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

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

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

Виконав здобувач вищої освіти
IV курсу, групи КБ-21
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Ліподат А.Р.
«____» _____ 2025 р.

Керівник проекту
доктор технічних наук, професор
_____ Смірнов О.А.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Освітній ступінь *бакалавр*
Галузь знань ._12 _____*"*Інформаційні технології*"*_____
Спеціальність *125 "*Кібербезпека*"*
Освітньо-професійна (освітньо-наукова) програма _*"*Кібербезпека*"*_

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

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

Ліподату Антону Романовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки передачі конфіденційних даних у мережі за протоколом DNSSEC*

2. Керівник роботи *Смірнов Олексій Анатолійович, докт. техн. наук, професор*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 57-02 від 17.01.2025 року

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

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

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 2025 р.

КАЛЕНДАРНИЙ ПЛАН

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

Дата видачі завдання
« 17 » січня 2025 р.

Підпис керівника
_____ Смірнов О.А.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Ліподат А.Р.
(прізвище та ініціали)

АНОТАЦІЯ

Ліподат А.Р. Програмне забезпечення системи кібербезпеки передачі конфіденційних даних у мережі за протоколом DNSSEC. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

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

Метою розробки є програмне забезпечення системи кібербезпеки передачі конфіденційних даних у мережі за протоколом DNSSEC.

Результат роботи – програмна реалізація системи кібербезпеки передачі конфіденційних даних у мережі за протоколом DNSSEC.

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

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

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

Програму розроблено в середовищі Builder C++.

Ключові слова: кібербезпека, DNSSEC

ABSTRACT

Lipodat A.R. Software for the cybersecurity system of confidential data transmission in the network using the DNSSEC protocol. 125 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed, which is intended for the cybersecurity system of confidential data transmission in the network using the DNSSEC protocol.

The purpose of the development is the software for the cybersecurity system of confidential data transmission in the network using the DNSSEC protocol.

The result of the work is the software implementation of the cybersecurity system of confidential data transmission in the network using the DNSSEC protocol.

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

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

The program can be used on PCs with Windows 10/11.

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

Keywords: cybersecurity, DNSSEC

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

АС	–	автоматизована система
ЕЦП	–	електронний цифровий підпис
ІзОД	–	інформація з обмеженим доступом
ОС	–	операційна система
ПЕОМ	–	персональна електронно-обчислювальна машина
ПО	–	програмне забезпечення
ППС	–	прикладна програмна система
СКЗІ	–	система конфіденційного захисту інформації

ВСТУП

Актуальність теми. Для зв'язку з іншим користувачем по Інтернету необхідно ввести відповідну адресу в комп'ютері: імені або номер. Дана адреса повинна бути унікальною, щоб комп'ютери могли знайти один одного. ICANN здійснює координацію цих унікальних ідентифікаторів по усьому світі. Без цього не існувало б глобального Інтернету. При уведенні імені системі спочатку необхідно перетворити його в числову адресу, а вже потім вона може встановити з'єднання. Дана система називається доменною системою імен (Domain Name System – DNS) і вона перетворить імена, наприклад www.kntu.kr.ua, у числа, називані адресами Інтернет-протоколу. Організація ICANN координує систему адресації для забезпечення унікальності всіх адрес. Недавно в DNS були виявлені уразливі місця, які дозволяють зловмисникові перехопити процес пошуку за іменем людини або вузла в Інтернеті . Метою атаки є захват контролю над сеансом, щоб, наприклад, направити користувача на веб-сайт зловмисника для одержання облікового запису й пароля. Дані уразливості привели до необхідності впровадження технології, називаної "Розширена безпека DNS" (DNS Security Extensions – DNSSEC), для захисту даної частини інфраструктури Інтернету. DNSSEC (Domain Name System Security Extensions) – набір розширень IETF протоколу DNS, що дозволяють мінімізувати атаки, пов'язані з підміною DNS-адреси при дозволі доменних імен. Він спрямований на надання DNS-клієнтам (англ. термін resolver) автентичних відповідей на DNS-запити (або автентичну інформацію про факт відсутності даних) і забезпечення їхньої цілісності. При цьому використовується криптографія з відкритим ключем. Не забезпечується доступність даних і конфіденційність запитів. Забезпечення безпеки DNS критично важливо для інтернет-безпеки в цілому.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки передачі конфіденційних даних у мережі за протоколом DNSSEC.

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

– Огляд існуючих систем передачі конфіденційних даних у мережі за протоколом DNSSEC.

– Дослідження системи кібербезпеки передачі конфіденційних даних у мережі за протоколом DNSSEC.

– Програмна реалізація системи кібербезпеки передачі конфіденційних даних у мережі за протоколом DNSSEC.

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

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки передачі конфіденційних даних у мережі за протоколом DNSSEC, 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

Представлено розробку Інтернет-протоколу під назвою Domain Name System Security Extensions (DNSSEC) для захисту інфраструктури Інтернету. DNSSEC комерційно запускає .se, організація, відповідальна за шведський домен верхнього рівня, і розгортає європейський постачальник послуг інфраструктури Інтернету Reseaux IP Européens Network Coordination Center. DNS – це розподілена ієрархічна система, яка підтримується набором організацій і організацій на низці платформ і конфігурацій, що відображають імена, які легко запам’ятовуються, на IP-адреси. Структура DNS доменів і субдоменів виражається в одній із двох метафор, які є листом і вузлом і батьківським потомком. DNSSEC запевняє користувачів, що ресурс даних підтверджено як заявлене джерело, а зіставлення імені з IP-адресою є точним.

1.2 Область застосування

DNSSEC розшифровується як «Розширення безпеки системи доменних імен». Це функція безпеки для системи доменних імен (DNS), яка перевіряє інформацію DNS (наприклад, IP-адресу) доменного імені. Використовуючи криптографічні цифрові підписи, технологія DNSSEC гарантує, що кінцевий користувач отримує доступ до фактичного веб-сайту або інших служб, які відповідають доменному імені. Іншими словами, DNSSEC не дозволяє зловмисникам перенаправляти кінцевих користувачів (на рівні DNS) на підроблений веб-сайт або службу.

DNSSEC захищає від атак DNS-спуфінгу «людина посередині» та «отруєння кешу», забезпечуючи криптографічну перевірку інформації DNS до того, як DNS-сервер перенаправить кінцевого користувача на веб-сайт.

Коли користувачі заходять на веб-сайт, використовуючи його доменне ім’я, наприклад, http://www.example.sg, системний DNS-розпізнавач спочатку запитує IP-адресу веб-сайту. Коли DNS-розпізнавач (наприклад, розпізнавач ISP) робить запит, зловмисник може обманом змусити розпізнавач прийняти фальшиву IP-адресу. Це відоме як атака «людина посередині». Більшість резолверів DNS також кешують повернуту IP-адресу, щоб пришвидшити відповіді на майбутні запити для того самого доменного імені від того самого користувача чи інших користувачів. Таким чином, якщо зловмиснику вдалося змусити розпізнавач DNS прийняти фальшиву IP-адресу, фальшива IP-адреса тепер кешується розпізначем DNS. Ця атака відома як «отруєння кешу». Коли інші користувачі роблять наступні запити до того самого домену (наприклад, інші користувачі того самого провайдера), DNS-розпізнавач перенаправлятиме їх на фальшиву IP-адресу. Це тому, що ці інші користувачі отримали кешовану та неправильну IP-адресу замість законної IP-адреси веб-сайту.

Як працює DNSSEC?

DNSSEC використовує криптографічні підписи для створення «ланцюжка довіри». DNSSEC використовує цей «ланцюг довіри», щоб підтвердити, що інформація, яку отримують користувачі, походить від правильних серверів DNS. Якщо DNSSEC не може перевірити інформацію, він відхиляє інформацію. Таким чином, якщо користувачі відвідують веб-сайт, захищений DNSSEC, і відповідь DNS змінюється хакером (через атаку "людина посередині"), розпізнавач DNS із підтримкою DNSSEC або програма може виявити підроблену інформацію та відкинути її.

Пояснення високого рівня: (Ланцюжок довіри)

1. Клієнт запитує DNS-запис від свого локального рекурсивного сервера.

2. Локальний рекурсивний сервер отримує запис DNS разом із відкритими ключами авторитетного сервера.

3. Локальний рекурсивний сервер перевіряє відкриті ключі офіційного сервера через запис DS, що зберігається на сервері верхнього рівня.

4. Локальний рекурсивний сервер отримує відкриті ключі для сервера TLD.

5. Локальний рекурсивний сервер перевіряє відкриті ключі сервера TLD через запис DS, що зберігається на кореневому сервері.

6. Локальний рекурсивний сервер отримує відкриті ключі для кореневого сервера.

7. Локальний рекурсивний сервер перевіряє кореневий сервер.

8. DNS-запит повертається після перевірки всіх серверів.

Повний опис пояснення високого рівня: (ланцюжок довіри)

1. Клієнт запитує DNS-запис від свого локального рекурсивного сервера.

2. Локальний рекурсивний сервер отримує запис DNS разом із відкритими ключами авторитетного сервера.

3. Локальний рекурсивний сервер перевіряє відкриті ключі офіційного сервера через запис DS, що зберігається на сервері верхнього рівня.

4. Локальний рекурсивний сервер отримує відкриті ключі для сервера TLD.

5. Локальний рекурсивний сервер перевіряє відкриті ключі сервера TLD через запис DS, що зберігається на кореневому сервері.

6. Локальний рекурсивний сервер отримує відкриті ключі для кореневого сервера.

7. Локальний рекурсивний сервер перевіряє кореневий сервер.

8. DNS-запит повертається після перевірки всіх серверів.

DNSSEC усуває ризики безпеки в протоколі DNS, додаючи автентифікацію для відповідей, отриманих від серверів DNS, запобігаючи підробці DNS, отруєнню кешу та викраденню.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки передачі конфіденційних даних у мережі за протоколом DNSSEC, 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

Розглянемо програмні комплекси, які реалізують технології забезпечення передачі конфіденційних даних у мережах.

Крипто-Експрес

Програмний продукт «Крипто-Експрес» призначений для захисту інформації від несанкціонованого доступу, забезпечення цілісності переданих даних, підтвердження дійсності електронного цифрового підпису й захисти електронного документа від підробки.

Простий і зручний інтерфейс програмного продукту забезпечує комфортну роботу із ключовими носіями й сертифікатами ключів підпису.

Додаток використовує засоби криптографічного захисту інформації відповідно до законодавства Російської Федерації й міжнародних стандартів.

Додаток інтегрується в стандартні засоби операційних систем Microsoft Windows. Продукт має автоматичну систему відновлень.

ПП "Крипто Експрес" дозволяє здійснювати:

І. Роботу з ЕЦП (у форматі стандартизованого підписаного повідомлення PKCS#7, у тому числі можливість роботи з підписаними повідомленнями, що включають вихідні дані, так і не включають – «відділена» підпис):

– Підпис файлів з довільними даними довільного розміру (обмеження накладає тільки розмір носія інформації).

– Додавання підпису в підписане повідомлення.

– Додавання підпису, що візує, у підписане повідомлення (у тому числі можливість візувати підпис на будь-якому рівні «дерева» підписів).

– Перевірка підписів у повідомленні (так само можлива перевірка підписів, що візують, на будь-якому рівні «дерева» підписів).

– Видалення довільного підпису з підписаного повідомлення.

– Робота зі штампами часу (TSP): одержання штампів часу і їхня перевірка.

– Робота з «удосконаленим електронним цифровим підписом (CAdES).

II. Роботу із зашифрованими повідомленнями (у форматі стандартизованого зашифрованого повідомлення PKCS#7):

– Шифрування файлів з довільними даними довільного розміру (обмеження накладає тільки розмір носія інформації).

– Розшифровка зашифрованих повідомлень.

III. Супутні можливості:

– Виклик реалізованих операцій з контекстного меню Провідника Windows.

– Можливість установлювати сертифікати із ключових носіїв, з файлів у сховище сертифікатів Windows.

– Максимально спрощений інтерфейс, оформлений у вигляді «майстрів».

– Висока швидкість роботи.

– Система автоматичної перевірки відновлень, легкість самого процесу відновлення.

– Установка й відновлення продукту не вимагають прав адміністратора системи.

Криптопровайдер « LISSI-CSP»

Програмний комплекс захисту інформації, що дозволяє використовувати російські криптографічні алгоритми в операційних системах Microsoft через стандартний інтерфейс CryptoAPI. Як криптографічне ядро використовує сертифіковане СКЗІ «ЛІРССЛ» («LirSSL»).

Криптопровайдер LISSI-CSP забезпечує:

– шифрування даних за ДСТ 28147-89 у режимах простої заміни, гаммування, гаммування зі зворотним зв'язком і в режимі зчеплення блоків;

– формування/перевірку електронного цифрового підпису (ЕЦП) відповідно до ДСТ Р 34.10-2001;

– контроль цілісності даних за допомогою обчислення имитовставки за ДСТ 28147-89;

– обчислення значення хеш-функції відповідно до ДСТ Р 34.11-94;

– генерацію ключової інформації;

– підтримку протоколів SSL v3 і TLS v1 з використанням російських криптоалгоритмів.

СКЗІ «ЛІРССЛ» («LirSSL»)

Призначено для використання криптографічних перетворень на базі російських алгоритмів у прикладному ПЗ. Програмний інтерфейс СКЗІ «LirSSL» повністю сполучимо з API OpenSSL. СКЗІ «LirSSL» є крос-платформним і функціонує під керуванням операційних систем MS Windows (включаючи MS Windows Vista), Sun Solaris, Linux, FreeBSD, IBM AIX, HP HP/UX і QNX, Mac OS.

СКЗІ «LirSSL» успішно пройшло сертифікаційні випробування ФСБ Росії по класах КС1 і КС2 (сертифікат відповідності реєстраційний номер СФ/ 114-2055, СФ/ 114-2056).

ПАК «ЛІССІ УЦ»

Призначений для створення центрів, що засвідчують, електронного цифрового підпису (ЕЦП) в автоматизованих системах захищеного електронного документообігу на основі технології відкритого розподілу ключів (PKI), у тому числі в органах державного керування й організаціях Російської Федерації. ПАК «ЛІССІ УЦ» успішно пройшов сертифікаційні випробування по «Вимогах до інформаційної безпеки центрів, що засвідчують,» по класах захисту КС1, КС2 і має позитивний висновок ФСБ Росії.

ПК « LirVPN-FW»

Міжмережний екран з можливість створення захищений віртуальних корпоративних мереж (ЗВКС). ПК " LirVPN-FW" (Сертифікат відповідності реєстраційний номер №1980 від 15.12.2009р.) функціонує під керуванням операційних систем MS Windows (включаючи Vista), Linux. ПК " LirVPN-FW", розроблений відповідно до вимог до міжмережних екранів, викладеним у РД Держтехкомісії Росії "Засобу обчислювальної техніки. Міжмережні екрани. Захист від несанкціонованого доступу до інформації. Показники захищеності від несанкціонованого доступу до інформації" для третього класу захищеності.

КСКЗІ « LISSI-SNC»

Комплекс засобів криптографічного захисту інформації LISSI-SNC забезпечує захист від несанкціонованого доступу шляхом автентифікації користувачів на сертифікатах ЕЦП і захист мережного трафіка в продуктах компанії SAP, що функціонують на базі Web Application Server, по протоколі SNC з використанням російських криптографічних алгоритмів.

КСКЗІ « LISSI-SSF»

Комплекс засобів криптографічного захисту інформації LISSI-SSF являє собою комплекс засобів підтримки протоколу SSF у продуктах компанії SAP AG і забезпечує криптографічний захист електронних документів за рахунок використання механізмів ЕЦП для підпису документів і зберігання документів у зашифрованому виді з використанням російських криптоалгоритмів.

ПК « SMS-FW»

Програмний комплекс SMS-FW як засіб захисту від несанкціонованого доступу розділяє захищену й публічну мережі й не приводить до появи можливостей для доступу по будь-яких мережних протоколах з публічної мережі (включаючи мережу Інтернет) у захищену обчислювальну мережу, а також можливості доступу користувачів захищеної обчислювальної мережі в публічну мережу (включаючи мережу Інтернет).

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 конфіденційних даних у мережі за протоколом DNSSEC.

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

Споконвічно система доменних імен не мала механізмів захисту від підміни інформації у відповіді сервера, тому що в часи розробки (на початку 1980-х років) погрози безпеки сучасного Інтернету не були актуальними. При цьому клієнти судили про вірність отриманої інформації винятково по двобайтовому ідентифікаторі запиту. Таким чином, зловмисникові було потрібно перебрати 65536 значень, щоб «отруїти кеш». Це означало, що дані в системі DNS ушкоджені (навмисно або через помилку), а DNS-сервер кешує їх для оптимізації швидкодії (кеш стає «отруєним») і починає надавати ці неавтентичні дані своїм клієнтам. Ще в 1990 році Стів Белловін (англ. Steve Bellovin) виявив серйозні недоліки в безпеці. Дослідження в цій області почалися й проводилися із часів публікації доповіді в 1995 році [1].

Підписання й перевірка даних DNS створюють додаткові витрати, що негативно позначається на продуктивності мережі й серверів. Приміром, у середньому зона DNSSEC (сукупність доменних імен певного рівня вхідних у конкретний домен) в 7-10 разів перевищує по розмірі саму систему DNS. Генерація й перевірка підписів віднімає значний час ЦПУ. Підписи й ключі займають на порядок більше місця на диску й в оперативній пам'яті, чим самі дані.

Принцип роботи

Перевірка дійсності даних в DNSSEC

Принцип роботи DNSSEC заснований на використанні цифрових підписів. В адміністратора є записи про відповідність імені домену й IP-адреси. DNSSEC ставить кожній з них у строгу відповідність спеціальну, строго певну послідовність символів, що являє собою цифровий підпис. Головна особливість

цифрового підпису в тому, що є відкриті, привселюдно доступні методи перевірки вірогідності підпису, а от генерування підпису для довільних даних вимагає наявності в розпорядженні секретного ключа, що підписує. Тому перевірити підпис може кожний учасник системи, але підписати нові або змінені дані на практиці може тільки той, у кого є секретний ключ.

У теорії ніщо не забороняє зломщикові обчислити секретний ключ і підібрати підпис, однак на практиці для досить складного й довгого ключа таку операцію не виконати за розумний час при наявності існуючих обчислювальних засобів і математичних алгоритмів.

При наявності секретного ключа обчислення цифрового підпису не представляє складності. Така робота асиметричних систем шифрування з відкритим ключем, що лежать в основі алгоритмів цифрового підпису.

Delegation of Signing (DS)-запис містить хеш доменного імені спадкоємця і його ключа KSK. Запис DNSKEY містить публічну частину ключа і його ідентифікатори (ID, тип і використовувана хеш-функція).

KSK (Key Signing key) означає ключ підпису ключа (ключ довгострокового користування), а ZSK (Zone Signing Key) означає ключ підпису зони (ключ короткочасного користування). За допомогою KSK верифікується ZSK, яким підписується коренева зона. ZSK кореневої зони створює компанія VeriSign, що технічно відповідає за поширення кореневої зони DNS. По прийнятої ICANN процедурі ZSK кореневої зони обновляється чотири рази в рік.

Проблеми впровадження й недоліки

Впровадження DNSSEC сильно затрималося з таких причин, як:

1. Розробка назад сумісного стандарту, який можна масштабувати до розміру інтернету.

2. Розбіжності між ключовими гравцями із приводу того, хто повинен володіти доменами верхнього рівня (.com, .net).

3. DNS сервери й клієнти повинні підтримувати DNSSEC.

4. Оновлені DNS-клієнти, що вміють працювати з DNSSEC, повинні використовувати TCP.

5. Кожний клієнт повинен одержати хоча б один довірений відкритий ключ до того моменту, як він почне використовувати DNSSEC.

6. Зростання навантаження на мережу через серйозно (в 6-7 разів) зрослого трафіку від запитів.

7. Зростання навантаження на процесор сервера через потребу генерації й перевірки підписів, так що може знадобитися заміна деяких недостатньо потужних DNS-серверів.

8. Збільшення вимог для сховищ інформації про адресацію, тому що підписані дані займають набагато більше місця.

9. Потрібно створювати, тестувати і допрацьовувати програмне забезпечення як серверної, так і клієнтської частини, що вимагає часу й випробувань у масштабах усього інтернету.

10. Stub-DNS-клієнти не захищені від отруєння кеша – валідація відбувається на стороні рекурсивного сервера, клієнт одержує тільки відповідь із виставленим битому AD.

11. Різко зросла небезпека атаки DNS Amplification.

Більша частина цих проблем рішається технічним інтернет-співтовариством.

3.2 Розробка структурної схеми

Принцип роботи DNSSec той же, що й у цифрового підпису. Тобто закритим ключем підписуємо, відкритим звіряємо.

Особливість полягає в тому, що DNSSec використовує два типи ключів – одним підписується зона (ZSK, zone signing key), іншим підписується набір ключів (KSK, key signing key). Зроблено це з таких міркувань: зона може бути досить великою щоб удалося підібрати закритий ключ, тому його треба міняти

частіше, та й зробити його можна більш коротким, щоб зони підписувалися швидше; KSK же використовується для невеликих обсягів даних, тому його можна й достовірніше зробити й міняти більш рідко. Тим більше, що хеш від відкритої частини KSK потрібно відправити в батьківську зону, що хотілося б робити не занадто часто.

ZSK

За допомогою цього ключа підписуються всі набори записів у зоні (RRSET), крім точок делегування. Тобто, допустимо у вас є зона kntu.kr.ua і в ній такий шматок:

```
$ORGIGIN kntu.kr.ua.
@       SOA dns.kntu.kr.ua ns.kntu.kr.ua {
                Serial
                Refresh
                Retry
                Expire
                nTTL
                }
        NS ns.kntu.kr.ua.
        NS ns1.kntu.kr.ua.
        DNSKEY  256 3 5 (
                                AwEAAfETe4xtZy3Ml+9NceWE0zTUWk5WgTs5
                                ogRJ1fVu5U2QmBb+t3lt4BrZObLRj2
                                HcMmneVC4C3IdgluAi6Jpj7hgRZIbb8les
                                Lai0/wOo/byPvNi5T0OQp3vgXyhoBdWxl
                                zghFU3eQSAnWF0x/c323rt0ird7v5
                                ) ; key id = 38754
         DNSKEY  257 3 5 (
                                AwEAAdanjnt82rOd97sDS5+QH+w1pKnRJ/
                                b8gmuRBC91q5f2Yi5zzYKi9+gENlqx/1MN
                                jAFXJ1Fvtf0pJYKjmkiBJoHdZoEVRnJz8ODx
                                FYgFX/fx+SBKs3ZioaHP3CEdZQ4k3kut6o
                                tawolvHfErSwnJT/3IhAplXDHZrLmwXgWU3M
                                PvMhnJRR9jd7S4f3WF10oq+3qPkBbJrqx3x
                                RydCSaZYfVu5U2QmBb+t3lt4BB8j8jOLS
                                zv2lufa6P8f0TJxtcpx/t9IfvUyWHmr9H7r
                                inl4k8xDTVvaPnmBScxbuBc=
                                ) ; key id = 23179
doo     IN A 127.0.0.1
```

```
        IN A 127.0.0.2
        IN MX mail
foo      NS ns1.test.ru.
        NS ns2.test.ru.
bar     NS ns1.test.ru.
        NS ns2.test.ru.
        DS 4915 5 1 180DC61CD4A2772DC02EC18934AA4C024D77DC11
        DS   4915   5   2   03EE2B29404BDF6D8891C0E0C89F714FE865E1D59A621F6FB4642648
4BC8C852
```

І ви вирішили цю зону підписати. Зроблене буде наступне:

1. Створиться послідовність підписаних записів.
2. Додадуться записи NSEC (про це трохи нижче).
3. Підпишуться записи SOA, набір NS для kntu.kr.ua, адресні й MX запису для doo, кожна NSEC RR і кожний набір DS.
4. Залежно від налаштувань програмного забезпечення також може бути підписаний набір DNSKEY.
5. Точки делегування, тобто NS запису для дочірніх зон, підписані не будуть.

KSK

Цим ключем підписується набір DNSKEY записів.

Крім того, від відкритої частини KSK береться хеш, що надалі відправляється в батьківську зону. У вищенаведеному прикладі це DS запису (Delegation Signer).

У випадку з кореневою зоною передбачається, що відкрита частина ключа буде повідомлена DNS-клієнтові вручну. А щоб при зміні ключа не обновляти його щораз вручну, існують механізми його автоматичного відновлення, наприклад в BIND'a є опція managed-keys.

Очевидно, що закриту частину KSK варто зберігати як зіницю ока – по-перше інакше зникає весь зміст DNSSec, по-друге у випадку компрометації швидко перемінити KSK не вийде.

Next SECure

Ну, от підписали ми зону, але однаково хтось сторонній може додати в неї свою інформацію й, навіть непідписана, вона буде віддаватися сервером нарівні з підписаною.

Щоб такого не відбувалося, при підписі зони доменні імена сортуються за абеткою, до кожного з них додається запис NSEC, у якій вказується яке наступне доменне імені захищене і якого запису для нього присутні в зоні. Остання NSEC запис указує на SOA.

Якщо авторитетний сервер повертає відповідь NXDOMAIN (RCODE=3) або NODATA (RCODE=0), то у відповіді обов'язково повинна бути присутнім NSEC запис. Приклад такої відповіді:

```
; <<>> Di 9.7.3 <<>> jhbczxccva.se +dnssec
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 8766
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 6, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;jhbczxccva.se.                  IN      A
;; AUTHORITY SECTION:
se.                             300     IN      SOA in-the-rye.nic.se. registry-
default.nic.se. 2011053104 1800 1800 864000 7200
se.                             300     IN      RRSIG   SOA 5 1 172800 20110613123930
20110531090447 24825 se. JJGWAvmc/Bq7t5z7iTQGLr9c0LzQkdwpNFsrIClJctZUn/Z3YN2EMVQ
0r6DvufTGk1L8JMvTaxk43ZmvasFeNNzfUFjr+td8Mv9h7FF5kTfEO5
R7JMh4j0Kxl/Gy4j+Ofcm0ZiCZTtcnNdHwCIHaVpz9KA6uIubnlJNSLw YXI=
```

Тому що відповідь NXDOMAIN завжди супроводжується SOA записом, то в підписаній зоні вертається SOA і підпис для неї.

```
se.                             300     IN      RRSIG   NSEC 5 1 7200 20110613011140
20110530130445 24825 se. dKL3iGCxNI51FSNjx4p0NmAMtjhJVqLeAShrRH0rRmd0Ej1nAH/Z/ip
zn7Pq+7j6PguNPfEU4ySHf8BTprvmod60J//Asi9/2ymadcNk5VXg
fJD1DMBpnCK1aUq8ieJFsMQuMrrYRkhy0TdrCxGtZmTCpOOxfOMtmKR rc=
se.                             300     IN      NSEC 0-0.se. NS SOA TXT RRSIG NSEC DNSKEY
```

Ця NSEC запис показує, що зазначене імені не попадає під маску.

```
jhbatar.se.             300      IN       RRSIG    NSEC 5 2 7200 20110613032526
20110530130445  24825  se.  DWXq1Zlzu9w0McNHWjpLO55H08rkWjtBDd8TewUCYnlj//1oXEvVc
ORT9AxXoz9TMOEku2wFGydce5zs4PZLwnEC+ieXu3ri/B0S533XpmKe
6CgQTda6maCvo8d1gOc2nIbd7zKjdOl2ujMVJKCb6Bv3yoy4Wj3gka Ufk=
jhbatar.se.             300     IN      NSEC    jhbeagleklubb.se. NS RRSIG NSEC
```

А вже ця NSEC запис говорить про те, що доменне імені не виявлене.

Недолік NSEC очевидний – хто завгодно може одержати вміст зони просто опитавши для кожного імені цей запис. У зв'язку із цим механізм був дороблений і з'явилися записи NSEC3, у яких доменні імена хешуються.

NSEC3 для обчислення хеша може використовувати сіль, крім солі можна задати кількість ітерацій. Варто помітити, що збільшення кількості ітерацій приводить до збільшення навантаження як на DNS-клієнт, так і на авторитетний сервер, причому на останній більшою мірою. Це відбувається через те, що для повернення NXDOMAIN, авторитетний сервер повинен обчислити хеш, і не один. Процес описаний в RFC 5155.

Крім того, NSEC3 запис має поле прапорів, якого немає в NSEC. На даний момент визначений тільки один прапор – OPT-OUT, завдяки якому існує можливість підписувати не всю зону (що при більших розмірах може бути досить тривалим процесом), а тільки ті доменні імена, для яких є DS запису.

З однієї сторони OPT-OUT дозволяє сильно скоротити час підпису, однак при його використанні, наявність у зловмисника доступу до файлу зони дозволяє йому додати незахищений запис, що надалі може доставити проблеми.

Механізм роботи

Допустимо ми хочемо довідатися адресу test.bar.kntu.kr.ua:

1. Ми запитуємо доменне імені в DNS-клієнта.

2. DNS-клієнт виставляє біт DO і запитує test.bar.kntu.kr.ua у кореневого сервера.

3. DNS-клієнт знає, що коренева доменна зона підписана – у нього зазначений ключ або хеш ключа для неї (так званий trust-anchor), тому він запитує в кореневого сервера DNSKEY запису для кореневої зони й звіряє отримане з наявним.

4. Кореневий сервер не знає такого доменного імені, та й взагалі максимум що йому відомо – на яких серверах розташовується зона com, він і повідомляє адреси цих серверів нашому DNS-клієнтові разом з підписаної DS записом для зони com.

5. DNS-клієнт валідує DS запис отриманим і перевіреним ZSK ключем кореневої зони.

6. Тепер DNS-клієнт знає, що зона com підписана, так що він запитує в її DNS сервера DNSKEY і валідує їх, після чого цікавиться про bar.kntu.kr.ua. Природно, що сервер зони com про таких не знає, йому тільки відомо, що зона kntu.kr.ua живе на ns.kntu.kr.ua і ns1.kntu.kr.ua, саме це він і відповідає DNS-клієнтові разом з – так-так – DS записом.

7. Так DNS-клієнт вибудував ланцюжок довіри до kntu.kr.ua, де він довідається сервери імен зони bar.kntu.kr.ua і її DS.

8. Зрештою DNS-клієнт ітеративно довідається адреси DNS серверів, відповідальних за bar.kntu.kr.ua, іде до них і нарешті-те одержує бажане, валідує всю інформацію й віддає нам адресний запис, виставивши у відповіді біт AD.

При неможливості щось провалідувати DNS-клієнт поверне servfail.

Надавані ефекти

Вихідний трафік авторитетного DNS сервера збільшується приблизно в 4 рази.

Розмір файлу зони після підпису (без OPT-OUT) виростає в 6-7 разів.

Збільшення довжини ключа приводить до помітного зниження qps DNS-клієнта, на авторитетний сервер впливає в меншому ступені.

Навпаки діє збільшення кількості ітерацій при використанні NSEC3.

DNSSec приводить до значного збільшення DNS відповіді й, отже, необхідно щоб все мережне встаткування коректно працювало з DNS пакетами більше 512 байт.

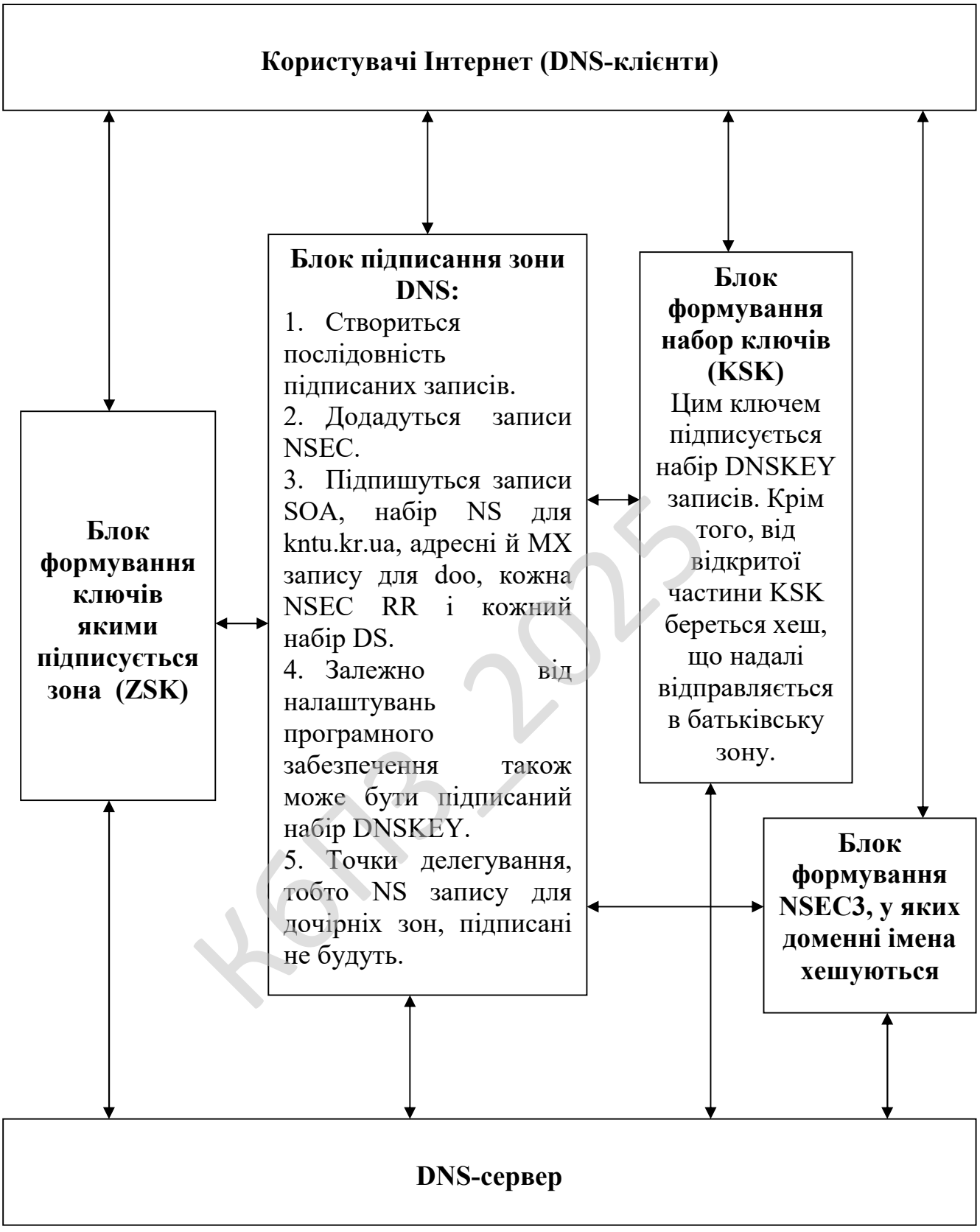

Рисунок 3.1 – Структурна схема система

Обґрунтування необхідності DNSSec

По-перше, треба враховувати створювані ефекти; по-друге, потрібно організувати надійне сховище для ключів; по-третє, стежити за ротацією ключів; по-четверте, стежити за строком придатності підписів; в-п'ятих, DNSSec підсилює ефект amplified ddos.

Все це є платою за те, щоб бути впевненим в інформації, одержуваній від авторитетних DNS серверів.

3.3 Розробка функціональної схеми

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

Розширення безпеки DNS, зроблені в DNSSEC (RFC 2535), пропонують служби, що виконують перевірку джерела даних і цілісності даних. Ці служби дозволяють шифрувати цифрові підписи із застосуванням закритих ключів і відправляти їх у вигляді записів ресурсів з DNS-сервера, на якому розміщені підписані (сумісні з DNSSEC) зони, до засобів дозволу, де дійсність записів ресурсів перевіряється із застосуванням відкритих ключів. Цифрові підписи й відкриті ключі додаються в підписану зону у вигляді записів ресурсів.

Важливо помітити, що розширення DNSSEC не призначені для усунення проблем безпеки на DNS-сервері; вони використовуються як метод захисту даних імені домену, що пересилаються за допомогою DNS. Закритий і відкритий ключі зв'язуються з конкретними зонами, а не з DNS-серверами, на яких ці зони розміщені. Якщо безпека DNS-серверів, на яких розміщаються зазначені зони, порушена, у засобів дозволу імен зберігається можливість безпечної перевірки дійсності записів ресурсів із цих зон.

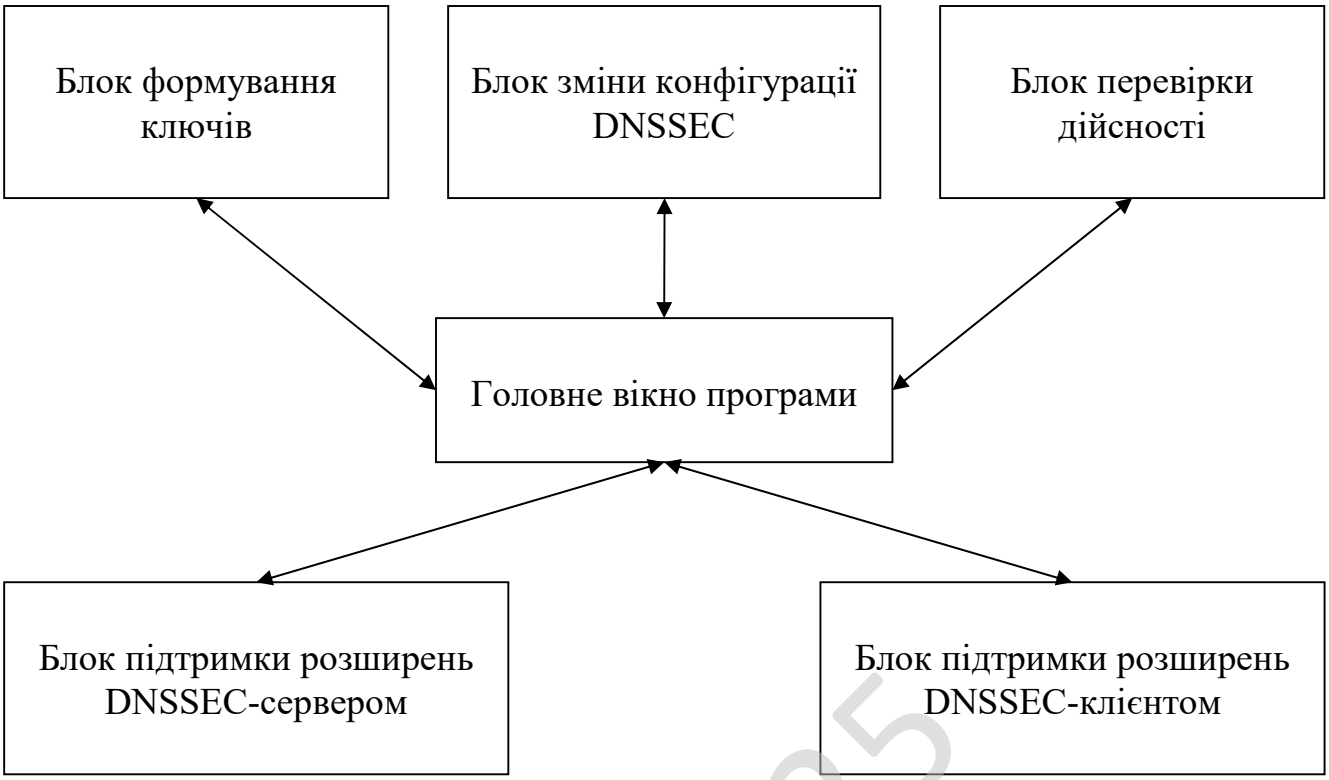

Рисунок 3.2 – Функціональна схема системи

Використання розширень безпеки DNS (DNSSEC)

Служба DNS системи Windows Server 2012 забезпечує базову підтримку протоколу розширень безпеки DNS (DNSSEC) відповідно до документа RFC 2535.

Підтримувані в цей час функції дозволяють DNS-серверам діяти в якості додаткових DNS-серверів для існуючих безпечних зон, сумісних з DNSSEC.

Служба DNS підтримує зберігання й завантаження специфічних DNSSEC записів ресурсів (RR).

У цей час DNS-сервер не здатний підписувати зони й записи ресурсів (створюючи криптографічні цифрові підписи) або перевіряти запису ресурсів SIG. У розширеннях DNSSEC застосовуються записи ресурсів KEY, SIG і NXT.

Підтримка розширень DNSSEC-сервером

При завантаженні зони, що містить запису ресурсів DNSSEC, DNS-сервер завантажує ці записи разом із записами інших типів, наявними в даній зоні. При

одержанні зонної передачі, що містить запису ресурсів DNSSEC (SIG, KEY, NXT), DNS-сервер заносить ці записи в сховище зони (файл дані зони або службу Active Directory) разом з іншими записами ресурсів.

Коли DNS-сервер одержує запит або відповідь, що містить записи ресурсів DNSSEC, він не перевіряє цифрові підписи, але кешує відповідь і використовує його для наступних запитів. Коли DNS-сервер одержує запит на запис ресурсів, що перебуває в зоні, що містить також запису ресурсів DNSSEC, він приєднує до відповіді відповідні записи DNSSEC.

Якщо в підписаній зоні є записи ресурсів для імені власника, у тому числі запис CNAME для зазначеного імені, DNS-сервер повертає запису ресурсів DNSSEC, пов'язані з іменем власника, і псевдонім запису ресурсу CNAME. DNS-сервер не перешкоджає поверненню запису ресурсу CNAME і не повертає запис ресурсу SIG для канонічного імені. Замість цього він повертає запис ресурсу SIG для псевдоніма.

Підтримка розширень DNSSEC-клієнтом

DNS-клієнт не виконує читання й збереження ключа для довіреної зони й, отже, не здійснює ніяких дій, пов'язаних із застосуванням криптографії, здійсненням перевірки дійсності або довіри. Коли засіб дозволу імен ініціює запит DNS і відповідь містить запису ресурсів DNSSEC, програми, виконувані на DNS-клієнті, повертають ці записи й кешують їх у такий же спосіб, як і будь-які інші записи ресурсів. Саме в такому ступені DNS-клієнти Windows 10; підтримують розширення DNSSEC. Коли DNS-клієнт одержує запис ресурсу SIG, що ставиться до набору записів ресурсів (RRset), він не надсилає додатковий запит на одержання пов'язаної з нею запису KEY або інших записів DNSSEC.

Засоби дозволу імен не проводять перевірку дійсності записів ресурсів шляхом перевірки інформації цифрового підпису, що втримується в записі ресурсу SIG. DNS-клієнт не містить ніякої інформації, що показує, для яких записів ресурсів була проведена перевірка дійсності й у якому ступені вони пройшли цю перевірку.

Коли засіб дозволу імен одержує відповідь або виконує операцію запиту, вона не розпізнає в заголовку запиту біт відключення перевірки (Checking Disabled, CD), що у розширеннях DNSSEC служить ознакою того, що перевірка дійсності даних проведена сервером відповідно до його політики, і не встановлює в заголовку запиту біт перевірки дійсності даних (Authentic Data, AD), що у розширеннях DNSSEC служить ознакою того, що неперевірені дані прийнятні для засобу дозволу імен.

Зміна конфігурації DNSSEC

Щоб змінити конфігурацію DNSSEC:

– Відкрийте редактор реєстру.

– У редакторі реєстру перейдіть у наступний розділ:

`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\DNS\Parameters`

– Додайте наступний запис DWORD: `EnableDnsSec`

Виконайте одну з дій.

– Щоб виключити запису ресурсів DNSSEC у відповідях на запити, відмінних від відповідей на запити записів ресурсів SIG, KEY або NXT, задайте значення 0x0. Відповідні записи ресурсів будуть включені тільки у відповіді на запити записів ресурсів SIG, KEY або NXT.

– Щоб включити записи ресурсів DNSSEC в усі відповіді на запити (відповідно до RFC 2535), задайте значення 0x2.

– Щоб включити записи ресурсів DNSSEC тільки в тих випадках, коли вихідний клієнтський запит містив запис ресурсу OPT (відповідно до RFC 2671), задайте значення 0x1 або зовсім не створюйте значення. DNS-сервер діє однаково, якщо задано значення 0x1 або якщо запис відсутня в реєстрі.

Для виконання даної процедури необхідно бути членом групи "Адміністратори" на локальному комп'ютері або одержати відповідні повноваження шляхом делегування. Якщо комп'ютер приєднаний до домену, цю процедуру можуть виконувати члени групи "Адміністратори домену". По міркуваннях безпеки для виконання цієї процедури рекомендується використовувати команду "Запуск від імені".

Щоб запустити редактор реєстру, натисніть кнопку Пуск, виберіть команду Виконати, уведіть regedit і натисніть кнопку ОК.

Значення запису реєстру EnableDnsSec визначає, включає або виключає DNS-сервер запису ресурсів DNSSEC при одержанні запитів.

Перевірка дійсності

У розширеннях DNSSEC для кожної зони є відкритий і закритий ключі, застосовувані для шифрування й розшифровки цифрових підписів. Зашифрованою, або безпечною, зоною називається зона DNS, що має як закритий, так і відкритий ключі. Якщо набір записів у зоні підписаний із застосуванням закритого ключа, засобу дозволу імен, що містять відкритий ключ цієї зони, можуть перевірити, чи був набір записів, отриманий від цієї зони, належним чином авторизований.

Завдяки застосуванню закритого ключа зони для підпису кожного набору записів ресурсів у цій зоні, для кожного імені домену в зазначеній зоні (наприклад, kntu.kr.ua) є закритий ключ. Цифровий підпис для набору записів додається в зону у вигляді нового запису ресурсу, SIG. Якщо DNS-сервер відповідає позитивно на запит імені DNS, він посилає запитані записи ресурсів і запис ресурсу SIG, що відповідає запитаному ім'ю. Засоби дозволу імен, що мають інформацію про відкритий ключ, пов'язаному із запитаним іменім, одержують запис ресурсу SIG і за допомогою відкритого ключа перевіряють правильність записів ресурсів. Відкритий ключ зони зберігається в новому типі запису ресурсів, KEY. Перш ніж засіб дозволу імен зможе перевірити дійсність записів ресурсів SIG, йому повинні бути надані записи ресурсу KEY.

Розширення DNSSEC перевіряють, чи дійсно запису, отримані засобом дозволу імен з безпечної зони, відправлені із цієї зони. За допомогою розширень DNSSEC засіб дозволу імен перевіряє, чи вірно, що IP-адреса домена kntu.kr.ua отримана з дійсної зони kntu.kr.ua.

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

3.4 Розробка діаграми процесів

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

– Інтерфейс ПЗ.

– Налаштування ПЗ.

– Налаштування параметрів передачі даних.

– Параметри налаштування з’єднання.

– Встановлення з'єднання з віддаленим ПК.

– Формування пакету даних для передачі.

– Шифрування пакету даних за протоколом SSL.

– Передача зашифрованого пакету даних по мережі.

– Прийом зашифрованого пакету даних.

– Розшифрування прийнятого пакету даних за протоколом SSL.

– Обробка прийнятого пакету даних.

Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування).

Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи. Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

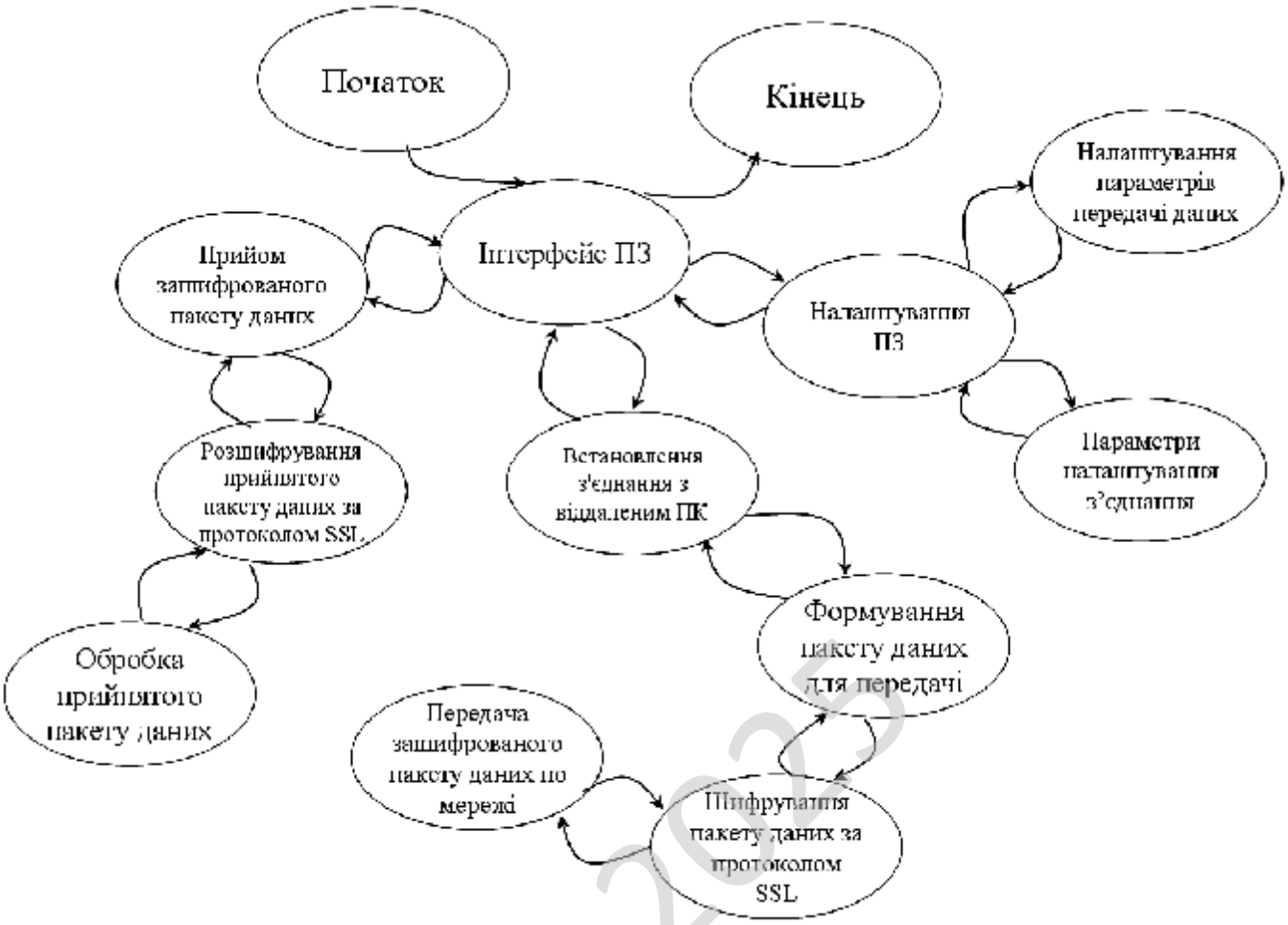

Рисунок 3.3 – Діаграма взаємодії процесів

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно розробка блок-схем. На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

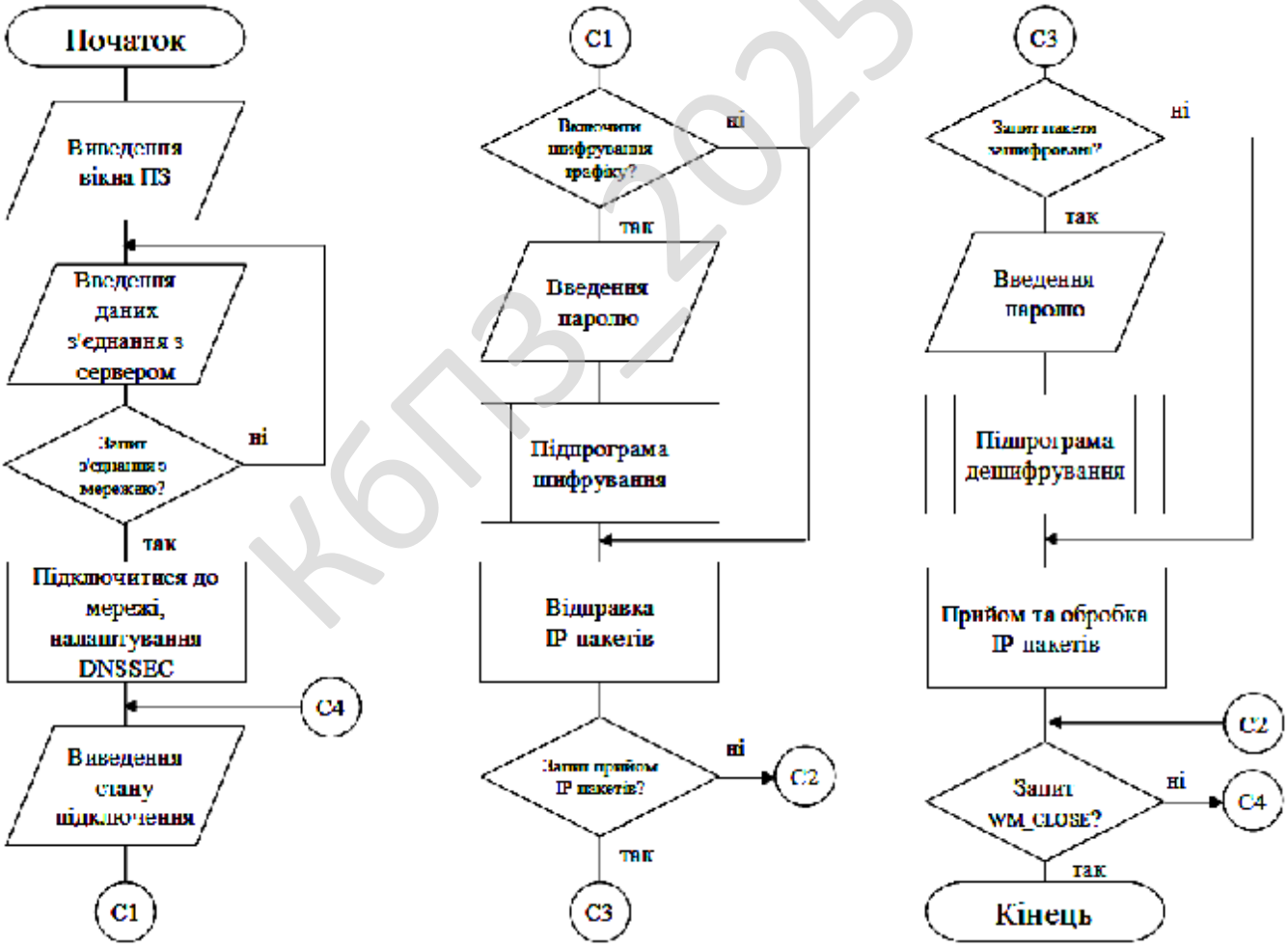

Рисунок 4.1 – Блок схема основної програми

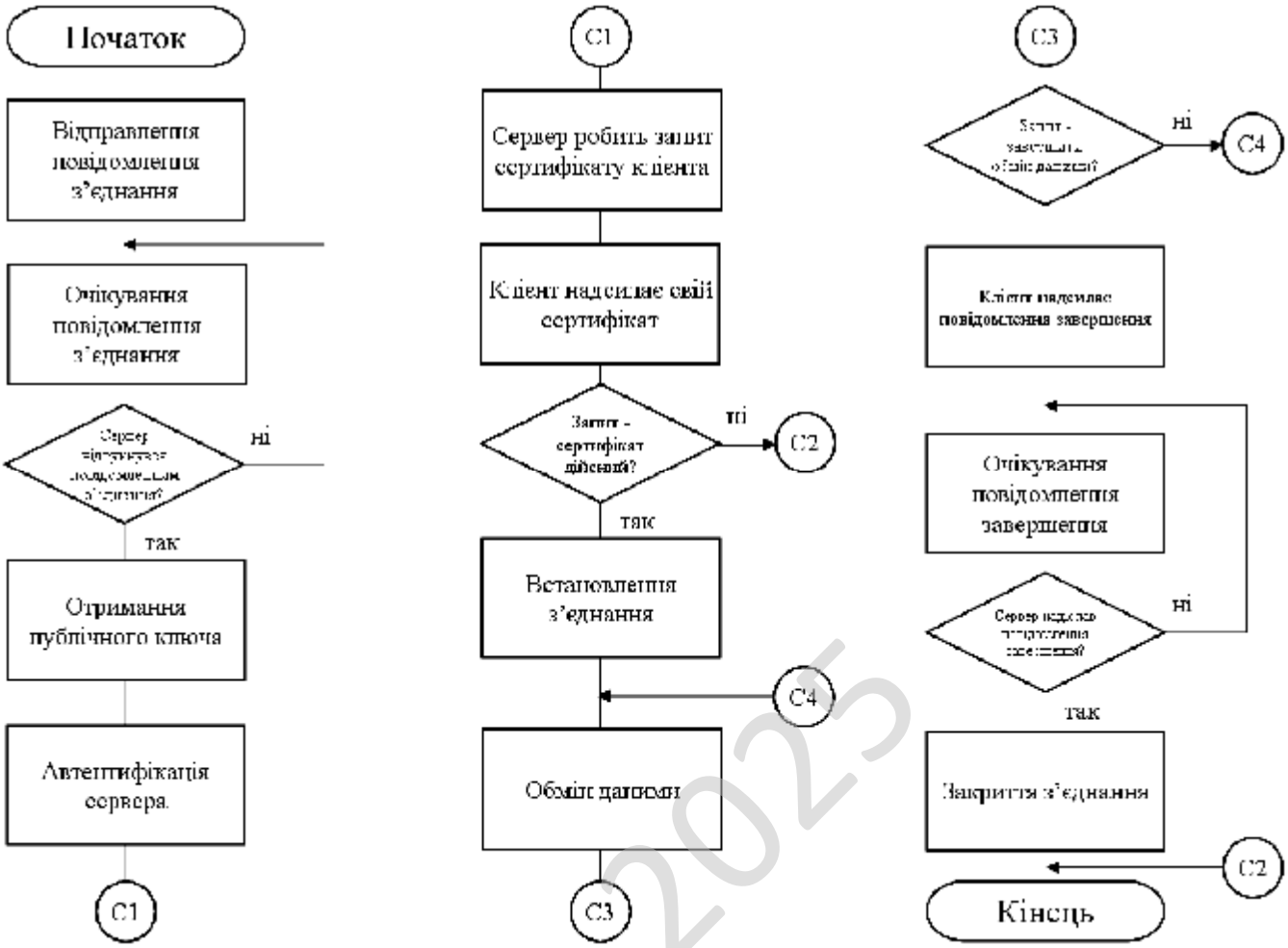

Рисунок 4.2 – Блок схема підпрограми

З якої видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ.

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

Опис алгоритмів функціонування системи

Розглянемо розробку графу діяльностей системи. При розробці використовувались концепції діаграм діяльності. Тобто в UML, візуальне

представлення графу діяльностей.

Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій.

Це фундаментальна одиниця визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів. Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії.

Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності.

Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності. Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

UML це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем.

UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація. UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм. Різні види діаграм які підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний код.

Основною причиною використання мови UML є спілкування розробників між собою. Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту.

UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об’єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки.

Практично усі CASE-засоби (програми автоматизації процесу аналізу і проектування) мають підтримку UML. Моделі розроблені в UML, дозволяють значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи. Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

– Керівникам проектів, які керують розподілом завдань і контролем за проектом.

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

– Бізнес-аналітикам, які досліджують реальну систему і здійснюють інжиніринг і реінжиніринг бізнесу компанії.

– Програмістам які реалізовують модулі інформаційної системи.

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності. Використання побудованої моделі при модифікаціях системи дає можливість усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

Розглянемо принципи роботи протоколу SSL та DNSSEC

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 аутентификації 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 4.3.

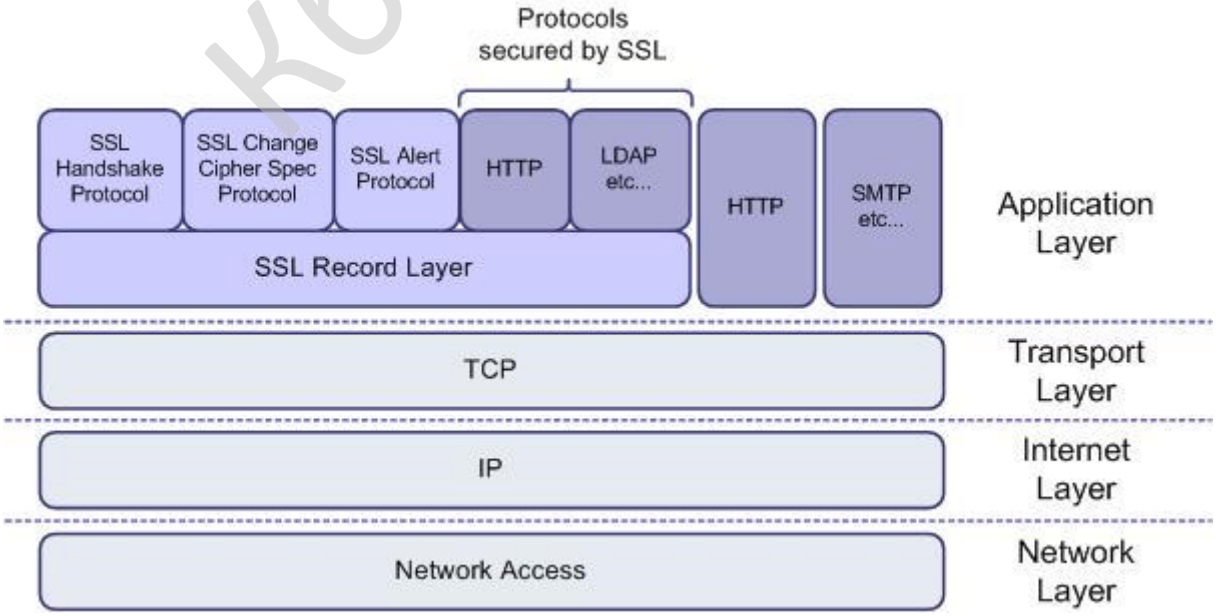

Рисунок 4.3 – Підпротоколи SSL

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 вигляді цей процес працює приблизно так, як показано на рисунку 4.4. 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.

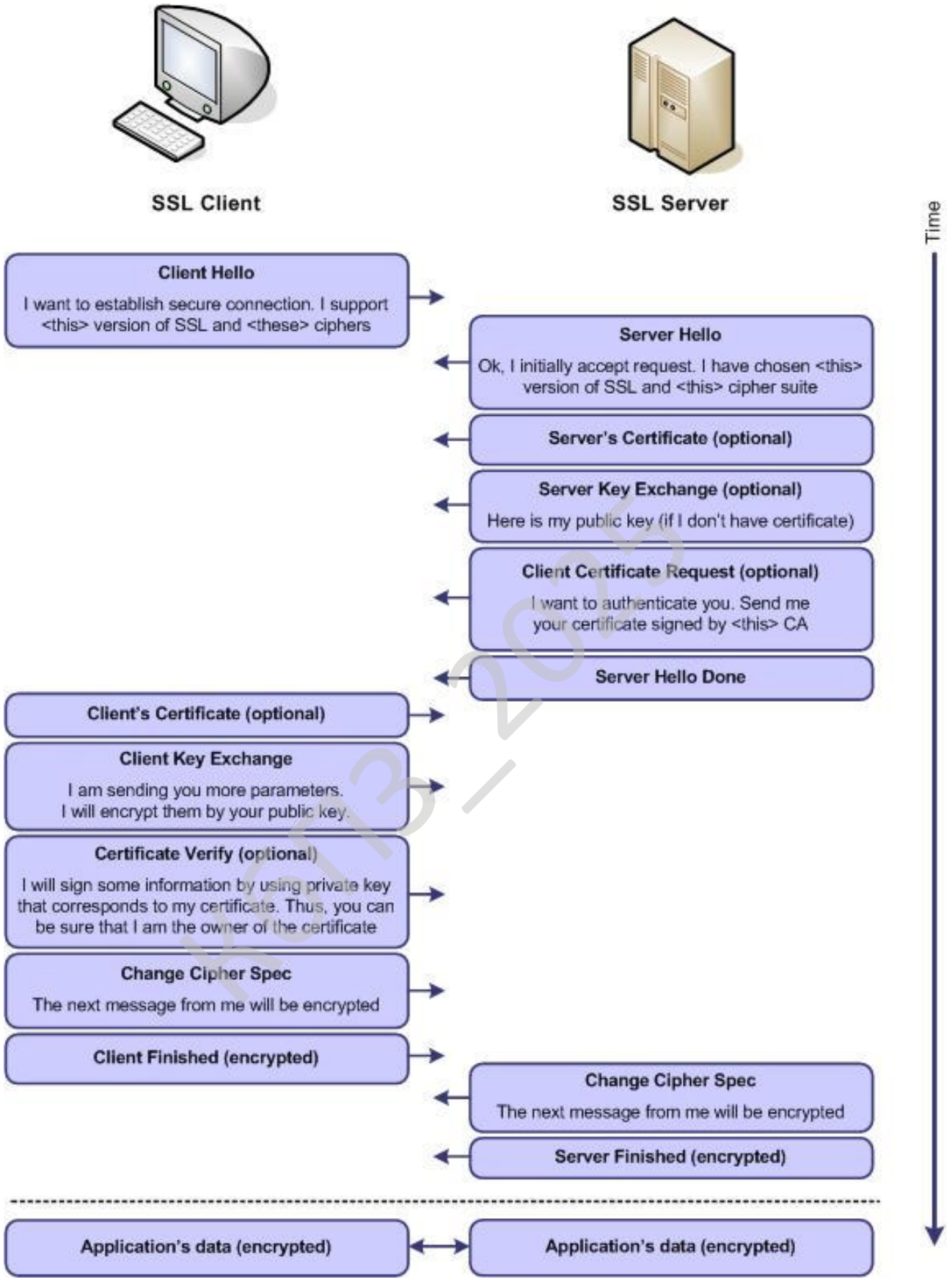

Рисунок 4.4 – Встановлення SSL з'єднань, крок за кроком

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).

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

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). Стан сесії характеризується рядом параметрів, наведених в табл. 4.1.

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 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.

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

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.

Розглянемо принципи роботи DNSSEC

DNSSEC (Domain Name System Security Extensions) – набір специфікацій IETF, що забезпечують безпеку інформації, що надається засобами DNS в IP-мережах.

Він забезпечує DNS-клієнтам (резолверам) аутентифікацію даних DNS або аутентифікацію інформації про факт відсутності даних та їх цілісність. Не забезпечується доступність даних і конфіденційність запитів.

Убезпечення інформації DNS особливо важливо для Інтернет-безпеки в цілому. Поширенню DNSSEC значною мірою перешкоджали:

– Відсутність сумісного з попередніми стандарту, який можна масштабувати до розміру Інтернету.

– Складність застосування реалізацій DNSSEC у величезній кількості DNS серверів і клієнтів.

– Розбіжності між ключовими гравцями з приводу того, хто повинен володіти доменами першого рівня (.com, .net тощо).

– Подолання передбачуваної складності та застосування DNSSEC.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою алгоритму FEAL – блоковий шифр, запропонований Акіхіро Симідзу і Седзі Міягуті.

У ньому використовуються 64-бітовий блок і 64-бітовий ключ. Його ідея полягає і в тому, щоб створити алгоритм, подібний DES, але з більш сильною функцією етапу. Використовуючи менше етапів, цей алгоритм міг би працювати швидше. На жаль, дійсність виявилася далекою від цілей проекту.

Як вхід процесу шифрування використовується 64-бітовий блок відкритого тексту. Спочатку блок даних підлягає операції XOR з 64 бітами ключа. Потім блок даних розщеплюється на ліву і праву половини. Об'єднання лівої і правої половин за допомогою XOR утворює нову праву половину. Ліва половина і нова права половина проходять через N етапів (спочатку 4). На кожному етапі половина об'єднується за допомогою функції F[1] з 16 бітами ключа і за допомогою XOR – з лівою половиною, створюючи нову праву

половину. Вихідна права половина (на початок етапу) стає новою лівою половиною. Після N етапів (ліва і права половини не переставляти після N-го етапу) ліва половина знову об'єднується з допомогою XOR з правою половиною, утворюючи нову праву половину, потім ліва і права об'єднуються разом в 64-бітове ціле. Блок даних об'єднується за допомогою XOR з іншими 64 бітами ключа і алгоритм завершується.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ яке зображено на рисунку 5.1. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні розділи: Меню програми; Вкладки підрозділів; Функціональні кнопки.

Розроблене програмне забезпечення реалізує метод створення самопідписаних SSL сертифікатів. Для системи передачі конфіденційних даних у мережі за протоколом DNSSEC. Сертифікат – спеціальний тип сертифіката, підписаний самим його автором. Технічно даний тип нічим не відрізняється від сертифіката, завіреного підписом центра сертифікатів, тільки замість передачі на підпис в центр сертифікатів користувач створює свою власну сигнатуру.

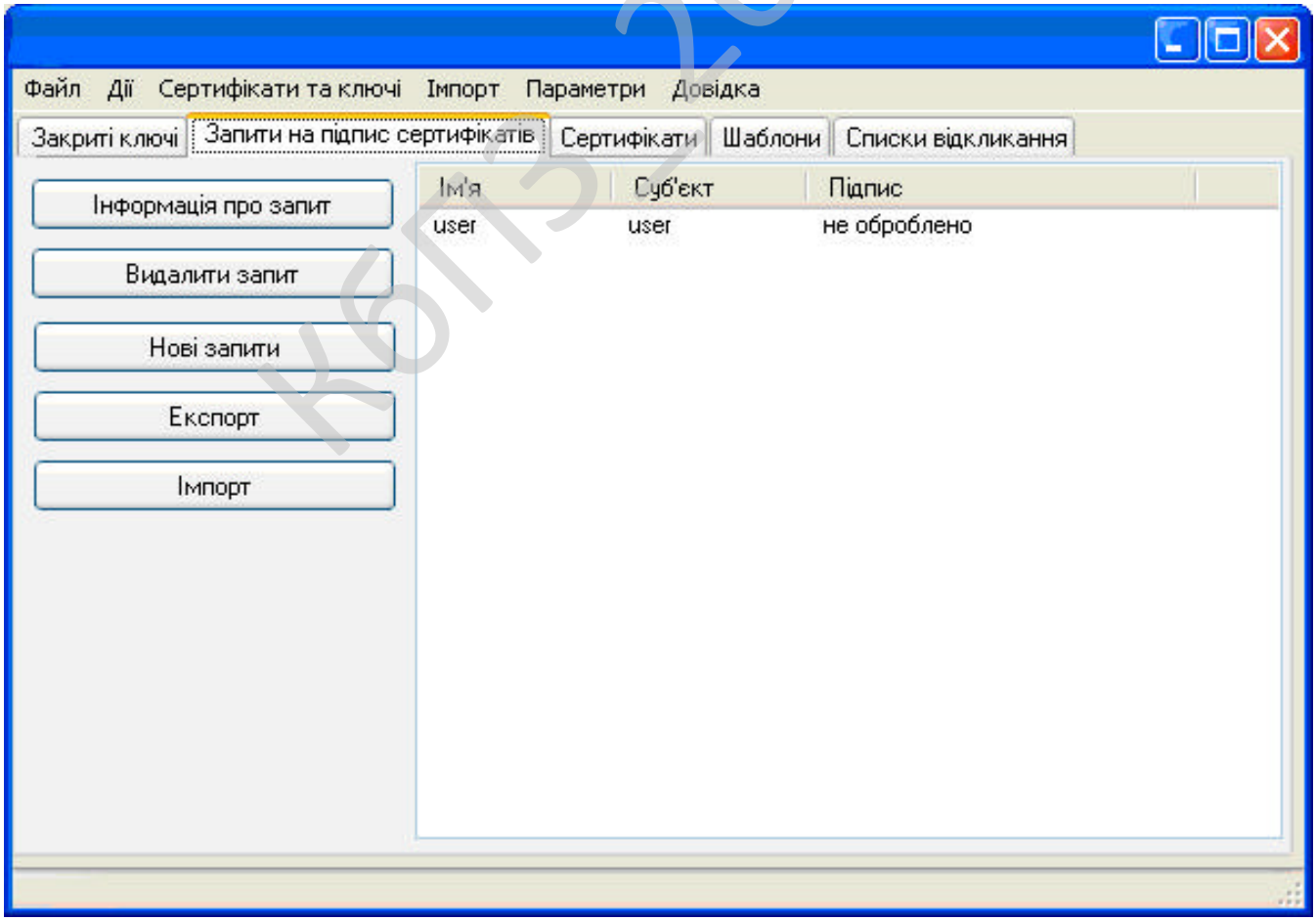

Рисунок 5.1 – Головне вікно ПЗ

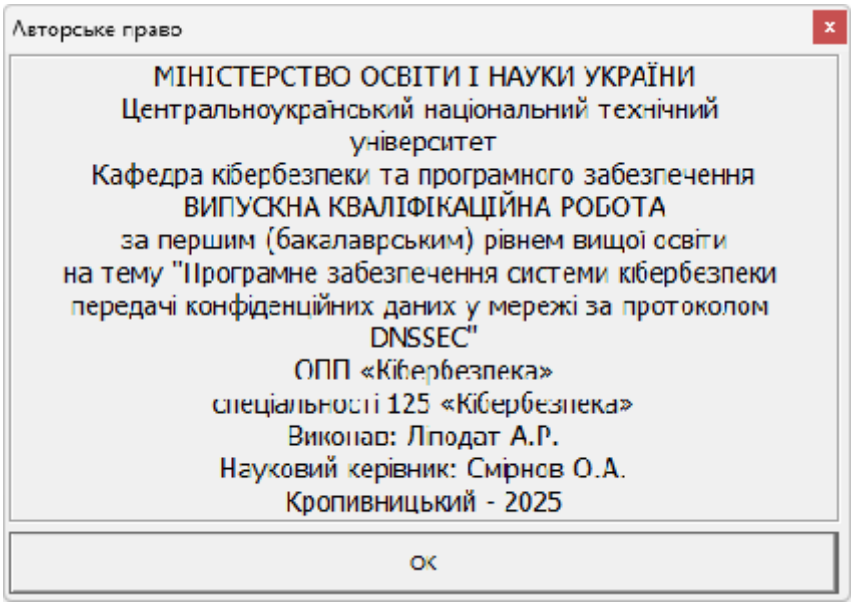

Рисунок 5.2 – Авторське право

Обрано умови розповсюдження – commercial software. Програмне забезпечення, створене комерційною організацією з метою отримання прибутку від його використання іншими, наприклад, шляхом продажу копій.

Далеко не завжди, але як правило терміни критично важливих змін в комерційних продуктах значно менше, ніж у некомерційних проектів. Це пов'язано з тим, що над комерційним продуктом працюють цілі групи розробників і ця робота є їх основним заняттям. Розробникам-початківцям як правило доводиться шукати додаткові способи заробітку, і це збільшує час, що витрачається на доповнення і зміни програм. Так як основним рушійним фактором створення комерційного ПЗ є одержання прибутку, то комерційні програмні продукти першими заповнюють вільні ніші та пропонують варіанти вирішення завдань відразу по мірі виявлення вакууму в будь-якому секторі ринку.

Окремий вид комерційних програм, коли їх розробка оплачується безпосередньо замовником. Такі програми найчастіше позбавлені всіх переваг комерційних продуктів, оскільки мають обмежений бюджет, але більш адаптовані до вимог замовника, ніж аналоги.

6 ОСНОВНІ ВИСНОВКИ

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

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 конфіденційних даних у мережі за протоколом DNSSEC.

– Досліджена система передачі конфіденційних даних у мережі за протоколом DNSSEC.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки передачі конфіденційних даних у мережі за протоколом DNSSEC.

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

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 конфіденційних даних у мережі за протоколом

DNSSEC. 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 Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм FEAL.

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

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

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

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

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

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

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

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

7. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in

software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56

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

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

10. 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,

11. 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.

12. 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.

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

14. Smirnov O., Kuznetsov A., Lokotkova I., Kuznetsova T., Florov S., Lebid O. «Using Orthogonal Signals to Hide Information in Images». *4 IEEE*

International Conference on Advanced Information and Communication Technologies (AICT) - 2021, Lviv, Ukraine, September 21-25, 2021. P. 255-260.

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

16. 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.*

17. 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.

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

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

20. 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.

21. 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.

22. 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.

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

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

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

26. 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.

27. 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.

28. 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.

29. 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.

30. 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.

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

32. 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.

33. 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.

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

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

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

37. Smirnov, O., Kuznetsov, A., Stefanovych, O., Gorbenko, Y., Krasnobaev, V., Kuznetsova K. «Information Hiding Using 3D-Printing Technology», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P.701-706.

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

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

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

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

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

43. 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.

44. 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.

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

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

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

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

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

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

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки передачі конфіденційних даних у мережі за протоколом DNSSEC.

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. № 57-02 від 17.01.2025 року).

3 Мета та призначення розробки

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

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 конфіденційних даних у мережі за протоколом DNSSEC;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище 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 – 53 аркуші.

8 Етапи розробки

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

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

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

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

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

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

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

_____ Смірнов О.А.

Програмне забезпечення системи кібербезпеки передачі конфіденційних даних у мережі за протоколом DNSSEC

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 35

Літера: РП

Кропивницький – 2025 року

Файл password_form_DNSSEC.cpp - вікно введення паролів

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

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

Файл password_form_DNSSEC.h - бібліотека для файлу password_form_DNSSEC.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
```

Основна программа

Файл main_form_DNSSEC.cpp основної програми

```
#include <vcl.h>
#pragma hdrstop
// ---------------------------------------------------------------------------
#include "main_form_DNSSEC.h"
#include "about_form_DNSSEC.h"
#include "password_form_DNSSEC.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-A 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 DNSSEC
        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_DNSSEC.h - бібліотека для файлу main_form_DNSSEC.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_DNSSEC.cpp", MainForm);
USEFORM("about_form_DNSSEC.cpp", AboutForm);
USEFORM("password_form_DNSSEC.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 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,
// генерируемое 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 DDNSSEC.cpp - шифрування DNSSEC

```
#pragma once

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

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

CDDNSSEC::CDDNSSEC(int DNSSECLen)
{
        this->UpdateProgress = 0;
            Initialize(DNSSECLen);
}

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

void CDDNSSEC::Initialize(int DNSSECLen)
{
      // Фіксуємо розмір хеша
      DNSSECGammaLen  = DNSSECLen;

      // Виділяємо пам'ять під властивості класу шифрування DNSSEC
      prgSourceData = new int[DNSSECGammaLen];
      prgDNSSEC       = new int[DNSSECGammaLen];

      // 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];
    }
}

CDDNSSEC::~CDDNSSEC()
{
```

```
	delete [] prgSourceData;

	delete [] prgDNSSEC;

	delete [] prgCodeTbl;
}

// XOR-Подібна операція для цілочислених операндів
inline int CDDNSSEC::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 CDDNSSEC::InitSubkey()
{
	for (int i = 0; i < INT_LEN; i++) // Ініціалізуємо структуру
	{
		subkey.Checksums.rgChecksum[i] = 0;
	}
}

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

// Обертання блоку вихідних даних
void CDDNSSEC::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 DNSSEC-Гами в його кінці,
		// (він був "витиснутий" уліво й з'явився праворуч)
		prgSourceData[sourceDataLen - 1] = temp;
	}
}

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

```
}

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

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

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

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

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

// Нормування індексу в масиві DNSSEC-гами
void CDDNSSEC::NormBegCode(int &begCode)
{
	begCode = HFDouble(begCode, DNSSECGammaLen, nTry2++);
}

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

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

		// Таблична підстановка
		prgDNSSEC[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 prgDNSSEC) з початку
            }
            i++;
      }
}

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

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

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

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

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

   IB = prgDNSSEC[DNSSECGammaLen - 1];
   PB = prgDNSSEC[0];
   prgDNSSEC[DNSSECGammaLen - 1] = Encode(IB, PB);
}

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

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

```
// Кодування DNSSEC-гами
void CDDNSSEC::DNSSECCode()
{
	int begCode = 0;
	iSubkey++;

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

	// Обчислюємо значення індексу початку кодування (BegCode)
	begCode += (unsigned char)(prgDNSSEC[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 DNSSEC-гами
	MoveCode();

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

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

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

	// Буфер ОС по даним підключа з контрольних сум

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

			j = 0;
		}

		// Блок кодування гами ключа за допомогою підключа

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

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

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

```
      }
}

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

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

      // Переносимо вихідні дані в масив int (він і буде шифруватися DNSSEC)
      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 prgDNSSEC
      PrepareSourceData(sourceLen);

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

            // Кодування DNSSEC-гами XOR-подібною операцією
            DNSSECCode();

            // Накладення гами підключа
            ImposeSubkeyGamma();

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

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

Файл DDNSSEC.h - бібліотека для файлу DDNSSEC.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 DNSSEC блоку даних
class CDDNSSEC
{
public:
      CDDNSSEC();
      CDDNSSEC(int DNSSECLen);
      CDDNSSEC(int DNSSECLen, void (*UpdateProgress)(int progress));
      virtual ~CDDNSSEC();

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

private:

      int DNSSECGammaLen;
      TSubkey subkey;

      int rgFeed[SUBKEY_LEN];

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

      void Initialize(int DNSSECLen);
      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 DNSSECCode();
        void SubkeyCode();
        void ImposeSubkeyGamma();
        void (*UpdateProgress)(int progress);
};
```