*@badmail.com» придушить кожне повідомлення від будь-якого користувача домену «badmail.com», а «frank@*» придушить кожне повідомлення від кожного з ім'ям «frank», не залежно від домену, звідки прийшло повідомлення.

Remove. Натисніть цю кнопку, щоб видалити елемент, що ви вибрали в списку Currently Suppressed Addresses.

Add. Натисніть цю кнопку, щоб додати зазначеного користувача до списку придушення.

Опції

Refuse to accept mail during SMTP session. Коли цей прапорець відзначений, то пошта, адресована з адрес, що придушуються, обраному домену, буде відхилена під час сеансу SMTP. Пошта з адрес, що придушуються, спрямована на цей домен, ніколи не буде збережена на вашім сервері, навіть у тимчасових робочих файлах. Якщо цей прапорець відсутній, то повідомлення будуть прийматися, але переноситися після прийому до черги для поганих

повідомлень. Ця функціональність установлюється в межах домену; не можна встановити її для адрес, що придушуються, для опції «All Domains».

Inform sender when their mail is rejected. Якщо цей прапорець відзначений, то до відправника повідомлення, що придушується, буде спрямоване ввічливе повідомлення, що проінформує його про те, що його повідомлення було вилучено. Ця функціональність надається на основі домену.

Zemana AntiLogger

Zemana AntiLogger – програмний набір антивірусних утиліт призначених для боротьби з різноманітним шпигунським програмним забезпеченням.

Рисунок 2.2 – Інтерфейс користувача Zemana AntiLogger

Zemana AntiLogger успішно захищає від троянів, кейлоггерів, різного spyware, а також SSL-троянів, що спеціалізуються на крадіжці банківської інформації. В обов'язки AntiLogger також входять захист системних компонентів від впровадження шкідливого коду.

Програма легка в освоєнні, має дружній інтерфейс і не конфліктує з іншими антивірусними засобами. Російська мова підтримується частково, "місцями", втім, чого цілком вистачає для того, щоб розібратися в і без того нескладних настроюваннях програми.

На перший погляд може здатися, що Zemana AntiLogger є не що інше, як звичайний антивірусний сканер, однак це не так. Функція сканування, у звичному для більшості користувачів розумінні в даній програмі відсутній. Захист базується на роботі проактивних модулів, що контролюють процеси зміни у файловій системі, будь те інсталяція програмного забезпечення або редагування файлу. При цьому по запевненню розроблювачів, AntiLogger прекрасно розуміє яка зміна викликана корисним, а яке шкідливим додатком. Відповідно, будь-яка підозріла діяльність на комп'ютері буде відразу відстежена й блокована.

Не менш ефективно AntiLogger захищає від крадіжки особистої інформації при відвідуванні різних веб-сайтів, у тому числі що використовують HTTPS/SSL з'єднання. SSL шифрування дозволить вам безпечно здійснювати банківські операції й проводити онлайн покупки.

До складу Zemana AntiLogger входить три основних модулі, не вважаючи настроювань: Захист, Завдання й Служби.

1. Завдання. Тут ви можете ознайомитися з балкою подій, переглянути вміст карантину, а також управляти списком виключень, у який програма заносити підозрілі додатки.

2. Служби. Скориставшись інструментами цього модуля, ви можете перевірити відновлення, зв'язатися зі службою технічної підтримки, а також обновити ліцензію. До речі, програма є платною, але протягом 15-ти днів ви можете тестувати її без яких-небудь обмежень.

3. Захист. Модуль включає шість утиліт або інструментів (називайте, як хочете), призначених для рішення конкретного завдання. Отут все просто. Інструмент може вити або включений, або виключений. Перелічимо їх.

– Anti-KeyLogger запобігає відстеженню кейлоггерами натискання клавіш і передач даних зловмиснику.

– Anti-ScreenLogger захищає систему від шпигунських програм що відслідковують те, що відбувається на екрані за допомогою регулярного зняття скриншотів.

– Аналогічним образом працює Anti-WebcamLogger, запобігаючи захватові даних з веб-камеры.

– Anti-ClipboardLogger. Утиліта призначена для захисту буфера обміну.

– Захист системи. Утиліта, що запобігає впровадження в довірені компоненти системи шкідливого коду.

Ще однією особливістю Zemana AntiLogger є самозахист від ненавмисного відключення. Завершення роботи програми можливо лише після введення каптчи в спеціальне віконце, що з'являється щораз при спробі закрити додаток.

Програма зручна й проста у використанні, споживає відносно мало ресурсів, не вимагає постійної уваги до себе. Установлюючи Zemana AntiLogger треба розуміти, що ця програма є всього лише додатковим засобом забезпечення безпеки комп'ютера. Заміною повноцінним антивірусам вона служити не може.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Оскільки потрібно розробити просту та легку у користуванні програму, яка б виконувалась під операційною системою Windows, то для її реалізації я обрав Builder C++. Існує велике число бібліотек написаних під Builder C++ , тому це одна з важливих причин вибору мови програмування. Середовище Builder C++ досить просте в користуванні, його вихідний код значно менше по об’єму в

порівнянні з Delphi чи деякими іншими програмами такого типу. Досить легко організувати взаємодію між модулями програм, об’єктно-орієнтований підхід дає можливість значно скоротити код програми, а отже і час його виконання.

На заміну старого розробленого набору елементів управління у Builder C++ інтегрована бібліотека візуальних компонентів VCL, представлених на палітрі компонентів. Після переносу на форму методом перетягування (drag-and-drop) компоненти відразу становляться діючими об’єктами вашої програми. Окрім типізованих інтерфейсних елементів Windows (кнопки, смуги прокручування, редагуємі текстові області, прості та комбіновані списки, та інше) у бібліотеку включені елементи підтримки діалогових вікон, обслуговування баз даних та багато іншого. Можливо не тільки модифікувати поведінку існуючих компонентів, але і будувати нові.

Builder C++ підтримує останні розширення стандарту мови C++ та забезпечує швидку компіляцію та складання 32-розрядних програм для Windows. Результуючі програми оптимізовані з точки зору швидкості виконання програм та затрат пам’яті. Зручний відладгоджувальник (з асемблерним вікном, можливістю крокового виконання, завдання точок зупинки, трасування та інше) повністю інтегрований у систему проектування. Дизайнер форм, редактор коду, інспектор об’єктів та інші інструменти зостаються доступними під час виконання програми, саме через це вносити зміни до коду можна прямо у процесі відлагодження.

Дизайнер форм, Інспектор об'єктів і інші засоби залишаються доступними під час роботи програми, тому вносити зміни можна в процесі відлагодження.

Builder C++ поставляється в трьох варіантах: Standard (стандартний), Professional (для професіоналів розробників, орієнтованих на мережеву архітектуру) і Client/Server Suite (для розробки систем в архітектурі клієнт/сервер). Останні два варіанти доповнюють стандартний початковими текстами візуальних компонентів, різномасштабним словником даних, новими

функціями мови запитів SQL для бази даних, пакетом підтримки систем Internet, службою моніторингу програм, а також рядом інших засобів.

Builder C++ підтримує зв'язок з різними базами даних 3-х видів: dBASE і Paradox: Sybase, Oracle, InterBase і Informix; Excel, Access, FoxPro і Btrieve. Механізм BDE (Borland Database Engine) додає обслуговуванню зв'язків з базами даних дивовижну простоту і прозорість. Провідник Database Explorer дозволяє зображати зв'язки і об'єкти баз даних графічно. Використовуючи компоненти баз даних, я побудував електронний записник згідно таблиці dBASE за півгодини роботи на комп'ютері. Спадкоємство готових форм і їх "підгонка" під специфічні вимоги помітно скорочують тимчасові витрати на вирішення подібних завдань.

Довідкова служба Builder C++ надавала мені допомогу в цій і багатьох інших подібних ситуаціях. Є повний опис кожного управляємого компонента, включаючи списки властивостей і методів, а також численні приклади. Виклад матеріалу в книзі був значно покращуваний і систематизований завдяки відомостям, почерпнутим мною з довідкової служби.

Завдяки засобам управління проектами, двосторонній інтеграції додатку і синхронізації між засобами візуального і текстового редагування, а також вбудованому відладнику (з асемблерним вікном прокрутки, покрокового виконання, точок останову, трасуванням і тому подібне) – Builder C++ корпорації Borland надає собою вражаюче середовище розробки, яка, мабуть, витримає конкурентну боротьбу з такими модними продуктами як Developer Studio фірми Microsoft.

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 передачі даних у мережі за протоколом SSL/TLS.

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

Для забезпечення безпечної передачі даних по мережі Інтернет використовується криптографічний протокол SSL (Secure Sockets Layer – протокол захищених сокетів).

При роботі із цього протоколу створюється захищене з'єднання між клієнтом і сервером. По стандарті протокол працює на 443-м порту. Але адміністратор завжди може перемінити номер порту щоб сховати від такого зловмисника як ти свої сервіси.

SSL споконвічно розроблена компанією Netscape Communications і зараз підтримується всіма популярними браузерами. Для роботи з SSL замість префіксу http: застосовується префікс https:.

SSL представляє із себе 128 бітову систему шифрування даних з відкритим ключем. Для кожного сервера в інтернеті, що використовує дану систему шифрування, створюються 2 ключі, один суспільний, до якого мають доступ всі користувачі сервісу, другий установлюється на сервері й недоступний ні для кого. Ці 2 ключі залежать один від іншого й друг без друга змісту не мають.

Для того щоб включити на своєму сервері підтримку SSL необхідно знайти ключі й сертифікат. Вартість коливається від 30$ до 500$ у залежності від виду сертифіката. Сертифікат призначений для підтвердження приналежності ключів даному серверу або домену.

Протокол Secure Sockets Layer (SSL) використовує комбінацію відкритого ключа й симетричного ключа шифрування. Шифрування симетричного ключа набагато швидше, ніж шифрування з відкритим ключем; Однак шифрування з відкритим ключем забезпечує кращі методи перевірки дійсності. Сеанс SSL

завжди починається з обміну повідомленнями, називаний SSL-підтвердження. Підтвердження дозволяє пройти перевірку дійсності клієнта за допомогою технології відкритого ключа сервера й потім дозволяє співробітничати в створенні симетричних ключів, використовуваних для швидкого шифрування, розшифровки й виявлення підробки під час сеансу, якому треба за клієнта й сервера. При необхідності підтвердження також дозволяє клієнтові проходити перевірку дійсності на сервері.

Нижче наведені кроки, необхідні для підтвердження SSL (Зверніть увагу, що наступні кроки припускають використання шифрів, перераховані в комплекти шифрів з обмін ключами RSA: DES, Triple DES, RC4, RC2):

– Клієнт відправляє серверу номер версії клієнта SSL, шифр параметри, специфічні для сеансу даних і інші відомості, сервер для зв'язку із клієнтом, за допомогою протоколу SSL.

– Сервер відправляє клієнтові номер версії сервера SSL, шифр параметри, специфічні для сеансу даних і інші відомості, які клієнт повинен зв'язатися із сервером за протоколом SSL. Сервер також відправляє свій власний сертифікат, і якщо клієнт запитує ресурс сервера, що вимагає перевірки дійсності клієнта, сервер запитує сертифікат клієнта.

– Клієнт використовує відомості, що відправляються сервером для перевірки дійсності сервера. Якщо сервер не пройшов перевірку дійсності, користувач попередження проблеми й інформувати, що не вдається встановити підключення до шифрування й перевірки дійсності. Якщо сервер може бути виконана перевірка дійсності, клієнт переходить до кроку 4.

– При використанні всі дані, створені в підтвердженні цих клієнта (при сприянні сервера, залежно від шифрування використовується) створює pre-master секрет для сеансу, шифрує його за допомогою відкритого ключа сервера (отриманий із сертифіката сервера, відправлені на кроці 2), а потім відправляє серверу зашифрований секретний код pre-master.

– Якщо сервер вимагає перевірки дійсності клієнта (необов'язковий крок у підтвердженні), клієнт також підписує інший фрагмент даних, унікальний для цього підтвердження й відомі із сервера й клієнта. У цьому випадку клієнт відправляє підписаних даних і сертифікат клієнта й сервера й зашифрований секретний код pre-master.

– Якщо сервер вимагає перевірки дійсності клієнта, сервер намагається перевірити дійсність клієнта. Якщо клієнт не може бути перевірений, сеанс закінчується. Якщо клієнт може успішно перевірку дійсності, сервер використовує свій закритий ключ для розшифровки pre-master секрет і виконує ряд дій (які також виконує клієнта, починаючи з того ж секрету pre-master) для створення головного секрету.

– І клієнт і сервер використовують головний секрет для створення ключів сеансів, які симетричні ключі, використовувані для шифрування й розшифровки інформації, переданих під час сеансу SSL і перевірки її цілісності (тобто, для виявлення змін у даних між часом, воно було відправлено й одержання через з'єднання SSL).

– Клієнт відправляє повідомлення на сервер, повідомляючи його, що наступні повідомлення від клієнта будуть зашифровані за допомогою ключа сеансу. Потім він відправляє окремі (зашифровані) повідомлення про те, що клієнт підтвердження завершення.

– Сервер відправляє повідомлення клієнтові, повідомляючи його, що майбутнім повідомлення із сервера буде зашифрований ключ сеансу. Потім він відправляє окремі (зашифровані) повідомлення про те, що сервер підтвердження завершення.

– SSL-підтвердження завершена й починається сеанс. Клієнт і сервер використовують ключі сеансу для шифрування й розшифровки даних, які вони відправляти один одному й перевірити його цілісність.

– Це умова нормальної роботи безпечного каналу. Заново в будь-який час, через внутрішній або зовнішній стимул (втручання користувача або

автоматизації), обидві сторони можуть встановити зв'язок, у цьому випадку процес повторюється.

Приведемо опис сертифікатів, які використовуються в сучасних системах.

SSL сертифікати RapidSSL

RapidSSL®

SSL сертифікати RapidSSL® забезпечують 128-і або 256-і – бітний захист. Сертифікати RapidSSL® сумісні з усіма популярними інтернет-браузерами. Низька ціна й висока надійність роблять сертифікати RapidSSL® одними з найбільш популярних.

Додаткова інформація про SSL сертифікат:

– Верифікація – Домену.

– Час видачі SSL сертифіката – 5 хвилин.

– Безкоштовна перевидача SSL сертифіката – так (не більше 5 разів).

– Зелений рядок браузера – Немає.

– Перевірка роботи сертифіката – Немає.

– Страховка сертифіката – 10,000$.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Так.

– Сумісність із браузерами – 99.3%.

– Захист субдоменів – Немає.

– Можливість установки SSL сертифіката на різних серверах – Так.

RapidSSL® WildCard

SSL сертифікати RapidSSL® Wildcard забезпечують 128-і або 256-і – бітний захист, а також дозволяють захистити всі субдомени на вашому сайті. Сертифікати RapidSSL® сумісні з усіма популярними інтернет-браузерами. Дане рішення ідеально підходить для великих комерційних порталів і сайтів з більшим числом субдоменів.

Додаткова інформація про SSL сертифікат:

– Верифікація – Домену.

– Час видачі SSL сертифіката – 5 хвилин.

– Безкоштовна перевидача SSL сертифіката – так (не більше 5 разів).

– Зелений рядок браузера – Немає.

– Перевірка роботи сертифіката – Немає.

– Страховка сертифіката – 10,000$.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Так.

– Сумісність із браузерами – 99.3%.

– Захист субдоменів – Так.

– Можливість установки SSL сертифіката на різних серверах – Так.

SSL сертифікати Geotrust

GeoTrust QuickSSL® Basic

SSL сертифікати QuickSSL® Basic забезпечують 128-і або 256-і – бітний захист. Сертифікати QuickSSL® Basic сумісні з усіма популярними інтернет-браузерами. Низька ціна й висока надійність роблять сертифікати QuickSSL® Basic одними з найбільш популярних.

Додаткова інформація про SSL сертифікат:

– Верифікація – Домену.

– Час видачі SSL сертифіката – 8 хвилин.

– Безкоштовна перевидача SSL сертифіката – так.

– Зелений рядок браузера – Немає.

– Перевірка роботи сертифіката – Немає.

– Страховка сертифіката – 100,000$.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Так.

– Сумісність із браузерами – 99.3%.

– Захист субдоменів – Немає.

– Можливість установки SSL сертифіката на різних серверах – Так.

GeoTrust QuickSSL® Premium

SSL сертифікати QuickSSL® Premium SSL Certificate забезпечують 256-і – бітний захист. Сертифікати QuickSSL® Basic сумісні з усіма популярними інтернет-браузерами, мобільними пристроями й смартфонами. Емітент – Equifax Secure Certificate Authority.

Додаткова інформація про SSL сертифікат:

– Верифікація – Домену.

– Час видачі SSL сертифіката – 8 хвилин.

– Безкоштовна перевидача SSL сертифіката – так (не більше 5 разів).

– Зелений рядок браузера – Немає.

– Перевірка роботи сертифіката – Немає.

– Страховка сертифіката – 100,000$.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Так.

– Сумісність із браузерами – 99.3%.

– Захист субдоменів – Немає.

– Можливість установки SSL сертифіката на різних серверах – Так.

GeoTrust True BusinessID Wildcard

1. Для одержання SSL сертифікат ви повинні бути зареєстровані як юридична особа.

2. При генерації CSR запиту переконаєтеся, що в поле Organization ви ввели коректну назву компанії Англійською мовою (відповідно до статутних документів). Не можна використовувати в написанні спеціальні символи << >> або / “ “.

3. В WHOIS доменного ім'я для якого замовляється SSL сертифікат повинне бути ідентична назва компанії й контактні дані (Якщо WHOIS інформація схована, її варто відкрити).

4. Підготувати перекладені на англійську мову й завірені нотаріально реєстраційні документи (Свідчення про реєстрацію компанії, копію статутних

документів). Також можуть знадобитися копії рахунків за телефонний зв'язок і комунальні послуги.

Додаткова інформація про SSL сертифікат:

– Верифікація – Компанії.

– Час видачі SSL сертифіката – 1-2 дні.

– Безкоштовна перевидача SSL сертифіката – так (не більше 5 разів).

– Зелений рядок браузера – Немає.

– Перевірка роботи сертифіката – Немає.

– Страховка сертифіката – 10,000$.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Так.

– Сумісність із браузерами – 99.3%.

– Захист субдоменів – Так.

– Можливість установки SSL сертифіката на різних серверах – Так.

GeoTrust TrueBusinessID Multi-Domain

1. Для одержання SSL сертифікат ви повинні бути зареєстровані як юридична особа.

2. При генерації CSR запиту переконаєтеся, що в поле Organization ви ввели коректну назву компанії Англійською мовою (відповідно до статутних документів). Не можна використовувати в написанні спеціальні символи << >> або / “ “.

3. В WHOIS доменного ім'я для якого замовляється SSL сертифікат повинне бути ідентична назва компанії й контактні дані (Якщо WHOIS інформація схована, її варто відкрити).

4. Підготувати перекладені на англійську мову й завірені нотаріально реєстраційні документи (Свідчення про реєстрацію компанії, копію статутних документів). Також можуть знадобитися копії рахунків за телефонний зв'язок і комунальні послуги.

Додаткова інформація про SSL сертифікат:

– Верифікація – Компанії.

– Час видачі SSL сертифіката – 1-2 дні.

– Безкоштовна перевидача SSL сертифіката – так (не більше 5 разів).

– Зелений рядок браузера – Немає.

– Перевірка роботи сертифіката – Немає.

– Страховка сертифіката – 100,000$.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Так.

– Сумісність із браузерами – 99.3%.

– Захист субдоменів – Немає.

– Можливість установки SSL сертифіката на різних серверах – Так.

GeoTrust TrueBusinessID EV Multi-Domain

1. Для одержання SSL сертифікат ви повинні бути зареєстровані як юридична особа.

2. При генерації CSR запиту переконаєтеся, що в поле Organization ви ввели коректну назву компанії Англійською мовою (відповідно до статутних документів). Не можна використовувати в написанні спеціальні символи << >> або / “ “.

3. В WHOIS доменного ім'я для якого замовляється SSL сертифікат повинне бути ідентична назва компанії й контактні дані (Якщо WHOIS інформація схована, її варто відкрити).

4. Підготувати перекладені на англійську мову й завірені нотаріально реєстраційні документи (Свідчення про реєстрацію компанії, копію статутних документів). Також можуть знадобитися копії рахунків за телефонний зв'язок і комунальні послуги.

Додаткова інформація про SSL сертифікат:

– Верифікація – Розширена.

– Час видачі SSL сертифіката – 1-10 днів.

– Безкоштовна перевидача SSL сертифіката – так (не більше 5 разів).

– Зелений рядок браузера – Так.

– Перевірка роботи сертифіката – Немає.

– Страховка сертифіката – 500,000$.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Так.

– Сумісність із браузерами – 99.3%.

– Захист субдоменів – Немає.

– Можливість установки SSL сертифіката на різних серверах – Так.

GeoTrust True BusinessID with EV Certificate

GeoTrust True BusinessID with EV Certificate

1. Для одержання SSL сертифікат ви повинні бути зареєстровані як юридична особа.

2. При генерації CSR запиту переконаєтеся, що в поле Organization ви ввели коректну назву компанії Англійською мовою (відповідно до статутних документів). Не можна використовувати в написанні спеціальні символи << >> або / “ “.

3. В WHOIS доменного ім'я для якого замовляється SSL сертифікат повинне бути ідентична назва компанії й контактні дані (Якщо WHOIS інформація схована, її варто відкрити).

4. Підготувати перекладені на англійську мову й завірені нотаріально реєстраційні документи (Свідчення про реєстрацію компанії, копію статутних документів). Також можуть знадобитися копії рахунків за телефонний зв'язок і комунальні послуги.

Додаткова інформація про SSL сертифікат:

– Верифікація – Розширена.

– Час видачі SSL сертифіката – 1-10 днів.

– Безкоштовна перевидача SSL сертифіката – так.

– Зелений рядок браузера – Так.

– Перевірка роботи сертифіката – Немає.

– Страховка сертифіката – 500,000$.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Немає.

– Сумісність із браузерами – 99.3%.

– Захист субдоменів – Немає.

– Можливість установки SSL сертифіката на різних серверах – Немає.

SSL сертифікати Thawte

Thawte SSL123

SL сертифікати Thawte SSL123 – сертифікат початкового рівня. Забезпечує захист даних при передачі між клієнтом і сервером.

Додаткова інформація про SSL сертифікат:

– Верифікація – Домену.

– Час видачі SSL сертифіката – 8 хвилин.

– Безкоштовна перевидача SSL сертифіката – так (не більше 5 разів).

– Зелений рядок браузера – Немає.

– Перевірка роботи сертифіката – Немає.

– Страховка сертифіката – немає.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Так.

– Сумісність із браузерами – 99.3%.

– Захист субдоменів – Немає.

– Можливість установки SSL сертифіката на різних серверах – Немає.

Thawte SSL Web Server

Сертифікат Thawte SSL Web Server забезпечить безпека передачі даних при здійсненні електронних угод між Вашим сайтом і клієнтами. При видачі відбувається повна перевірка домену й компанії, на которуюэтот сертифікат виписаний.Більше немає приводу для занепокоєння! Жоден шахрай у світі не зуміє перехопити передачу даних.

Опис і особливості. Thawte SSL Web Server підтримує 128/ 256-бітне шифрування. Воно розроблено як для захисту веб-сайтів, так і для захисту мереж Intranet, VPN-Мереж і Інтернет.

Процедура одержання Сертифіката включає строгу перевірку на дійсність доменного імені й компанії (Замовлення сертифіката доступне тільки для юридичних осіб). А це високий ступінь довіри до сайту.

Висока сумісність – підтримка всіма основними веб-браузерами. SSL Сертифікат відповідає 1024-бітному галузевому стандарту.

Додаткова інформація про SSL сертифікат:

– Верифікація – Компанії.

– Час видачі SSL сертифіката – 1-2 дні.

– Безкоштовна перевидача SSL сертифіката – так (не більше 5 разів).

– Зелений рядок браузера – Немає.

– Перевірка роботи сертифіката – Немає.

– Страховка сертифіката – немає.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Так.

– Сумісність із браузерами – 99.3%.

– Захист субдоменів – Немає.

– Можливість установки SSL сертифіката на різних серверах – Немає.

Thawte SSL Web Server with EV

SSL Web Server EV сертифікат – передбачає найвищий стандарт автентифікації сайту компанії.

– Опис і особливості. SSL Web Server EV дозволяє здійснити 256-, 128-, 56-, або 40- бітне шифрування.

– SSL Web Server EV – це високий ступінь довіри – зелений адресний рядок браузера, що забезпечується EV (Extended Validation) сертифікатом.

– Найвища сумісність із браузерами.

Замовлення сертифіката доступне тільки для юридичних осіб.

Додаткова інформація про SSL сертифікат:

– Верифікація – Розширена.

– Час видачі SSL сертифіката – 1-10 днів.

– Безкоштовна перевидача SSL сертифіката – так (не більше 5 разів).

– Зелений рядок браузера – Так.

– Перевірка роботи сертифіката – Немає.

– Страховка сертифіката – немає.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Так.

– Сумісність із браузерами – 99.3%.

– Захист субдоменів – Немає.

– Можливість установки SSL сертифіката на різних серверах – Немає.

Thawte Code Signing

Code Signing сертифікати для Microsoft ® Authenticode ® При підписі коду від Thawte, ви можете гарантувати користувачам, що ваш код і вміст є безпечним для завантаження, і захистити ваш самий коштовний актив бізнесу – вашу репутацію. Підпис коду автентифікацію джерела коду й підтверджує цілісність змісту розподіленої мережі. Thawte ® Code Signing сертифікат Microsoft ® Authenticode ® (багатоцільовий) забезпечує максимальну гнучкість у єдиний сертифікат для підписання коду розроблені на декількох платформах. Цифровий підпис 32 – і 64-розрядних користувальницького режиму (EXE, CAB, DLL, OCX, MSI і XPI файлів).

Цифровий підпис коду для Microsoft ® Office 2000, Microsoft VBA, Netscape маркування об'єктів і Marimba Channel Signing

Додаткова інформація про SSL сертифікат:

– Верифікація – -.

– Час видачі SSL сертифіката – -.

– Безкоштовна перевидача SSL сертифіката – -.

– Зелений рядок браузера – Немає.

– Перевірка роботи сертифіката – Немає.

– Страховка сертифіката – -.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Немає.

– Сумісність із браузерами – -.

– Захист субдоменів – Немає.

– Можливість установки SSL сертифіката на різних серверах – Немає.

Thawte SGC SuperCerts

Сертифікат від Thawte (SuperCerts) дозволить Вам забезпечити 128-бітове шифрування для всіх ваших клієнтів. Ключ такої довжини неможливо розкрити за розумний проміжок часу (менш терміну дії сертифіката), а це головний плюс!

Додаткова інформація про SSL сертифікат:

– Верифікація – Компанії.

– Час видачі SSL сертифіката – 1-2 дні.

– Безкоштовна перевидача SSL сертифіката – так (не більше 5 разів).

– Зелений рядок браузера – Немає.

– Перевірка роботи сертифіката – Немає.

– Страховка сертифіката – немає.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Так.

– Сумісність із браузерами – 99.3%.

– Захист субдоменів – Немає.

– Можливість установки SSL сертифіката на різних серверах – Немає.

Thawte SSL Webserver Wildcard

Додаткова інформація про SSL сертифікат:

– Верифікація – Компанії.

– Час видачі SSL сертифіката – 1-2 дні.

– Безкоштовна перевидача SSL сертифіката – так (не більше 5 разів).

– Зелений рядок браузера – Немає.

– Перевірка роботи сертифіката – Немає.

– Страховка сертифіката – немає.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Так.

– Сумісність із браузерами – 99.3%.

– Захист субдоменів – Так.

– Можливість установки SSL сертифіката на різних серверах – Немає.

SSL сертифікати VeriSign

VeriSign Secure Site

Додаткова інформація про SSL сертифікат:

– Верифікація – Компанії.

– Час видачі SSL сертифіката – 2 дні.

– Безкоштовна перевидача SSL сертифіката – так (не більше 5 разів).

– Зелений рядок браузера – Немає.

– Перевірка роботи сертифіката – Так.

– Страховка сертифіката – 1,000,000$.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Так.

– Сумісність із браузерами – 99.3%.

– Захист субдоменів – Немає.

– Можливість установки SSL сертифіката на різних серверах – Немає.

VeriSign Secure Site EV

1. Для одержання SSL сертифікат ви повинні бути зареєстровані як юридична особа.

2. При генерації CSR запиту переконаєтеся, що в поле Organization ви ввели коректну назву компанії Англійською мовою (відповідно до статутних документів). Не можна використовувати в написанні спеціальні символи << >> або / “ “.

3. В WHOIS доменного ім'я для якого замовляється SSL сертифікат повинне бути ідентична назва компанії й контактні дані (Якщо WHOIS інформація схована, її варто відкрити).

4. Підготувати перекладені на англійську мову й завірені нотаріально реєстраційні документи (Свідчення про реєстрацію компанії, копію статутних документів). Також можуть знадобитися копії рахунків за телефонний зв'язок і комунальні послуги.

Додаткова інформація про SSL сертифікат:

– Верифікація – Розширена.

– Час видачі SSL сертифіката – 5 днів.

– Безкоштовна перевидача SSL сертифіката – так (не більше 5 разів).

– Зелений рядок браузера – Так.

– Перевірка роботи сертифіката – Так.

– Страховка сертифіката – 1,500,000$.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Так.

– Сумісність із браузерами – 99.3%.

– Захист субдоменів – Немає.

– Можливість установки SSL сертифіката на різних серверах – Немає.

VeriSign Secure Site Pro

Додаткова інформація про SSL сертифікат:

– Верифікація – Компанії.

– Час видачі SSL сертифіката – 2 дні.

– Безкоштовна перевидача SSL сертифіката – так (не більше 5 разів).

– Зелений рядок браузера – Немає.

– Перевірка роботи сертифіката – Так.

– Страховка сертифіката – 1,250,000$.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Так.

– Сумісність із браузерами – 99.3%.

– Захист субдоменів – Немає.

– Можливість установки SSL сертифіката на різних серверах – Немає.

VeriSign Secure Site Pro EV

Додаткова інформація про SSL сертифікат:

– Верифікація – Розширена.

– Час видачі SSL сертифіката – 5 днів.

– Безкоштовна перевидача SSL сертифіката – так (не більше 5 разів).

– Зелений рядок браузера – Так.

– Перевірка роботи сертифіката – Так.

– Страховка сертифіката – 1,500,000$.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Так.

– Сумісність із браузерами – 99.3%.

– Захист субдоменів – Немає.

– Можливість установки SSL сертифіката на різних серверах – Немає.

VeriSign Code Signing

Додаткова інформація про SSL сертифікат:

– Верифікація – -.

– Час видачі SSL сертифіката – -.

– Безкоштовна перевидача SSL сертифіката – -.

– Зелений рядок браузера – Немає.

– Перевірка роботи сертифіката – Немає.

– Страховка сертифіката – -.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Немає.

– Сумісність із браузерами – -.

– Захист субдоменів – Немає.

– Можливість установки SSL сертифіката на різних серверах – Немає.

SSL сертифікати Comodo

Comodo Positive SSL

Comodo Positive SSL – SSL сертифікат початкового рівня, що дозволяє підтвердити вірогідність адреси сайту (Domain validation SSL). У роботі з Comodo Positive SSL, в адресному рядку браузера буде відображений жовтий замок, що дозволить відвідувачам бути впевненими у тому, що з'єднання зашифроване (128 біт), а угоди захищені. Сертифікати Comodo Positive SSL ідеально підходять для інтернет-проектів початкового рівня або не більших інтернет-магазинів.

Для одержання SSL сертифіката Comodo Positive SSL не потрібне надання документів, що дозволяє швидко одержати сертифікат – після оплати й внесення даних сертифіката (CSR запиту) – SSL сертифікат готовий до використання в плині 5 хвилин!

Comodo Positive SSL підтримуються 99,3% браузерів і підходять для організації прийому оплати по пластикових картах VISA/MASTER CARD на вашому сайті.

Додаткова інформація про SSL сертифікат:

– Верифікація – Домену.

– Час видачі SSL сертифіката – 5 хвилин.

– Безкоштовна перевидача SSL сертифіката – так.

– Зелений рядок браузера – Немає.

– Перевірка роботи сертифіката – Так.

– Страховка сертифіката – 10,000$.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Так.

– Сумісність із браузерами – 99.3%.

– Захист субдоменів – Немає.

– Можливість установки SSL сертифіката на різних серверах – Так.

Comodo InstantSSL

Додаткова інформація про SSL сертифікат:

– Верифікація – Компанії.

– Час видачі SSL сертифіката – 1-2 дні.

– Безкоштовна перевидача SSL сертифіката – так.

– Зелений рядок браузера – Немає.

– Перевірка роботи сертифіката – Так.

– Страховка сертифіката – 10,000$.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Так.

– Сумісність із браузерами – 99.3%.

– Захист субдоменів – Немає.

– Можливість установки SSL сертифіката на різних серверах – Так.

Comodo InstantSSL Premium

1. Для одержання SSL сертифікат ви повинні бути зареєстровані як юридична особа.

2. При генерації CSR запиту переконаєтеся, що в поле Organization ви ввели коректну назву компанії Англійською мовою (відповідно до статутних документів). Не можна використовувати в написанні спеціальні символи << >> або / “ “.

3. В WHOIS доменного ім'я для якого замовляється SSL сертифікат повинне бути ідентична назва компанії й контактні дані (Якщо WHOIS інформація схована, її варто відкрити).

4. Підготувати перекладені на англійську мову й завірені нотаріально реєстраційні документи (Свідчення про реєстрацію компанії, копію статутних документів). Також можуть знадобитися копії рахунків за телефонний зв'язок і комунальні послуги.

Додаткова інформація про SSL сертифікат:

– Верифікація – Компанії.

– Час видачі SSL сертифіката – 1-2 дні.

– Безкоштовна перевидача SSL сертифіката – так.

– Зелений рядок браузера – Немає.

– Перевірка роботи сертифіката – Так.

– Страховка сертифіката – 250,000$.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Так.

– Сумісність із браузерами – 99.3%.

– Захист субдоменів – Немає.

– Можливість установки SSL сертифіката на різних серверах – Так.

Comodo InstantSSL Pro

Додаткова інформація про SSL сертифікат:

– Верифікація – Компанії.

– Час видачі SSL сертифіката – 1-2 дні.

– Безкоштовна перевидача SSL сертифіката – так.

– Зелений рядок браузера – Немає.

– Перевірка роботи сертифіката – Так.

– Страховка сертифіката – 100,000$.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Так.

– Сумісність із браузерами – 99.3%.

– Захист субдоменів – Немає.

– Можливість установки SSL сертифіката на різних серверах – Так.

Comodo Multi-Domain Certificate

Comodo Multi-Domain Certificate

1. Для одержання SSL сертифікат ви повинні бути зареєстровані як юридична особа.

2. При генерації CSR запиту переконаєтеся, що в поле Organization ви ввели коректну назву компанії Англійською мовою (відповідно до статутних

документів). Не можна використовувати в написанні спеціальні символи << >> або / “ “.

3. В WHOIS доменного ім'я для якого замовляється SSL сертифікат повинне бути ідентична назва компанії й контактні дані (Якщо WHOIS інформація схована, її варто відкрити).

4. Підготувати перекладені на англійську мову й завірені нотаріально реєстраційні документи (Свідчення про реєстрацію компанії, копію статутних документів). Також можуть знадобитися копії рахунків за телефонний зв'язок і комунальні послуги.

Додаткова інформація про SSL сертифікат:

– Верифікація – Компанії.

– Час видачі SSL сертифіката – 1-2 дні.

– Безкоштовна перевидача SSL сертифіката – так.

– Зелений рядок браузера – Немає.

– Перевірка роботи сертифіката – Так.

– Страховка сертифіката – немає.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Немає.

– Сумісність із браузерами – 99.3%.

– Захист субдоменів – Немає.

– Можливість установки SSL сертифіката на різних серверах – Так.

Comodo EV Multi-Domain

1. Для одержання SSL сертифікат ви повинні бути зареєстровані як юридична особа.

2. При генерації CSR запиту переконаєтеся, що в поле Organization ви ввели коректну назву компанії Англійською мовою (відповідно до статутних документів). Не можна використовувати в написанні спеціальні символи << >> або / “ “.

3. В WHOIS доменного ім'я для якого замовляється SSL сертифікат повинне бути ідентична назва компанії й контактні дані (Якщо WHOIS інформація схована, її варто відкрити).

4. Підготувати перекладені на англійську мову й завірені нотаріально реєстраційні документи (Свідчення про реєстрацію компанії, копію статутних документів). Також можуть знадобитися копії рахунків за телефонний зв'язок і комунальні послуги.

Додаткова інформація про SSL сертифікат:

– Верифікація – Розширена.

– Час видачі SSL сертифіката – 1-7 днів.

– Безкоштовна перевидача SSL сертифіката – так.

– Зелений рядок браузера – Так.

– Перевірка роботи сертифіката – Так.

– Страховка сертифіката – 250,000$.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Так.

– Сумісність із браузерами – 99.3%.

– Захист субдоменів – Немає.

– Можливість установки SSL сертифіката на різних серверах – Так.

Comodo EV SGC Certificate

Comodo EV SGC Certificate

1. Для одержання SSL сертифікат ви повинні бути зареєстровані як юридична особа.

2. При генерації CSR запиту переконаєтеся, що в поле Organization ви ввели коректну назву компанії Англійською мовою (відповідно до статутних документів). Не можна використовувати в написанні спеціальні символи << >> або / “ “.

3. В WHOIS доменного ім'я для якого замовляється SSL сертифікат повинне бути ідентична назва компанії й контактні дані (Якщо WHOIS інформація схована, її варто відкрити).

4. Підготувати перекладені на англійську мову й завірені нотаріально реєстраційні документи (Свідчення про реєстрацію компанії, копію статутних документів). Також можуть знадобитися копії рахунків за телефонний зв'язок і комунальні послуги.

Додаткова інформація про SSL сертифікат:

– Верифікація – Розширена.

– Час видачі SSL сертифіката – 1-7 днів.

– Безкоштовна перевидача SSL сертифіката – так.

– Зелений рядок браузера – Так.

– Перевірка роботи сертифіката – Так.

– Страховка сертифіката – 250,000$.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Так.

– Сумісність із браузерами – 99.3%.

– Захист субдоменів – Немає.

– Можливість установки SSL сертифіката на різних серверах – Так.

Comodo EV SSL Certificate

1. Для одержання SSL сертифікат ви повинні бути зареєстровані як юридична особа.

2. При генерації CSR запиту переконаєтеся, що в поле Organization ви ввели коректну назву компанії Англійською мовою (відповідно до статутних документів). Не можна використовувати в написанні спеціальні символи << >> або / “ “.

3. В WHOIS доменного ім'я для якого замовляється SSL сертифікат повинне бути ідентична назва компанії й контактні дані (Якщо WHOIS інформація схована, її варто відкрити).

4. Підготувати перекладені на англійську мову й завірені нотаріально реєстраційні документи (Свідчення про реєстрацію компанії, копію статутних документів). Також можуть знадобитися копії рахунків за телефонний зв'язок і комунальні послуги.

Додаткова інформація про SSL сертифікат:

– Верифікація – Розширена.

– Час видачі SSL сертифіката – 1-7 днів.

– Безкоштовна перевидача SSL сертифіката – так.

– Зелений рядок браузера – Так.

– Перевірка роботи сертифіката – Так.

– Страховка сертифіката – 250,000$.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Так.

– Сумісність із браузерами – 99.3%.

– Захист субдоменів – Немає.

– Можливість установки SSL сертифіката на різних серверах – Так.

Comodo Positive SSL Wildcard

Comodo Positive SSL Wildcard – SSL сертифікат початкового рівня, що дозволяє підтвердити вірогідність адреси сайту (Domain validation SSL) – основного домену й усього його субдоменів. Positive SSL Wildcard є одним з найбільш доступних Wildcard SSL сертифікатів, які представлені в Інтернет. Сертифікати Comodo Positive SSL Wildcard ідеально підходять для інтернет-проектів початкового рівня або не більших інтернет-магазинів.

Для одержання SSL сертифіката Comodo Positive SSL Wildcard не потрібне надання документів, що дозволяє швидко одержати сертифікат – після оплати й внесення даних сертифіката (CSR запиту) – SSL сертифікат готовий до використання в плині 5 хвилин!

Comodo Positive SSL Wildcard підтримуються 99,3% браузерів і підходять для організації прийому оплати по пластикових картах VISA/MASTER CARD на вашому сайті.

Додаткова інформація про SSL сертифікат:

– Верифікація – Домену.

– Час видачі SSL сертифіката – 5 хвилин.

– Безкоштовна перевидача SSL сертифіката – так.

– Зелений рядок браузера – Немає.

– Перевірка роботи сертифіката – Так.

– Страховка сертифіката – 10,000$.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Так.

– Сумісність із браузерами – 99.3%.

– Захист субдоменів – Так.

– Можливість установки SSL сертифіката на різних серверах – Так.

Comodo Premium Wildcard

Додаткова інформація про SSL сертифікат:

– Верифікація – Компанії.

– Час видачі SSL сертифіката – 1-2 дні.

– Безкоштовна перевидача SSL сертифіката – так.

– Зелений рядок браузера – Немає.

– Перевірка роботи сертифіката – Так.

– Страховка сертифіката – 250,000$.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Так.

– Сумісність із браузерами – 99.3%.

– Захист субдоменів – Так.

– Можливість установки SSL сертифіката на різних серверах – Так.

Comodo SGC SSL Wildcard

Додаткова інформація про SSL сертифікат:

– Верифікація – Компанії.

– Час видачі SSL сертифіката – 1-2 дні.

– Безкоштовна перевидача SSL сертифіката – так.

– Зелений рядок браузера – Немає.

– Перевірка роботи сертифіката – Так.

– Страховка сертифіката – 250,000$.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Так.

– Сумісність із браузерами – 99.3%.

– Захист субдоменів – Так.

– Можливість установки SSL сертифіката на різних серверах – Так.

Comodo SGC SSL

1. Для одержання SSL сертифікат ви повинні бути зареєстровані як юридична особа.

2. При генерації CSR запиту переконаєтеся, що в поле Organization ви ввели коректну назву компанії Англійською мовою (відповідно до статутних документів). Не можна використовувати в написанні спеціальні символи << >> або / “ “.

3. В WHOIS доменного ім'я для якого замовляється SSL сертифікат повинне бути ідентична назва компанії й контактні дані (Якщо WHOIS інформація схована, її варто відкрити).

4. Підготувати перекладені на англійську мову й завірені нотаріально реєстраційні документи (Свідчення про реєстрацію компанії, копію статутних документів). Також можуть знадобитися копії рахунків за телефонний зв'язок і комунальні послуги.

Додаткова інформація про SSL сертифікат:

– Верифікація – Компанії.

– Час видачі SSL сертифіката – 1-2 дні.

– Безкоштовна перевидача SSL сертифіката – так.

– Зелений рядок браузера – Немає.

– Перевірка роботи сертифіката – Так.

– Страховка сертифіката – 250,000$.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Так.

– Сумісність із браузерами – 99.3%.

– Захист субдоменів – Немає.

– Можливість установки SSL сертифіката на різних серверах – Так.

Comodo Unified Communications

SAN – Subject Alternative Name, також відомі як UCC – Єдині Комунікаційні Сертифікати. Одним сертифікатом SAN можна захистити трохи доменні імен, назви домену, IP-Адреси, а шлюз і межсетевой екран пристрою імен вузлів.

Сертифікат доступний тільки для юридичних осіб. Кожне додаткове доменне ім'я оплачується окремо. Ціна одного додаткового домену: 650 грн/рік.

Додаткова інформація про SSL сертифікат:

– Верифікація – Компанії.

– Час видачі SSL сертифіката – 1-2 дні.

– Безкоштовна перевидача SSL сертифіката – так.

– Зелений рядок браузера – Немає.

– Перевірка роботи сертифіката – Немає.

– Страховка сертифіката – немає.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Так.

– Сумісність із браузерами – 99.3%.

– Захист субдоменів – Немає.

– Можливість установки SSL сертифіката на різних серверах – Так.

Comodo Secure Email Certificate

Додаткова інформація про SSL сертифікат:

– Верифікація -.

– Час видачі SSL сертифіката -.

– Безкоштовна перевидача SSL сертифіката -.

– Зелений рядок браузера – Немає.

– Перевірка роботи сертифіката – Немає.

– Страховка сертифіката -.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Немає.

– Сумісність із браузерами -.

– Захист субдоменів – Немає.

– Можливість установки SSL сертифіката на різних серверах – Немає.

SSL сертифікати GlobalSign

GlobalSign DomainSSL Certificate

Додаткова інформація про SSL сертифікат:

– Верифікація – Домену.

– Час видачі SSL сертифіката – 10 хвилин.

– Безкоштовна перевидача SSL сертифіката – так.

– Зелений рядок браузера – Немає.

– Перевірка роботи сертифіката – Так.

– Страховка сертифіката – 10,000$.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Так.

– Сумісність із браузерами – 99.3%.

– Захист субдоменів – Немає.

– Можливість установки SSL сертифіката на різних серверах – Так.

GlobalSign OrganizationSSL

1. Для одержання SSL сертифікат ви повинні бути зареєстровані як юридична особа.

2. При генерації CSR запиту переконаєтеся, що в поле Organization ви ввели коректну назву компанії Англійською мовою (відповідно до статутних документів). Не можна використовувати в написанні спеціальні символи << >> або / “ “.

3. В WHOIS доменного ім'я для якого замовляється SSL сертифікат повинне бути ідентична назва компанії й контактні дані (Якщо WHOIS інформація схована, її варто відкрити).

4. Підготувати перекладені на англійську мову й завірені нотаріально реєстраційні документи (Свідчення про реєстрацію компанії, копію статутних документів). Також можуть знадобитися копії рахунків за телефонний зв'язок і комунальні послуги.

Додаткова інформація про SSL сертифікат:

– Верифікація – Компанії.

– Час видачі SSL сертифіката – 1-2 дні.

– Безкоштовна перевидача SSL сертифіката – так.

– Зелений рядок браузера – Немає.

– Перевірка роботи сертифіката – Так.

– Страховка сертифіката – 10,000$.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Так.

– Сумісність із браузерами – 99.3%.

– Захист субдоменів – Немає.

– Можливість установки SSL сертифіката на різних серверах – Так.

GlobalSign EV SSL Certificate

1. Для одержання SSL сертифікат ви повинні бути зареєстровані як юридична особа.

2. При генерації CSR запиту переконаєтеся, що в поле Organization ви ввели коректну назву компанії Англійською мовою (відповідно до статутних документів). Не можна використовувати в написанні спеціальні символи << >> або / “ “.

3. В WHOIS доменного ім'я для якого замовляється SSL сертифікат повинне бути ідентична назва компанії й контактні дані (Якщо WHOIS інформація схована, її варто відкрити).

4. Підготувати перекладені на англійську мову й завірені нотаріально реєстраційні документи (Свідчення про реєстрацію компанії, копію статутних документів). Також можуть знадобитися копії рахунків за телефонний зв'язок і комунальні послуги.

Додаткова інформація про SSL сертифікат:

– Верифікація – Розширена.

– Час видачі SSL сертифіката – 3-4 дні.

– Безкоштовна перевидача SSL сертифіката – так.

– Зелений рядок браузера – Так.

– Перевірка роботи сертифіката – Так.

– Страховка сертифіката – 250,000$.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Так.

– Сумісність із браузерами – 99.3%.

– Захист субдоменів – Немає.

– Можливість установки SSL сертифіката на різних серверах – Так.

SSL сертифікати DigiCert

DigiCert SSL Plus

Додаткова інформація про SSL сертифікат:

– Верифікація – Компанії.

– Час видачі SSL сертифіката – 1 день.

– Безкоштовна перевидача SSL сертифіката – так.

– Зелений рядок браузера – Немає.

– Перевірка роботи сертифіката – Так.

– Страховка сертифіката – 1,000,000$.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Так.

– Сумісність із браузерами – 99.3%.

– Захист субдоменів – Немає.

– Можливість установки SSL сертифіката на різних серверах – Так.

DigiCert EV Certificate

1. Для одержання SSL сертифікат ви повинні бути зареєстровані як юридична особа.

2. При генерації CSR запиту переконаєтеся, що в поле Organization ви ввели коректну назву компанії Англійською мовою (відповідно до статутних документів). Не можна використовувати в написанні спеціальні символи << >> або / “ “.

3. В WHOIS доменного ім'я для якого замовляється SSL сертифікат повинне бути ідентична назва компанії й контактні дані (Якщо WHOIS інформація схована, її варто відкрити).

4. Підготувати перекладені на англійську мову й завірені нотаріально реєстраційні документи (Свідчення про реєстрацію компанії, копію статутних документів). Також можуть знадобитися копії рахунків за телефонний зв'язок і комунальні послуги.

Додаткова інформація про SSL сертифікат:

– Верифікація – Розширена.

– Час видачі SSL сертифіката – 1 день.

– Безкоштовна перевидача SSL сертифіката – так.

– Зелений рядок браузера – Так.

– Перевірка роботи сертифіката – Так.

– Страховка сертифіката – 1,000,000$.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Так.

– Сумісність із браузерами – 99.3%.

– Захист субдоменів – Немає.

– Можливість установки SSL сертифіката на різних серверах – Немає.

DigiCert WildCard Certificate

1. Для одержання SSL сертифікат ви повинні бути зареєстровані як юридична особа.

2. При генерації CSR запиту переконаєтеся, що в поле Organization ви ввели коректну назву компанії Англійською мовою (відповідно до статутних документів). Не можна використовувати в написанні спеціальні символи << >> або / “ “.

3. В WHOIS доменного ім'я для якого замовляється SSL сертифікат повинне бути ідентична назва компанії й контактні дані (Якщо WHOIS інформація схована, її варто відкрити).

4. Підготувати перекладені на англійську мову й завірені нотаріально реєстраційні документи (Свідчення про реєстрацію компанії, копію статутних документів). Також можуть знадобитися копії рахунків за телефонний зв'язок і комунальні послуги.

Додаткова інформація про SSL сертифікат:

– Верифікація -.

– Час видачі SSL сертифіката -.

– Безкоштовна перевидача SSL сертифіката -.

– Зелений рядок браузера – Немає.

– Перевірка роботи сертифіката – Немає.

– Страховка сертифіката -.

– Встановлення безкоштовної електронної печаті на сайт (логотипу) (Site Seal) – Немає.

– Сумісність із браузерами -.

– Захист субдоменів – Немає.

– Можливість установки SSL сертифіката на різних серверах – Немає.

3.2 Розробка структурної схеми

Для здійснення SSL з'єднання необхідно, щоб сервер мав інстальований цифровий сертифікат. Цифровий сертифікат – це файл, що унікальним образом ідентифікує користувачів і сервери. Це свого роду електронний паспорт, що проводить автентифікацію сервера до того, як установлюється сеанс SSL з'єднання. Звичайно цифровий сертифікат незалежно підписується й засвідчується третьою стороною, що гарантує його вірогідність. У ролі такої третьої сторони виступають центри сертифікації, зокрема, компанія Thawte. Компанія Thawte є найбільш відомим у світі центром сертифікації.

Протокол SSL забезпечує захищений обмін даними за рахунок сполучення двох наступних елементів.

Автентифікація

Цифровий сертифікат прив'язаний до конкретного домену мережі Інтернет, а центр сертифікації проводить перевірки, що підтверджують дійсність організації, а вже потім створює й підписує цифровий сертифікат для цієї організації. Такий сертифікат може бути встановлений тільки на той домен web-сервера, частка якого він пройшов автентифікацію, що й дає користувачам мережі Інтернет необхідні гарантії.

Перетворення інформації з метою збереження конфіденційності

Перетворення інформації з метою збереження конфіденційності – це процес перетворення інформації, яка не читається для всіх, крім конкретного одержувача. Воно ґрунтується на необхідні для електронної комерції гарантіях конфіденційності передачі інформації й неможливості її фальсифікації.

Одна з ознак того, чи має web-сайт SSL сертифікат, ви знайдете в рядку стану (status bar) браузера. Зверніть увагу на значок у вигляді замочку.

Іншу ознаку ви знайдете в рядку адрес. Якщо між браузером і web-сервером установлюється захищене з'єднання, то префікс адреси «http» зміниться на «https».

Також можливо одержати додаткову інформацію про ступінь захищеності конкретного SSL сеансу. У браузері IE просто наведіть курсор миші на замочок, і Ви побачите бітність (довжину) ключа перетворення інформації з метою збереження конфіденційності.

У браузері Netscape подвійне клацання по значку замка покаже вам SSL сертифікат. Бітність ключа перетворення інформації з метою збереження конфіденційності Ви знайдете на першій закладці сертифіката.

SSL сертифікат web-сервера дозволяє відвідувачам Вашого сайту бачити наступну інформацію (детальна інформація про сертифікат представлена на закладці «Состав»):

– Домен мережі Інтернет, для якого випущений цей сертифікат. Ця інформація дозволяє переконатися, що даний SSL сертифікат web-сервера був випущений саме для вашого домену й домену (http://www.mydomain.com/).

– Власник сертифіката. Ця інформація є додатковою гарантією, оскільки відвідувач може побачити ім'я того, з ким веде бізнес.

– Місто, де зареєстроване компанія-власник сертифіката. Ця інформація ще раз переконує відвідувача, що він має справу з реальною організацією.

– Термін дії сертифіката. Ця інформація особливо важлива, оскільки показує користувачеві, що ваш цифровий сертифікат діючий.

При відвідуванні ж web-сайту з діючим сертифікатом користувач буде проінформований про те, що даний сайт має цифровий сертифікат від центра сертифікації, такого як компанія Thawte, і всі дані, які користувач надає даному сайту, будуть зашифровані. Перевіряючи сертифікат, клієнт може переконатися у тому, що web-сайт належить реальній зареєстрованій компанії, а також що він звертається до доменного ім'я, яким володіє саме ця компанія.

Коли ви встановлюєте мережне з'єднання із захищеним web-сервером, таким як https://www.ssl.ua, сервер повинен спочатку сам автентифікувати клієнтський web-браузер, що здійснюється за допомогою цифрового сертифіката, після чого встановлюється захищене з'єднання.

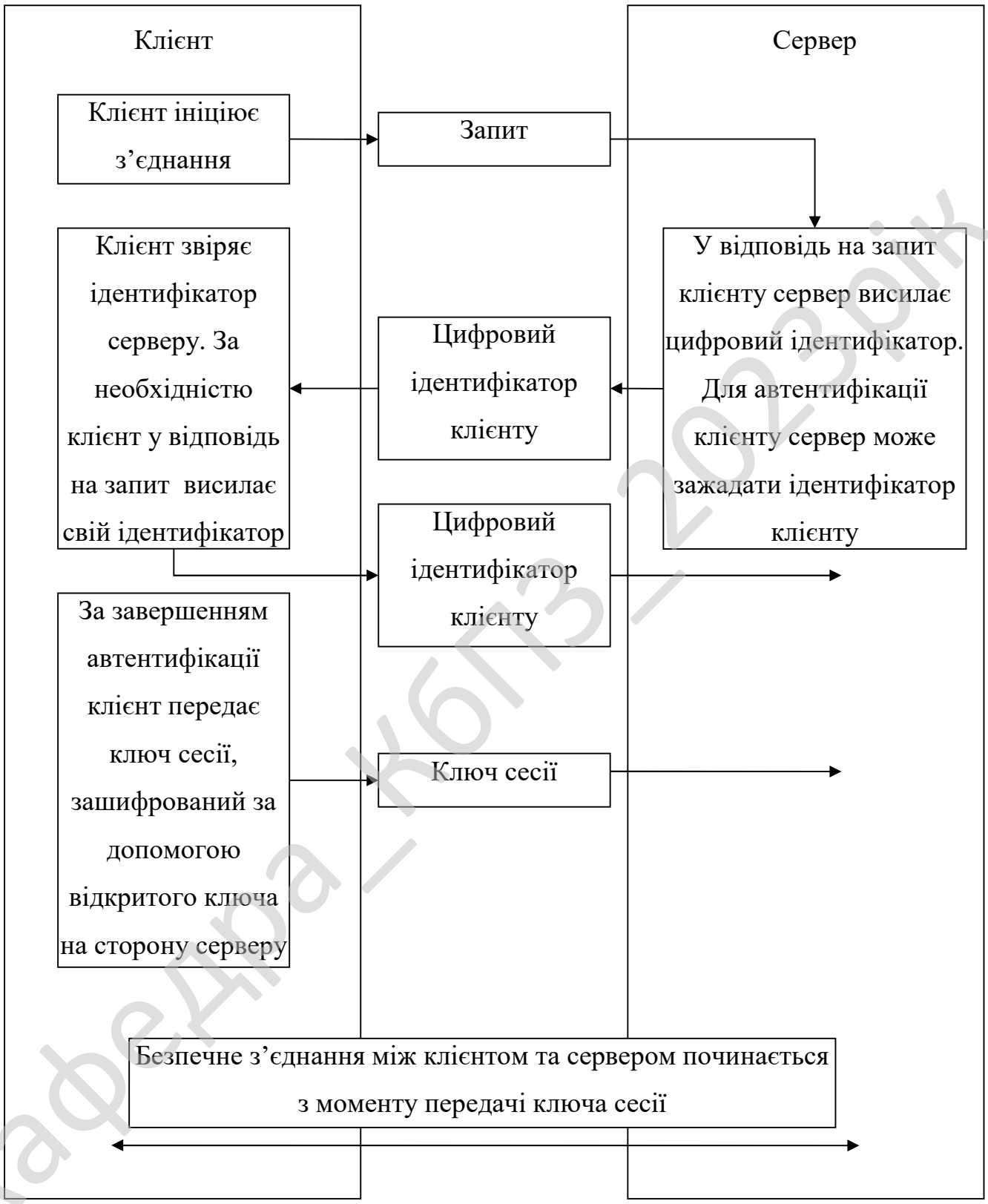

Рисунок 3.1 – Структурна схема системи

Структурна схема показує кроки, необхідні для установки захищеної сесії за протоколом SSL.

У ході цієї процедури web-браузер перевіряє, щоб:

– доменне ім'я в сертифікаті відповідало тому домену, від якого йде запит на захищене з'єднання;

– сертифікат не був прострочений;

– сертифікації, що підписала сертифікат домену, входив до числа довірених вашого web-браузера.

Етапи цієї процедури проходять без пауз, так що користувач «не почуває» цієї внутрішньої роботи. Сертифікат служить електронним документом, незалежним образом завіряється третьою стороною, такий як компанія Thawte, що гарантує, що домен належить саме цій реально існуючій компанії. Діючий сертифікат дає користувачеві гарантію конфіденційності при передачі минулого автентифікацію вузлу Інтернет. Передача інформації здійснюється захищеним образом.

Відкритий і секретний ключі

Коли Ви запитуєте сертифікат для домену своєї компанії, то запускаєте на своєму сервері процес генерації пари ключів – відкритого й секретного. Таємний ключ установлюється на сервері й при цьому дуже важливо, щоб ніхто крім Вас не мав доступу до нього. На основі секретного ключа створюється цифровий підпис, що є своєрідною електронною печаткою Вашої компанії. Якщо Ви втратите секретний ключ, то більше не зможете використовувати свій сертифікат. Необхідно створювати резервні копії секретного ключа, що є невід'ємним елементом керування безпекою інформації.

Парний відкритий ключ установлюється на web-сервер і є частиною цифрового сертифіката. Відкритий і секретний ключі зв'язані математичною закономірністю, але не ідентичні. Бажаючий установити з Вами захищений сеанс (за допомогою SSL протоколу) клієнт звертається до відкритого ключа вашого сертифіката й за допомогою відкритого ключа шифрує інформацію, що

посилається Вам. Цей процес відбувається миттєво й непомітно для користувача. Розшифрувати інформацію можна тільки за допомогою секретного ключа сервера. Це дає гарантію клієнтові, що ніяка третя сторона не одержить доступ до його інформації.

Рішення про застосування SSL сертифіката виходить із важливості забезпечення конфіденційної передачі даних у мережі Інтернет. Наприклад, при проведенні через ваш web-сайт фінансових транзакцій питання про застосування SSL сертифіката самоочевидно. Якщо ви обробляєте значиму персональну інформацію про клієнтів, таку як номер соціального страхування або іншу подібну інформацію, то застосування SSL сертифіката має серйозні підстави. Особливо якщо питання конфіденційності й інформаційної безпеки ваших клієнтів мають високий пріоритет.

У сфері бізнесу застосування SSL сертифікатів гарантує клієнтам, що ризик витоку інформації при її передачі через відкриті мережі виключений. Саме по собі це вже дає очевидну перевагу, адже багато видів комерційної діяльності будуються на основі довіри до мережних партнерів. Тому якщо ваш успіх у бізнесі залежить від установлення довірчих відносин із клієнтами й проведення онлайнових транзакцій, то застосування SSL сертифіката стає життєво необхідним.

Цифрові сертифікати використовуються на більшості web-серверів і прийняті як довірені в більшості браузерів, так що можете бути впевнені, що ви здобуваєте цифровий сертифікат, що дає клієнтові впевненість, що він спілкується саме з Вами й одержує від Вас неспотворені дані. Така впевненість необхідна при проведенні онлайнових транзакцій.

Значимість автентифікації web-сервера

Інформація – це основа життєзабезпечення вашого бізнесу. Щоб забезпечити цілісність (захист від фальсифікації) і конфіденційність, необхідно ідентифікувати тих, з ким ви маєте справу, а також бути впевненими в надійності одержуваних даних. Автентифікація допоможе вам установити довіру між

сторонами при проведенні будь-яких видів транзакцій. До числа засобів забезпечення надійності ставиться запобігання наступних явищ:

– «Спуфінг» (імітація з'єднання) – Відносно дешеві послуги web-дизайнерів і легкість, з якої можуть бути скопійовані вже існуючі сторінки, дозволяють створювати нелегальні web-сайти, які виглядають «офіційно» і на вид представляють організацію. Реально ж це пастка, щоб незаконно одержати, скажемо, номера кредитних карт.

– Несанкціоновані дії – Конкурент або скривджений користувач можуть змінити ваш web-сайт так, щоб він подавав помилкову інформацію або відмовлявся обслуговувати потенційних клієнтів.

– Неправомочне розголошення інформації – Коли транзакції здійснюються «відкритим текстом», хакер може їх перехоплювати, щоб одержати інформацію, важливу для ваших клієнтів.

– Фальсифікація даних – Зміст транзакції може бути перехоплене й зловмисно або випадково в процесі передачі змінено. Імена користувачів, номера кредитних карт і фінансова інформація, передана «відкритим текстом» занадто уразлива для втручання з боку.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно.

Алгоритми, що використовуються в SSL

– Для обміну ключами й перевірки їхньої дійсності застосовуються: RSA, Diffie-Hellman, ECDH, SRP, PSK.

– Для автентифікації: RSA, DSA, ECDSA.

– Для симетричного перетворення інформації з метою збереження конфіденційності: RC2, RC4, IDEA, DES, Triple DES або AES, Camellia.

– Для хеш-функцій: SHA, MD5, MD4 і MD2.

Алгоритми перетворення інформації з метою збереження конфіденційності:

– Обмін ключами й перевірка їх дійсності: RSA, Diffie-Hellman, ECDH, SRP, PSK.
– Автентифікація: RSA, DSA, ECDSA.
– Симетричне шифрування: RC2, RC4, IDEA, DES, Triple DES, AES, Camellia.
– Хеш-функція: SHA, MD5, MD4, MD2.

Сертифікати:

1. SSL сертифікати RapidSSL.
2. SSL сертифікати Geotrust.
3. SSL сертифікати Thawte.
4. SSL сертифікати VeriSign.
5. SSL сертифікати Comodo.
6. SSL сертифікати GlobalSign.
7. SSL сертифікати DigiCert.

Інтерфейс користувача системи передачі даних у мережі за протоколом SSL

Блок реалізації алгоритму системи передачі даних у мережі за протоколом SSL

Протокол ESP

Протокол AH

Алгоритм шифрування симетричним алгоритмом

Алгоритм автентифікації

Область інтерпретації (DOI)

Управління ключами (ISAKMP/Oakley)

Рисунок 3.2 – Функціональна схема системі

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 перетворення інформації з метою збереження конфіденційності. 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 перетворення інформації з метою збереження конфіденційності. 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 перетворення інформації з метою збереження конфіденційності для декодування інформації транспортного рівня.

Заголовок AH

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

Формат AH досить простий і складається з 96-бітового заголовка й даних змінної довжини, що складаються з 32-бітових слів. Назви полів досить ясно відбивають їхній зміст: Next Header указує на наступний заголовок, Payload Len представляє довжину пакета, SPI є покажчиком на контекст безпеки й Sequence Number Field містить послідовний номер пакета.

На відміну від алгоритмів обчислення контрольної суми, застосовуваних у протоколах передачі інформації з линиям зв'язку, що комутуються або по каналах локальних мереж і орієнтованих на виправлення випадкових помилок середовища передачі, механізми забезпечення цілісності даних у відкритих телекомунікаційних мережах повинні мати засоби захисту від внесення цілеспрямованих змін. Одним з таких механізмів є спеціальне застосування алгоритму MD5: у процесі формування AH послідовно обчислюється хеш-функція від об'єднання самого пакета й деякого попередньо погодженого ключа, а потім від об'єднання отриманого результату й перетвореного ключа. Даний механізм застосовується за замовчуванням з метою забезпечення всіх реалізацій IPv6, принаймні, одним загальним алгоритмом, не підданим експортним обмеженням.

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

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

Швидкий режим простіше основного, і узгодження SA для IPsec здійснюється за допомогою трьох пакетів. IPsec-ключі створюються за допомогою простих операцій піднесення в ступінь переданих в основному режимі даних. У швидкому режимі погодяться також алгоритми перетворення інформації з метою збереження конфіденційності 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 перетворення інформації з метою збереження конфіденційності 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.

Сертифікати, які застосовуються в розробленій системі

1. SSL сертифікати RapidSSL:

– RapidSSL®.

– RapidSSL® WildCard.

2. SSL сертифікати Geotrust:

– GeoTrust QuickSSL® Basic.

– GeoTrust QuickSSL® Premium.

– GeoTrust True BusinessID Wildcard.

– GeoTrust TrueBusinessID Multi-Domain.

– GeoTrust TrueBusinessID EV Multi-Domain.

– GeoTrust True BusinessID with EV Certificate.

3. SSL сертифікати Thawte:

– Thawte SSL123.

– Thawte SSL Web Server.

– Thawte SSL Web Server with EV.

– Thawte Code Signing.

– Thawte SGC SuperCerts.

– Thawte SSL Webserver Wildcard.

4. SSL сертифікати VeriSign:

– VeriSign Secure Site.

– VeriSign Secure Site EV.

– VeriSign Secure Site Pro.

– VeriSign Secure Site Pro EV.

– VeriSign Code Signing.

5. SSL сертифікати Comodo:

– Comodo Positive SSL.

– Comodo InstantSSL.

– Comodo InstantSSL Premium.

– Comodo InstantSSL Pro.

– Comodo Multi-Domain Certificate.

– Comodo EV Multi-Domain.

– Comodo EV SGC Certificate.

– Comodo EV SSL Certificate.

– Comodo Positive SSL Wildcard.

– Comodo Premium Wildcard.

– Comodo SGC SSL Wildcard.

– Comodo SGC SSL.

– Comodo Unified Communications.

– Comodo Secure Email Certificate.

6. SSL сертифікати GlobalSign:

– GlobalSign DomainSSL Certificate.

– GlobalSign OrganizationSSL.

– GlobalSign EV SSL Certificate

7. SSL сертифікати DigiCert:

– DigiCert SSL Plus.

– DigiCert EV Certificate.

– DigiCert WildCard Certificate.

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

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання бакалаврського проектування, наведена на рисунку 3.3. Першим процесом, який завантажується у системі, є процес виведення головного вікна програми.

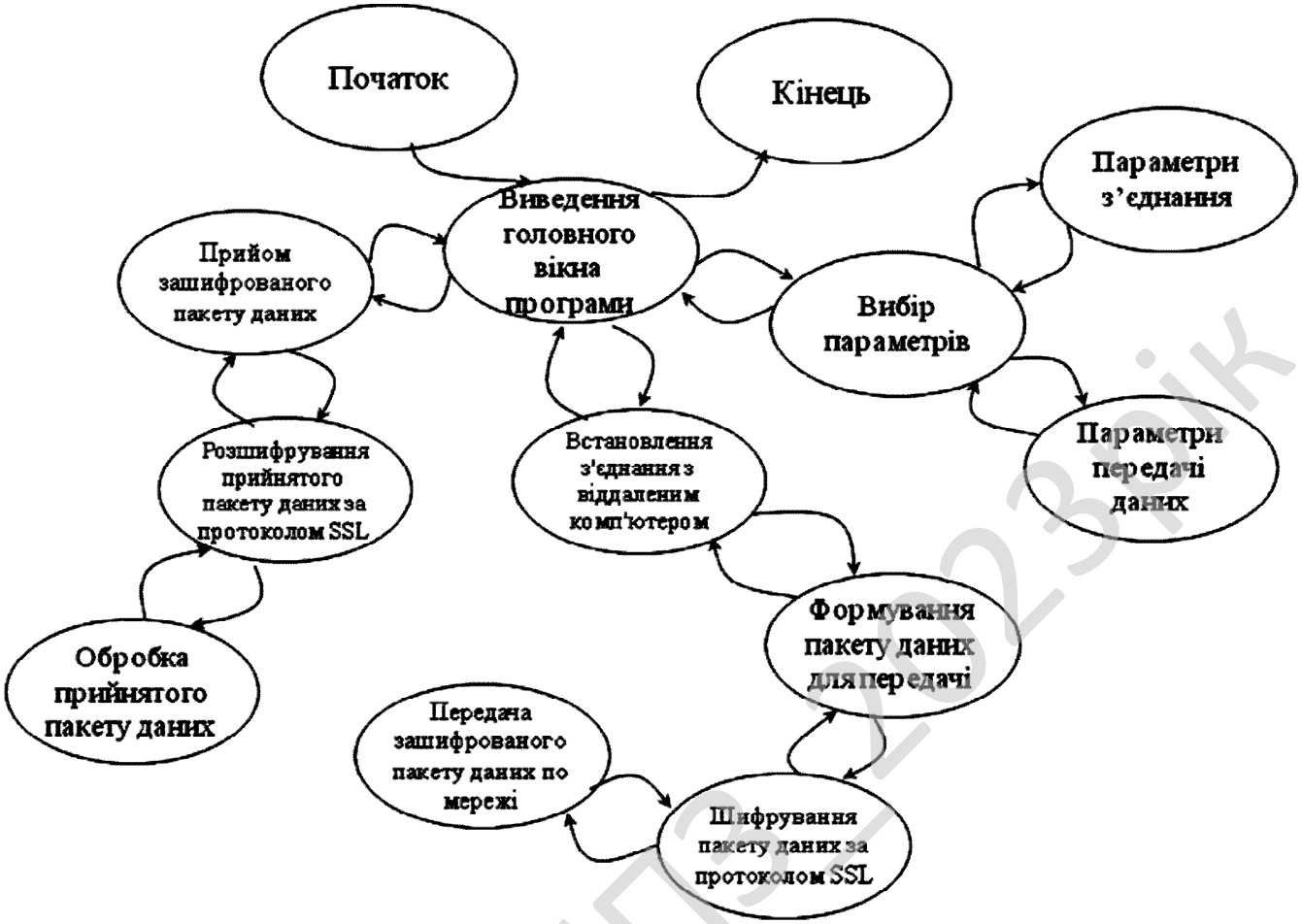

Рисунок 3.3 – Діаграма взаємодії процесів

Процес виведення головного вікна програми взаємодіє з наступними процесами:

– Процес вибору параметрів.

– Процес встановлення з'єднання з віддаленим комп'ютером.

– Процес прийому зашифрованого пакету даних.

Процес вибору параметрів взаємодіє з наступними процесами:

– Процес вибору параметрів з'єднання.

– Процес вибору параметрів передачі даних.

Процес встановлення з'єднання з віддаленим комп'ютером взаємодіє з процесом формування пакету передачі даних, який, у свою чергу взаємодіє з процесом шифрування пакету даних за протоколом SSL, який, у свою чергу

взаємодіє з процесом передачі зашифрованого пакету даних по мережі.

Процес прийому зашифрованого пакету даних взаємодіє з процесом розшифрування прийнятого пакету даних за протоколом SSL, який, у свою чергу взаємодіє з процесом обробки прийнятого пакету даних.

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рисунку 4.1 наведено блок-схему роботи основної програми. Робота основної програми складається з виконання наступних кроків:

Крок 1. Виведення основного вікна програми.

Крок 2. Введення параметрів з'єднання з сервером.

Крок 3. Якщо користувач обирає дію з'єднання з мережею, то програма переходить до виконання наступних кроків, інакше повертаємося на крок 2.

Крок 4. Відбувається підключення до мережі.

Крок 5. Виведення стану поточного підключення до мережі.

Крок 6. Якщо користувач обирає дію включити шифрування трафіку, то програма виконує кроки 7-8 та переходить до наступних, інакше відразу переходить до кроку 9.

Крок 7. Введення паролю користувача.

Крок 8. Запуск підпрограми шифрування IP- пакетів за технологією SSL.

Крок 9. Відправка IP-пакетів по захищеному з'єднанню.

Крок 10. Якщо відбувається прийом IP-пакетів, то програма виконує кроки 11-14, інакше переходить до кроку 15.

Крок 11. Якщо прийняті пакети зашифровані, то програма переходить до виконання наступних кроків, інакше відразу переходить до виконання кроку 14.

Крок 12. Введення паролю користувача.

Крок 13. Запуск підпрограми дешифрування IP-пакетів за технологією SSL.

Крок 14. Прийом та обробка IP-пакетів.

Крок 15. Якщо користувач обирає дію вихід з програми, то захищене з'єднання вимикається та закривається вікно програми, інакше повертаємося до кроку 5.

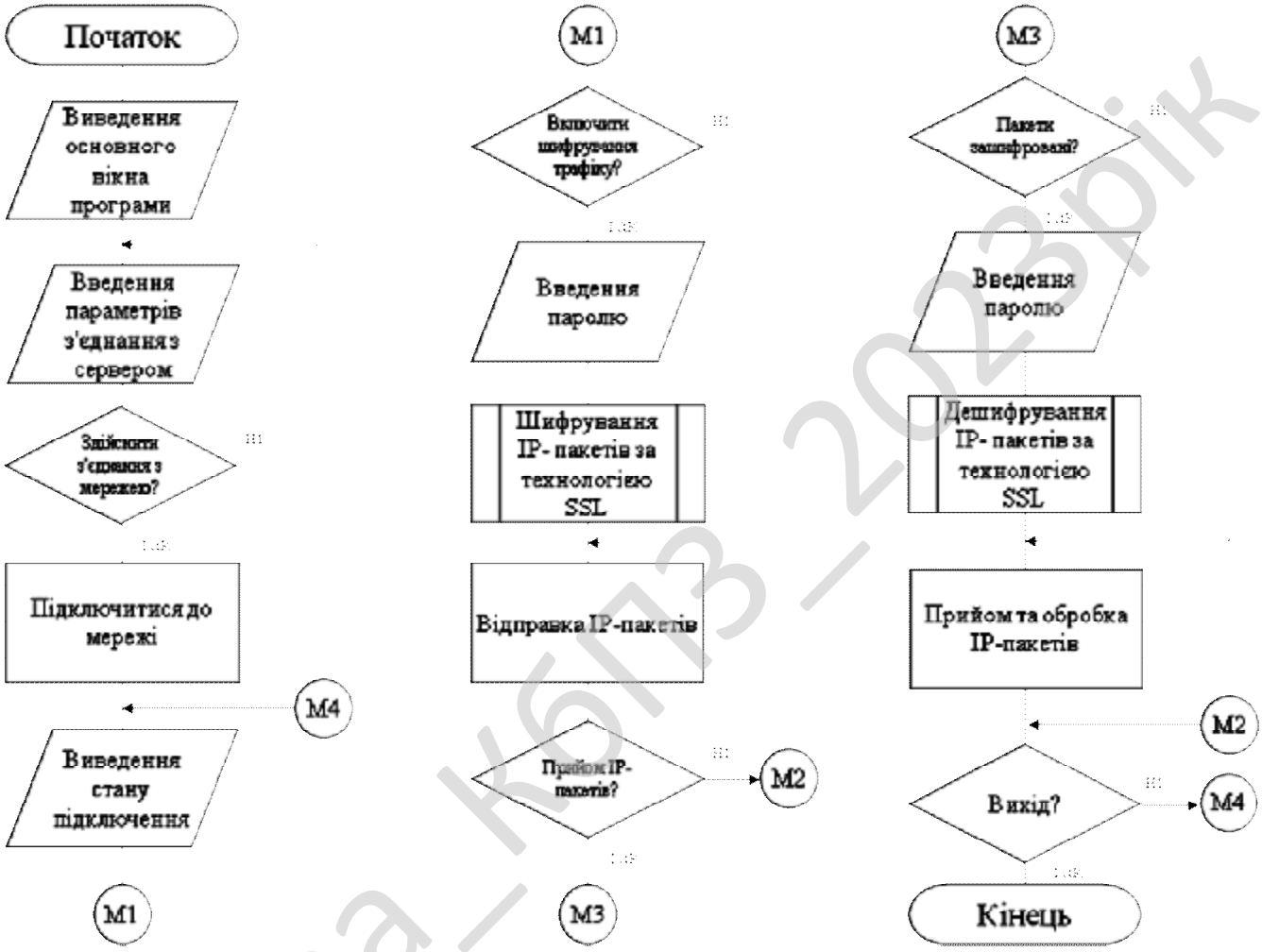

Рисунок 4.1 – Блок-схема основної програми

Розглянемо детальніше принцип роботи протоколу SSL.

Принципи роботи протоколу SSL

Потокол SSL реалізує наступні функції:

– Конфіденційність з'єднання. Після попереднього діалогу визначається секретний ключ, що використовується для симетричної криптографії (наприклад, DES або rc4).

– Партнери ідентифікують один одного за допомогою асиметричних

криптографічних методів (наприклад, RSA або DSS).

– Забезпечення надійності з'єднання. Пересилання містить у собі контроль цілісності повідомлень із застосуванням коду аутентификации MAC (Message Autentification Code) і хеш функцій (SHA або MD5).

SSL – це багаторівневий протокол і складається із чотирьох під-протоколів:

– SSL Handshake Protocol.

– SSL Change Cipher Spec Protocol.

– SSL Alert Protocol.

– SSL Record Layer.

Місця вищевказаних протоколів, відповідно до моделі стека протоколів TCP/IP показані на рисунку 4.2.

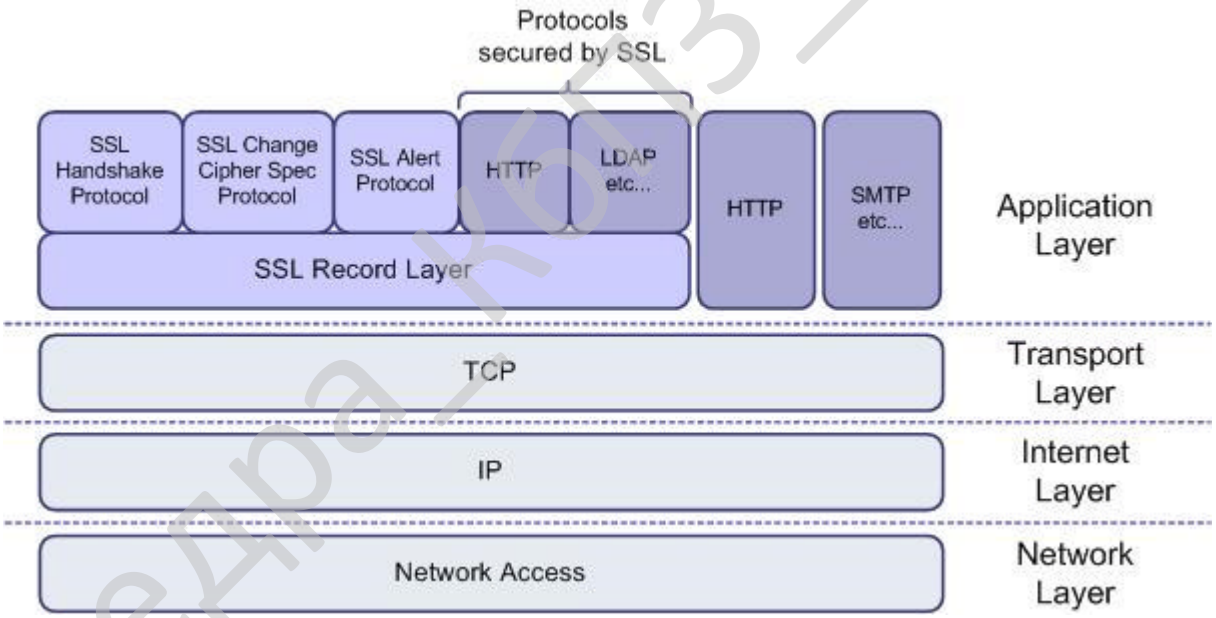

Рисунок 4.2 – Підпротоколи SSL у моделі TCP/IP

Як видно з рисунку, SSL перебуває на рівні додатків моделі TCP/IP. Завдяки цьому, SSL може бути розгорнутим майже на будь-який ОС, що підтримує TCP/IP, без якого-небуть виправлення ядра системи або TCP/IP стека. У результаті SSL має переваги перед іншими протоколами, такими як IPSec (IP

Security Protocol), що вимагає підтримки модифікованого TCP/IP стека ядром системи. Крім того, SSL легко пропускають брандмауери, проксі й NAT без проблем.

На рисунку 4.3 показаний спрощений покроковий процес встановлення SSL з'єднання між клієнтом (звичайно веб-браузер) і сервером (найчастіше SSL веб-сервер).

Отже, процес встановлення кожного нового SSL з'єднання починається з обміну параметрами шифрування, а потім (опціонально) відбувається аутентифікація серверів (через SSL Handshake Protocol). Якщо «рукопотискання» вдалося, і обидві сторони погодилися на той самий алгоритм шифрування й ключі шифрування, використовувані додатки (звичайно HTTP, але може бути й інший) можуть бути передані по зашифрованому каналу (використовується SSL Record Layer).

Якщо розглядати реальну ситуацію, то процес, описаний вище, виглядає набагато складніше. Щоб уникнути непотрібних «рукостискань» деякі параметри шифрування кешуються. При цьому можуть бути надіслані попереджуючі повідомлення. Також можуть бути змінені блоки шифру. Як би те не було, не залежно від тонкостей специфікації SSL, у загальному видгляді цей процес працює приблизно так, як показано на рисунку 4.3.

У рамках протоколу SSL визначені чотири види шифрування:

- digitally-signed (шифрування електронного підпису);
- stream-ciphered (потокове шифрування);
- block-ciphered (блокове шифрування);
- publick-key-encrypted (шифрування за допомогою відкритого ключа).

Електронний підпис потребує використання хешування до шифрування. В RSA-підписі 36-байтова структура двох хеш-функцій SHA і MD5 шифрується за допомогою закритого ключа. В DSS 20-байтовий блок хеш-функції SHA безпосередньо передається на обробку алгоритму цифрового підпису без додаткового хешування.

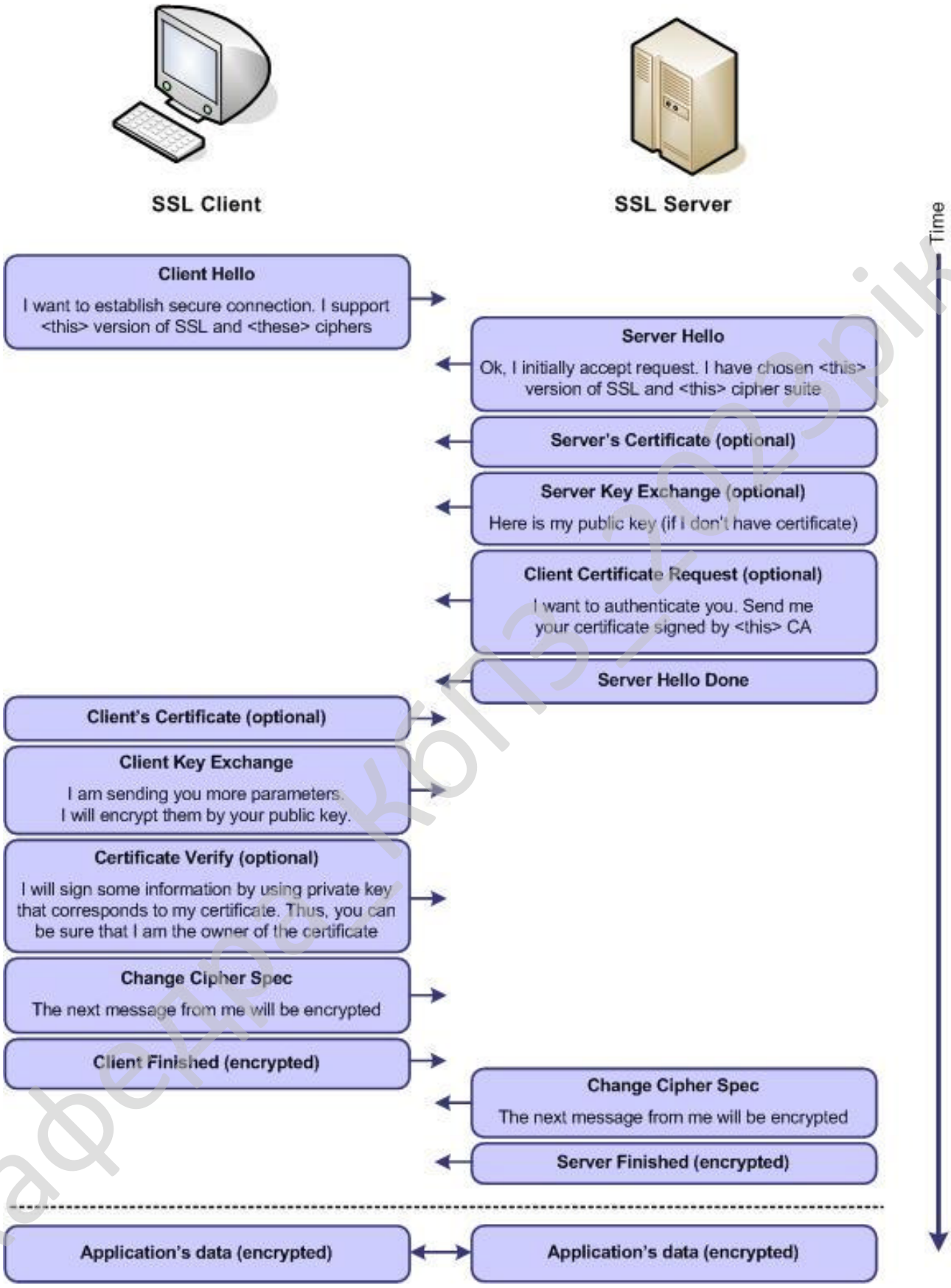

Рисунок 4.3 – Встановлення SSL з'єднань, крок за кроком

При потоковому шифруванні вихідний текст піддається обробці з використанням операції XOR за допомогою криптографічного ключа еквівалентної довжини, створюваного за допомогою генератора псевдовипадкових чисел.

У блоковому шифруванні кожний блок вихідного тексту перетворюється в рівний йому за розміром шифрований текст. Звичайно розмір блоку дорівнює 64 байтам. Якщо необхідно вихідний текст доповнюється до необхідного розміру блоку нулями.

Шифрування з використанням відкритого ключа припускає дешифрування із застосуванням секретного ключа або навпаки (ключі, що утворюють пару, симетричні з погляду свого використання).

Протокол SSL припускає послідовний перехід клієнта й сервера з одного стану в інший. Кожна процедура реалізована в строго певному стані об'єкта. Діалогова частина протоколу SSL дозволяє координувати роботу машин станів клієнта й сервера. Логічно будь-який стан представляється двічі, у якості робітника (operating) стану й розглянутого стану (pending). Передбачено, крім того, стани читання й запису. Коли клієнт або сервер одержує повідомлення **change cipher spec**, він копіює розглянутий стан у поточний стан читання. При посилці повідомлення **change cipher spec** клієнт або сервер копіює розглянутий стан у поточний стан запису. Коли діалог узгодження завершений, клієнт і сервер обмінюються повідомленнями **change cipher spec**, після чого взаємодіють один з одним, використовуючи погоджену специфікацію шифрування. Протокол SSL допускає будь-яке число з'єднань між клієнтом і сервером у рамках однієї сесії. Дозволена також реалізація довільного числа сесій паралельно. Стан сесії характеризується рядом параметрів, наведених в табл. 4.1.

Таблиця 4.1 – Параметри стану сесії

Параметр	Опис параметра
session identifier	Довільна послідовність байтів, обрана сервером щоб ідентифікувати активну сесію
peer sertificate	Сертифікат партнера – X509.v3. Цей елемент може бути дорівнює нулю
compression method	Алгоритм, використовуваний для стиску інформації перед шифруванням
cipher spec	Специфікація алгоритму шифрування даних (наприклад, нуль або DES) і алгоритм MAC (наприклад MD5 або SHA). Вона визначає криптографічні атрибути, такі як hash_size.
master secret	48-байтовий секретний код, загальний для клієнта й сервера
is resumable	Прапор, що вказує, що сесія може використовуватися для формування нових з'єднань

Стан з'єднання характеризується параметрами наведеними в табл. 4.2.

Рівень записів протоколу ssl здійснює фрагментацію повідомлень, розбиваючи їх на блоки 2^{12} байт або коротше. Всі записи стискуються з використанням погодженого для даної сесії алгоритму архівації. Стиск завжди повинен створюватися без втрат і не може збільшувати довжину вмісту більш ніж на 1024 байта. Всі записи захищаються за допомогою шифрування й алгоритму MAC, заданого в поточній специфікації cipherspec.

У процесі діалогу (handshake protocol) фіксуються наступні атрибути: версія протоколу, ідентифікатор сесії, шифровой набір і метод стиснення інформації. Коли діалог узгодження завершений, партнери мають загальний секретний код, що використовується для шифрування записів і для обчислення автентифікаційних кодів MAC. Методики шифрування й обчислення MAC задані в специфікації cipherspec. mac обчислюється до шифрування основних даних. Протокол діалогу є одним із клієнтів високого рівня протоколу записів SSL.

Таблиця 4.2 – Параметри стану з'єднання

Параметр	Опис параметра
server and client random	Послідовність байтів, обирана сервером і клієнтом для кожного з'єднання
server write mac secret	Секретний код, використовуваний в MAC-Операціях з даними, записаними сервером
client write MAC secret	Секретний код, використовуваний в MAC-Операціях з даними, записаними клієнтом
server write key	Ключ шифрування даних шифруємих сервером і дешифрованих клієнтом
client write key	Ключ шифрування даних шифруємих клієнтом і дешифрованих сервером
initialization vectors	Коли використовується блоковий шифр у режимі CBC, для кожного ключа підтримується ініціалізаційний вектор (IV). Це поле встановлюється першим у процесі стартового діалогу.
sequence numbers	Кожна зі сторін підтримує свої номери один по одному для переданих і отриманих повідомлень для кожного із з'єднань. Коли партнер посилає або одержує повідомлення **change cipher spec**, відповідне число, що характеризує номер обнуляється. Значення номера не може перевищувати 264-1.

Для блокових шифрів (RC2 або DES) шифрування й MAC-функції перетворять структури sslcompressed.fragment у структури sslciphertext.fragment.

На рисунку 4.4 наведено блок-схему роботи підпрограми обміну IP-пакетами за протоколом SSL. ЇЇ робота складається з наступних кроків:

Крок 1. Клієнт ініціює діалог посилкою повідомлення CLIENT-HELLO.

Крок 2. Програма переходить у стан очікування.

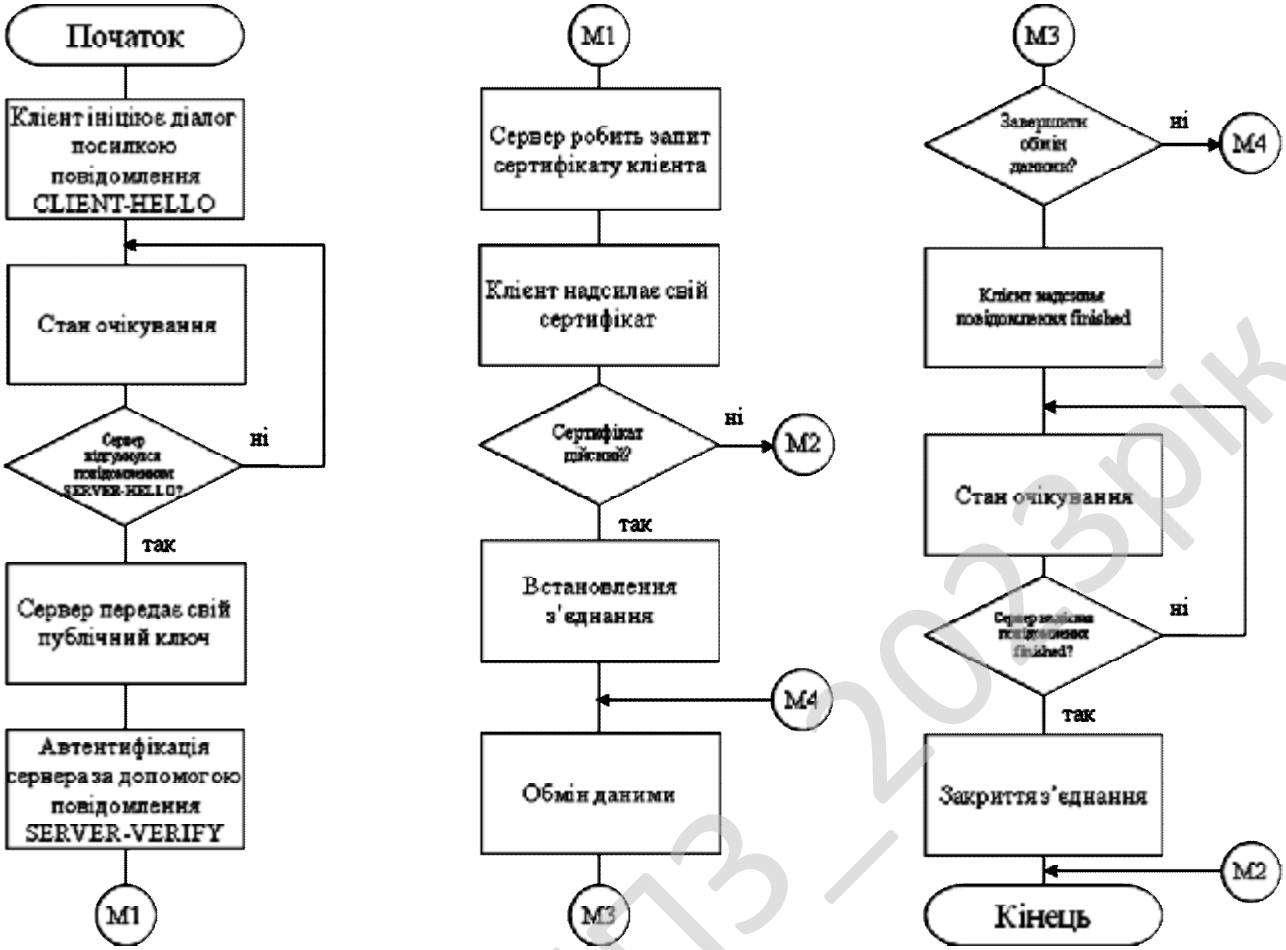

Рисунок 4.4 – Блок-схема підпрограми обміну IP-пакетами за протоколом SSL

Крок 3. Якщо сервер не відгукнувся повідомленням SERVER-HELLO, повертаємося на крок 2, інакше виконуємо наступні кроки.

Крок 4. Сервер передає свій публічний ключ.

Крок 5. Автентифікація сервера за допомогою повідомлення SERVER-VERIFY.

Крок 6. Сервер робить запит сертифікату клієнта.

Крок 7. Клієнт надсилає свій сертифікат.

Крок 8. Якщо сертифікат клієнта дійсний, то переходимо до виконання наступних кроків, інкше програма завершує виконання, з'єднання не створюється.

Крок 9. Встановлення з’єднання.

Крок 10. Обмін даними по захищеному з'єднанні у мережі.

Крок 11. Якщо клієнт бажає завершити обмін даними, то переходимо до виконання наступних кроків, інакше повертаємося на крок 10.

Крок 12. Клієнт надсилає повідомлення finished.

Крок 13. Програма переходить у стан очікування.

Крок 14. Якщо сервер надіслав повідомлення finished, то переходимо до виконання наступного кроку, інакше повертаємося на крок 13.

Крок 15. Закриття захищеного з'єднання.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою Sinople – симетричний блоковий криптоалгоритм, побудований на основі незбалансованої «мережі Фейстеля». Алгоритм розроблено у 2003 році.

Основні вимоги до алгоритму при його розробці:

– Можливість програмної і апаратної реалізації.

– Висока швидкість.

– Простота.

– Низькі вимоги до пам'яті.

– Високий рівень безпеки.

Алгоритм заснований на 32-розрядних операціях і має 64 раунду, серед яких два типи – C і D. D раунди спроектовані для досягнення максимальної дифузії, C раунди – для досягнення перемішування. F-функція D раунду використовує один з елементів блоку даних (D[3]) та поточного з'єднання (K[r]) для трансформації 3-х елементів блоку даних. F-функція C раунду, навпаки, використовує перші три елемента блоку даних і поточний з'єднання (K[r]) для трансформації останнього елемента блоку даних (D[3]). Раунди D-типу виконуються до раундів C-типу. Додавання ключів з даними проводиться тільки

через таблиці замін. Операції XOR (додавання по модулю 2) обов'язково поєднуються з операціями ADD (додавання по модулю 2^{32}).

Таблиці замін спочатку запозичені з алгоритму MARS і містять 512 32-розрядних елементів, проте були жорстко проаналізовано на предмет посилення.

Ключове розклад було спроектовано з урахуванням вимог:

– Простота

– Використовується та ж процедура, що і при шифруванні та розшифрування

– Установка ключа займає менше часу, ніж зашифрування

– Виключення еквівалентних ключів

– Виключення слабких ключів

Алгоритм, згідно із заявою авторів, стійкий до лінійного і диференціального аналізу.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розроблене програмне забезпечення реалізує метод створення самопідписаних SSL сертифікатів. Самозавірений (самопідписаний) сертифікат – спеціальний тип сертифіката, підписаний самим його автором. Технічно даний тип нічим не відрізняється від сертифіката, завіреного підписом центра сертифікатів (ЦС), тільки замість передачі на підпис в ЦС користувач створює свою власну сигнатуру. Приклади вікон програми наведені на рисунках 5.1–5.5.

Інструкція роботи з розробленою програмою

1. Створюємо свій центр сертифікації. За допомогою нього створюємо секретні ключі й сертифікати, підписані нашим центром сертифікації. Кореневий сертифікат міститься на сервер. Клієнтський ключ і сертифікат заноситься в браузер клієнта. При спробі підключитися до захищеного ресурсу, відбувається перевірка пари ключів. Якщо все добре, то створюється захищений канал між сервером і браузером, у якому дані «упаковуються» до криптографічного протоколу SSL, тим самим забезпечуючи захист цих даних.

2. Створюємо базу даних майбутнього центра сертифікації, де будуть зберігатися сертифікати користувачів.

3. Потім необхідно створити кореневий сертифікат і ключ центра сертифікації.

Вкладка *Закриті ключі-> Нові ключі* -> У поле *«Ім'я»* вводимо CA і натискаємо кнопку *«Створити»*. Тим самим відбулося створення закритого ключа центра сертифікації.

4. Створюємо сертифікат до закритого ключа ЦС.

Вкладка *Сертифікати -> Новий сертифікат*, у вікні, що відкриється, у *поле «Шаблон для нового сертифікату»* вибираємо *«[default] CA»*, натискаємо *кнопку «Застосувати»*. Цими діями відбувся вибір шаблону для

створення самопідписаного кореневого сертифікату. Далі переходимо на вкладку *Суб'єкти*. Там необхідно заповнити поля, які будуть занесені в кореневий сертифікат. Приклад заповнення представлений на рисунку 5.1.

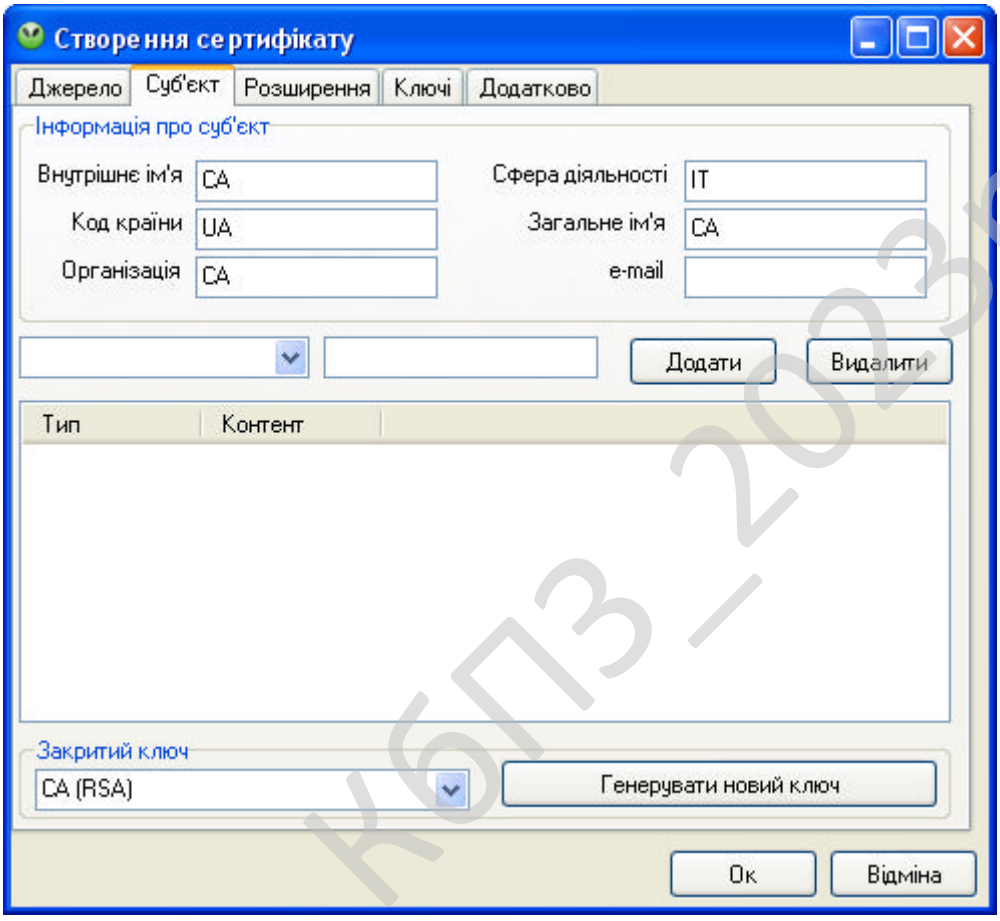

Рисунок 5.1 – Вікно створення сертифікату

Слід звернути увагу на поле «Закритий ключ». У ньому повинне бути зазначене ім'я закритого ключа ЦС, на який робиться кореневий сертифікат.

Переходимо на вкладку *Розширення* і натискаємо кнопку *«Затвердити»*, далі кнопку *«ОК»*. Тепер центр сертифікації створено.

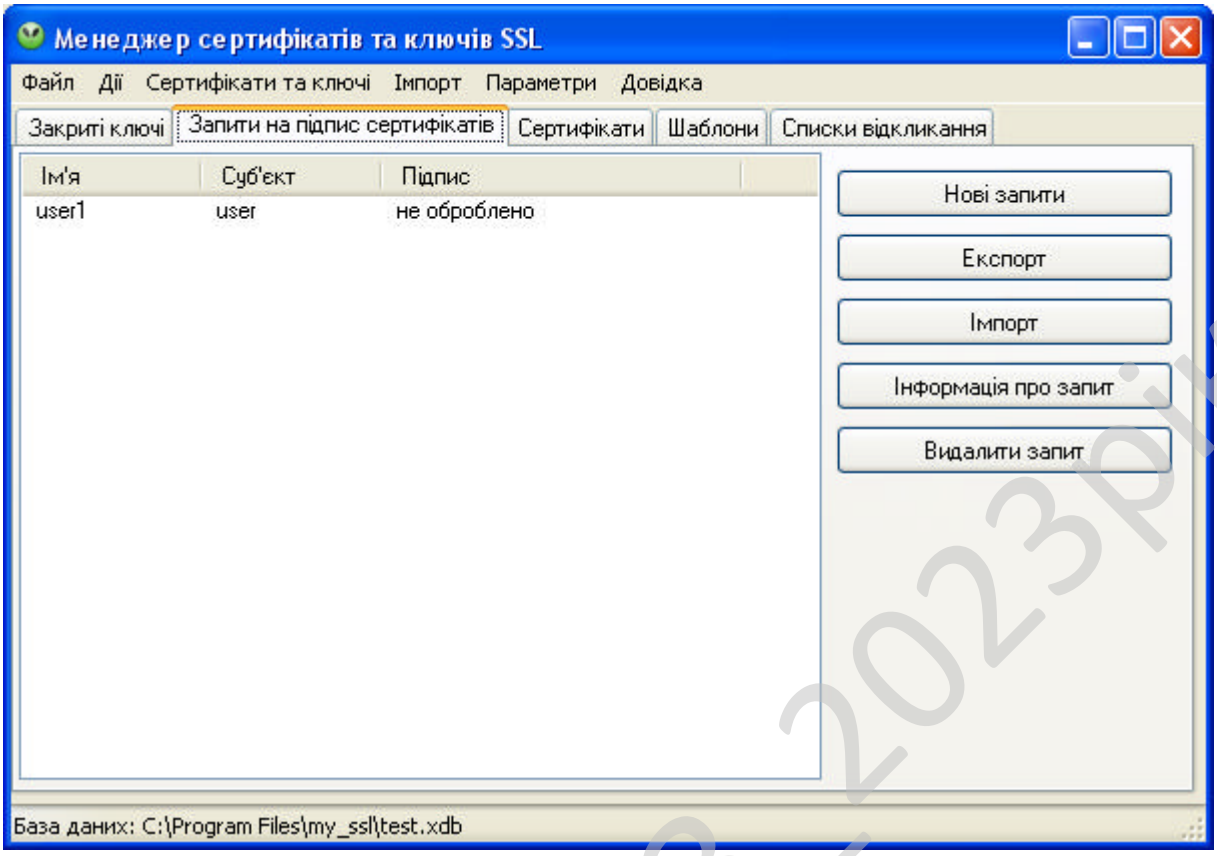

Рисунок 5.2 – Вікно менеджера сертифікатів та ключів SSL (вкладка «Запити на підпис сертифікатів»)

5. Необхідно створити ключі й сертифікати користувача й сервера. Робиться це наступним чином. Вкладка *Запити на підпис сертифікатів -> Новий запит*, у вікні, що з'явилося, переходимо на вкладку *Суб'єкт* і заповнюємо за аналогією з кореневим сертифікатом. Далі натискаємо кнопку *«Генерувати новий ключ»* і цим створюємо закритий ключ, після чого натискаємо кнопку *«ОК»*.

Натискаємо на створеному запиті правою кнопкою миші й вибираємо *Підпис*. У вікні, що з'явиться, у полі *Підпис* вибираємо *Використовувати цей сертифікат для підписання*. У полі введення з'являється кореневий сертифікат ЦС, натискаємо *«ОК»*. Отже, клієнтський ключ і сертифікат підписані центром сертифікації.

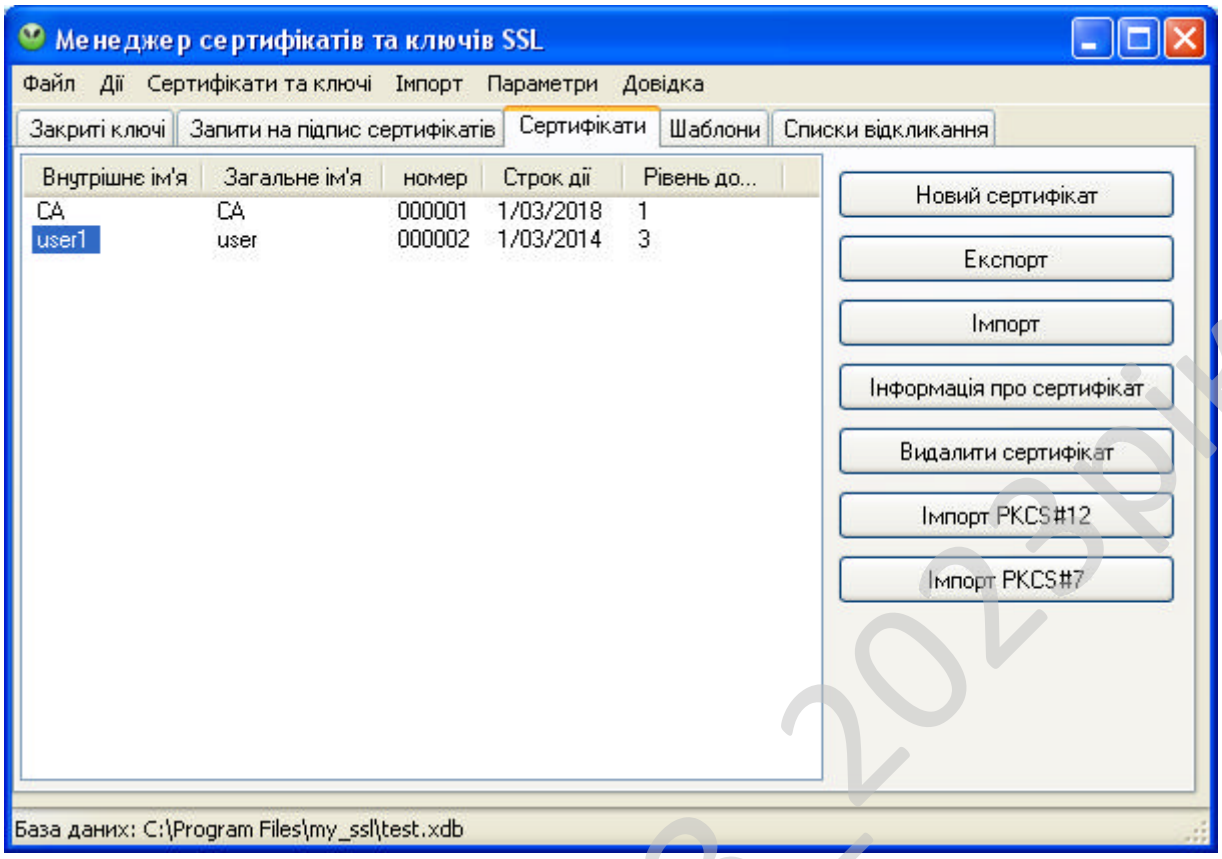

Рисунок 5.3 – Вікно менеджера сертифікатів та ключів SSL (вкладка «Сертифікати»)

6. У такий же спосіб створюємо ключ і сертифікат для сервера.

7. Експортуємо закриті ключі сервера й користувача у форматі *PEM*. Сертифікат користувача експортуємо у форматі *«PKCS #12 with Certificate chain»*.

PKCS#12 – один зі стандартів сімейства Public-Key Cryptography Standards (PKCS), опублікованих RSA Laboratories. Він визначає файловий формат, використовуваний для зберігання секретних ключів у супроводі із сертифікатами, захищений за допомогою заснованого на паролі симетричного ключа.

При експорті програма запросить пароль на файл (рис. 5.4).

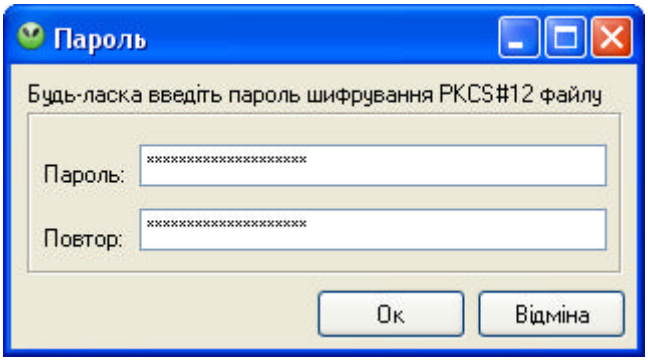

Рисунок 5.4 – Вікно введення пароля

Вводимо придуманий пароль. Наступного разу програма запросить його ввести при імпорті файлу в браузер користувача.

8. Сертифікат центра сертифікації й сервера експортуємо у формат «CRT».

У підсумку в нас повинно з'явитися:

– *CA.crt* – корнєвій сертифікат центра сертифікації;

– *Server.crt* – сертифікат сервера;

– *Server.pem* – закритий ключ сервера;

– *User.pkcs12* – закритий ключ + сертифікат клієнта.

– 9. Тепер слід налаштувати сервер.

Приклад файлу конфігурації *stunnel.conf*:

```
; Certificate/key is needed in server mode and optional in client mode
; Шляхи до закритого ключа й сертифіката сервера
cert = /usr/local/etc/stunnel/server.crt
key = /usr/local/etc/stunnel/server.pem
; Protocol version (all, SSLv2, SSLv3, TLSv1)
sslVersion = SSLv3
; Some security enhancements for UNIX systems - comment them out on Win32
chroot = /var/tmp/stunnel
setuid = stunnel
setgid = nogroup
pid = /stunnel.pid
; Some performance tunings
socket = l:TCP_NODELAY=1
socket = r:TCP_NODELAY=1
; Authentication stuff
```

```
verify = 2
; CApath is located inside chroot jail
CApath = /certs
; It's often easier to use CAfile
; Шлях до кореневого сертифіката центра сертифікації
CAfile = /usr/local/etc/stunnel/ca.crt
; Some debugging stuff useful for troubleshooting
debug = 7
output = /var/log/stunnel.log
; Настроювання
[https]
accept = 443
connect = 192.168.1.1:80
```

10. Залишилося імпортувати *User.pkcs12* і *CA.crt* у браузер користувача. При імпорті *User.pkcs12* програма запросить ввести пароль, той самий який було вказано при експорті з нашого центра сертифікації.

Після виконання вищеперерахованих дій, встановлюється захищений канал передачі даних між браузером і нашим ресурсом у локальній мережі.

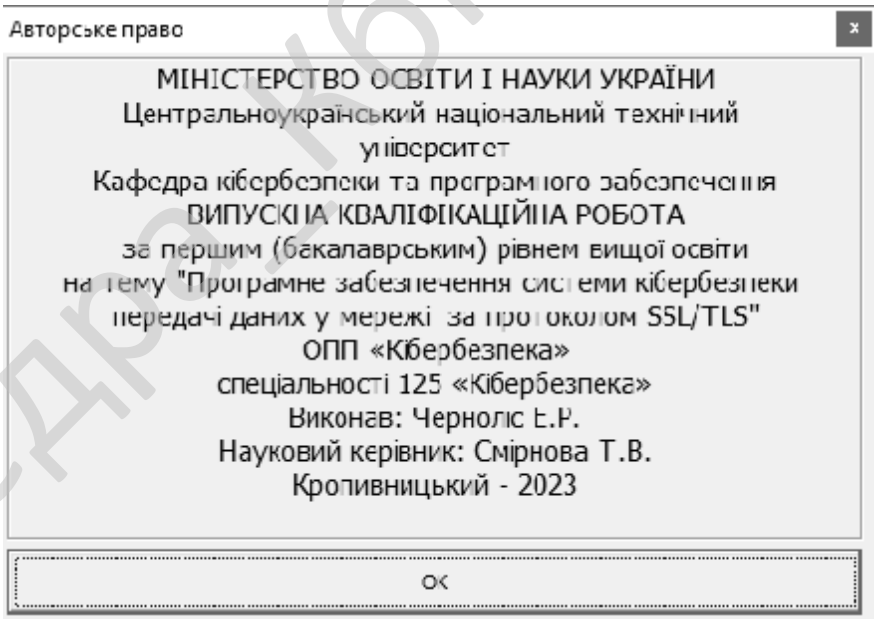

Рисунок 5.5 – Вікно «Про програму…»

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки передачі даних у мережі за протоколом SSL/TLS.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем передачі даних у мережі за протоколом SSL/TLS.

– Досліджена система передачі даних у мережі за протоколом SSL/TLS.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки передачі даних у мережі за протоколом SSL/TLS.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання передачі даних у мережі за протоколом SSL/TLS.

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 Builder C++. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи кібербезпеки передачі даних у мережі за протоколом SSL/TLS. 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 Sinople.

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. 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, pp. 1-12. **(Scopus).**

2. Smirnov O., Smirnova T., Anas M. Al-Oraiqat, Drieiev O., Polishchuk L., Sheroz Khan, Yassin M. Y. Hasan, Aladdein M. Amro, Hazim S. AlRawashdeh «Method for Determining Treated Metal Surface Quality Using Computer Vision Technology». *Sensors (Basel, Switzerland)* Volume 22, Issue 16, 6223, 2022. (**Scopus**).

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. **(Scopus).**

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. **(Scopus).**

5. Smirnov O., Kuznetsov A., Kryvinska N., Kiian A., Kuznetsova K. «Full Non-Binary Constant-Weight Codes». *SN Computer Science,* Vol 2, 337, 2021. https://doi.org/10.1007/s42979-021-00739-w **(Scopus).**

6. Smirnov O., Kovalenko O., Kovalenko A., Kavun S. «Quantitative Risk Assessment Method Development in the Context of the SDLC-model». *2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T),* 2021, pp. 203-208, doi: 10.1109/PICST54195.2021.9772143 **(Scopus).**

7. Smirnov O., Neskorodieva T., Fedorov E., Rymar P. «Neural Network Modeling Method of Transformations Data of Audit Production with Returnable Waste». *CEUR Workshop Proceedings* Volume 3101, 2021, Pages 192-207. **(Scopus).**

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.* **(Scopus)**.

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. **(Scopus)**.

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

11. Smirnov O.A., 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. **(Scopus)**.

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. **(Scopus)**.

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. **(Scopus)**.

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. **(Scopus)**.

15. 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. **(Scopus)**.

16. 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. **(Scopus)**.

17. Smirnov, O., Drieieva, H., Drieiev, O., Polishchuk, Y., Brzhanov, R., Aleksander, M. «Method of fractal traffic generation by a model of generator on the graph». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 366-379. **(Scopus)**.

18. 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. **(Scopus)**.

19. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 633-645. **(Scopus)**.

20. 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., **(Scopus)**.

21. 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. (**Scopus)**.

22. 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. **(Scopus)**.

23. 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. **(Scopus)**.

24. 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. **(Scopus)**.

25. 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. **(Scopus)**.

26. Smirnov, O., Kuznetsov, A., Prokopovych-Tkachenko, D. «Hiding Data in Images Using a Pseudo-Random Sequence». *ISCI'2020: Information Security in Critical Infrastructures.* ***Collective monograph***. Edited by Ivan D. Gorbenko, Victor A. Krasnobayev and Alexandr A. Kuznetsov. ASC Academic Publishing, USA, 2020. pp. 46-59. – ISBN: 978-1-7362833-0-1 (Hardback), ISBN: 978-1-7362833-1-8 (Ebook).

27. Smirnov, O., Kuznetsov, A., Shekhanin, K., Chepurko, I. Detecting Hidden Information in FAT. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures**. Collective monograph**. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 412-429. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

28. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. **Collective monograph**. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

29. 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.

30. Смірнов О.А., Дрєєва Г.М., «Метод генерування фрактального трафіку за допомогою моделі генератора на графі» у Інформаційна безпека та інформаційні технології: монографія / за заг. ред. В. С. Пономаренка. – Х. : Вид. Рожко С.Г. 2019. С. 123-139.

31. Смирнов А.А., Коваленко А.В. Комплекс математических моделей технологии тестирования WEB-приложений. Інформаційні технології: сучасний стан та перспективи: монографія / За загальною редакцією В.С. Пономаренка. – Х.: ТОВ «ДІСА ПЛЮС», 2018. – 461 с.

32. Смирнов А.А., Коваленко А.В. Разработка метода управления рисками разработки программного обеспечения. Інформаційні технології: проблеми та перспективи: монографія / За загальною редакцією В.С. Пономаренка. – Х.: Видавець Рожко С.Г., 2017. – 447 с.

33. 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. 2022.

34. 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.

35. 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.

36. Смірнов О.А., Смірнова Т.В., Буравченко К.О., Кравченко С.С., Горбов В.О., «Хмарна система підтримки прийняття рішень технологічного

процесу відновлення поверхонь конструкцій і деталей машин». *Сучасні інформаційні системи*. 2021. Т. 5, № 4. С. 79-95

37. Смирнов А., Кузнецов А., Кузнецова Т. «Шумоподобные дискретные сигналы для асинхронных систем кодового разделения радиоканалов». *Радиотехника*, № 2(205), 175–183. 2021.

38. 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.

39. Смірнов, О.А., Полігенько О.О., Одарченко Р.С., Терещенко Л.Ю.Усік П.С., «Інформаційна технологія та програмне забезпечення для підвищення ефективності планування підсистеми базових станцій стільникового зв'язку». *Проблеми телекомунікацій*. № 1(26). С. 83-96. 2020.

40. Смирнов А.А, Кузнецов А.А., Киян А.С., Кузнецова Е.А. «Сокрытие данных на основе адресации шумоподобных сигналов». *Всеукраїнський міжвідомчий науково-технічний збірник "Радіотехніка"* – Харьків: ХНУРЕ. – 2020. – Вип. 203. – С. 38-49.

41. Смирнов А.А., Дудан А.В., Смирнова Т.В. «Формализация структуры технологического процесса электродугового напыления». *Сборник научных трудов «Актуальные вопросы машиноведения».* Объединенный институт машиностроения Национальной Академии Наук Беларуси. №9. С. 308-312, 2020.

42. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у комп'ютерних системах операторів стільникового зв'язку» *Вісник Черкаського державного технологічного університету. Технічні науки.* №4. С. 103-110. 2020.

43. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», *Кібербезпека: освіта, наука, техніка*. № 3(7). С. 43-62. 2020.

44. А.А. Смирнов, Т.В. Смирнова, А.Н. Дреев, А.В. Дудан. «Оптимизация технологического процесса восстановления и упрочнения поверхностей с заданными характеристиками в виде облачного сервиса». Вестник Полоцкого государственного университета. Серия В, Промышленность. Прикладные науки. Республика Беларусь - 2020. - № 3. - С. 50-61.

45. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». *Центральноукраїнський науковий вісник. Технічні науки.* № 2(33). с. 161-172, 2019.

46. О.А. Смірнов, Т.В. Смірнова, О.М. Дрєєв, Є.К. Солових, «Методи оптимізації технологічних процесів відновлення сталевих покриттів», *Shipbuilding & marine infrastructure / Суднобудування і морська інфраструктура* № 1 (11). с. 48-57, 2019.

47. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., «Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей». Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

48. Смірнов О.А., Смірнова Т.В., Солових Є.К., Дрєєв О.М., «Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей». Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

49. Смірнов О.А., Смірнова Т.В.,. Дрєєв О.М, «Експертна система оптимізації процесу відновлення та зміцнення поверхонь деталей типу «вал» електродуговим напиленням», Системи управління, навігації та зв'язку, № 2 (54). с. 149-154, 2019.

50. Смірнов О.А., Смірнов С.А., Поліщук Л.І., Смірнова Т.В., Конопліцька-Слободенюк О.К. Метод формування антивірусного захисту даних з використанням безпечної маршрутизації метаданих. Кібербезпека: освіта, наука, техніка. – Том 3 № 3. – Київ: КУ ім. Бориса Грінченка. – 2019. – С. 63-87.

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 передачі даних у мережі за протоколом SSL/TLS.

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. № 14-02 від 5.01.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи кібербезпеки передачі даних у мережі за протоколом SSL/TLS.

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 передачі даних у мережі за протоколом SSL/TLS;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Builder C++.

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 – 89 аркушів.

8 Етапи розробки

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

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

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

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

11.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 6.06.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Смірнова Т.В.

Програмне забезпечення системи кібербезпеки передачі даних у мережі за протоколом SSL/TLS

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 35

Літера: РП

Кропивницький – 2023 року

Основна программа

Файл main_form_SSL.cpp основної програми

```
#include <vcl.h>
#pragma hdrstop
// ---------------------------------------------------------------------------
#include "main_form_SSL.h"
#include "about_form_SSL.h"
#include "password_form_SSL.h"
// ---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
// ---------------------------------------------------------------------------
TMainForm *MainForm;
// ---------------------------------------------------------------------------
__fastcall TMainForm::TMainForm(TComponent* Owner)
        : TForm(Owner)
{
        bIsServer = false;
        bProtection = false;
        bUsingKeybdPassword = true;
        lPasswordSize = 0;
        prgbPasswordBuffer = NULL;
        prgbSendBuffer = NULL;
        prgbPasswordBuffer2 = NULL;
        pInKeyIPsec = NULL;
}
// ---------------------------------------------------------------------------
// Змінює стан Listen...
void __fastcall TMainForm::ChangeListenCondition(TObject *Sender)
{
        if (FileListenItem->Checked)
        {
               //... міняємо іконку на протилежну
               ListenSpeedButton->Glyph = ListenOffImage->Glyph;
        } else
        {
             ListenSpeedButton->Glyph = ListenOnImage->Glyph;
        }
        FileListenItem->Checked = !FileListenItem->Checked;

        if (FileListenItem->Checked)
        {
                ClientSocket->Close();
                ServerSocket->Open();
                StatusBar->Panels->Items[0]->Text = " Mode: Server";
                StatusBar->Panels->Items[2]->Text = " Connected to:";
        }else
        {
                if (ServerSocket->Active)
                {
                        ServerSocket->Close();
                }
                StatusBar->Panels->Items[0]->Text = " Mode: Idle";
                StatusBar->Panels->Items[2]->Text = " Connected to:";
        }
}
// ---------------------------------------------------------------------------
void __fastcall TMainForm::FileListenItemClick(TObject *Sender)
{
        // Змінюємо стан Listen...
        ChangeListenCondition(Sender);
}
// ---------------------------------------------------------------------------
```

```
void __fastcall TMainForm::ListenSpeedButtonClick(TObject *Sender)
{
        // Змінюємо стан Listen...
        ChangeListenCondition(Sender);
}
// ---------------------------------------------------------------------------
void __fastcall TMainForm::FileExitItemClick(TObject *Sender)
{
        // Закриваємо програму
        Close();
}
// ---------------------------------------------------------------------------
void __fastcall TMainForm::HelpAboutItemClick(TObject *Sender)
{
        // Відображаємо діалог About
        AboutForm->ShowModal();
}
// ---------------------------------------------------------------------------
void __fastcall TMainForm::ConnectTo(TObject *Sender)
{
  // Робимо запит IP-Адреси...
  if (InputQuery("Connect to...", "Address Name:", Server))
  {
     // Якщо рядок був не порожня...
     if (Server.Length() > 0)
     {
       // Якщо в цей момент були на коннекті, рвемо з'єднання...
       if (ClientSocket->Active)
       {
                ClientSocket->Close();
       }
        // Ініціалізуємо параметри сокету...
        ClientSocket->Host = Server;
        ClientSocket->Open();

        FileListenItem->Checked = false;
        ListenSpeedButton->Glyph=MainForm->ListenOffImage->Glyph;
        StatusBar->Panels->Items[0]->Text = " Mode: Client";
     }
  }
}
// ---------------------------------------------------------------------------
void __fastcall TMainForm::FileConnectItemClick(TObject *Sender)
{
        ConnectTo(Sender);
}
// ---------------------------------------------------------------------------
void __fastcall TMainForm::ConnectSpeedButtonClick(TObject *Sender)
{
        ConnectTo(Sender);
}
// ---------------------------------------------------------------------------
void __fastcall TMainForm::FormClose(TObject *Sender, TCloseAction &Action)
{
        ServerSocket->Close();
        ClientSocket->Close();

        // Очищаємо список підключень...
        WipeVCL(Sender);

        // Вивільняємо ресурси
        FreeObjects();
}
// ---------------------------------------------------------------------------
void __fastcall TMainForm::ClientSocketConnect(TObject *Sender,
      TCustomWinSocket *Socket)
{
      StatusBar->Panels->Items[2]->Text = " Connected to: " + Socket-
>RemoteHost;
```

```
}
// ---------------------------------------------------------------------------
void __fastcall TMainForm::Disconnect(TObject *Sender)
{
        // Закриваємо всі з'єднання...
        ClientSocket->Close();
        ServerSocket->Close();

        FileListenItem->Checked = false;
        ListenSpeedButton->Glyph = ListenOffImage->Glyph;

        StatusBar->Panels->Items[0]->Text = " Mode: Idle";
        bProtection = false;
        StatusBar->Panels->Items[1]->Text = " Protection: NO";
        StatusBar->Panels->Items[2]->Text = " Connected to:";

        // Вивільняємо ресурси
        FreeObjects();
}
// ---------------------------------------------------------------------------
void __fastcall TMainForm::FileDisconnectItemClick(TObject *Sender)
{
        Disconnect(Sender);
}
// ---------------------------------------------------------------------------
void __fastcall TMainForm::DisconnectSpeedButtonClick(TObject *Sender)
{
        Disconnect(Sender);
}
// ---------------------------------------------------------------------------
void __fastcall TMainForm::ClientSocketRead(TObject *Sender,
      TCustomWinSocket *Socket)
{
        // Якщо вікно додатка згорнуте в трей, указуємо на те, що
        // прийшли нові зашифровані повідомлення...
        if (bInTray)
        {
                IconBlinkTimer->Enabled = true;
        }

        // Якщо працюємо в незахищеному режимі
        if (!bProtection)
        {
                ChatMemo->Lines->Add(TimeToStr(Time()) + " (+) " + Socket-
>ReceiveText());
        } else
        {
                // Приймаємо дані...
                Socket->ReceiveBuf(prgbSendBuffer,(EncryptionBlockSize +
INT_LENGTH));
                //... і декодуємо
                DecodeSendBuffer(Sender);
        }
}
// ---------------------------------------------------------------------------
void __fastcall TMainForm::ServerSocketClientRead(TObject *Sender,
      TCustomWinSocket *Socket)
{
        // Якщо вікно додатка згорнуте в трей, указуємо на те, що
        // прийшли нові зашифровані повідомлення...
        if (bInTray)
        {
                IconBlinkTimer->Enabled = true;
        }

        // Якщо працюємо в незахищеному режимі
        if (!bProtection)
        {
```

```
                ChatMemo->Lines->Add(TimeToStr(Time()) + " (+) " + Socket-
>ReceiveText());
        } else
        {
                Socket->ReceiveBuf(prgbSendBuffer,(EncryptionBlockSize +
INT_LENGTH));
                DecodeSendBuffer(Sender);
        }
}
// ---------------------------------------------------------------------------
void __fastcall TMainForm::ServerSocketAccept(TObject *Sender,
      TCustomWinSocket *Socket)
{
        bIsServer = true;
        StatusBar->Panels->Items[2]->Text = " Connected to: " + Socket-
>RemoteAddress;
}
// ---------------------------------------------------------------------------
void __fastcall TMainForm::ClientSocketError(TObject *Sender,
      TCustomWinSocket *Socket, TErrorEvent ErrorEvent, int &ErrorCode)
{
        ChatMemo->Lines->Add("Error connecting to: " + Server);
        StatusBar->Panels->Items[0]->Text = " Mode: Idle";
        StatusBar->Panels->Items[2]->Text = " Connected to:";
        ErrorCode = 0;
}
// ---------------------------------------------------------------------------
// Підготовляє буфер для відправлення
void __fastcall TMainForm::EncodeSendBuffer(TObject *Sender)
{
        // Крок 1 - Конвертуємо дані з Edit-А в char* і додаємо
        //         після записаного рядка випадкові дані
        ConvertAnsiStringToChar(ChatEdit->Text, prgbSendBuffer);

        long int i;

        for (i = (ChatEdit->Text.Length() + 1); i < EncryptionBlockSize; i++)
        {
                prgbSendBuffer[i] = random(256);
        }

        // Крок 2 - Вибираємо випадковий індекс у межах парольного файлу
        //         і "намотуємо", починаючи з його, 1024 бт. парольних даних,
        //         заповнюючи prgbPasswordBuffer2
        long int lRandPos = random(lPasswordSize);

        long int j = lRandPos;

        for (i = 0; i < EncryptionBlockSize; i++)
        {
                prgbPasswordBuffer2[i] = prgbPasswordBuffer[j];
                j++;
                if (j == lPasswordSize)
                {
                        j=0;
                }
        }

        // Крок 3 - Шифрування даних у вихідному буфері
        pInKeyIPsec->Encrypt(prgbSendBuffer, EncryptionBlockSize,
prgbPasswordBuffer2);

        // Крок 4 - Дописуємо в кінець буфера індекс,
        // с якого "намотували" 1 кб. парольних даних
        IntToRGB.IntVar = lRandPos;

        j = EncryptionBlockSize;

        for (i = 0; i < INT_LENGTH; i++)
```

```
        {
                prgbSendBuffer[j] = IntToRGB.rgbVar[i];
                j++;
        }
}
// ---------------------------------------------------------------------------
// Декодує буфер після приймання
void __fastcall TMainForm::DecodeSendBuffer(TObject *Sender)
{
        long int i, j = EncryptionBlockSize;

        // Крок 1 - Довідаємося випадкову позицію в парольному файлі
        for (i = 0; i < INT_LENGTH; i++)
        {
                IntToRGB.rgbVar[i] = prgbSendBuffer[j];
                j++;
        }

        unsigned long int ulRandPos = IntToRGB.IntVar;

        // Якщо парольні файли не відповідають один одному по розміру приховуємо
цей факт
        if (ulRandPos >= lPasswordSize)
        {
                ulRandPos = random(lPasswordSize);
        }

        // Крок 2 - "намотуємо" дані з парольного буфера

        j = ulRandPos;

        for (i = 0; i < EncryptionBlockSize; i++)
        {
                prgbPasswordBuffer2[i] = prgbPasswordBuffer[j];
                j++;
                if (j == lPasswordSize)
                {
                        j=0;
                }
        }

        // Крок 3 - декодуємо дані
        pInKeyIPsec->Decrypt(prgbSendBuffer, EncryptionBlockSize,
prgbPasswordBuffer2);

        // Крок 4 - Після декодування переносимо дані з буфера в рядок
        //          для висновку в ChatMemo
        i = 0;

        // Установлюємо розмір рядка із запасом
        asDecodedStr.SetLength(EncryptionBlockSize);

        while (1)
        {
                asDecodedStr[i + 1]=prgbSendBuffer[i];
                if (
                                (prgbSendBuffer[i] == '\0')
                        ||
                                (i == (EncryptionBlockSize - 1))
                    )
                {
                        break;
                }
                i++;
        }

        asDecodedStr.SetLength((i + 1));

        // Додаємо рядок в Memo
```

```
        ChatMemo->Lines->Add(TimeToStr(Time()) + " (+) " + asDecodedStr);
}
// ---------------------------------------------------------------------------
void __fastcall TMainForm::ChatEditKeyDown(TObject *Sender, WORD &Key,
      TShiftState Shift)
{
        if (Key == VK_RETURN)
        {
                if (bIsServer)
                {
                        try
                        {
                                ServerSocket->Socket->Connections[0]->Connected;
                        }
                        catch (...)
                        {
                               ServerSocket->Close();
                               ChatMemo->Lines->Add("Error: Client is not
accessible!");
                               // Перший виклик скидає стан на режим Idle
                               ChangeListenCondition(Sender);
                               // Другий виклик повертає режим Server
                               ChangeListenCondition(Sender);
                               return;
                        }
                        // Якщо працюємо в незахищеному режимі...
                        if (!bProtection)
                        {
                                ServerSocket->Socket->Connections[0]->SendText(
                                ChatEdit->Text);
                        } else
                        {
                                // Підготовляємо буфер для відправлення
                                EncodeSendBuffer(Sender);
                                // Тепер відправляємо підготовлений
prgbSendBuffer
                                ServerSocket->Socket->Connections[0]->SendBuf(
                                prgbSendBuffer,(INT_LENGTH +
EncryptionBlockSize));
                        }
                }
                else
                {
                if (ClientSocket->Socket->Connected)
                {
                     // Якщо працюємо в незахищеному режимі...
                     if (!bProtection)
                     {
                                ClientSocket->Socket->SendText(ChatEdit-
>
                                Text);
                     } else
                     {
                        // Підготовляємо буфер для відправлення
                        EncodeSendBuffer(Sender);
                        // Тепер відправляємо підготовлений
prgbSendBuffer
                        ClientSocket->Socket->SendBuf(prgbSendBuffer,
                        (INT_LENGTH+EncryptionBlockSize));

                     }
                } else
                {
                        ClientSocket->Close();
                        ChatMemo->Lines->Add("Error: Server is not
accessible!");
                        StatusBar->Panels->Items[0]->Text = " Mode:
Idle";
```

```
                                StatusBar->Panels->Items[2]->Text = " Connected
to:";
                                return;
                        }
                }
                ChatMemo->Lines->Add(TimeToStr(Time()) + " (<) " + ChatEdit-
>Text);

                int i;

                // Спочатку повністю затираємо рядок в ChatEdit...
                for (i=1; i <= ChatEdit->Text.Length(); i++)
                {
                        ChatEdit->Text[i] = 0x00;
                }

                // ... а потім забираємо весь текст у ньому
                ChatEdit->Text = "";
        }
}
// ---------------------------------------------------------------------------
// Видає повідомлення про помилку відкриття файлу
void __fastcall TMainForm::FileErrorMessage(char *szFilename,int ErrKind)
{
        std::stringstream msg;
        msg << "Can't open " << szFilename << (ErrKind == TO_READ ? " to read!"
:
         " to write!");
        MessageDlg(msg.str().c_str(), mtError, TMsgDlgButtons() << mbOK, 0);
}
// ---------------------------------------------------------------------------
void __fastcall TMainForm::ConvertAnsiStringToChar(AnsiString asStr, char *pCh)
{
        int i = 1;
        while (i <= asStr.Length())
        {
                pCh[ i-1] = asStr[i];
                i++;
        }
        pCh[i - 1] = '\0';
}
// ---------------------------------------------------------------------------
// Повертає довжину файлу
long __fastcall TMainForm::FileLength(char *szFileName)
{
            int fHandle;
            long int lFileSize;

            fHandle = open(szFileName,O_RDONLY);
            lFileSize = filelength(fHandle);
            close(fHandle);

            return lFileSize;
}
// ---------------------------------------------------------------------------
// Забезпечує ініціалізацію:
// 1 - Файлового покажчика
// 2 - Змінної, що зберігає довжину файлу
// У цілому: бере на себе функції коректного відкриття файлу для читання
FILE* __fastcall TMainForm::OpenFileToRead(char *szFilename,long &lFileSize)
{
            FILE *fHandle;

            // Перевіряємо файл на можливість коректного відкриття...
            if((fHandle = fopen(szFilename,"rb")) == NULL)
            {
                 // Якщо відкрити не можна - виводимо повідомлення про
помилку...
                 FileErrorMessage(szFilename,TO_READ);
```

```
                // ... і виходимо з функції
                return NULL;
            }

            // ... якщо файл відкрився - все ОК, але ще необхідно зробити
            // визначення його розміру, для цього потрібно перевідкрити його
            // в іншому режимі, тому його тимчасово закриваємо...
            fclose(fHandle);

            // ... і одержуємо розмір файлу.
            lFileSize = FileLength(szFilename);

            // А от і відкриття файлу для роботи з ним...
            fHandle = fopen(szFilename, "rb");

            return fHandle;
}
// ---------------------------------------------------------------------------
void __fastcall TMainForm::SetPasswordFileSpeedButtonClick(TObject *Sender)
{
        // Якщо запропоновано одержати пароль із клавіатури...
        if (bUsingKeybdPassword)
        {
                // 1) Відображаємо форму уведення пароля...
                if(
                        (PasswordDlg->ShowModal() == mrOk)
                    &&
                        (PasswordDlg->Password->Text.Length() != 0)
                   )
                {
                        // Установлюємо розмір парольного буфера
                        lPasswordSize = MAX_ENCRYPTION_BLOCK_SIZE;

                        // Створюємо необхідні об'єкти
                        AllocateObjects();

                        //...довідаємося його довжину...
                        int PasswordLen = PasswordDlg->Password->Text.Length();

                        //...копіюємо в парольний буфер...
                        for (int i = 1; i <= PasswordLen; i++)
                        {
                                prgbPasswordBuffer[i - 1] = PasswordDlg-
>Password->Text[i];
                                PasswordDlg->Password->Text[i] = 0x00;
                        }

                        //...і хешуємо пароль
                        pDHash->Hash(prgbPasswordBuffer, PasswordLen);

                        PasswordDlg->Password->Text = "";

                        // Указуємо, що захист використовується
                        bProtection = true;
                        StatusBar->Panels->Items[1]->Text = " Protection: YES";
                }

                return;
        }

        if (MainForm->OpenDialog->Execute())
        {
                // Спочатку вивільняємо ресурси, виділені під старий
                // парольний файл
                FreeObjects();
                bProtection=false;
                StatusBar->Panels->Items[1]->Text = " Protection: NO";

                // Задаємо ім'я максимальної довжини
```

```
                char *szPasswordFile = new char [MAX_NAME_LENGTH];
                ConvertAnsiStringToChar(OpenDialog->FileName,szPasswordFile);

                // Відкриваємо парольний файл
                FILE* fPassword = OpenFileToRead(szPasswordFile,lPasswordSize);

                // Якщо парольний файл не відкрився, виходимо з функції
                if (!fPassword)
                {
                        delete [] szPasswordFile;
                        return;
                }

                // а інакше виділяємо пам'ять під дані парольного файлу
                // і читаємо його в буфер
                AllocateObjects();

                fread(prgbPasswordBuffer,lPasswordSize,1,fPassword);
                fclose(fPassword);

                delete [] szPasswordFile;

                // Указуємо, що захист використовується
                bProtection = true;
                StatusBar->Panels->Items[1]->Text = " Protection: YES";
        }
}
 // ---------------------------------------------------------------------------
-
void __fastcall TMainForm::FileSetPasswordFileItemClick(TObject *Sender)
{
SetPasswordFileSpeedButtonClick(Sender);
}

// ---------------------------------------------------------------------------
// Ліквідує дані переданого масиву char з міркувань
// безпеки
void __fastcall TMainForm::WipeData (char *prgbData, long lDataSize)
{
            long int i;

            for (i = 0; i < lDataSize; i++)
            {
                 prgbData[i] = (char)0x00;
            }
}
// ---------------------------------------------------------------------------
void __fastcall TMainForm::AllocateObjects()
{
        prgbPasswordBuffer = new char[lPasswordSize];
        // Якщо не вдалося виділити пам'ять
        if (!prgbPasswordBuffer)
        {
                MessageDlg("Нмає записів в базі даних паролів" ,
                mtError, TMsgDlgButtons() << mbOK, 0);
                return;
        }

        prgbSendBuffer = new char[(MAX_ENCRYPTION_BLOCK_SIZE + INT_LENGTH)];
        if (!prgbSendBuffer)
        {
                MessageDlg("Не може розподілитися пам’ять під prgbSendBuffer!" ,
                mtError, TMsgDlgButtons() << mbOK, 0);
                return;
        }

        prgbPasswordBuffer2 = new char[MAX_ENCRYPTION_BLOCK_SIZE];
        if (!prgbPasswordBuffer2)
        {
```

```
                MessageDlg("Не може розподілитися пам’ять під
prgbPasswordBuffer2!" ,
                mtError, TMsgDlgButtons() << mbOK, 0);
                return;
        }

        // Створюємо екземпляр класу шифрування SSL
        pDHash = new CDHash(MAX_ENCRYPTION_BLOCK_SIZE);
        if (!pDHash)
        {
                MessageDlg("Не може розподілитися пам’ять під pDHash!" ,
                mtError, TMsgDlgButtons() << mbOK, 0);
                return;
        }

        // Створюємо екземпляр класу шифрування
        pInKeyIPsec = new CInKeyIPsec;
        if (!pInKeyIPsec)
        {
                MessageDlg("Не може розподілитися пам’ять під pInKeyIPsec!" ,
                mtError, TMsgDlgButtons() << mbOK, 0);
                return;
        }

        // Необхідно для шифрування
        randomize();
}
// ---------------------------------------------------------------------------
void __fastcall TMainForm::FreeObjects()
{
        // Вивільняємо ресурси...
        if (prgbPasswordBuffer)
        {
                WipeData(prgbPasswordBuffer,lPasswordSize);
                delete [] prgbPasswordBuffer;
                prgbPasswordBuffer = NULL;
        }

        if (prgbSendBuffer)
        {
                WipeData(prgbSendBuffer,(EncryptionBlockSize+INT_LENGTH));
                delete [] prgbSendBuffer;
                prgbSendBuffer = NULL;
        }

        if (prgbPasswordBuffer2)
        {
                WipeData(prgbPasswordBuffer2,EncryptionBlockSize); // char* data
                delete [] prgbPasswordBuffer2;
                prgbPasswordBuffer2 = NULL;
        }

        if (pDHash)
        {
                delete pDHash;
                pDHash = NULL;
        }

        if (pInKeyIPsec)
        {
                delete pInKeyIPsec;
                pInKeyIPsec = NULL;
        }
 }
// ---------------------------------------------------------------------------
void __fastcall TMainForm::FormCreate(TObject *Sender)
{
        // Установлюємо початковий стан блоку шифрування
        EncryptionBlockSize = 1024;
```

```
        ChatEdit->MaxLength = EncryptionBlockSize;

        // Завантажуємо іконки в ImageList
        IconsImageList->Add(NormalIconImage->Picture->Bitmap, NULL);
        IconsImageList->Add(BlinkIconImage->Picture->Bitmap, NULL);
}
// ---------------------------------------------------------------------------
void __fastcall TMainForm::IconBlinkTimerTimer(TObject *Sender)
{
        // Перекидаємо індекс іконки в масиві іконок
        TrayIcon->IconIndex ^= 0x01;
}
// ---------------------------------------------------------------------------
void __fastcall TMainForm::TrayIconClick(TObject *Sender)
{
        // Якщо звернулися до додатка...
        bInTray = false;

        // ...відключаємо таймер...
        IconBlinkTimer->Enabled = false;
        // ... і повертаємо нормальну іконку
        TrayIcon->IconIndex = 0x00;
}
// ---------------------------------------------------------------------------
void __fastcall TMainForm::TrayIconMinimize(TObject *Sender)
{
        // Указуємо, що додаток у треї...
        bInTray = true;
}
// ---------------------------------------------------------------------------
void __fastcall TMainForm::WipeVCL(TObject *Sender)
{
        int i,j;

        // Спочатку повністю затираємо рядок в ChatEdit...
        for (i=1; i <= ChatEdit->Text.Length(); i++)
        {
                ChatEdit->Text[i] = 0x00;
        }

        // Потім займаємося ChatMemo...
        for (j=0; j < ChatMemo->Lines->Count; j++)
        {
                // Затираємо всі символи в кожному рядку...
                for (i=1; i <= (ChatMemo->Lines->operator [](j).Length()); i++)
                {
                        ChatMemo->Lines->operator [](j)[i] = 0x00;
                }
        }

}
// ---------------------------------------------------------------------------
void __fastcall TMainForm::EditClearClick(TObject *Sender)
{
        // Очищаємо список підключень...
        WipeVCL(Sender);
        ChatMemo->Clear();
        ChatEdit->Text = "";
}
// ---------------------------------------------------------------------------
void __fastcall TMainForm::EditTCPPortClick(TObject *Sender)
{
        // Викликаємо діалог установки порту...
          if (InputQuery("Set to...", "TCP Port:", Port))
          {
             unsigned int TCPPort = StrToInt(Port);
             if (
                                (Port.Length() > 0)
                        &&
```

```
                                ((TCPPort > 0)
                        &&
                                (TCPPort <= 65535))
                )
                {
                        ClientSocket->Port = TCPPort;
                        ServerSocket->Port = TCPPort;
                        StatusBar->Panels->Items[3]->Text = " Port: " + Port;
                }
          }
}
// ---------------------------------------------------------------------------
void __fastcall TMainForm::N8192bitClick(TObject *Sender)
{
        // Якщо пункт меню ще не виділений, виділяємо
        if (!((TMenuItem*)Sender)->Checked)
        {
                EncryptionBlockSize = 1024;
                ChatEdit->MaxLength = EncryptionBlockSize;

                ((TMenuItem*)Sender)->Checked = true;
        }
}
// ---------------------------------------------------------------------------
void __fastcall TMainForm::N16384bitClick(TObject *Sender)
{
        // Якщо пункт меню ще не виділений, виділяємо
        if (!((TMenuItem*)Sender)->Checked)
        {
                EncryptionBlockSize = 2048;
                ChatEdit->MaxLength = EncryptionBlockSize;

                ((TMenuItem*)Sender)->Checked = true;
        }
}
// ---------------------------------------------------------------------------
void __fastcall TMainForm::N24576bitClick(TObject *Sender)
{
         // Якщо пункт меню ще не виділений, виділяємо
        if (!((TMenuItem*)Sender)->Checked)
        {
                EncryptionBlockSize = 3072;
                ChatEdit->MaxLength = EncryptionBlockSize;

                ((TMenuItem*)Sender)->Checked = true;
        }
}
// ---------------------------------------------------------------------------
void __fastcall TMainForm::N32768bitClick(TObject *Sender)
{
        // Якщо пункт меню ще не виділений, виділяємо
        if (!((TMenuItem*)Sender)->Checked)
        {
                EncryptionBlockSize = 4096;
                ChatEdit->MaxLength = EncryptionBlockSize;

                ((TMenuItem*)Sender)->Checked = true;
        }
}
// ---------------------------------------------------------------------------
void __fastcall TMainForm::PasswordModeClick(TObject *Sender)
{
         // Якщо пункт меню ще не виділений, виділяємо
        if (!((TMenuItem*)Sender)->Checked)
        {
                ((TMenuItem*)Sender)->Checked = true;
                 bUsingKeybdPassword = true;
        } else
        {
```

```
                ((TMenuItem*)Sender)->Checked = false;
                 bUsingKeybdPassword = false;
        }
}
// ---------------------------------------------------------------------------
void __fastcall TMainForm::ProtectionResetClick(TObject *Sender)
{
        // Указуємо, що захист відключений...
        bProtection = false;
        StatusBar->Panels->Items[1]->Text = " Protection: NO";

        // Вивільняємо ресурси...
        FreeObjects();
}
// ---------------------------------------------------------------------------
void __fastcall TMainForm::ClearClick(TObject *Sender)
{
        // Очищаємо список підключень...
        WipeVCL(Sender);
        ChatMemo->Clear();
        ChatEdit->Text = "";

        // ...відключаємо таймер...
        IconBlinkTimer->Enabled = false;
        // ... і повертаємо нормальну іконку
        TrayIcon->IconIndex = 0x00;
}
// ---------------------------------------------------------------------------
void __fastcall TMainForm::ExitClick(TObject *Sender)
{
        Close();
}
```

Файл main_form_SSL.h - бібліотека для файлу main_form_SSL.cpp

```
      #ifndef about_form
      #define about_form
      // ---------------------------------------------------------------------------
------
      #include <Classes.hpp>
      #include <Controls.hpp>
      #include <StdCtrls.hpp>
      #include <Forms.hpp>
      #include <ExtCtrls.hpp>
      #include <Graphics.hpp>
      // ---------------------------------------------------------------------------
------
      class TAboutForm : public TForm
      {
      __published:      // IDE-менеджер компонентів
              TButton *OKButton;
              TImage *AboutImage;
              TBevel *AboutBevel;
              TStaticText *VersionStaticText;
              TStaticText *CopyrightStaticText;
              TStaticText *ProtectionVersionStaticText;
              TStaticText *ProtectionVersionStaticText2;
              void __fastcall OKButtonClick(TObject *Sender);
      private:    // задає користувач
      public:           // Задає користувач
              __fastcall TAboutForm(TComponent* Owner);
      };
      // ---------------------------------------------------------------------------
------
      extern PACKAGE TAboutForm *AboutForm;
      // ---------------------------------------------------------------------------
------
      #endif
```

Файл Project1.cpp основної програми

```
// ---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
// ---------------------------------------------------------------------------
USEFORM("main_form_SSL.cpp", MainForm);
USEFORM("about_form_SSL.cpp", AboutForm);
USEFORM("password_form_SSL.cpp", PasswordDlg);
// ---------------------------------------------------------------------------
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
        try
        {
                 Application->Initialize();
                 Application->CreateForm(__classid(TMainForm), &MainForm);
                 Application->CreateForm(__classid(TAboutForm), &AboutForm);
                 Application->CreateForm(__classid(TPasswordDlg), &PasswordDlg);
                 Application->Run();
        }
        catch (Exception &exception)
        {
                 Application->ShowException(&exception);
        }
        catch (...)
        {
                 try
                 {
                         throw Exception("");
                 }
                 catch (Exception &exception)
                 {
                         Application->ShowException(&exception);
                 }
        }
        return 0;
}
// ---------------------------------------------------------------------------
```

Файл password_form_SSL.cpp - вікно введення паролів

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "password_form_SSL.h"
//---------------------------------------------------------------------------
#pragma resource "*.dfm"
TPasswordDlg *PasswordDlg;
//---------------------------------------------------------------------------
__fastcall TPasswordDlg::TPasswordDlg(TComponent* AOwner)
        : TForm(AOwner)
{
}
//---------------------------------------------------------------------------
```

Файл password_form_SSL.h - бібліотека для файлу password_form_SSL.cpp

```
// ---------------------------------------------------------------------------
i
#ifndef password_form
#define password_form
// ---------------------------------------------------------------------------
i
#include <vcl\Buttons.hpp>
#include <vcl\StdCtrls.hpp>
#include <vcl\Controls.hpp>
#include <vcl\Forms.hpp>
#include <vcl\Graphics.hpp>
#include <vcl\Classes.hpp>
#include <vcl\SysUtils.hpp>
#include <vcl\Windows.hpp>
#include <vcl\System.hpp>
// ---------------------------------------------------------------------------
i
class TPasswordDlg : public TForm
{
__published:
        TLabel *PasswordLabel;
      TEdit *Password;
      TButton *OKBtn;
      TButton *CancelBtn;
private:
public:
      virtual __fastcall TPasswordDlg(TComponent* AOwner);
};
// ---------------------------------------------------------------------------
i
extern PACKAGE TPasswordDlg *PasswordDlg;
// ---------------------------------------------------------------------------
i
#endif
```

Файл inKeyIPsec.cpp - шифрування IPsec

```
#pragma once

#include "inKeyIPsec.h"

CInKeyIPsec::CInKeyIPsec()
{
    this->UpdateProgress = NULL;

    Initialize();
}

CInKeyIPsec::CInKeyIPsec(void (*UpdateProgress)(int progress))
{
    // Підписуємося на функтор-оброблювач відновлення прогресу
    this->UpdateProgress = UpdateProgress;

    Initialize();
}

void CInKeyIPsec::Initialize()
{
      // Ініціалізуємо масив "бітами", які будуть брати участь
    // в операціях виділення відповідних бітів
    rgbBitMask[0] = (char)0x01;
    rgbBitMask[1] = (char)0x02;
    rgbBitMask[2] = (char)0x04;
    rgbBitMask[3] = (char)0x08;
    rgbBitMask[4] = (char)0x10;
    rgbBitMask[5] = (char)0x20;
    rgbBitMask[6] = (char)0x40;
    rgbBitMask[7] = (char)0x80;

    // Ініціалізуємо множину векторів MVS - 56 штук
    prgubMVS = new unsigned char[MVS_NUM * NBITS];
}

CInKeyIPsec::~CInKeyIPsec()
{
    // Деініціалізуємо множину векторів MVS
      if (prgubMVS)
      {
            delete [] prgubMVS;
      }
}

////////////////////////////////////////////////////////////////////////////
// Генерується множина векторів, призначених для зміни
// порядку проходження "стовпців". // РЕКУРСИВНА ЧАСТИНА АЛГОРИТМУ
void CInKeyIPsec::GenerateMVS(char rgbBank[NBITS],int bankLen,
                              char rgbMVVector[NBITS])
{
      int i, j;

      char rgbSavedBank[NBITS];

      // Якщо ми використовували всі символи для генерування векторів,
      // те продовжувати генерування не представляється можливим
      if (bankLen == 0)
      {
            // ТУТ ---> ЗАПИС ГОТОВОГО ВЕКТОРА В БЛОК ВЕКТОРІВ
            for (j = 0; j < NBITS; j++)
            {
                  prgubMVS[cMVS * NBITS + j] = rgbMVVector[j];
            }
            cMVS++;
```

```
                return;
        }

        // Цей цикл породжує множину рекурсивних викликів,
        // додаючи один символ у вектор, і передаючи його долілиць по
        // рекурсії. Так, наприклад, перша гілка рекурсії
        // створить вектор "01234567".

        // Зберігаємо вихідний стан переданого банку
        for (j = 0; j < bankLen; j++)
        {
                rgbSavedBank[j] = rgbBank[j];
        }

        for (i = 0; i < bankLen; i++)
        {
           // Кожний з векторів, одержуваних у різних рекурсивних
           // піддеревах буде відрізнятися від іншого на 1 символ на цієї
           // стадії
           rgbMVVector[NBITS - bankLen] = rgbBank[i];

           // Тепер, коли ОДИН ІЗ СИМВОЛІВ ВИЛУЧЕНИЙ,
           // банк повинен зменшитися на ЦЕЙ СИМВОЛ і
           // потім надійти в скороченому виді в
           // подальшу рекурсію.
           // Переносимо символи вліво, затираючи "дірку"
           // символами, що залишилися в банку
           for (j = i; j < (bankLen - 1); j++)
           {
                     rgbBank[j] = rgbBank[j + 1];
           }

           // РЕКУРСІЯ
           GenerateMVS(rgbBank, (bankLen - 1), rgbMVVector);

           // Тепер відновлюємо той стан банку, у якому
           // він був переданий у рекурсивну функцію
           for (j = 0; j < bankLen; j++)
           {
                     rgbBank[j] = rgbSavedBank[j];
           }
        }
}

////////////////////////////////////////////////////////////////////////////
// Генерується множина векторів, призначених для зміни
// порядку проходження "стовпців".
// BankLen - кількість символів у масиві Bank для генерування
// кожного вектора в масиві. Щораз, коли черговий символ
// породжує піддерево комбінацій, він зникає. Таким чином,
// генеріруемое піддерево вже не буде містити свій
// корінь у жодному з піддерев.
void CInKeyIPsec::StartToGenerateMVS()
{
        // ! ПОТЕНЦІЙНА КРАПКА МОДИФІКУВАННЯ АЛГОРИТМУ
        char rgbBank[NBITS] = {(char)0x00,(char)0x01,(char)0x02,(char)0x03,

(char)0x04,(char)0x05,(char)0x06,(char)0x07};
        // ! ПОТЕНЦІЙНА КРАПКА МОДИФІКУВАННЯ АЛГОРИТМУ

        char rgbMVVector[NBITS];

        // Довжина початкового банку символів
        int bankLen = NBITS;

        // Лічильник кількості записаних векторів для переміщення
        // "стовпців"
        cMVS = 0;
```

```
	// Рекурсивна функція генерування множини векторів
	// для зміни порядку проходження стовпців -
	// заповнює кожний вектор значеннями 0..7
	GenerateMVS(rgbBank, bankLen, rgbMVVector);
}

////////////////////////////////////////////////////////////////////////////
// Ліквідує дані переданого масиву char з міркувань
// безпеки
void CInKeyIPsec::WipeData(char *prgbData, long int lDataSize)
{
	long int i;

	for (i = 0; i < lDataSize; i++)
	{
		prgbData[i] = (char)0x00;
	}
}

////////////////////////////////////////////////////////////////////////////
// Повертає стан необхідного біта з переданого байта
inline char CInKeyIPsec::Getbit(char byte, int nbit)
{
	byte &= rgbBitMask[nbit];

	if (byte != 0)
	{
		return 1;
	} else
	{
		return 0;
	}
}

////////////////////////////////////////////////////////////////////////////
// Установлює стан необхідного біта в переданий байт,
// яке закодовано байтом. Повертає модифікований байт
// !!! Увага !!! Перед накладенням він повинен бути встановлений в 0!
inline char CInKeyIPsec::Putbit(char byte, int nbit, char bit)
{
	if (bit != 0)
	{
		byte |= rgbBitMask[nbit];
	}

	return byte;
}

////////////////////////////////////////////////////////////////////////////
// Перекодування вихідних даних в "рядок"
void CInKeyIPsec::ConvertSourceToBitData(char *prgbSourceBuffer,
                                                    long int lSourceSize, char
*prgbBitData)
{
   long int i, j;

   int nbit;

   // Обробляємо кожний із символів вихідної гами...
   for (i = 0, j = 0; i < lSourceSize; i++)
   {
           // У байті 8 біт, витягаємо кожного...
      for (nbit = 0; nbit < NBITS; nbit++,j++)
      {
             prgbBitData[j] = Getbit(prgbSourceBuffer[i], nbit);
      }
   }
}
```

```
////////////////////////////////////////////////////////////////////////////
// Перекодування дані "рядка" у блок вихідних даних
void CInKeyIPsec::ConvertBitDataToSource(char *prgbSourceBuffer,
                                                    long int lSourceSize, char
*prgbBitData)
{
   long int i, j = 0;

   int nbit;

   // Ураховуємо довісок
   if (bitDataPosition != 0)
   {
            j = NBITS * lSourceSize;
   }

   // Всі вихідні байти повинні бути повернуті на місця...
   for (i = 0; i < lSourceSize; i++)
   {
          // У байті 8 біт, повертаємо на місце кожний...
        for (nbit = 0; nbit < NBITS; nbit++,j++)
        {
              prgbSourceBuffer[i] = Putbit(prgbSourceBuffer[i], nbit,
prgbBitData[j]);
        }
   }
}
////////////////////////////////////////////////////////////////////////
// Обертає prgBitData (ліквідація циклів перемішування IPSec)
void CInKeyIPsec::RotateBitData (long int lSourceSize,char *prgbPasswordBuffer,
                                          char *prgbBitData, long int L, int
mode)
{
      long int i, j;
      int sPartIndex;

      // Індекс, з якого починається "друга частина" даних
      sPartIndex = (unsigned char)prgbPasswordBuffer[L] + 1;

      // Напрямок обертання прямо залежить від режиму роботи
      // програми - шифрування або розшифровка IPSec
      switch (mode)
      {
            case ENCRYPT:
            {
                  // Переносимо "другу частину" даних на перше місце
                  for (i = sPartIndex, j = 0; i < (NBITS * lSourceSize);
i++, j++)
                  {
                        prgbBitData[(bitDataPosition ^ 0x01) * NBITS *
lSourceSize + j] = prgbBitData[bitDataPosition * NBITS * lSourceSize + i];
                  }

                  // Переносимо "першу частину" даних IPSec на друге місце

                  for (i = (sPartIndex - 1); j < (NBITS * lSourceSize); i-
i--, j++)
                  {
                        prgbBitData[(bitDataPosition ^ 0x01) * NBITS *
lSourceSize + j] = prgbBitData[bitDataPosition * NBITS * lSourceSize + i];
                  }

                  // Перекидаємо "покажчик" на дані
                  bitDataPosition ^= 0x01;

                  break;
            }
```

```
            case DECRYPT:
            {
                        // Переносимо "першу частину" даних на перше місце
                        for (i = (NBITS * lSourceSize - sPartIndex), j =
(sPartIndex - 1); j >= 0; i++, j-j--)
                        {
                              prgbBitData[(bitDataPosition ^ 0x01) * NBITS *
lSourceSize + j] = prgbBitData[bitDataPosition * NBITS * lSourceSize + i];
                        }

                        // Переносимо "другу частину" даних на друге місце

                        for (i = 0, j = sPartIndex; j < (NBITS * lSourceSize);
i++, j++)
                        {
                              prgbBitData[(bitDataPosition ^ 0x01) * NBITS *
lSourceSize + j] = prgbBitData[bitDataPosition * NBITS * lSourceSize + i];
                        }

                        // Перекидаємо "покажчик" на дані
                        bitDataPosition ^= 0x01;
            }
      }
}

////////////////////////////////////////////////////////////////////
// УСТАНОВЛЮЄ ІНДЕКС iMVS і РЕАЛІЗУЄ ЙОГО (перемішування бітів)
void CInKeyIPsec::MixBitData(long int lSourceSize, char *prgbPasswordBuffer,
                                    char *prgbBitData, long int L, long int R,
int mode)
{
      int iMVS; // Індекс у масиві покажчиків на char-вектори
                  // перестановок

      int nbit, nbit2 = 0;

      long int i, lIndexTarget, lIndexSource;

      // Цей union необхідний для формування індексу в MVS
      // Індекс складається із двох байтів парольного блоку
      // При наступній ітерації відбувається зсув "рамки зчитування"
      // двох байт по буфері парольного файлу
      static TCharToInt charToInt;

      // Ініціалізуємо "вихідний" блок даних в union-е
      // даними, узятими з парольного блоку даних
      charToInt.rgbVariable[0] = prgbPasswordBuffer[L];
      charToInt.rgbVariable[1] = prgbPasswordBuffer[R];

      // Здійснюємо перетворення Char[2] ---> Int
      iMVS = charToInt.intVariable;

      // Нормалізуємо індекс ВИБОРУ ПОТОЧНОГО ВЕКТОРА
      if (iMVS > (MVS_NUM - 1))
      {
            iMVS -= MVS_NUM;
      }

      // РЕАЛІЗУЄМО ВЕКТОР ПЕРЕСТАНОВКИ ДАНИХ:
      // Необхідно ВСІ N-Е   Б І Т Ы    вихідного блоку
      // помістити в N-ий  "С Т О В П Е Ц Ь" блоку-близнюка
      for (nbit = 0; nbit < NBITS; nbit++)
      {
            // N-Ых бітів стільки, скільки байтів у блоці вихідних даних
            for (i = 0; i < lSourceSize; i++)
            {
                  // При перенесенні даних завжди пишемо в масив-близнюк
                  // НОРМАЛЬНА АДРЕСАЦІЯ - ЧИТАННЯ ПО РЯДКАХ -
nbit*lSourceSize+i
```

```
                  // ТРАНСПОНОВАНА АДРЕСАЦІЯ - ЧИТАННЯ ПО СТОВПЦЯХ -
nbit*i+lSourceSize

                  // Режим адресації прямо залежить від режиму роботи програми -
                  // шифрування або розшифровка
                  switch (mode)
                  {
                        case ENCRYPT:
                        {
                              // Беремо по векторі - пишемо підряд
                              lIndexSource = (long int)((prgubMVS[iMVS * NBITS +
nbit]) * lSourceSize + i);
                              lIndexTarget = nbit * lSourceSize + i;

                              break;
                        }

                        case DECRYPT:
                        {
                              // Беремо підряд - пишемо по векторі
                              lIndexSource = nbit * lSourceSize + i;
                              lIndexTarget = (long int)prgubMVS[iMVS * NBITS +
nbit] * lSourceSize + i;
                        }
                  }

                  // Ураховуємо можливий зсув на другу частину масиву
                  // "бітових" даних
                  // Якщо читаємо із другої частини масиву, те
                  // bitDataPosition != 0, отже, необхідно
                  // додати "довесок" до "вихідного" індексу...
                  if (bitDataPosition != 0)
                  {
                        lIndexSource += (NBITS * lSourceSize);

                  } else
                  // ... а якщо bitDataPosition==0, то для
                  // коректного запису в більше високий шар
                  // також вносимо "довісок"
                  {
                        lIndexTarget += (NBITS * lSourceSize);
                  }

                  // Переміщаємо дані
                  prgbBitData[lIndexTarget] = (prgbBitData[lIndexSource] ^
Getbit(prgbPasswordBuffer[i], nbit2));

                  nbit2++;

                  if (nbit2 >= NBITS)
                  {
                        nbit2 = 0;
                  }
            }
      }
      // "Перекидаємо" покажчик на один із двох логічних підмасивів
      // у масиві prgbBitData
      bitDataPosition ^= (char)0x01;
}

//////////////////////////////////////////////////////////////////////
// Шифрування вихідного блоку даних парольним блоком даних
void CInKeyIPsec::Encrypt(char *prgbSourceBuffer, long int lSourceSize,
                          char *prgbPasswordBuffer)
{
      long int i;

      // Курсори в блоці парольних даних
      long int L, R;
```

```
      char *prgbBitData = new char[2 * NBITS * lSourceSize];


      int mode = ENCRYPT;

    // ГЕНЕРУЄМО MVS - Move Vector
      StartToGenerateMVS();

    // Споконвічно дані лежать на початку масиву (після перекодування)
    bitDataPosition = 0;

      // Курсор R у парольному буфері встає на своє місце...
      R = (lSourceSize - 1);
      // а L - на своє.
      L = 0;

      // Перекодування вихідних даних в "бітове" відображення
      ConvertSourceToBitData(prgbSourceBuffer, lSourceSize, prgbBitData);

      // Очищаємо буфер вихідних даних, щоб забезпечити коректне
      // накладення бітових даних з prgbBitData
      WipeData(prgbSourceBuffer, lSourceSize);

      ////////////////////////////////////////
      // Обробка даних: ШИФРУВАННЯ IPSec
      ////////////////////////////////////////
      // Працюємо доти, поки всі парольні дані IPSec не використані
      for (i = 0; i < lSourceSize; i++)
      {
            // Установка вектора перестановок (циклічний виклик приводить
            // до зміни порядку проходження "бітів")
            MixBitData(lSourceSize, prgbPasswordBuffer,
                              prgbBitData, L, R, mode);

            // Обертаємо бітові дані першим алгоритмом
            RotateBitData(lSourceSize, prgbPasswordBuffer,
                              prgbBitData, L, mode);

            L++;

            R-R--;

            // Обновляємо прогрес
            if (UpdateProgress != NULL)
            {
                  UpdateProgress(i);
            }
      }

      // Перекодування даних з "бітової" форми у вихідну
      ConvertBitDataToSource(prgbSourceBuffer, lSourceSize, prgbBitData);

      // Знищуємо дані...
      WipeData(prgbBitData, 2 * NBITS * lSourceSize);

      if (prgbBitData)
      {
            delete [] prgbBitData;
            prgbBitData = NULL;
      }
}

////////////////////////////////////////////////////////////////////
// Шифрування вихідного блоку даних IPSec парольним блоком даних
void CInKeyIPsec::Decrypt(char *prgbSourceBuffer, long int lSourceSize,
                              char *prgbPasswordBuffer)
{
      long int i;
```

```
	// Курсори в блоці парольних даних
	long int L, R;

	char *prgbBitData = new char[2 * NBITS * lSourceSize];

	int mode = DECRYPT;

// ГЕНЕРУЄМО MVS - Move Vector
	StartToGenerateMVS();

// Споконвічно дані лежать на початку масиву (після перекодування)
bitDataPosition = 0;

	// Курсор R у парольному буфері встає на своє місце...
	L = (lSourceSize - 1);
	// а L - на своє.
	R = 0;

	// Перекодування вихідних даних в "бітове" відображення
	ConvertSourceToBitData(prgbSourceBuffer, lSourceSize, prgbBitData);

	// Очищаємо буфер вихідних даних, щоб забезпечити коректне
	// накладення бітових даних з prgbBitData
	WipeData(prgbSourceBuffer, lSourceSize);

	////////////////////////////////
	// Обробка даних: РОЗШИФРОВКА IPSec
	////////////////////////////////
	// Працюємо доти, поки всі парольні дані IPSec не використані
	for (i = 0; i < lSourceSize; i++)
	{
		// Обертаємо бітові дані першим алгоритмом
		RotateBitData(lSourceSize, prgbPasswordBuffer,
				prgbBitData, L, mode);

		// Установка вектора перестановок (циклічний виклик приводить
		// до зміни порядку проходження "бітів")
		MixBitData(lSourceSize, prgbPasswordBuffer,
				prgbBitData, L, R, mode);

		L-L--;

		R++;

		// Обновляємо прогрес
		if (UpdateProgress != NULL)
		{
			UpdateProgress(i);
		}
	}

	// Перекодування даних з "бітової" форми у вихідну
	ConvertBitDataToSource(prgbSourceBuffer, lSourceSize, prgbBitData);

	// Знищуємо дані...
	WipeData(prgbBitData, 2 * NBITS * lSourceSize);

	if (prgbBitData)
	{
		delete [] prgbBitData;
		prgbBitData = NULL;
	}
}
```

Файл inKeyIPsec.h - бібліотека для файлу inKeyIPsec.cpp

```
#pragma once

#define MVS_NUM 40320  // Кількість перестановок у векторі з 8-і
                              // символів
#define NBITS 8        // MVS_NUM = NBITS! (факторіал)

#define ENCRYPT 1      // Режим: ШИФРУВАТИ
#define DECRYPT 2      // Режим: РОЗШИФРУВАТИ

//////////////////////////////////////////////////////////////////////////
// Цей union необхідний для формування індексу в MVS
// Індекс складається із двох байтів парольного блоку
// При наступній ітерації відбувається зсув "рамки зчитування"
// двох байт по буфері парольного файлу
typedef union
{
      int intVariable;
      char rgbVariable[2];
} TCharToInt;

class CInKeyIPsec
{
public:
      CInKeyIPsec();
      CInKeyIPsec(void (*UpdateProgress)(int progress));
      virtual ~CInKeyIPsec();

      void Encrypt(char *prgbSourceBuffer, long int lSourceSize,
                           char *prgbPasswordBuffer);
      void Decrypt(char *prgbSourceBuffer, long int lSourceSize,
                           char *prgbPasswordBuffer);
private:
      unsigned char *prgubMVS;
    char rgbBitMask[NBITS];
      int cMVS;
      int bitDataPosition;
      void Initialize();
      void GenerateMVS(char rgbBank[NBITS], int bankLen,
                                    char rgbMVVector[NBITS]);
      void StartToGenerateMVS();
      void WipeData(char *prgbData, long int lDataSize);
      inline char Getbit(char byte, int nbit);
      inline char Putbit(char byte, int nbit, char bit);
      void ConvertSourceToBitData(char *prgbSourceBuffer,
                                    long int lSourceSize, char *prgbBitData);
      void ConvertBitDataToSource(char *prgbSourceBuffer,
                                    long int lSourceSize, char *prgbBitData);
      void RotateBitData(long int lSourceSize, char *prgbPasswordBuffer,
                        char *prgbBitData, long int L,int mode);
      void MixBitData(long int lSourceSize, char *prgbPasswordBuffer,
                     char *prgbBitData, long int L, long int R, int mode);
      void (*UpdateProgress)(int progress);
};
```

Файл DSSL.cpp - шифрування SSL

```
#pragma once

#include "DSSL.h"
#include <math.h>

CDSSL::CDSSL()
{
        this->UpdateProgress = 0;
            Initialize(1024);
}

CDSSL::CDSSL(int SSLLen)
{
        this->UpdateProgress = 0;
            Initialize(SSLLen);
}

CDSSL::CDSSL(int SSLLen, void (*UpdateProgress)(int progress))
{
      // Підписуємося на функтор-оброблювач відновлення прогресу
    this->UpdateProgress = UpdateProgress;
      Initialize(SSLLen);
}

void CDSSL::Initialize(int SSLLen)
{
      // Фіксуємо розмір хеша
      SSLGammaLen  = SSLLen;

      // Виділяємо пам'ять під властивості класу шифрування SSL
      prgSourceData = new int[SSLGammaLen];
      prgSSL       = new int[SSLGammaLen];

      // Таблиця перекодування блоку вихідних даних
      int rgCodeTbl[256] =
      {
          34,254, 82,184,160,96,242,222,55,209,212,126,146, 23, 33,43,
          39,134, 59,128,236, 38,155,170, 69,172,252,238, 47,121, 228,
            183,203,135,165,166, 60, 98,7,207,120,189,210,8,226, 41, 72,
            253,71,24,171,196,101,168,169,186,0,46,68, 95,237,65,53,208,
            211, 83,114,157,144,32,193,143,36,250,75,234,49,167,125,141,
              58, 10,103,198,151,109, 37,112, 84,231,224,185,138,152, 94,
            99,139,22,191,136,111,162,227,118,26,102,12,50,132,21,76,213,
            73,197,16,119,48,140,110,54,45,4,148,192,205,158,124,214,180,
              217,230, 86,6, 28,221,107, 2,42,220,201,133, 74, 64, 20,129,
            159,92,66,246,13,216,40, 62,291,31, 3,85,206,145,79,154,142,
            204,117,223,195,244, 90,87,116,104,161,56, 179,77,225,150,5,
              194,174,251,229,115,57,249,215, 19,255,199,147, 70,149, 35,
            245, 63, 11,97,30,27,240,80,122,106,108, 78,173,182,61,130,
              88,219, 25,137, 15,175,190,241,181,239,153,232,1,177,233,14,
          51,164,200, 31,156,247,187, 89, 81,176,188,105,243,127, 17,
            202, 67, 44,235, 9,18,100,52,113,218,123, 93, 91,163,178,248
      };

    prgCodeTbl = new int[ASC_TABLE_SIZE];

    // Переносимо дані зі статичного масиву в динамічний
    for (int i = 0; i < ASC_TABLE_SIZE; i++)
    {
            prgCodeTbl[i] = rgCodeTbl[i];
    }
}

CDSSL::~CDSSL()
{
```

```
	delete [] prgSourceData;

	delete [] prgSSL;

	delete [] prgCodeTbl;
}

// XOR-Подібна операція для цілочислених операндів
inline int CDSSL::Encode(int IB, int PB)
{
	int OB;

	OB = IB - (prgCodeTbl[(unsigned char)PB]);

	if (OB < 0)
	{
	   OB += ASC_TABLE_SIZE;
	}

	return OB;
}

// Ініціалізація структури підключа
void CDSSL::InitSubkey()
{
	for (int i = 0; i < INT_LEN; i++) // Ініціалізуємо структуру
	{
		subkey.Checksums.rgChecksum[i] = 0;
	}
}

// Один крок обертання блоку вихідних даних
void CDSSL::RotateStep(int sourceDataLen)
{
	for (int i = 0; i < (sourceDataLen - 1); i++)
	{
		prgSourceData[i] = prgSourceData[i + 1];
	}
}

// Обертання блоку вихідних даних
void CDSSL::RotateSourceData(int cNumRounds, int sourceDataLen)
{
	int temp;

	// Обертаємо вихідні дані
	for (int i = 0; i < cNumRounds; i++)
	{
		// Зберігаємо нульовий елемент пароля,тому що на його місце у
функції
		// обертання будуть записані дані, що уліво зміщаються
		temp = prgSourceData[0];

		// Один крок обертання вихідних даних
		RotateStep(sourceDataLen);

		// Відновлюємо збережений елемент SSL-Гами в його кінці,
		// (він був "витиснутий" уліво й з'явився праворуч)
		prgSourceData[sourceDataLen - 1] = temp;
	}
}

// Нормування кількості раундів обертання
void CDSSL::NormcNumRounds(int &cNumRounds, int sourceDataLen)
{
	while (cNumRounds > sourceDataLen)
	{
		cNumRounds -= sourceDataLen;
	}
```

```
}

// Нормування індексу таблиці підстановок
inline void CDSSL::NormISubstTbl(int &iSubstTbl)
{
	if (iSubstTbl >= ASC_TABLE_SIZE)
	{
		iSubstTbl -= ASC_TABLE_SIZE;
	}
}

// SSL-Функція, заснована на діленні з остачею
inline int CDSSL::HF1(int k, int size)
{
	return (k % size);
}

// SSL-Функція, заснована на діленні з остачею
// (доповнення до функції HF1 для подвійного шифрування SSL)
inline int CDSSL::HF1_2(int k, int size)
{
	return 1 + (k % (size - 2));
}

// Подвійне шифрування SSL
inline int CDSSL::HFDouble(int k, int size, int numberOfTry)
{
	return (HF1(k, size) + numberOfTry * HF1_2(k, size)) % size;
}

// Нормування індексу в масиві підключа
void CDSSL::NormISubkey(int &iSubkey)
{
	iSubkey = HFDouble(iSubkey, SUBKEY_LEN, nTry1++);
}

// Нормування індексу в масиві SSL-гами
void CDSSL::NormBegCode(int &begCode)
{
	begCode = HFDouble(begCode, SSLGammaLen, nTry2++);
}

// Підготовка блоку вихідних даних до шифруванню SSL
void CDSSL::PrepareSourceData(int sourceDataLen)
{
	int i = 0, j = 0, cNumRounds, iCodeTbl;

	// Циклічне копіювання блоку вихідних даних у масив
	// prgSSL
	while (i < SSLGammaLen)
	{
		// Індекс у таблиці підстановок
		iCodeTbl = prgSourceData[j];

		// Таблична підстановка
		prgSSL[i] = prgCodeTbl[iCodeTbl];

		// Збільшуємо індекс у масиві вих. дан.
		j++;

		// Перевіряємо на закінчення вих. дан.
		if (j >= sourceDataLen)
		{
			// Обертати вих. дан. з одного елемента безглуздо
			if (sourceDataLen > 1)
			{
				// Первісна кількість зрушень вих. дан.
				cNumRounds = (prgSourceData[0] ^ i);
```

```
                        // Нормування кількості кроків обертання вих. дан.
                        NormcNumRounds(cNumRounds, sourceDataLen);

                        // Обертання вих. дан.
                        RotateSourceData(cNumRounds, sourceDataLen);
                  }
                  j = 0; // Обнуляем j (починаємо копіювати вих. дан. з
                         // prgSourceData в prgSSL) з початку
            }
            i++;
      }
}

// Обчислення підключа (використовується для кодування SSL-Гами
// prgSSL. Підключ обчислюється на основі контрольних сум гами
// prgSSL, що підлягає кодуванню)
void CDSSL::CalculateSubkey()
{
      for (int i = 0; i < SSLGammaLen; i++)
      {
            subkey.Checksums.rgChecksum[0] += (prgSSL[i] * prgSSL[i] ^ prgSSL[i]
* i * i * i);
            subkey.Checksums.rgChecksum[1] += (prgSSL[i] * prgSSL[i] * prgSSL[i]
^ prgSSL[i] + i);
            subkey.Checksums.rgChecksum[2] += (prgSSL[i] * prgSSL[i] ^
prgSSL[i]);
            subkey.Checksums.rgChecksum[3] += (prgSSL[i] * prgSSL[i] ^ prgSSL[i]
* i * i);
      }
}

// Перший раунд кодування - починаючи з BegCode і до кінця
void CDSSL::FirstCodeRound(int begCode)
{
      int IB, PB;

      for (int i = begCode; i < (SSLGammaLen - 1); i++)

      {
            IB = prgSSL[i];    // Кодуємий елемент потоку
            PB = prgSSL[i + 1]; // елемент, Що Кодує, потоку
            prgSSL[i] = Encode(IB, PB);
      }
}

// Кодування першого елемента SSL-Гами останнім - реалізуємо
// перенос ОС на початок
inline void CDSSL::MoveCode()
{
   int IB, PB;

   IB = prgSSL[SSLGammaLen - 1];
   PB = prgSSL[0];
   prgSSL[SSLGammaLen - 1] = Encode(IB, PB);
}

// Другий раунд кодування - з початку й до BegCode
void CDSSL::SecondCodeRound(int begCode)
{
      int i, IB, PB;

      // Другий раунд кодування - з початку й до BegCode
      for (i = 0; i < (begCode - 1); i++)
      {
            IB = prgSSL[i];      // Кодуємий елемент потоку
            PB = prgSSL[i + 1];  // елемент, Що Кодує, потоку
            prgSSL[i] = Encode(IB, PB);
      }
}
```

```
// Кодування SSL-гами
void CDSSL::SSLCode()
{
      int begCode = 0;
      iSubkey++;

      // Нормування індексу в масиві підключа
      NormISubkey(iSubkey);

      // Обчислюємо значення індексу початку кодування (BegCode)
      begCode += (unsigned char)(prgSSL[0] + subkey.rgubSubkey[iSubkey]);

    // Нормуємо параметр автокодування
    NormBegCode(begCode);

      // Перший раунд зв'язування - починаючи з BegCode і до кінця
    FirstCodeRound(begCode);

      // Кодування першого елемента останнім - реалізуємо
      // перенос "ОС" на початок SSL-гами
      MoveCode();

      // Другий раунд кодування - з початку й до BegCode
      SecondCodeRound(begCode);
}

// Реалізація ОС при накладенні підключа на SSL-Гаму
void CDSSL::SubkeyCode()
{
      for (int i = 0; i < SUBKEY_LEN; i++)
      {
            subkey.rgubSubkey[i] ^= (unsigned char)rgFeed[i];
      }
}

// Накладення гами підключа на SSL-Гаму
void CDSSL::ImposeSubkeyGamma()
{
      int IB, PB;

      // Буфер ОС по даним підключа з контрольних сум

      // Накладення контрольної суми
      for (int i = 0, j = 0; i < SSLGammaLen; i++, j++)
      {
            if (j >= SUBKEY_LEN)
            {
                  // Реалізація ОС - кодування підключа unSubkey.Subkey[],
                  // який складається з контрольних сум prgSSL, блоком вихідних
                  // даних
                  SubkeyCode();

                  j = 0;
            }

            // Блок кодування гами ключа за допомогою підключа

            // Кодуємий байт основного ключа
            IB = prgSSL[i];
            // байт, що кодує, з підключа
            PB = (int)subkey.rgubSubkey[j];

            // Результат кодування
            prgSSL[i] = Encode(IB, PB);

            // Формуємо буфер ОС
            rgFeed[j] = prgSSL[i];
      }
```

```
}

// Функція шифрування SSL даних
void CDSSL::SSL(char *prgbSourceData, int sourceLen)
{
    int cRounds;

    iSubkey = 0;
      nTry1 = 0;
      nTry2 = 0;

      // Переносимо вихідні дані в масив int (він і буде шифруватися SSL)
      for (int i = 0; i < sourceLen; i++)
      {
            prgSourceData[i] = (unsigned char)prgbSourceData[i];
      }

      // Ініціалізуємо структуру, що містить масив контрольних сум
      // і вхідну в об'єднання Subkey
      InitSubkey();

      // Підготовка вих. дан.: циклічне копіювання його в prgSSL
      PrepareSourceData(sourceLen);

      // Шифрування SSL блоку вихідних даних
      for (cRounds = 0; cRounds < SSLGammaLen; cRounds++)
      {
            // Обчислення підключа
            CalculateSubkey();

            // Кодування SSL-гами XOR-подібною операцією
            SSLCode();

            // Накладення гами підключа
            ImposeSubkeyGamma();

            // Обновляємо прогрес
            if (UpdateProgress != NULL)
            {
                  UpdateProgress(cRounds);
            }
      }

      // Записуємо результат в "цільовий" масив
      for (int i = 0; i < SSLGammaLen; i++)
      {
            prgbSourceData[i] = (char)prgSSL[i];
      }
}
```

Файл DSSL.h - бібліотека для файлу DSSL.cpp

```
#pragma once

#define ASC_TABLE_SIZE 256 // Довжина ASCII-Таблиці
#define SUBKEY_LEN 16      // Довжина підключа unSubkey.Subkey[]
#define MODKEY_LEN 3       // Довжина ключа для кодування модуля
#define INT_LEN 4        // Довжина в байтах змінної типу int
#define NULL 0

// Об'єднання для перетворення чотирьох контрольних сум у підключ,
// складається з 16 байт
typedef union
{
      struct TChecksums // Поле масиву з 4-ьох змінних типи int
      {
            int rgChecksum[4];
      } Checksums;

      unsigned char rgubSubkey[SUBKEY_LEN];  // ...використовується як масив
                                                  // unsigned char з
16
                                             // елементів (4*4=16)
} TSubkey;

// Клас шифрування SSL блоку даних
class CDSSL
{
public:
      CDSSL();
      CDSSL(int SSLLen);
      CDSSL(int SSLLen, void (*UpdateProgress)(int progress));
      virtual ~CDSSL();

      // Функція шифрування SSL даних
      void SSL(char *prgbSourceData, int SourceLen);

private:

      int SSLGammaLen;
      TSubkey subkey;

      int rgFeed[SUBKEY_LEN];

      int *prgSourceData;
      int *prgSSL;
      int iSubkey;
      int *prgCodeTbl;
      int nTry1;
      int nTry2;

      void Initialize(int SSLLen);
      inline int Encode(int IB, int PB);
      void InitSubkey();
      void RotateStep(int sourceDataLen);
      void RotateSourceData(int cNumRounds, int sourceDataLen);
      void NormcNumRounds(int &cNumRounds, int sourceDataLen);
      inline void NormISubstTbl(int &iSubstTbl);
    inline int HF1(int k, int size);
      inline int HF1_2(int k, int size);
      inline int HFDouble(int k, int size, int numberOfTry);
      void NormISubkey(int &iSubkey);
      void NormBegCode(int &begCode);
      void PrepareSourceData(int sourceDataLen);
      void CalculateSubkey();
      void FirstCodeRound(int begCode);
      void MoveCode();
```

```
        void SecondCodeRound(int begCode);
        void SSLCode();
        void SubkeyCode();
        void ImposeSubkeyGamma();
        void (*UpdateProgress)(int progress);
};
```