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

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2024 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки для управління файлами в ОС Windows на основі технології EFS”

Виконав здобувач вищої освіти
IV курсу, групи КБ-20
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Андрусик Б.М.
«____» _____ 2024 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Смірнов С.А.
«____» _____ 2024 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

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

Спеціальність *125 “*Кібербезпека*”*

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

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2024 року

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

Андрусику Богдану Миколайовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки для управління файлами в ОС Windows на основі технології EFS*

2. Керівник роботи *Смірнов Сергій Анатолійович, канд. техн. наук, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 135-02 від 01.04.2024 року

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

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи кібербезпеки для управління файлами в ОС Windows на основі технології EFS*

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

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи кібербезпеки в промислову експлуатацію.

6. Висновки

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

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

7. Дата видачі завдання « 17 » січня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2024 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2024 р.	
3.	Розробка моделі компонента	20.03.2024 р.	
4.	Розробка структур даних	25.03.2024 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2024 р.	
6.	Програмування алгоритмів	10.04.2024 р.	
7.	Оформлення ПЗ	17.04.2024 р.	
8.	Попередній захист роботи	23.05.2024 р.	

Дата видачі завдання
« 17 » січня 2024 р.

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

Завдання прийнято до виконання
« 17 » січня 2024 р.

Підпис здобувача
_____ Андрусик Б.М.
(прізвище та ініціали)

АНОТАЦІЯ

Андрусик Б.М. Програмне забезпечення системи кібербезпеки для управління файлами в ОС Windows на основі технології EFS. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2024.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки для управління файлами в ОС Windows на основі технології EFS.

Метою розробки є програмне забезпечення системи кібербезпеки для управління файлами в ОС Windows на основі технології EFS.

Результат роботи – програмна реалізація системи кібербезпеки для управління файлами в ОС Windows на основі технології EFS.

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 RAD Studio Delphi.

Ключові слова: кібербезпека, EFS

ABSTRACT

Andrusyk B.M. Cybersecurity system software for file management in Windows OS based on EFS technology. 125 Cyber security. Central Ukrainian National Technical University. Kropyvnytskyi. 2024.

In this final qualification work for the first (bachelor) level of higher education, software is developed, which is intended for the cyber security system for managing files in the Windows OS based on EFS technology.

The goal of the development is the software of the cyber security system for managing files in the Windows OS based on the EFS technology.

The result of the work is a software implementation of a cyber security system for file management in the Windows OS based on EFS technology.

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

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

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

The program was developed in the RAD Studio Delphi environment.

Keywords: cyber security, EFS

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 2
ВСТУП 3
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 5
1.1 Призначення системи 5
1.2 Область застосування 6
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 7
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти 7
2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки та мови програмування 13
2.3 Розгорнута постановка завдання 19
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 21
3.1 Опис функціонування системи 21
3.2 Розробка структурної схеми 30
3.3 Розробка функціональної схеми 34
3.4 Розробка діаграми процесів 40
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ 42
4.1 Розробка блок-схем та опис алгоритмів функціонування системи 42
4.2 Захист розробленого програмного забезпечення 63
5 ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ 65
6 ОСНОВНІ ВИСНОВКИ 69
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 71

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

НЖМД	–	накопичувач на жорсткому магнітному диску
ПЗП	–	постійний запам’ятовуючий пристрій
ЦП	–	центральний процесор
DES	–	симетричний алгоритм управління файлами з метою збереження конфіденційності,
EFS	–	система управління файлами з метою збереження конфіденційності, даних на диску
FSRTL	–	бібліотека часу виконання файлової системи
IPSec	–	протокол захисту даних
LPC	–	шина
NTFS	–	файлова система
PKI	–	інфраструктура відкритих ключів
RSA	–	асиметричний алгоритм управління файлами з метою збереження конфіденційності
WebDAV	–	протокол захисту даних

ВСТУП

Актуальність теми. Encrypting File System (EFS) – шифрована файлова система – це базова технологія що дозволяє управляти файлами, які зберігаються на томах з файловою системою NTFS, з метою збереження конфіденційності. Файлова система EFS інтегрована в NTFS, відрізняється простотою керування й стійкістю до атак. Користувачі можуть вибирати файли, якими потрібно управляти з метою збереження конфіденційності, але розшифровувати файли вручну перед використанням не потрібно – можна просто відкрити файл і змінити їх, як звичайно. Файли, управління якими реалізовано з метою збереження конфіденційності, будуть захищені навіть у тому випадку, якщо зловмисник одержить фізичний доступ до комп'ютера. Крім того, навіть користувачі, що мають право на доступ до комп'ютера (наприклад, адміністратори), не мають доступу до файлів, якими потрібно управляти з метою збереження конфіденційності, за допомогою файлової системи EFS іншими користувачами. Проте файлова система EFS підтримує призначені агенти відновлення. При їхньому правильному настроюванні можна гарантувати відновлення даних при необхідності. Файлова система EFS в Windows 10/11 і Windows Server 2022 була вдосконалена за рахунок наступних основних можливостей: підтримка зберігання ключів шифрування на смарт-картах; централізоване адміністрування політик захисту файлової системи EFS; шифрування клієнтського кеша (автономних файлів) для кожного користувача; шифрування системного файлу підкачування; спрощення відновлення ключів шифрування за допомогою майстра повторного створення ключів.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки для управління файлами в ОС Windows на основі технології EFS.

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

– Огляд існуючих систем для управління файлами в ОС Windows на основі технології EFS.

– Дослідження системи кібербезпеки для управління файлами в ОС Windows на основі технології EFS.

– Програмна реалізація системи кібербезпеки для управління файлами в ОС Windows на основі технології EFS.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі для управління файлами в ОС Windows на основі технології EFS.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки для управління файлами в ОС Windows на основі технології EFS, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система призначена для системи управління файлами в ОС Windows на основі технології EFS.

Шифрована файлова система (EFS) – це ефективний засіб управління файлами і папками, з метою збереження конфіденційності, на клієнтських комп'ютерах і віддалених файлових серверах, що дозволяє користувачам захищати дані від несанкціонованого доступу до них інших користувачів або зовнішніх зловмисників.

EFS корисно використовувати для управління файлами і папками, з метою збереження конфіденційності, на рівні користувача. Вона була вперше представлена в операційній системі Microsoft® Windows® 2000 і поліпшена в наступних випусках систем Windows.

Файлова система EFS може бути корисна для наступних груп користувачів:

– адміністраторів, фахівців із захисту даних, а також осіб, відповідальних за дотримання норм і правил, що забезпечують контроль доступу до конфіденційних відомостей;

– адміністраторів, відповідальних за сервери або портативні клієнтські комп'ютери з Windows 10/11®;

– користувачі спільно використовуваних комп'ютерів, що працюють із конфіденційними відомостями.

1.2 Область застосування

Перед впровадженням файлової системи EFS адміністраторам варто скласти план відновлення даних у випадку втрати ключів або сертифікатів. Файлова система EFS підтримує надійний механізм відновлення, у якому в даному випуску Windows реалізовані три великих зміни, зазначених нижче.

– Зміни агента відновлення ключів.

– Агент відновлення даних тепер може зберігатися на смарт-карті, що робить непотрібною автономну станцію відновлення й дає можливість віддаленого відновлення даних.

Ці зміни важливі в першу чергу для адміністраторів.

– Засіб ntbackup більше не входить до складу операційної системи. Замість цього в Windows Server® 2022 додана службова програма Robocopy, що дозволяє копіювати файли, управління якими реалізовано з метою збереження конфіденційності, файлової системи EFS без ключа розшифровки (копії, створювані таким чином, залишаються зашифрованими). Модуль SafeDocs у системі Windows Server 2022 підтримує резервне копіювання файлів шифрованої файлової системи.

Всі ці зміни можуть значно змінити план розгортання файлової системи EFS.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки для управління файлами в ОС Windows на основі технології EFS, 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

Advanced EFS Data Recovery

Програма призначена для одержання доступу до файлів і папок, управління якими з метою збереження конфіденційності, відбувається з використанням засобів Encrypting File System (EFS). Advanced EFS Data Recovery (AEFSDR) відновить доступ до даних, захищених EFS, в Windows 2000, XP, Server 2003/2008, Vista, Windows 10/11. У випадку, якщо диск із даними встановлений на іншому комп'ютері, відформатований або якщо деякі ключі шифрування ушкоджені, за допомогою AEFSDR ви однаково зможете відновити дані. Microsoft Encrypting File System (EFS) є складовою частиною NTFS і доступна в багатьох сучасних версіях операційної системи Windows. EFS здійснює повне й прозоре для користувача шифрування файлів, забезпечуючи тим самим надійний захист від несанкціонованого доступу до файлів навіть у тому випадку, коли зловмисник заволодів комп'ютером або дисками, на яких зберігаються зашифровані дані. Втратити можливість доступу до даних, захищеним EFS, досить просто. Втрата можлива у випадку установки Windows поверх старої системи, переформатування або зміни розділів диска, а також при установці диска із захищеними даними в інший комп'ютер. Advanced EFS Data Recovery ефективно відновлює захищені засобами EFS файли навіть у тих випадках, коли всі інші методи розшифровки й відновлення не діють. Сканування жорсткого диска на низькому рівні й перегляд збережених даних сектор за сектором дозволяє Advanced EFS Data Recovery відновлювати файли, управління якими реалізовано з метою збереження конфіденційності, й папки навіть у тому

випадку, коли деякі ключі шифрування ушкоджені або безповоротно загублені. AEFSDR допомагає вирішувати проблеми, що виникли внаслідок помилок адміністрування EFS, таких як видалення облікових записів користувачів, відсутність агентів відновлення (Data Recovery Agents) або їхнє неправильне конфігурування, некоректний перенос облікових записів в інший домен, а також перенос дисків із зашифрованими даними між комп'ютерами.

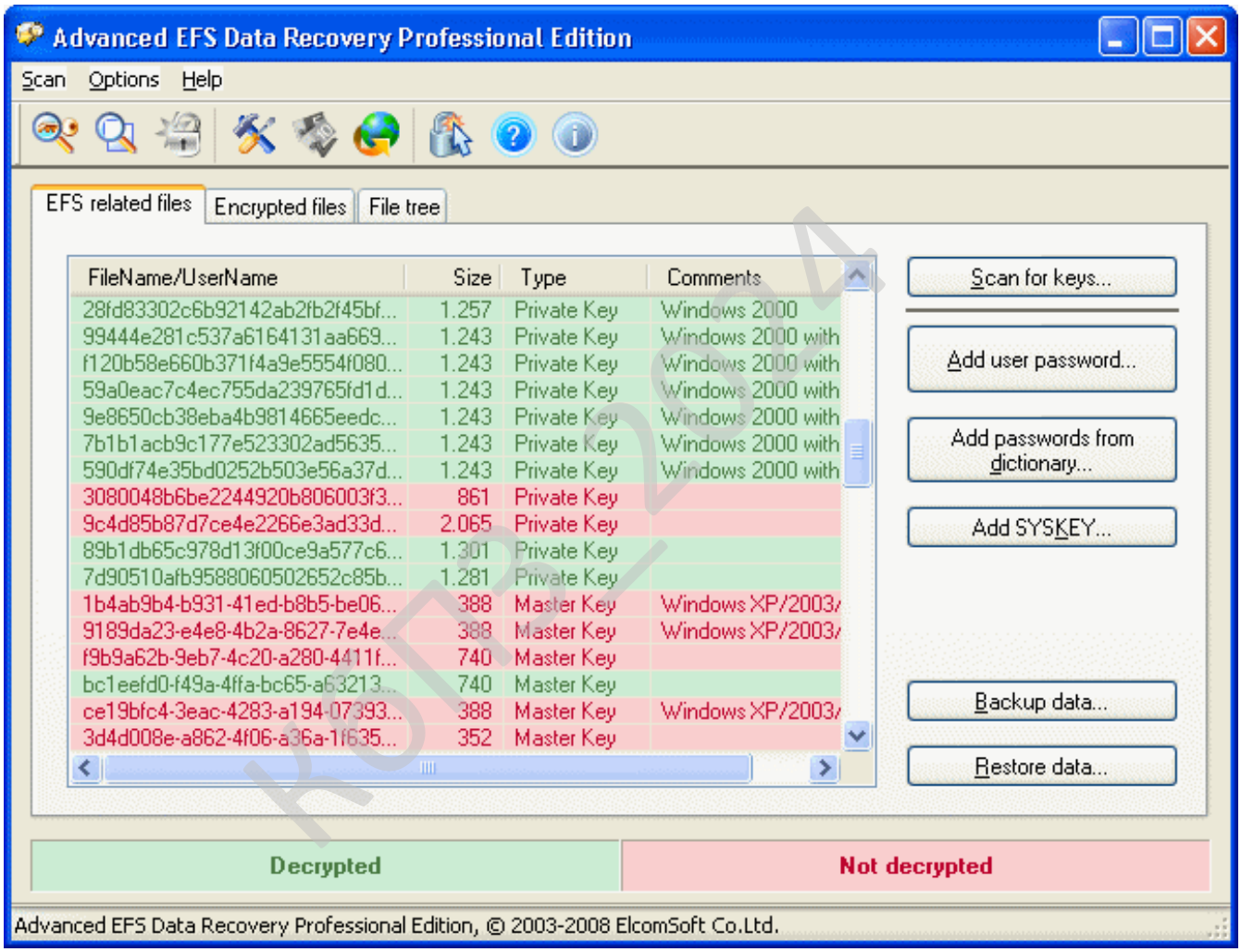

Рисунок 2.1 – Інтерфейс користувача Advanced EFS Data Recovery

Програма Advanced EFS Data Recovery працює на операційних системах:

– Windows 11

– Windows 10.

– Windows 8.

– Windows 7 x64.

– Windows 7.

Інтерфейс програми Advanced EFS Data Recovery доступний на українській, англійській, німецькій мовах. Ви можете скачати українську й англійську версії програми Advanced EFS Data Recovery зовсім безкоштовно.

Виконання переносу файлів і сертифікатів шифрованої файлової системи (EFS)

Щоб перенести шифровані файли, необхідно змінити поводження за замовчуванням одним з описаних нижче способів.

Якщо на кінцевому комп'ютері встановлена операційна система Windows Vista, перенос сертифікатів шифрованої файлової системи (EFS) буде виконаний автоматично. Однак за замовчуванням у випадку виявлення шифрованого файлу відбудеться збій засобу переносу користувальницького середовища (якщо не заданий параметр /efs). Таким чином, щоб перенести шифровані файли, при запуску програми ScanState необхідно вказати параметр /efs:copyraw. Потім при виконанні програми LoadState на кінцевому комп'ютері відбудеться автоматичний перенос шифрованого файлу й сертифіката файлової системи EFS.

Щоб перенести шифровані файли на комп'ютери з операційною системою Windows XP, необхідно також перенести сертифікат шифрованої файлової системи (EFS). За замовчуванням у випадку виявлення шифрованого файлу відбудеться збій засобу переносу користувальницького середовища (якщо не заданий параметр /efs). Щоб перенести шифровані файли, необхідно змінити поводження за замовчуванням. При переносі сертифікатів за допомогою засобу переносу користувальницького середовища потрібне втручання користувача як до, так і після переносу.

Перенести сертифікати EFS можна за допомогою програми Cipher.exe або оснащення «Сертифікати»:

– Виконання переносу сертифікатів EFS за допомогою програми Cipher.exe.

– Виконання переносу сертифікатів EFS за допомогою оснащення «Сертифікати».

Виконання переносу сертифікатів EFS за допомогою програми Cipher.exe

Щоб перенести сертифікати шифрованої файлової системи (EFS) за допомогою програми Cipher.exe, користувач, що є власником сертифіката, повинен виконати описану нижче процедуру.

1. Користувач, що є власником сертифіката, повинен увійти на вихідний комп'ютер і закрити всі програми.

2. Потім варто запустити програму Cipher.exe з командного рядка, використовуючи наступний синтаксис:

cipher /x [:Шлях_до_файлу_EFS] [Ім'я_файлу],

де Ім'я_файлу являє собою ім'я файлу без розширення, а Шлях_до_файлу_EFS – шлях до шифрованого файлу. Якщо задати значення параметра Шлях_до_файлу_EFS, буде створена резервна копія сертифіката (або сертифікатів) користувача, що використовувався для шифрування файлу. У противному випадку буде створена резервна копія поточного сертифіката EFS і EFS-Ключів. За вказівкою користувача програмою Cipher буде створений захищений паролем PFX-файл. Для одержання додаткових відомостей про програму Cipher.exe у командному рядку наберіть cipher /?.

3. Як тільки сертифікат збережений, адміністратор може зібрати дані про стан користувача за допомогою параметра командного рядка /efs:copyraw програми Scanstate. Наприклад:

scanstate \ \ fileserver \ migration \ mystore /efs:copyraw /i:migapp.xml /i:migsys.xml /i:miguser.xml /v:13/targetxp

4. Потім при необхідності адміністратор повинен установити на кінцевому комп'ютері операційну систему Windows XP і додатка, після чого

відновити користувальницьке середовище на кінцевому комп'ютері за допомогою програми LoadState.

5. Після завершення переносу користувач повинен увійти на кінцевий комп'ютер, перейти до резервної копії сертифіката й двічі клацнути PFX-файл.

6. Відновити сертифікат EFS можна за допомогою майстра імпорту. Користувачеві необхідно вказати свій пароль для сертифіката. Після завершення роботи майстри імпорту сертифікат буде відновлений.

Виконання переносу сертифікатів EFS за допомогою оснащення «Сертифікати»

Щоб перенести сертифікати шифрованої файлової системи (EFS) за допомогою оснащення «Сертифікати», користувач, що є власником сертифіката, повинен перед переносом експортувати його з вихідного комп'ютера. Після переносу користувач повинен імпортувати цей сертифікат на кінцевий комп'ютер. Якщо не виконати одну із цих процедур, файл на кінцевому комп'ютері буде як і раніше зашифрований.

Експорт сертифіката з вихідного комп'ютера

1. Користувач, що є власником сертифіката, повинен увійти на вихідний комп'ютер.

2. Потім у діалоговому вікні **Виконати** введіть **mmc**, щоб відкрити консоль керування MMC.

3. У меню **Файл** виберіть команду **Додати або видалити оснащення**.

4. Потім у діалоговому вікні **Додати або видалити оснащення** натисніть кнопку **Додати**.

5. У списку виберіть пункт **Сертифікати**, натисніть кнопку **Додати**, потім виберіть пункт **Мій обліковий запис**.

6. Натисніть кнопку **Готово**, виберіть **Закрити**, потім натисніть кнопку **ОК**.

7. Перейдіть до вузла **Сертифікати – Поточного користувача \ Особисті \ Сертифікати**.

8. Правою кнопкою миші клацніть сертифікат, якому необхідно перенести.

9. Виберіть пункт **Всі завдання**, а потім виберіть **Експорт**.

10. За допомогою **майстра експорту сертифікатів** можна зберегти сертифікат у такому місці, до якого можливий доступ з кінцевого комп'ютера (наприклад, на дискеті або в загальній папці). У відповідь на запит потрібно вказати, що закритий ключ варто експортувати разом із сертифікатом. По завершенні буде виведене повідомлення про успішне виконання експорту.

11. Тепер адміністратор може зібрати дані про стан користувача за допомогою параметра командного рядка /efs:copyraw програми Scanstate. Наприклад:

scanstate \ \ fileserver \ migration \ mystore /efs:copyraw /i:migapp.xml /i:migsys.xml /i:miguser.xml /v:13/targetxp

12. Потім при необхідності адміністратор повинен установити на кінцевому комп'ютері операційну систему Windows XP і додатка, після чого відновити стан користувача на кінцевому комп'ютері за допомогою програми LoadState.

Імпорт сертифіката на кінцевий комп'ютер

1. Користувач, що є власником сертифіката, повинен увійти на кінцевий комп'ютер.

2. Потім у діалоговому вікні **Виконати** введіть **mmc**, щоб відкрити консоль керування MMC.

3. У меню **Файл** виберіть команду **Додати або видалити оснащення**.

4. Потім у діалоговому вікні **Додати або видалити оснащення** натисніть кнопку **Додати**.

5. Зі списку виберіть пункт **Сертифікати**, натисніть кнопку **Додати**, потім виберіть пункт **Мій обліковий запис**.

6. Натисніть кнопку **Готово**, виберіть **Закрити**, потім натисніть кнопку **ОК**.

7. Перейдіть до вузла **Сертифікати – Поточного користувача \ Особисті**.

8. Правою кнопкою миші клацніть **Особисті**.

9. Виберіть пункт **Всі завдання**, а потім виберіть **Імпорт**.

10. Знайти експортований сертифікат можна за допомогою **майстра імпорту сертифікатів**. При пошуку сертифіката в списку, що **розкривається, Файли** типу варто вибрати пункт **файли обміну особистою інформацією (*.p12, *.pfx)**. Буде необхідно ввести пароль, заданий при експорті сертифіката з вихідного комп'ютера.

11. По завершенні буде виведене повідомлення про успішне виконання імпорту.

2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки та мови програмування

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

Delphi 10.4 Sydney

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

виправлень багів, поліпшення продуктивності середовища й бібліотек і багато чого крім того.

Основні можливості Delphi 10.4.1:

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

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

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

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

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

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

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

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

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

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

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

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

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

RAD Studio 10.4 Короткий огляд:

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

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

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

Істотне поліпшення Delphi Code Insight

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

Delphi Custom Managed Records

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

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

Єдине керування пам'яттю

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

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

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

Розширена підтримка бібліотек C++

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

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

Win 64-відладник і збирач для C++

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

Підвищення якості й швидкодії інструментів

- Велика кількість поліпшень STL від Dinkumware.
- Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.
- Поліпшена підтримка Cmake.
- Велика кількість виправлень для підвищення стабільності і якості.
- Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.
- Загальні вдосконалення в бібліотеці доступу до БД FireDAC, включаючи оновлені драйвера для FireBird, PostgreSQL і SQLite. Вибір статичного або динамічного підключення SQLite до застосунку.

Змінені стилі VCL для High DPI

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

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

Нові High DPI стилі й стилізація окремих VCL компонент

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

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

Поліпшена кроссплатформеність

- Додана підтримка Metal Driver GPU для macOS і iOS.
- Крім підтримки останнього iOS SDK, в RAD Studio 10.4 розроблювачі можуть задовольнити нові вимоги Apple до набору стартових екранів.
- Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.
- Користувачам редакцій Enterprise або Architect доступна повна інтеграція Fmxlinux з IDE для створення клієнтських застосунків Linux з GUI.
- Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.
- Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

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

Універсальний інсталятор для установки Online і Offline

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

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

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи кібербезпеки для управління файлами в ОС Windows на основі технології EFS.

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

В Windows Server® 2022 у файловій системі EFS реалізовані деякі важливі вдосконалення. У їхнє число входять:

– можливість зберігати сертифікати управління файлами з метою збереження конфіденційності, на смарт-картах;

– управління файлами з метою збереження конфіденційності, у клієнтському кеші для окремих користувачів;

– додаткові параметри групової політики;

– новий майстер повторного створення ключів.

Зберігання ключів на смарт-картах

Ключі й сертифікати управління файлами з метою збереження конфіденційності, файлової системи EFS можна зберігати на смарт-картах, що забезпечує більш надійний їхній захист. Це особливо корисно для захисту портативних комп'ютерів і робочих станцій із загальним доступом. Використання смарт-карт для зберігання ключів управління файлами з метою збереження конфіденційності, дозволяє також у деяких випадках поліпшити керування ключами у великих організаціях.

При зберіганні ключів файлової системи EFS на смарт-карті немає необхідності зберігати їх на жорсткому диску комп'ютера. Це поліпшує захист ключів, не дозволяючи іншим користувачам або зловмисникам, які можуть украсти комп'ютер, організовувати атаки на них.

Відмінні риси

В Windows Server 2022 і Windows 10/11 файлова система EFS підтримує зберігання закритих ключів користувачів на смарт-картах.

Кешування ключів

За допомогою параметрів групової політики можна настроїти файлову систему EFS для зберігання закритих ключів на смарт-картах у режимі без кешування або з кешуванням.

– Режим без кешування. Подібно традиційному способу роботи файлової системи EFS, у цьому режимі всі операції розшифровки, що вимагають використання закритого ключа користувача, виконуються на смарт-карті.

– Режим з кешуванням. У цьому режимі на основі закритого ключа користувача створюється симетричний ключ, кешуємий у захищеній пам'яті. Операції управління файлами з метою збереження конфіденційності, й розшифровки, виконувані за допомогою ключа користувача, заміняються відповідними симетричними криптографічними операціями з використанням похідного ключа. Це усуває необхідність підключення смарт-карти або використання процесора смарт-карт для кожної операції розшифровки, що значно підвищує продуктивність роботи.

Крім того, файлова система EFS надає політики, що дозволяють задавати необхідність застосування смарт-карт, а також управляти параметрами ключів користувачів і їх кешуванням.

Єдиний вхід за допомогою смарт-карти

Єдиний вхід за допомогою смарт-карти ініціюється в тих випадках, коли користувач намагається ввійти в систему за допомогою смарт-карти й при цьому виконується одна з наступних умов:

– у користувача немає на комп'ютері дійсного ключа управління файлами з метою збереження конфіденційності, файлової системи EFS, а параметри політики вимагають застосування смарт-карт для файлової системи EFS;

– у користувача є дійсний ключ управління файлами з метою збереження конфіденційності, файлової системи EFS, що зберігається на смарт-карті, за допомогою якої здійснюється вхід у систему.

Коли ініціюється єдиний вхід у систему, файлова система EFS кешує ПІН-код, уведений користувачем при вході в систему, а також використовує його для виконання своїх операцій. Тому в ході сеансу користувач не одержує від файлової системи EFS запитів ПІН-коду.

Якщо смарт-карта, за допомогою якої користувач увійшов у систему, витягає з модуля читання смарт-карт до виконання яких-небудь операцій управління файлами з метою збереження конфіденційності, єдиний вхід відключається. При спробі виконати першу операцію файлової системи EFS користувач одержує вказівку вставити смарт-карту й увести ПІН-код.

Підготовка до даної зміни

Щоб підготуватися до використання смарт-карт для зберігання сертифікатів EFS, варто вивчити наявну інфраструктуру відкритих ключів і спланувати використання в ній сертифікатів EFS. Якщо в організації не розгорнута інфраструктура відкритих ключів, використовувати смарт-карти для зберігання сертифікатів EFS неможливо.

Управління файлами з метою збереження конфіденційності, автономних файлів для окремих користувачів

Автономні копії файлів, що перебувають на віддалених серверах, також можна шифрувати з використанням файлової системи EFS. Якщо цей параметр включений, кожний файл, що перебуває в автономному кеші, шифрується за допомогою відкритого ключа користувача, що виконав кешування файлу. Таким чином, тільки цей користувач має доступ до файлу, і навіть локальні адміністратори не можуть прочитати файл без доступу до закритих ключів користувача.

Значення цієї можливості

Управління файлами з метою збереження конфіденційності, для окремих користувачів зміцнює безпека. Раніше будь-який користувач комп'ютера міг потенційно одержати доступ до будь-якого файлу, що перебуває в автономному кеші.

Відмінні риси

У минулому, управління файлами з метою збереження конфіденційності, виконувалося за допомогою системних ключів, через що користувачі могли читати автономні файли інших користувачів. Тепер це не так, тому що управління файлами з метою збереження конфіденційності, виконується за допомогою відкритого ключа кожного користувача.

Підготовка до даної зміни

Ознайомтеся з новими параметрами файлової системи EFS і виберіть ті з них, які відповідають вимогам організації до безпеки.

Розширені можливості настроювання файлової системи EFS з використанням групової політики

Використовуючи групову політику, можна централізовано контролювати й набудовувати політики захисту файлової системи EFS для всієї організації.

Щоб допомогти адміністраторам визначати й впроваджувати організаційні політики для файлової системи EFS, у групову політику були додані деякі нові параметри. У їхнє число входять параметри, що дозволяють задати необхідність використання смарт-карт для файлової системи EFS і необхідність управління файлами з метою збереження конфіденційності, файлу підкачування, указати мінімальну довжину ключів для файлової системи EFS, забезпечити управління файлами з метою збереження конфіденційності, папки документів користувача й заборонити використання сертифікатів, що самозаверяють.

Значення цієї можливості

Розширені можливості настроювання роблять роботу адміністраторів більше ефективної, дозволяючи їм набудовувати й контролювати політики файлової системи EFS у масштабі всієї організації.

Відмінні риси

Додаткові параметри підвищують ефективність групової політики.

Підготовка до даної зміни

Ознайомтеся з новими параметрами файлової системи EFS у груповій політиці й виберіть ті з них, які відповідають вимогам організації до безпеки.

Майстер повторного створення ключів файлової системи EFS

Майстер повторного створення ключів файлової системи EFS дозволяє користувачеві вибрати сертифікат для файлової системи EFS і перенести наявні файли, які будуть використовувати обраний сертифікат. З його допомогою можна також перевести користувачів з використання сертифікатів програм на смарт-карты. Адміністратори й самі користувачі можуть також використовувати цей майстер для відновлення даних. Цей спосіб ефективніше, ніж розшифровка й повторне управління файлами з метою збереження конфіденційності.

Значення цієї можливості

Майстер повторного створення ключів файлової системи EFS дозволяє використовувати просту покрокову процедуру вибору сертифікатів або переносу файлів.

Відмінні риси

При відкритті або відновленні файлів їхнє повторне управління файлами з метою збереження конфіденційності, автоматично не виконується. Майстер повторного створення ключів забезпечує користувачам високий ступінь гнучкості.

Підготовка до даної зміни

Натисніть на тестовому комп'ютері кнопку **Пуск**. У поле **Почати пошук** уведіть команду **rekeywiz** і натисніть клавішу УВЕДЕННЯ. Буде запущений майстер повторного створення ключів файлової системи EFS, з роботою якого можна буде ознайомитися.

Нові й змінені параметри

В описуваному випуску Windows Server 2022 додатковими параметрами файлової системи EFS можна управляти за допомогою групової політики. Параметри групової політики, описані в наведеній нижче таблиці, доступні в адміністративних шаблонах.

У таблиці 3.1 наведені прості описи параметрів.

Таблиця 3.1 – Опис параметрів файлової системи EFS

Шаблон і параметр	Шлях і опис	Значення за замовчуванням
GroupPolicy.admx – обробка політики відновлення EFS	Конфігурація комп'ютера \ Адміністративні шаблони \ Система \ Групова політика – визначає порядок відновлення політик управління файлами з метою збереження конфіденційності.	Не налаштований
EncryptFilesonMove.admx – не виконувати автоматичне управління файлами з метою збереження конфіденційності, переміщуваних у зашифровані папки.	Конфігурація комп'ютера \ Адміністративні шаблони \ Система \ – забороняє провідникові виконувати автоматичне управління файлами з метою збереження конфіденційності, переміщуваних у зашифровані папки.	Не налаштований
OfflineFiles.admx – шифрувати кеш автономних файлів.	Конфігурація комп'ютера \ Адміністративні шаблони \ Мережа \ Автономні файли \ – ця політика визначає, чи буде виконуватися управління файлами з метою збереження конфіденційності, автономних файлів.	Не налаштований
Search.admx – дозволити індексування шифрованих файлів.	Конфігурація комп'ютера \ Адміністративні шаблони \ Компонента Windows \ Пошук \ – цей параметр дозволяє індексування шифрованих файлів пошуком Windows.	Не налаштований

За допомогою консолі керування груповими політиками або локального редактора групових політик (secpol.msc) можна також настроїти параметри файлової системи EFS, зазначені нижче. Щоб переглянути або змінити ці параметри, розгорніть вузол **Політики відкритого ключа**, клацніть правою кнопкою миші елемент **Шифрована файлова система** й виберіть пункт **Властивості**. На вкладці **Загальні** можна настроїти загальні параметри й параметри сертифікатів. Доступні загальні параметри зазначені в таблиці 3.2.

Таблиця 3.2 – Загальні параметри

Параметр	Примітки	Значення за замовчуванням
Управління файлами з метою збереження конфіденційності, за допомогою файлової системи, що шифрує	Якщо цей параметр має значення "Заборонити", файлову систему EFS не можна використовувати на даному комп'ютері. Якщо цей параметр має значення "Дозволити" або не визначений, файлову систему EFS можна використовувати на даному комп'ютері.	Не визначений
Шифрувати вміст папки "Документи" користувача	Якщо цей параметр включений, папки "Документи" всіх користувачів комп'ютера будуть автоматично шифруватися за допомогою файлової системи EFS.	Відключений
Вимагати смарт-карту для EFS	Якщо цей параметр включений, використовувати сертифікати програм для файлової системи EFS неможливо.	Відключений

Продовження таблиці 3.2

Параметр	Примітки	Значення за замовчуванням
Створити кешуємий користувальницький ключ зі смарт-карти	Якщо цей параметр включений, то при виникненні необхідності використання смарт-карти для файлової системи EFS у ході користувальницького сеансу в перший раз створюється кешована версія необхідних ключів відповідно до процедури, описаної вище. Якщо він відключений, смарт-карта необхідна при кожному шифровані або розшифровці файлу, захищеного за допомогою сертифіката на смарт-карті.	Включений
Включити управління файлами з метою збереження конфіденційності, файлу підкачування	Якщо цей параметр включений, файл підкачування Windows буде шифруватися файловою системою EFS.	Відключений
Відображати повідомлення про архівацію ключа при створенні або зміні користувальницького ключа	Якщо цей параметр включений, при створенні або зміні ключа користувачеві буде виводитися пропозицію створити резервну копію його ключів EFS для відновлення.	Комп'ютер, приєднаний до домену: відключений Робоча станція або ізольований комп'ютер: включений

У розділі сертифікатів доступні параметри, зазначені в наведеній нижче таблиці.

Таблиця 3.3 – Параметри сертифікатів

Параметр	Примітки	Значення за замовчуванням
Дозволити EFS створювати сертифікати, що самозаверяють, коли центр сертифікації недоступний	Якщо цей параметр відключений, користувачі можуть використовувати файлову систему EFS тільки в тому випадку, якщо в них є сертифікати, видані центром сертифікації.	Включений
Розмір ключа для сертифікатів, що самозаверяють	Цей параметр дозволяє вибрати розмір ключа, що становить 1024, 2048, 4096, 8192 або 16 384 біт. Ключі великого розміру підвищують безпека, але можуть знижувати продуктивність.	2048
Шаблон EFS для автоматичних запитів сертифікатів	Цей параметр визначає ім'я шаблона сертифіката, використовуваного для запиту сертифіката EFS у центрі сертифікації.	Базове управління файлами з метою збереження конфіденційності, EFS

Параметри кеша сертифікатів EFS можна настроїти на вкладці **Кеш**.

Необхідність внесення змін в існуючий код

Для використання файлової системи EFS не потрібно вносити зміни в існуючий код.

Підготовка до розгортання

Перед включенням файлової системи EFS необхідно врахувати перераховані нижче вимоги.

– Визначить агент відновлення й процедуру відновлення.

– Ознайомтеся з новими параметрами EFS і з'ясуєте, як їх варто настроїти для виконання певних вимог до безпеки.

Доступність у різних випусках системи Windows Server 2022

Шифрована файлова система є невід'ємним елементом файлової системи й має однакові можливості у всіх випусках операційної системи Windows Server 2022. Є версії для 32-розрядних і 64-розрядних платформ.

Шифрована файлова система доступна в системах Windows 10/11® і може бути дуже корисна для забезпечення безпеки даних, що зберігаються на клієнтських комп'ютерах, особливо портативних.

3.2 Розробка структурної схеми

Структурна схема системи зображена на рисунку 3.1. Робота системи відбувається наступним чином. EFS працює, управляючи кожним файлом з метою збереження конфіденційності, за допомогою алгоритму симетричного управління файлами з метою збереження конфіденційності, що залежить від версії операційної системи й настроювань. При цьому використовується випадково згенерований ключ для кожного файлу, називаний **File Encryption Key** (FEK), вибір симетричного управління файлами з метою збереження конфіденційності, на даному етапі пояснюється його швидкістю й більшою надійністю стосовно асиметричного управління файлами з метою збереження конфіденційності.

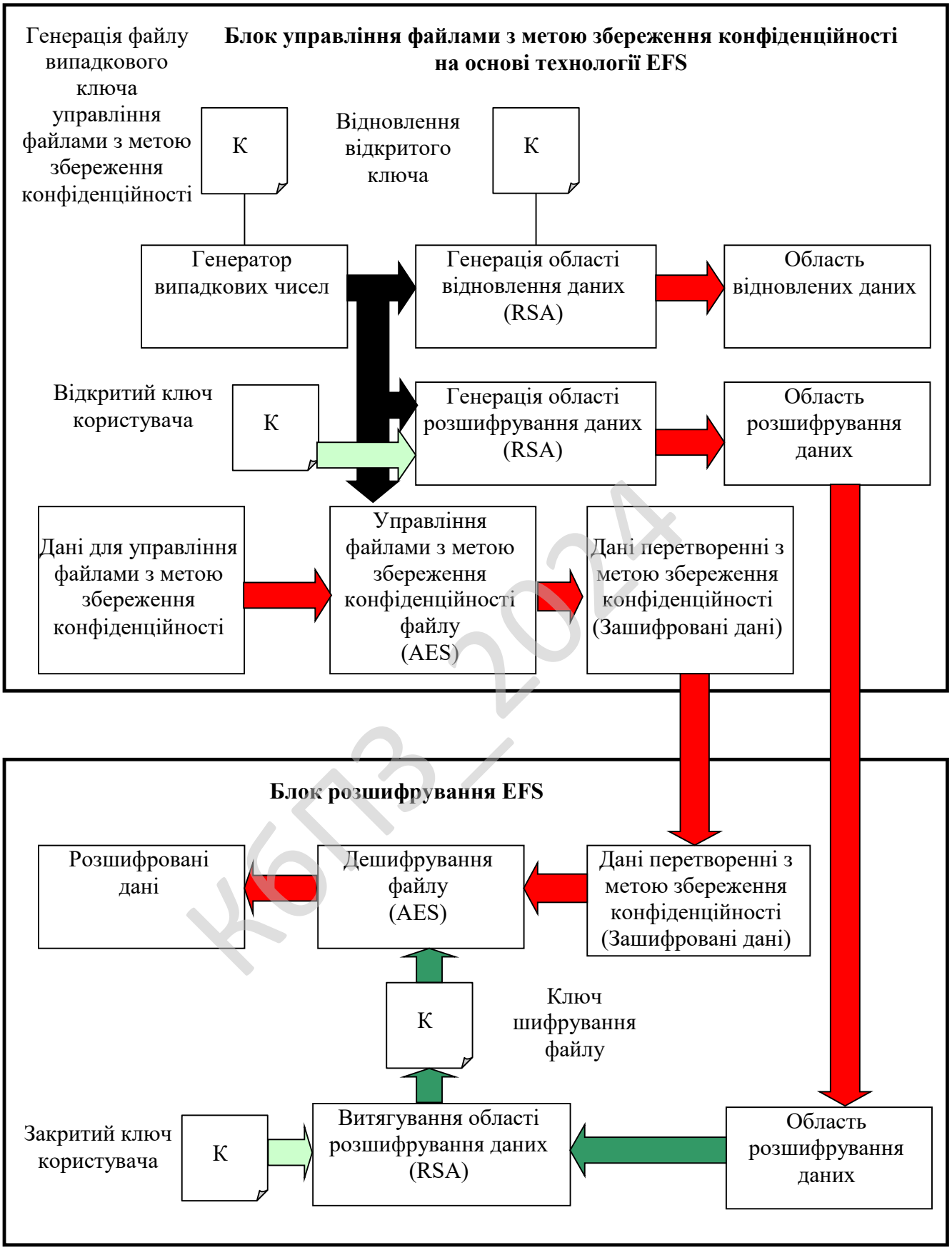

Рисунок 3.1 – Структурна схема системи

У даному бакалаврському проекті у якості симетричного алгоритму управління файлами з метою збереження конфіденційності, вибраний AES, у зв’язку з тим, що він є достатньо стійким та швидким.

FEK (випадковий для кожного файлу ключ симетричного управління файлами з метою збереження конфіденційності, AES) захищається шляхом асиметричного управління файлами з метою збереження конфіденційності, RSA, що використовує відкритий ключ користувача файл, який шифрує. RSA обраний тому, що він достатньо стійкий , для цих потреб, та виконується більш швидко, ніж інші алгоритми симетричного управління файлами з метою збереження конфіденційності. Зашифрований у такий спосіб ключ FEK зберігається в альтернативному потоці $EFS файлової системи NTFS. Для дешифрування даних, драйвер шифрованої файлової системи, прозоро для користувача, розшифровує FEK використовуючи закритий ключ RSA користувача, а потім і необхідний файл за допомогою розшифрованого файлового ключа AES.

Перед використанням можливостей управління файлами з метою збереження конфіденційності, EFS варто визначитися чи буде використовуватися **Агент відновлення даних**. Агентом відновлення називається користувач, уповноважений розшифровувати дані, зашифровані іншим користувачем, якщо користувач втратив закриті ключі сертифіката управління файлами з метою збереження конфіденційності, або обліковий запис користувача віддалений і потрібно відновити зашифровані дані. Як правило, Агентом відновлення вказується Адміністратор, але може бути призначений і інший користувач. Може бути створене трохи Агентів відновлення. Щоб призначити користувача Агентом відновлення, необхідно спочатку створити сертифікати Агента відновлення.

Автентифікація користувача й права доступу до ресурсів, що мають місце в NT працюють, коли операційна система завантажена, але при фізичному доступі до системи можливо завантажити іншу ОС щоб обійти ці обмеження. EFS використовує симетричне управління файлами з метою збереження конфіденційності, для захисту файлів, а також управління файлами з метою

збереження конфіденційності, засноване на парі відкритий/закритий ключ для захисту випадково згенерованого ключа управління файлами з метою збереження конфіденційності, для кожного файлу. За замовчуванням закритий ключ користувача захищений за допомогою управління файлами з метою збереження конфіденційності, користувальницьким паролем і захищеність даних залежить від стійкості пароля користувача.

При першому використанні можливостей EFS система створює цифрове посвідчення, що використовує для роботи із зашифрованими файлами. За замовчуванням (якщо інше не встановлено адміністратором) таке посвідчення має ім'я користувача й міститься в сховище "Особисті сертифікати".

З метою відновлення доступу до зашифрованих файлів після переустановки системи або внаслідок втрати закритого ключа варто зберегти в надійному місці закриті ключі Агентів відновлення або (якщо вони не призначені), закриті ключі всіх користувачів, що використовують EFS, експортувавши їх зі сховища "Особисті" оснащення "Сертифікати". Варто помітити, що якщо для локального користувача адміністратор спробує видалити пароль облікового запису, то користувач втратить всі особисті сертифікати, а відповідно й доступ до зашифрованих EFS файлам (при такій спробі буде видане відповідне попередження)

Шифрувати можна як окремі файли, так і цілі папки, при цьому якщо шифрується папка вже утримуюча файли, тобто можливість вибору, шифрувати тільки папку або папку й вкладені файли. Управління файлами з метою збереження конфіденційності, папки не означає що інші користувачі не зможуть переглядати вміст папки – вони лише не зможуть відкривати файли, управління якими реалізовано з метою збереження конфіденційності. Всі нові файли, збережені в зашифрованій папці або скопійовані в неї будуть автоматично зашифровані. Шифрувати папки більш зручно, і крім того безпечно, оскільки EFS має схему відновлення після аварійного збою (наприклад якщо під час операції управління файлами з метою збереження конфіденційності, відбулася критична помилка) яка передбачає створення незашифрованої архівної копії вихідного

файлу – при успішному завершенні операції управління файлами з метою збереження конфіденційності, архівна копія віддаляється, але може бути відновлена спеціальними програмами відновлення віддалених даних, що створює потенційну погрозу інформаційної безпеки. А при збереженні файлу в зашифрованій папці управління файлами з метою збереження конфіденційності, відбувається без створення такої резервної копії. Якщо все-таки шифрувалися одиночні файли, тобто можливість перезаписати кластери, що залишилися після зміни або видалення файлів на томах NTFS випадковими значеннями – для цього може бути використана команда cipher /w: шлях запущена з командного рядка або програми сторонніх розроблювачів.

3.3 Розробка функціональної схеми

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

Win64 API

Забезпечує інтерфейс програмування для управління файлами з метою збереження конфіденційності, відкритих файлів, дешифрування й відновлення закритих файлів, прийому й передачі закритих файлів без їхньої попередньої розшифровки. Реалізований у вигляді стандартної системної бібліотеки advapi64.dll.

Бібліотека часу виконання EFS (FSRTL)

FSRTL – це модуль усередині драйвера EFS, що здійснює зовнішні виклики NTFS для виконання різних операцій файлової системи, таких як читання, запис, відкриття зашифрованих файлів і каталогів, а також операцій управління файлами з метою збереження конфіденційності, дешифрування, відновлення даних при записі на диск і читанні з диска. Незважаючи на те, що драйвер EFS і FSRTL реалізовані у вигляді одного компонента, вони ніколи не взаємодіють прямо. Для обміну повідомленнями між собою вони використовують механізм викликів NTFS. Це гарантує участь NTFS у всіх файлових операціях.

Операції, реалізовані з використанням механізмів керування файлами, включають запис даних у файлові атрибути EFS (DDF і DRF) і передачу обчислених в EFS ключів FEK у бібліотеку FSRTL, тому що ці ключі повинні встановлюватися в контексті відкриття файлу. Такий контекст відкриття файлу дозволяє потім здійснювати непомітне управління файлами з метою збереження конфіденційності, й дешифрування при записі й зчитуванні файлів з диска.

Драйвер EFS

Цей компонент розташований логічно на вершині NTFS. Він взаємодіє із сервісом EFS, одержує ключі управління файлами з метою збереження конфіденційності, поля DDF, DRF і інші дані керування ключами. Драйвер передає цю інформацію в FSRTL (file system runtime library, бібліотека часу виконання файлової системи) для прозорого виконання різних файлових системних операцій (наприклад, відкриття файлу, читання, запис, додавання даних у кінець файлу).

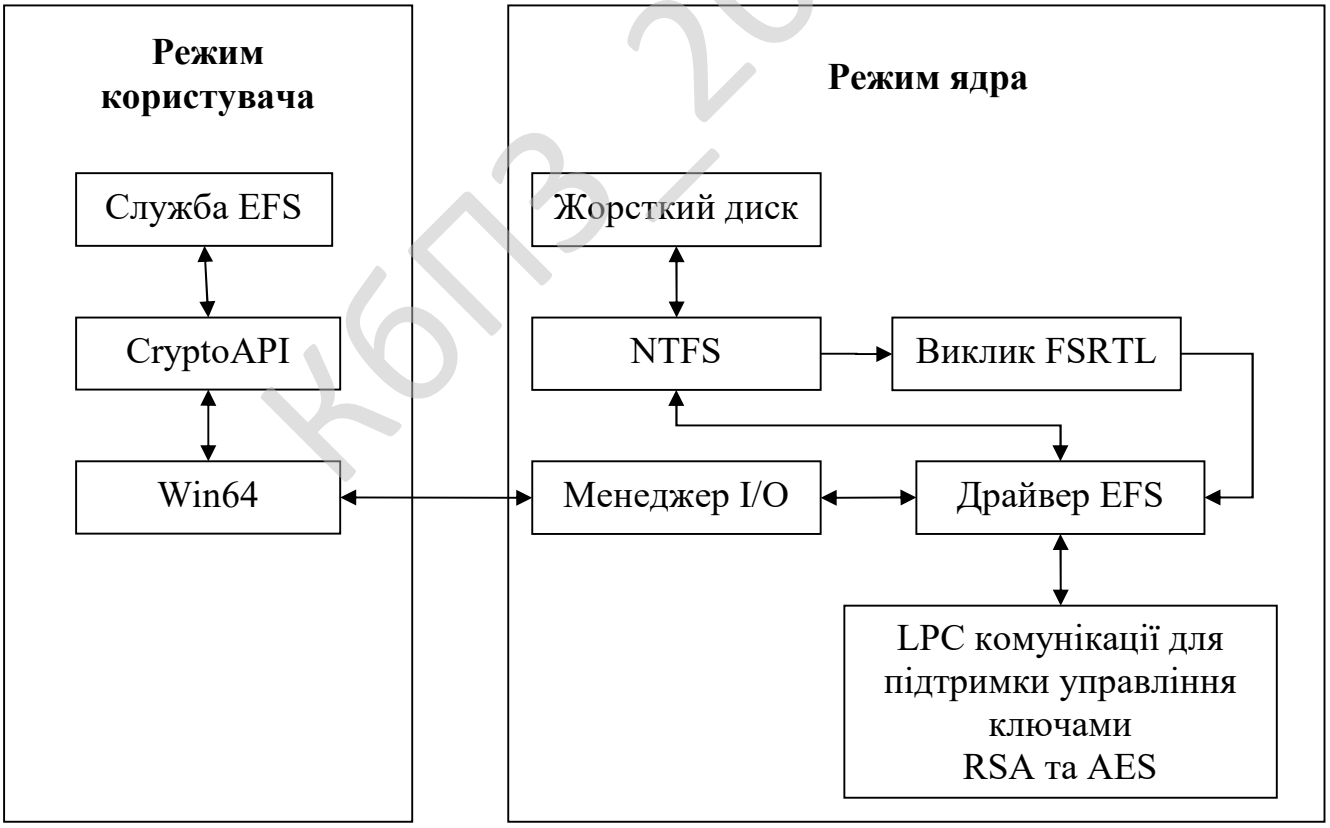

Рисунок 3.2 – Функціональна схема системи

Служба EFS

Служба EFS є частиною підсистеми безпеки. Вона використовує існуючий порт зв'язку LPC між LSA (Local security authority, локальні засоби захисту) і працюючої в kernel-mode монітором безпеки для зв'язку із драйвером EFS.

Шина LPC (Low Pin Count або LPC bus) використовується в IBM PC-сумісних персональних комп'ютерах для підключення пристроїв, що не вимагають великої пропускної здатності до ЦП. До таких пристроїв ставляться завантажувальне ПЗП й контролери «застарілих» низькопродуктивних інтерфейсів передачі даних, такі як послідовний і паралельний інтерфейси, інтерфейс підключення маніпулятора «миша» і клавіатури, НЖМД, а з недавнього часу й пристроїв зберігання криптографічної інформації. Зазвичай контролер шини LPC розташований у південному мосту на материнській платі.

Шина LPC була введена фірмою Intel в 1998 році для заміни шини ISA. Хоча LPC фізично сильно відрізняється від ISA, програмна модель периферійних контролерів, що підключаються через LPC, залишилася колишньою. Це дозволило без доробок використовувати на комп'ютерах з LPC ПЗ, розроблене для керування периферійними контролерами, які підключалися до шини ISA.

У режимі користувача служба EFS взаємодіє із програмним інтерфейсом CryptoAPI, надаючи ключі управління файлами з метою збереження конфіденційності, і забезпечуючи генерацію DDF і DRF. Крім цього, служба EFS здійснює підтримку інтерфейсу Win64 API.

RSA

Алгоритм RSA являє собою блоковий алгоритм управління файлами з метою збереження конфіденційності, де зашифровані й незашифровані дані є цілими між 0 і n-1 для деякого n.

Дані шифруються блоками, кожний блок розглядається як число, менше деякого числа n. Управління файлами з метою збереження конфіденційності, і дешифрування мають наступний вигляд для деякого незашифрованого блоку M та зашифрованого блоку C:

$$C = M^e \pmod n, \quad (3.1)$$

$$M = C^d \pmod n = (M^e)^d \pmod n = M^{ed} \pmod n. \quad (3.2)$$

Як відправник, так і одержувач повинні знати значення n. Відправник знає значення e, одержувач знає значення d. Таким чином, відкритий ключ є KU = {e, n} і закритий ключ є KR = {d, n}. При цьому повинні виконуватися наступні умови:

1. Можливість знайти значення e, d і n такі, що $M^{ed} = M \bmod n$ для всіх M < n.

2. Відносна легкість обчислення M^e й C^d для всіх значень M < n.

3. Неможливість визначити d, знаючи e та n.

Створення ключів

Вибрати прості p та q.

Обчислити $n = p \cdot q$.

Обчислити функцію Ейлера $\Phi(n) = (p - 1) * (q - 1)$.

Вибрати d взаємно просте з Φ(n): НСД (Φ(n), d) = 1; 1 < d < Φ(n).

Обчислити e: $d \cdot e \equiv 1 \bmod \Phi(n)$.

Відкритий ключ KU = {e, n} – публікується

Закритий ключ KR = {d, n} – тримається в таємниці

Примітка: p, q тримаються в таємниці, якщо вони стають відомі зловмиснику, то протокол дискредитовано.

Управління файлами з метою збереження конфіденційності,

Незашифрований текст: M < n.

Зашифрований текст: $C = M^e \pmod n$.

Дешифрування

Зашифрований текст: C < n.

Незашифрований текст: $M = C^d \pmod n$.

AES

Він є нетрадиційним блоковим шифром, оскільки не використовує мережу Фейштеля для криптоперетворень. Алгоритм представляє кожний блок кодуємих даних у вигляді двовимірного масиву байт розміром 4x4, 4x6 або 4x8 залежно від

установленої довжини блоку. Далі на відповідних етапах перетворення відбуваються або над незалежними стовпцями, або над незалежними рядками, або взагалі над окремими байтами в таблиці.

Всі перетворення в шифрі мають строге математичне обґрунтування. Сама структура й послідовність операцій дозволяють виконувати даний алгоритм ефективно як на 16-бітних так і на 64-бітних процесорах. У структурі алгоритму закладена можливість паралельного виконання деяких операцій, що на багатопроцессорних робочих станціях може ще підняти швидкість управління файлами з метою збереження конфіденційності, в 4 рази.

Алгоритм складається з деякої кількості раундів (від 10 до 14 – це залежить від розміру блоку й довжини ключа), у яких послідовно виконуються наступні операції :

ByteSub – Таблична підстановка 8x8 біт (рисунок 3.3).

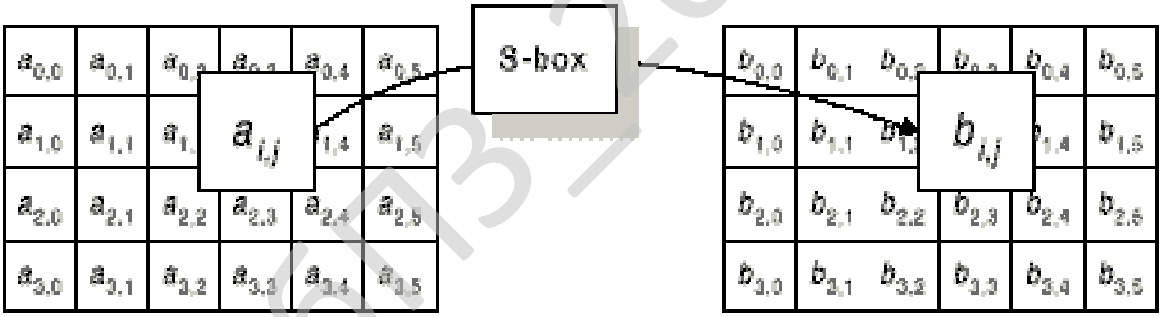

Рисунок 3.3 – Таблична підстановка 8x8 біт

ShiftRow – зрушення рядків у двовимірному масиві на різні зсуви (рисунок 3.4).

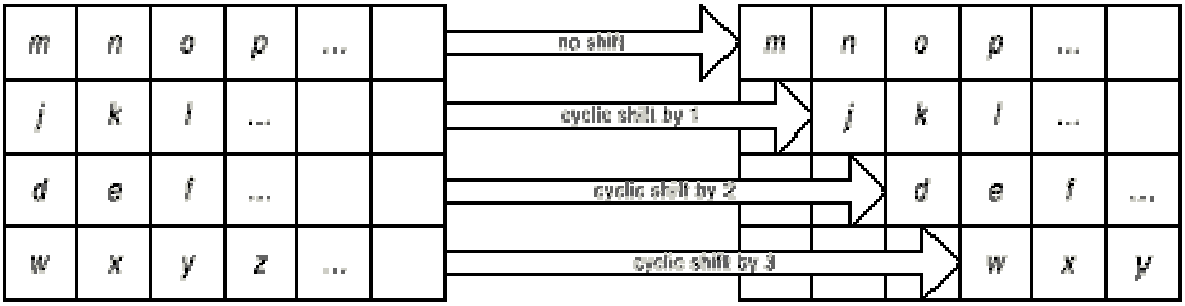

Рисунок 3.4 – Зрушення рядків у двовимірному масиві на різні зсуви

MixColumn – математичне перетворення, що перемішує дані усередині стовпця (рисунок 3.5).

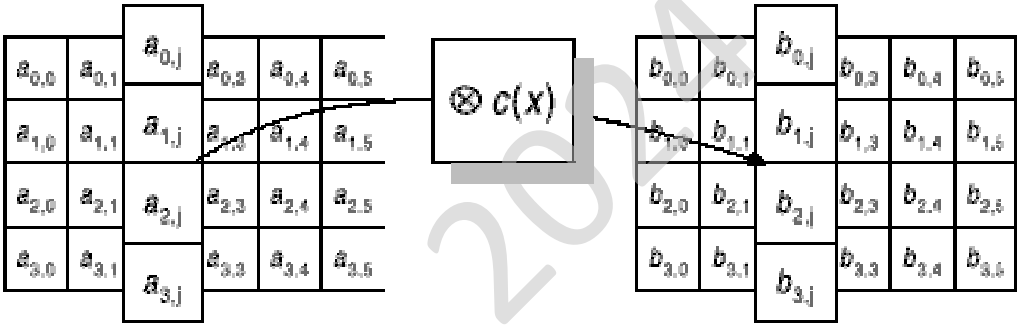

Рисунок 3.5 – Математичне перетворення, що перемішує дані усередині стовпця

AddRoundKey – додавання матеріалу ключа операцією XOR (рисунок 3.6).

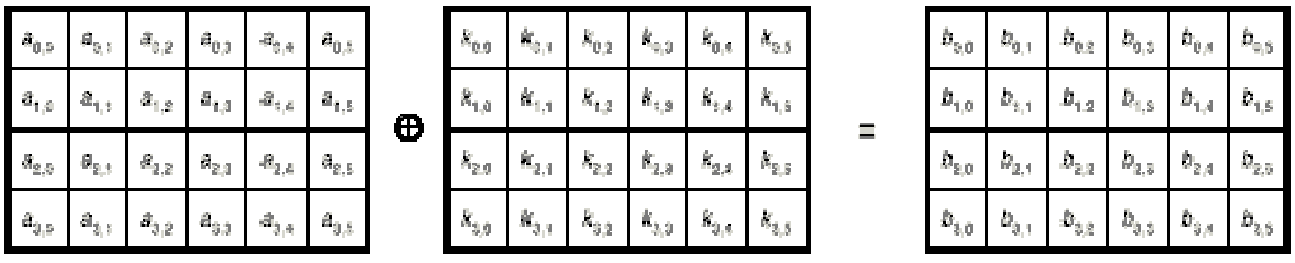

Рисунок 3.6 – Додавання матеріалу ключа операцією XOR

В останньому раунді операція перемішування стовпців відсутня, що робить всю послідовність операцій симетричною.

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

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання бакалаврського проектування, наведена на рисунку 3.7. Першим процесом, який завантажується у системі, є процес виведення головного вікна програми. Він взаємодіє з наступними процесами:

– Процес роботи з ключами шифрування.

– Процес виведення дерева каталогів та файлів.

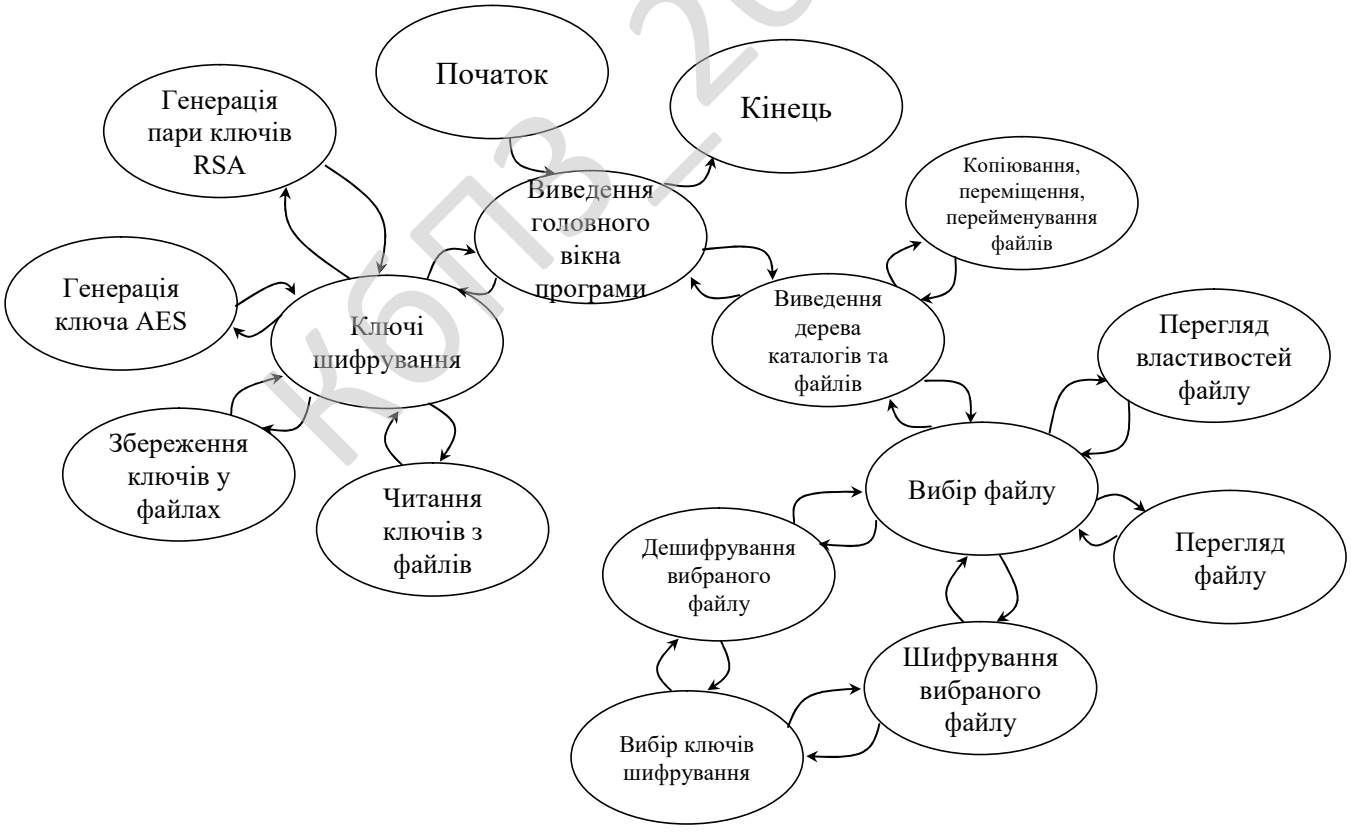

Рисунок 3.7 – Діаграма взаємодії процесів

Процес роботи з ключами шифрування взаємодіє з наступними процесами:

– Процес генерації пари ключів RSA.

– Процес генерації ключа AES.

– Процес збереження ключів у файлах.

– Процес читання ключів з файлів.

Процес виведення дерева каталогів та файлів взаємодіє з наступними процесами:

– Процес копіювання, переміщення, перейменування файлів.

– Процес вибору файлів.

Процес вибору файлів взаємодіє з наступними процесами:

– Процес перегляду властивостей файлу.

– Процес перегляду файлів.

– Процес шифрування вибраного файлу.

– Процес дешифрування вибраного файлу.

Останні два процеси взаємодіють з процесом вибору ключів шифрування.

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

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

4.1 Блок–схеми та опис алгоритмів функціонування системи

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

Спершу відбувається виведення основного вікна програми та виведення каталогів та файлів, розташованих на диску.

Після цього користувач вибирає, чи потрібно йому створювати ключі. Якщо потрібно, то виконується послідовність операцій:

– Генерація ключа AES.

– Генерація пари: закритий та відкритий ключі RSA.

– Збереження ключів у файлах.

Якщо користувачу не потрібно створювати ключі, йому пропонується завантажити файли з ключами у програму, для чого користувачеві потрібно вказати шлях до файлів із ключами програми.

Далі, якщо користувачеві потрібно дешифрувати файл, то, після вибору відповідної команди, відбувається наступна послідовність операцій:

– Вибір файлу.

– Вибір ключів шифрування.

– Дешифрування файлу завантаженими ключами шифрування.

– Виведення дешифрованого файлу.

– Збереження дешифрованого файлу.

Якщо користувачеві потрібно шифрувати файл, то відбуваються наступні операції:

– Вибір файлу.

– Вибір ключів шифрування.

– Шифрування файлу завантаженими ключами шифрування.

– Збереження зашифрованого файлу.

Далі користувач вирішує, чи продовжувати йому працювати із програмою, чи ні.

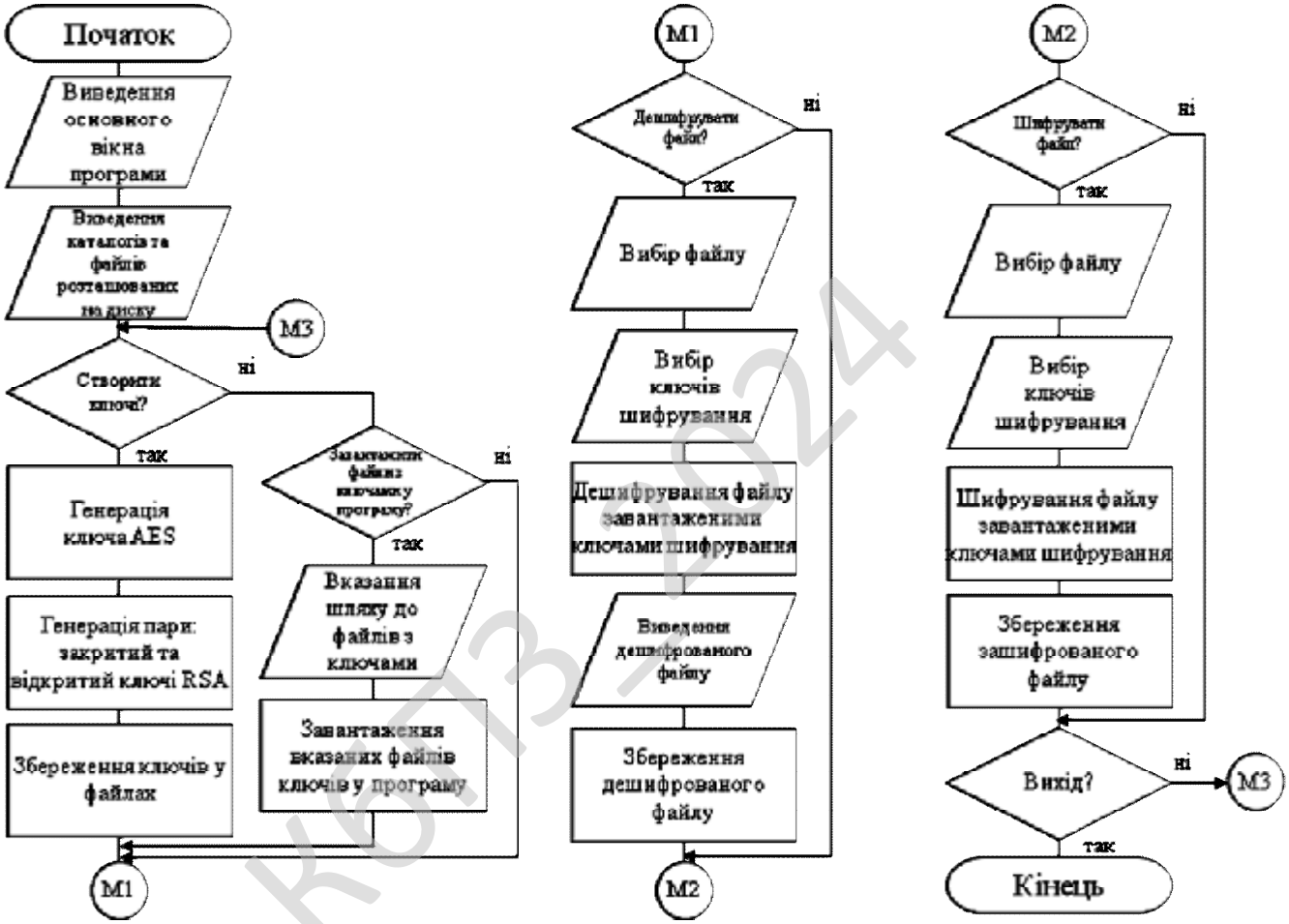

Рисунок 4.1 – Блок-схема основної програми

На рисунку 4.2 наведено блок-схему підпрограм шифрування/ дешифрування файлів.

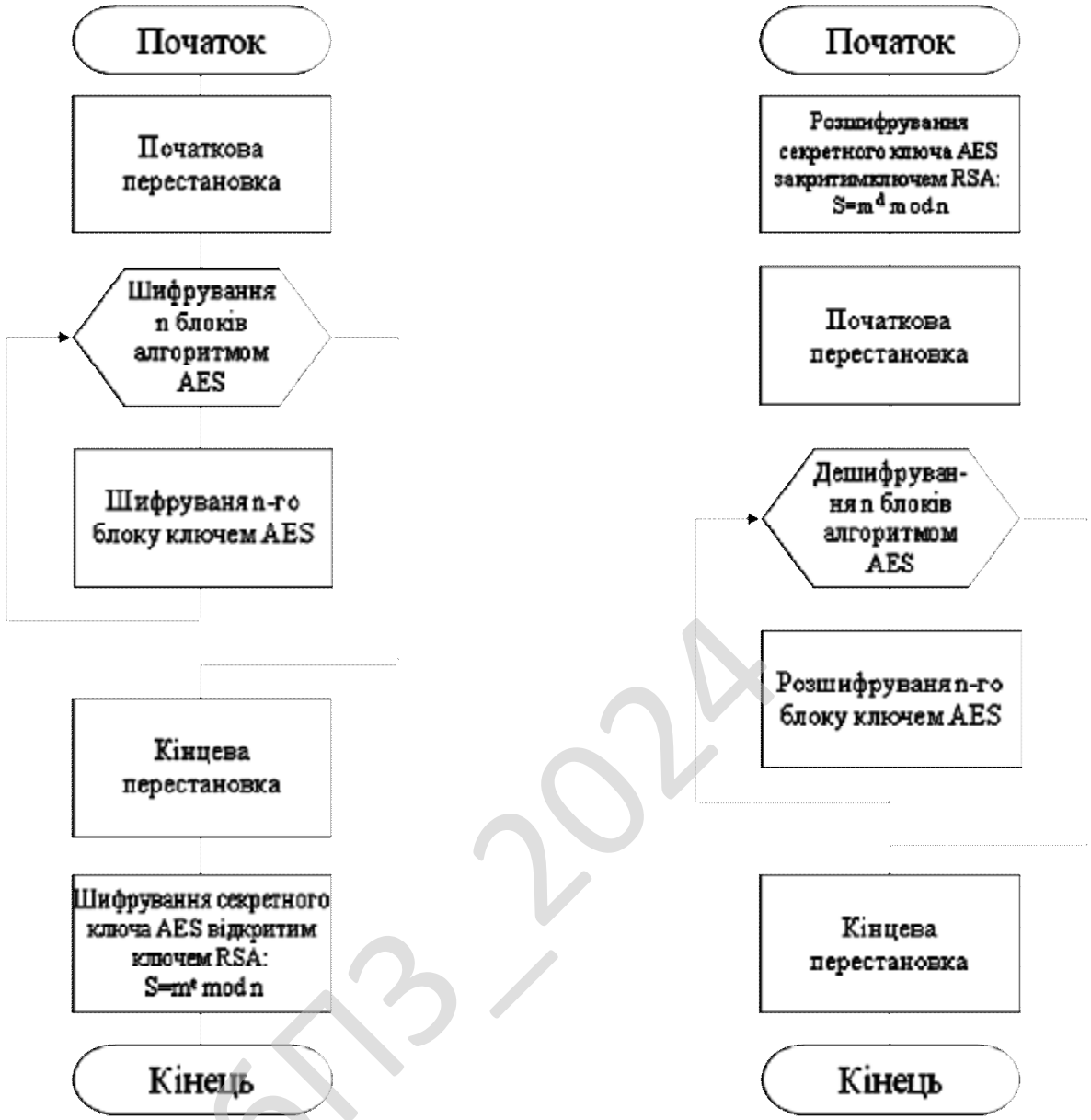

Рисунок 4.2 – Блок-схеми підпрограм шифрування/дешифрування файлів

Шифрування файлів подається на лівій блок-схемі в наступній послідовності операцій:

– Початкова перестановка.

– Шифрування n блоків алгоритмом AES, в яку входить шифрування кожного блоку ключем AES.

– Кінцева перестановка.

– Шифрування секретного ключа AES відкритим ключем RSA:

$S=m^d \bmod n$.

Після цього підпрограма завершує свою роботу.

Розшифрування файлів подається на правій блок-схемі в наступній послідовності операцій:

– Розшифрування секретного ключа AES закритим ключем RSA: $S=m^d \bmod n$.

– Початкова перестановка.

– Дешифрування n блоків алгоритмом AES, в яке входить дешифрування кожного блоку ключем AES.

– Кінцева перестановка.

Після цього підпрограма завершує свою роботу.

Нижче наведено програмний код, за допомогою якого відбувається шифрування/дешифрування алгоритмом AES:

```
Type
  TBitString = Array of Boolean;
  PBitString = ^TBitString;
  TSplitKeyParts = record
    C:TBitString;
    D:TBitString;
    end;
  TSplitKey = Array[0..16]Of TSplitKeyParts;
  TConcatKey = Array[0..15]Of TBitString;
  TIPKeyParts = record
    L:TBitString;
    R:TBitString;
    end;
  TIPKey = Array[0..16]OF TIPKeyParts;
//матриці перестановок
Const
AES_PC1:Array[0..55] Of Byte = (57,49,41,33,25,17,9,
                                1,58,50,42,34,26,18,
                                10,2,59,51,43,35,27,
                                19,11,3,60,52,44,36,
                                63,55,47,39,31,23,15,
                                7,62,54,46,38,30,22,
                                14,6,61,53,45,37,29,
                                21,13,5,28,20,12,4);
```

```
AES_PC2:Array[0..47] Of Byte = (14,17,11,24,1,5,
                                3,28,15,6,21,10,
                                23,19,12,4,26,8,
                                16,7,27,20,13,2,
                                41,52,31,37,47,55,
                                30,40,51,45,33,48,
                                44,49,39,56,34,53,
                                46,42,50,36,29,32);
AES_IP:Array[0..63] Of Byte = (58,50,42,34,26,18,10,2,
                               60,52,44,36,28,20,12,4,
                               62,54,46,38,30,22,14,6,
                               64,56,48,40,32,24,16,8,
                               57,49,41,33,25,17,9,1,
                               59,51,43,35,27,19,11,3,
                               61,53,45,37,29,21,13,5,
                               63,55,47,39,31,23,15,7);
AES_E:Array[0..47] Of Byte = (32,1,2,3,4,5,
                              4,5,6,7,8,9,
                              8,9,10,11,12,13,
                              12,13,14,15,16,17,
                              16,17,18,19,20,21,
                              20,21,22,23,24,25,
                              24,25,26,27,28,29,
                              28,29,30,31,32,1);
// S-блоки
S_BOXES:Array[0..7,0..3,0..15]Of Byte = (
((14,04,13,01,02,15,11,08,03,10,06,12,05,09,00,07),
  (00,15,07,04,14,02,13,01,10,06,12,11,09,05,03,08),
  (04,01,14,08,13,06,02,11,15,12,09,07,03,10,05,00),
  (15,12,08,02,04,09,01,07,05,11,03,14,10,00,06,13)),
((15,01,08,14,06,11,03,04,09,07,02,13,12,00,05,10),
   (03,13,04,07,15,02,08,14,12,00,01,10,06,09,11,05),
   (00,14,07,11,10,04,13,01,05,08,12,06,09,03,02,15),
   (13,08,10,01,03,15,04,02,11,06,07,12,00,05,14,09)),
((10,00,09,14,06,03,15,05,01,13,12,07,11,04,02,08),
   (13,07,00,09,03,04,06,10,02,08,05,14,12,11,15,01),
   (13,06,04,09,08,15,03,00,11,01,02,12,05,10,14,07),
   (01,10,13,00,06,09,08,07,04,15,14,03,11,05,02,12)),
((07,13,14,03,00,06,09,10,01,02,08,05,11,12,04,15),
   (13,08,11,05,06,15,00,03,04,07,02,12,01,10,14,09),
   (10,06,09,00,12,11,07,13,15,01,03,14,05,02,08,04),
   (13,15,00,06,10,01,13,08,09,04,05,11,12,07,02,14)),
```

```
((02,12,04,01,07,10,11,06,08,05,03,15,13,00,14,09),
   (14,11,02,12,04,07,13,01,05,00,15,10,03,08,09,06),
   (04,02,01,11,10,13,07,08,15,09,12,05,06,03,00,14),
   (11,08,12,07,01,14,02,13,06,15,00,09,10,04,05,03)),
((12,01,10,15,09,02,06,08,00,13,03,04,14,07,05,11),
   (10,15,04,02,07,12,09,05,06,01,13,14,00,11,03,08),
   (09,14,15,05,02,08,12,03,07,00,04,10,01,13,11,06),
   (04,03,02,12,09,05,15,10,11,14,01,04,06,00,08,13)),
((04,11,02,14,15,00,08,13,03,12,09,07,05,10,06,01),
   (13,00,11,07,04,09,01,10,14,03,05,12,02,15,08,06),
   (01,04,11,13,12,03,07,14,10,15,06,08,00,05,09,02),
   (06,11,13,08,01,04,10,07,09,05,00,15,14,02,03,12)),
((13,02,08,04,06,15,11,01,10,09,03,14,05,00,12,07),
   (01,15,13,08,10,03,07,04,12,05,06,11,00,14,09,02),
   (07,11,04,01,09,12,14,02,00,06,10,13,15,03,05,08),
   (02,01,14,07,04,10,08,13,15,12,09,00,03,05,06,11))
);
AES_P:Array[0..31] Of Byte = (16,7,20,21,
                              29,12,28,17,
                              1,15,23,26,
                              5,18,31,10,
                              2,8,24,14,
                              32,27,3,9,
                              19,13,30,6,
                              22,11,4,25);
AES_REVERSE_IP:Array[0..63] Of Byte = (40,8,48,16,56,24,64,32,
            39,7,47,15,55,23,63,31,
            38,6,46,14,54,22,62,30,
            37,5,45,13,53,21,61,29,
            36,4,44,12,52,20,60,28,
            35,3,43,11,51,19,59,27,
            34,2,42,10,50,18,58,26,
            33,1,41,9,49,17,57,25);
AES_LSH:Array[0..15] Of Byte = (1,1,2,2,2,2,2,2,1,2,2,2,2,2,2,1);
Function BinToInt(S:TBitString):Integer;
Function IntToBin(N:Integer;Precision:Integer=8):TBitString;
Function BinToStr(Bits:TBitString):String;
Function StrToBin(S:String):TBitString;
Function AnsiStrToBin(S:String; Zeroes:Boolean=True):TBitString;
Function BinToAnsiStr(Bits:TBitString):String;
Procedure CopyBits(Var Dest:TBitString; Source:TBitString; NBits:Integer);
Function ConcatBits(Bits:Array Of TBitString):TBitString;
```

```
Function DESEncode(S,Key:String):TBitString;
Function DESDecode(S,Key:String):TBitString;
Function GetPermutedKey(Key:TBitString):TBitString;
Function GetPermutedKey2(Key:TBitString):TBitString;
Function GetSplitKey(Key:TBitString):TSplitKey;
Function GetConcatKey(Key:TSplitKey):TConcatKey;
Function GetIPKey(M:TBitString; ConcatKey:TConcatKey):TIPKey;
Function GetF(R,K:TBitString):TBitString;
Function GetSBox(Index:Integer; T:TBitString):TBitString;
Function GetReverseIP(RL:TBitString):TBitString;
Procedure ReverseSubKeys(Var Keys:TConcatKey);
implementation
Function ConcatBits(Bits:Array Of TBitString):TBitString;
Var
I,C:Integer;
Begin
SetLength(Result,0);
For C:=0 To Length(Bits)-1 Do
  Begin
  SetLength(Result,Length(Result)+Length(Bits[C]));
  For I:=0 To Length(Bits[C])-1 Do
    Result[Length(Result)-Length(Bits[C])+I]:=Bits[C][I];
  End;
End;
Procedure CopyBits(Var Dest:TBitString; Source:TBitString; NBits:Integer);
Var
I:Integer;
Begin
SetLength(Dest,NBits);
For I:=0 To NBits-1 Do
  Dest[I]:=Source[I];
End;
Function BinToInt(S: TBitString): Integer;
Var
L,I:Integer;
Begin
Result:=0;
L:=Length(S);
IF L=0 Then
  Raise EConvertError.Create(спеціальна бітова строка довжиною 0 біт');
For I:=L-1 DownTo 0 Do
  Result:=Result+Ord(S[I])*Trunc(Power(2,L-I-1));
```

```
End;
Function IntToBin(N:Integer; Precision:Integer):TBitString;
Var
BitList:TList;
Bit:PBoolean;
Begin
SetLength(Result,0);
BitList:=TList.Create;
While N>0 Do
  Begin
  New(Bit);
  Bit^:=Boolean(N mod 2);
  BitList.Insert(0,Bit);
  N:=N div 2;
  End;
While BitList.Count<Precision Do
  Begin
  New(Bit);
  Bit^:=False;
  BitList.Insert(0,Bit);
  End;
For N:=0 To BitList.Count-1 Do
  Begin
  SetLength(Result,N+1);
  Bit:=BitList.Items[N];
  Result[N]:=Bit^;
  Dispose(Bit);
  End;
BitList.Free;
end;
Function AnsiStrToBin(S: String; Zeroes:Boolean):TBitString;
Var
Temp,B:TBitString;
L,I,J:Integer;
Begin
L:=0;
SetLength(Result,L);
SetLength(Temp,L);
SetLength(B,0);
For I:=1 To Length(S) Do
  Begin
  B:=IntToBin(Ord(S[I]));
```

```
  L:=L+Length(B);
  SetLength(Temp,L);
  For J:=0 To Length(B)-1 Do
    Temp[Length(Temp)-Length(B)+J]:=B[J];
  End;
Result:=Temp;
End;
Function BinToStr(Bits:TBitString):String;
Var
I,L:Integer;
Begin
Result:='';
L:=Length(Bits);
IF L=0 Then
  Raise EConvertError.Create('Спеціальна бітова строка довжиною 0 біт');
For I:=0 To L-1 Do
  IF Bits[I] Then Result:=Result+'1'
  Else Result:=Result+'0';
End;
Function StrToBin(S:String):TBitString;
Var
I:Integer;
Begin
SetLength(Result,0);
For I:=1 To Length(S) Do
  Begin
  IF (S[I]<>'1')And(S[I]<>'0') Then
    Raise EConvertError.Create(S+' некоректна бінарна строка');
  SetLength(Result,I);
  Result[I-1]:=Boolean(StrToInt(S[I]));
  End;
End;
Function BinToAnsiStr(Bits:TBitString):String;
Var
I:Integer;
B:TBitString;
Begin
Result:='';
SetLength(B,8);
I:=0;
While I<=Length(Bits)-8 Do
  Begin
```

```
  CopyMemory(B,Ptr(Integer(Bits)+I),8);
  Result:=Result+Char(BinToInt(B));
  Inc(I,8);
  End;
End;
Function GetPermutedKey(Key:TBitString):TBitString;
Var
   I:Integer;
Begin
   SetLength(Result,Length(AES_PC1));
   For I:=0 To Length(AES_PC1)-1 Do
     Result[I]:=Key[AES_PC1[I]-1];
End;
Function GetPermutedKey2(Key:TBitString):TBitString;
Var
   I:Integer;
Begin
   SetLength(Result,Length(AES_PC2));
   For I:=0 To Length(AES_PC2)-1 Do
     Result[I]:=Key[AES_PC2[I]-1];
End;
Function GetSplitKey(Key:TBitString):TSplitKey;
  Function LeftShift(Key:TBitString; N:Integer):TBitString;
  Var
  I,J:Integer;
  Temp:TBitString;
  Begin
  SetLength(Result,28);
  SetLength(Temp,28);
  For I:=0 To 27 Do
    Temp[I]:=Key[I];
  For J:=1 To N Do
    Begin
    For I:=1 To 27 Do
      Result[I-1]:=Temp[I];
    Result[27]:=Temp[0];
    For I:=0 To 27 Do
      Temp[I]:=Result[I];
    End;
  End;
Var
I,J:Integer;
```

```
Begin
For J:=1 To 16 Do
  Begin
  SetLength(Result[J].C,28);
  SetLength(Result[J].D,28);
  End;
CopyBits(Result[0].C,Key,28);
CopyBits(Result[0].D,TBitString(Integer(Key)+28),28);
For I:=1 To 16 Do
  Begin
  Result[I].C:=LeftShift(Result[I-1].C,AES_LSH[I-1]);
  Result[I].D:=LeftShift(Result[I-1].D,AES_LSH[I-1]);
  End;
End;
Function GetConcatKey(Key:TSplitKey):TConcatKey;
Var
I:Integer;
Temp:TBitString;
Begin
For I:=0 To 15 Do
  Begin
  SetLength(Result[I],56);
  Temp:=ConcatBits([Key[I+1].C,Key[I+1].D]);
  Result[I]:=GetPermutedKey2(Temp);
  End;
End;
Function GetIPKey(M:TBitString; ConcatKey:TConcatKey):TIPKey;
Var
I,J:Integer;
IP, F:TBitString;
Begin
For I:=0 To 16 Do
  Begin
  SetLength(Result[I].L,32);
  SetLength(Result[I].R,32);
  End;
SetLength(IP,64);
For I:=0 To Length(AES_IP)-1 Do
  IP[I]:=M[AES_IP[I]-1];
For I:=0 To 31 Do
  Result[0].L[I]:=IP[I];
For I:=32 To 63 Do
```

```
  Result[0].R[I-32]:=IP[I];
For I:=1 To 16 Do
  Begin
  Result[I].L:=Result[I-1].R;
  F:=GetF(Result[I-1].R,ConcatKey[I-1]);
  For J:=0 To 31 Do
    Result[I].R[J]:=Result[I-1].L[J] XOR F[J];
  End;
End;
Function GetF(R,K:TBitString):TBitString;
Var
I,J:Integer;
S,E,KE,F,T:TBitString;
Begin
   SetLength(E,48);
   For I:=0 To 47 Do
     E[I]:=R[AES_E[I]-1];
   SetLength(KE,48);
   For I:=0 To 47 Do
     KE[I]:=K[I] XOR E[I];
   SetLength(T,6);
   SetLength(F,0);
   SetLength(S,4);
   I:=0;
   While I<48 Do
     Begin
     For J:=0 To 6 Do
       T[J]:=KE[J+I];
     S:=GetSBox(I div 6,T);
     F:=ConcatBits([F,S]);
     I:=I+6;
     End;
   SetLength(Result,32);
   For I:=0 To 31 Do
     Result[I]:=F[AES_P[I]-1];
End;
Function GetSBox(Index:Integer; T:TBitString):TBitString;
Var
Val,Row,Col:Integer;
Temp:TBitString;
Begin
   SetLength(Result,4);
```

```
   SetLength(Temp,2);
   Temp[0]:=T[0];
   Temp[1]:=T[5];
   Row:=BinToInt(Temp);
   SetLength(Temp,4);
   CopyBits(Temp,TBitString(@T[1]),4);
   Col:=BinToInt(Temp);
   Val:=S_BOXES[Index,Row,Col];
   SetLength(Result,4);
   Result:=IntToBin(Val,4);
End;
Function GetReverseIP(RL:TBitString):TBitString;
Var
I:Integer;
Begin
   SetLength(Result,64);
   For I:=0 To Length(AES_REVERSE_IP)-1 Do
     Result[I]:=RL[AES_REVERSE_IP[I]-1];
End;
Procedure ReverseSubKeys(Var Keys:TConcatKey);
Var
   I,L:Integer;
   T:TBitString;
Begin
   SetLength(T,48);
   L:=Length(Keys);
   For I:=0 To (L-1)Div 2 Do
     Begin
     T:=Keys[I];
     Keys[I]:=Keys[(L-I)-1];
     Keys[(L-I)-1]:=T;
     End;
End;
Function DESEncode(S,Key:String):TBitString;
Var
   I:Integer;
   K:TBitString;
   M:TBitString;
   RL:TBitString;
   Kplus:TBitString;
   SplitKey:TSplitKey;
   ConcatKey:TConcatKey;
```

```
   IPKey:TIPKey;
Begin
   K:=AnsiStrToBin(Key);
   Kplus:=GetPermutedKey(K);
   SplitKey:=GetSplitKey(Kplus);
   ConcatKey:=GetConcatKey(SplitKey);
   M:=AnsiStrToBin(S);
   IPKey:=GetIPKey(M,ConcatKey);
   SetLength(RL,64);
   For I:=0 To 31 Do
     Begin
     RL[I]:=IPKey[16].R[I];
     RL[I+32]:=IPKey[16].L[I];
     End;
   RL:=GetReverseIP(RL);
   Result:=RL;
End;
Function DESDecode(S,Key:String):TBitString;
Var
   I:Integer;
   K:TBitString;
   M:TBitString;
   RL:TBitString;
   Kplus:TBitString;
   SplitKey:TSplitKey;
   ConcatKey:TConcatKey;
   IPKey:TIPKey;
Begin
   K:=AnsiStrToBin(Key);
   Kplus:=GetPermutedKey(K);
   SplitKey:=GetSplitKey(Kplus);
   ConcatKey:=GetConcatKey(SplitKey);
   ReverseSubKeys(ConcatKey);
   M:=AnsiStrToBin(S);
   IPKey:=GetIPKey(M,ConcatKey);
   SetLength(RL,64);
   For I:=0 To 31 Do
     Begin
     RL[I]:=IPKey[16].R[I];
     RL[I+32]:=IPKey[16].L[I];
     End;
   RL:=GetReverseIP(RL);
```

```
   Result:=RL;
   End;
end.
```

Генерація ключів RSA та шифрування ключа AES відбувається за допомогою наступного коду:

```
type
  TRSA_keys = class(TForm)
    Button1: TButton;
    Label3: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    Edit3: TEdit;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    Edit_E: TEdit;
    Edit_N: TEdit;
    Edit_D: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label4: TLabel;
    Button5: TButton;
    Edit1: TEdit;
    Label122: TLabel;
    Label12: TLabel;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

```
var
  RSA_keys: TRSA_keys;
   n, e, d, dp, dq, p, q, phi, one, two, gcd, temp, nilgint : TFGInt;
   test, signature : String;
   ok : boolean;
   st: string;
   tx1: text;
   tx2: text;
   tx3: text;
implementation
{$R *.dfm}
procedure TRSA_keys.Button1Click(Sender: TObject);
begin
// Генерація p та q
   Base256StringToFGInt('3557', p);
   Base256StringToFGInt('2579', q);
   PrimeSearch(p);
   PrimeSearch(q);
   FGIntToBase256String(p, st);
   FGIntToBase256String(q, st);
// Обчислення N
   FGIntMul(p, q, n);
   p.Number[1] := p.Number[1] - 1;
   q.Number[1] := q.Number[1] - 1;
   FGIntMul(p, q, phi);
   FGIntToBase10String(n, st);
   Edit_N.Text:=st;
// Обчислення E - ключ шифрування
//   Base10StringToFGInt('65537', e); // непарне число
//   Base10StringToFGInt('8171921', e);
   Base10StringToFGInt('14486581214143', e);
   Base10StringToFGInt('1', one);
   Base10StringToFGInt('2', two);
   FGIntGCD(phi, e, gcd);
   While FGIntCompareAbs(gcd, one) <> Eq Do
   Begin
      FGIntadd(e, two, temp);
      FGIntCopy(temp, e);
      FGIntGCD(phi, e, gcd);
   End;
   FGIntDestroy(two);
   FGIntDestroy(one);
```

```
   FGIntDestroy(gcd);
// Обчислення D – ключ дешифрування
   FGIntModInv(e, phi, d);
   FGIntModInv(e, p, dp);
   FGIntModInv(e, q, dq);
   p.Number[1] := p.Number[1] + 1;
   q.Number[1] := q.Number[1] + 1;
   FGIntDestroy(phi);
   FGIntDestroy(nilgint);
   FGIntToBase10String(e, st);
   Edit_E.Text:=st;
   FGIntToBase10String(d, st);
   Edit_D.Text:=st;
end;
//дешифрування ключа AES
procedure TRSA_keys.Button2Click(Sender: TObject);
var prom: string;
begin
   test := Edit3.Text;
   RSAEncrypt(test, e, n, test);
   Edit1.Text:=test;
end;
//запис ключів RSA у файли
procedure TRSA_keys.Button3Click(Sender: TObject);
begin
  AssignFile(tx1,'OpenKey.keys');
  AssignFile(tx2,'CloseKey.keys');
  ReWrite(tx1); CloseFile(tx1);
  ReWrite(tx2); CloseFile(tx2);
  Append(tx1);
  Append(tx2);
  FGIntToBase256String(n, st);
  ConvertBase256to64(st, st);
  WriteLn(tx1,st);
  WriteLn(tx2,st);
  FGIntToBase256String(e, st);
  ConvertBase256to64(st, st);
  WriteLn(tx1,st);
  FGIntToBase256String(d, st);
  ConvertBase256to64(st, st);
  WriteLn(tx2,st);
  CloseFile(tx1);
```

```
  CloseFile(tx2);
end;
//шифрування ключа AES
procedure TRSA_keys.Button4Click(Sender: TObject);
begin
 RSADecrypt(test, d, n, Nilgint, Nilgint, Nilgint, Nilgint, test);
       Edit3.Text:=test;
end;
//Запис ключа AES у файл
procedure TRSA_keys.Button5Click(Sender: TObject);
begin
AssignFile(tx3,'SecretKey.keys');
  ReWrite(tx3); CloseFile(tx3);
  Append(tx3);
  WriteLn(tx3,Edit1.Text);
  CloseFile(tx3);
end;
procedure TRSA_keys.FormCreate(Sender: TObject);
begin
Edit1.Text:=AES.EditDES.Text;
end;
```

Шифрування алгоритмом RSA реалізовано за допомогою наступного коду:

```
Procedure RSAEncrypt(P : String; Var exp, modb : TFGInt; Var E : String);
Procedure RSADecrypt(E : String; Var exp, modb, d_p, d_q, p, q : TFGInt; Var D :
String);
Procedure RSASign(M : String; Var d, n, dp, dq, p, q : TFGInt; Var S : String);
Procedure RSAVerify(M, S : String; Var e, n : TFGInt; Var valid : boolean);
Implementation
{$H+}
// Шифруємо рядок алгоритмом RSA, P^exp mod modb = E
Procedure RSAEncrypt(P : String; Var exp, modb : TFGInt; Var E : String);
Var
   i, j, modbits : longint;
   PGInt, temp, zero : TFGInt;
   tempstr1, tempstr2, tempstr3 : String;
Begin
   Base2StringToFGInt('0', zero);
   FGIntToBase2String(modb, tempstr1);
   modbits := length(tempstr1);
   convertBase256to2(P, tempstr1);
```

```
   tempstr1 := '111' + tempstr1;
   j := modbits - 1;
   While (length(tempstr1) Mod j) <> 0 Do tempstr1 := '0' + tempstr1;
   j := length(tempstr1) Div (modbits - 1);
   tempstr2 := '';
   For i := 1 To j Do
   Begin
      tempstr3 := copy(tempstr1, 1, modbits - 1);
      While (copy(tempstr3, 1, 1) = '0') And (length(tempstr3) > 1) Do
delete(tempstr3, 1, 1);
      Base2StringToFGInt(tempstr3, PGInt);
      delete(tempstr1, 1, modbits - 1);
      If tempstr3 = '0' Then FGIntCopy(zero, temp) Else
FGIntMontgomeryModExp(PGInt, exp, modb, temp);
      FGIntDestroy(PGInt);
      tempstr3 := '';
      FGIntToBase2String(temp, tempstr3);
      While (length(tempstr3) Mod modbits) <> 0 Do tempstr3 := '0' + tempstr3;
      tempstr2 := tempstr2 + tempstr3;
      FGIntdestroy(temp);
   End;
   While (tempstr2[1] = '0') And (length(tempstr2) > 1) Do delete(tempstr2, 1, 1);
   ConvertBase2To256(tempstr2, E);
   FGIntDestroy(zero);
End;
Дешифруємо рядок алгоритмом RSA, E^exp mod modb = D
// modb = p*q, d_p*e mod (p-1) = 1
// d_q*e mod (q-1)
Procedure RSADecrypt(E : String; Var exp, modb, d_p, d_q, p, q : TFGInt; Var D :
String);
Var
   i, j, modbits : longint;
   EGInt, temp, temp1, temp2, temp3, ppinvq, qqinvp, zero : TFGInt;
   tempstr1, tempstr2, tempstr3 : String;
Begin
   Base2StringToFGInt('0', zero);
   FGIntToBase2String(modb, tempstr1);
   modbits := length(tempstr1);
   convertBase256to2(E, tempstr1);
   While copy(tempstr1, 1, 1) = '0' Do delete(tempstr1, 1, 1);
   While (length(tempstr1) Mod modbits) <> 0 Do tempstr1 := '0' + tempstr1;
   If exp.Number = Nil Then
```

```
   Begin
      FGIntModInv(q, p, temp1);
      FGIntMul(q, temp1, qqinvp);
      FGIntDestroy(temp1);
      FGIntModInv(p, q, temp1);
      FGIntMul(p, temp1, ppinvq);
      FGIntDestroy(temp1);
   End;
   j := length(tempstr1) Div modbits;
   tempstr2 := '';
   For i := 1 To j Do
   Begin
      tempstr3 := copy(tempstr1, 1, modbits);
      While (copy(tempstr3, 1, 1) = '0') And (length(tempstr3) > 1) Do
delete(tempstr3, 1, 1);
      Base2StringToFGInt(tempstr3, EGInt);
      delete(tempstr1, 1, modbits);
      If tempstr3 = '0' Then FGIntCopy(zero, temp) Else
      Begin
         If exp.Number <> Nil Then FGIntMontgomeryModExp(EGInt, exp, modb, temp)
Else
         Begin
            FGIntMontgomeryModExp(EGInt, d_p, p, temp1);
            FGIntMul(temp1, qqinvp, temp3);
            FGIntCopy(temp3, temp1);
            FGIntMontgomeryModExp(EGInt, d_q, q, temp2);
            FGIntMul(temp2, ppinvq, temp3);
            FGIntCopy(temp3, temp2);
            FGIntAddMod(temp1, temp2, modb, temp);
            FGIntDestroy(temp1);
            FGIntDestroy(temp2);
         End;
      End;
      FGIntDestroy(EGInt);
      tempstr3 := '';
      FGIntToBase2String(temp, tempstr3);
      While (length(tempstr3) Mod (modbits - 1)) <> 0 Do tempstr3 := '0' +
tempstr3;
      tempstr2 := tempstr2 + tempstr3;
      FGIntdestroy(temp);
   End;
   If exp.Number = Nil Then
```

```
   Begin
      FGIntDestroy(ppinvq);
      FGIntDestroy(qqinvp);
   End;
   While (Not (copy(tempstr2, 1, 3) = '111')) And (length(tempstr2) > 3) Do
delete(tempstr2, 1, 1);
   delete(tempstr2, 1, 3);
   ConvertBase2To256(tempstr2, D);
   FGIntDestroy(zero);
End;
// підписуємо рядок RSA, M^d mod n = S
// n = p*q, dp*e mod (p-1) = 1
// dq*e mod (q-1)
Procedure RSASign(M : String; Var d, n, dp, dq, p, q : TFGInt; Var S : String);
Var
   MGInt, SGInt, temp, temp1, temp2, temp3, ppinvq, qqinvp : TFGInt;
Begin
   Base256StringToFGInt(M, MGInt);
   If d.Number <> Nil Then FGIntMontgomeryModExp(MGInt, d, n, SGInt) Else
   Begin
      FGIntModInv(p, q, temp);
      FGIntMul(p, temp, ppinvq);
      FGIntDestroy(temp);
      FGIntModInv(q, p, temp);
      FGIntMul(q, temp, qqinvp);
      FGIntDestroy(temp);
      FGIntMontgomeryModExp(MGInt, dp, p, temp1);
      FGIntMul(temp1, qqinvp, temp2);
      FGIntCopy(temp2, temp1);
      FGIntMontgomeryModExp(MGInt, dq, q, temp2);
      FGIntMul(temp2, ppinvq, temp3);
      FGIntCopy(temp3, temp2);
      FGIntAddMod(temp1, temp2, n, SGInt);
      FGIntDestroy(temp1);
      FGIntDestroy(temp2);
      FGIntDestroy(ppinvq);
      FGIntDestroy(qqinvp);
   End;
   FGIntToBase256String(SGInt, S);
   FGIntDestroy(MGInt);
   FGIntDestroy(SGInt);
End;
```

```
// Перевіряємо правильність алгоритму  RSA,
// Якщо M = S^e mod n тоді ok:=true інакше ok:=false
Procedure RSAVerify(M, S : String; Var e, n : TFGInt; Var valid : boolean);
Var
   MGInt, SGInt, temp : TFGInt;
Begin
   Base256StringToFGInt(S, SGInt);
   Base256StringToFGInt(M, MGInt);
   FGIntMod(MGInt, n, temp);
   FGIntCopy(temp, MGInt);
   FGIntMontgomeryModExp(SGInt, e, n, temp);
   FGIntCopy(temp, SGInt);
   valid := (FGIntCompareAbs(SGInt, MGInt) = Eq);
   FGIntDestroy(SGInt);
   FGIntDestroy(MGInt);
End;
```

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм Lucifer. Алгоритм Lucifer являє собою мережу перестановок і підстановок, його основні блоки нагадують блоки алгоритму DES. В DES результат функції f складається операцією XOR із входом попереднього раунду, утворюючи вхід наступного раунду. В S-блоках алгоритму Lucifer 4-бітові входи й виходи, вхід S-блоків являє собою перетасований вихід S-блоків попереднього раунду, входом S-блоків першого раунду служить відкритий текст. Для вибору використовуваного S-блоку із двох можливих використовується біт ключа. (Lucifer реалізує все це в єдиному T-блоці з 9 бітами на вході й 8 бітами на виході). На відміну від алгоритму DES, половини блоку між раундами не переставляються, та й саме поняття половини блоку в алгоритмі Lucifer не використовується. У цього алгоритму 16 раундів, 128-бітові блоки й більше проста, чим в DES, схема розгорнення ключа.

Блок тексту розглядається як ненегативне ціле число, або як кілька незалежних ненегативних цілих чисел. Довжина блоку завжди вибирається

рівною ступеню двійки. У алгоритмі Lucifer використовуються наступні типи операцій:

– Таблична підстановка, при якій група біт відображається в іншу групу біт. Це так звані S-box.

– Переміщення, за допомогою якого біти повідомлення переупорядковуються.

– Операція додавання по модулю 2, позначувана XOR або ⊕.

– Операція додавання по модулю 2^{32} або по модулю 2^{16}.

– Циклічне зрушення на деяке число біт.

Ці операції циклічно повторюються в алгоритмі, створюючи так звані раунди. Входом кожного раунду є вихід попереднього раунду й ключ, що отриманий по певному алгоритму із ключа шифрування K. Ключ раунду називається підключем. Алгоритм шифрування може бути представлений у такий спосіб:

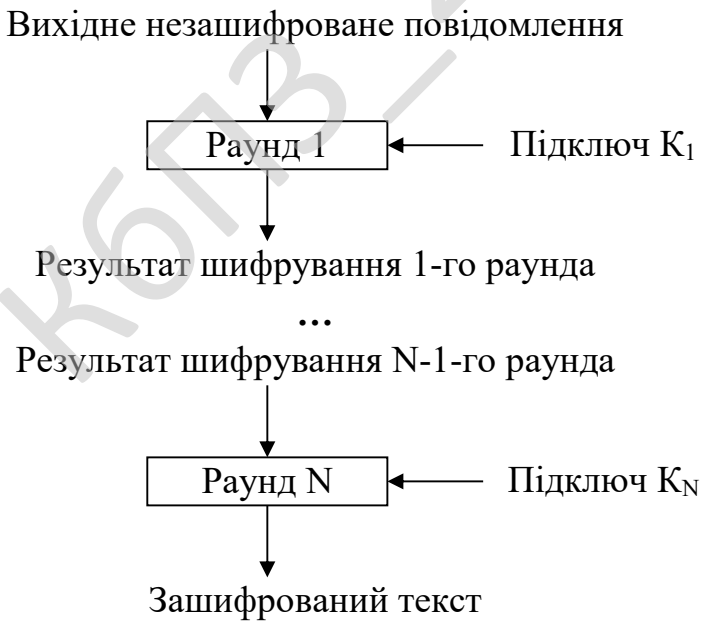

Рисунок 4.3 – Структура алгоритму алгоритмі Lucifer

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображене головне вікно програми. Як видно на цьому рисунку, до інтерфейсу користувача входять наступні блоки:

– Блок меню.

– Блок вікон відображення файлової структури.

– Блок кнопок швидкого доступу до елементів управління.

– Блок вікон відображення інформації про вибраний об’єкт.

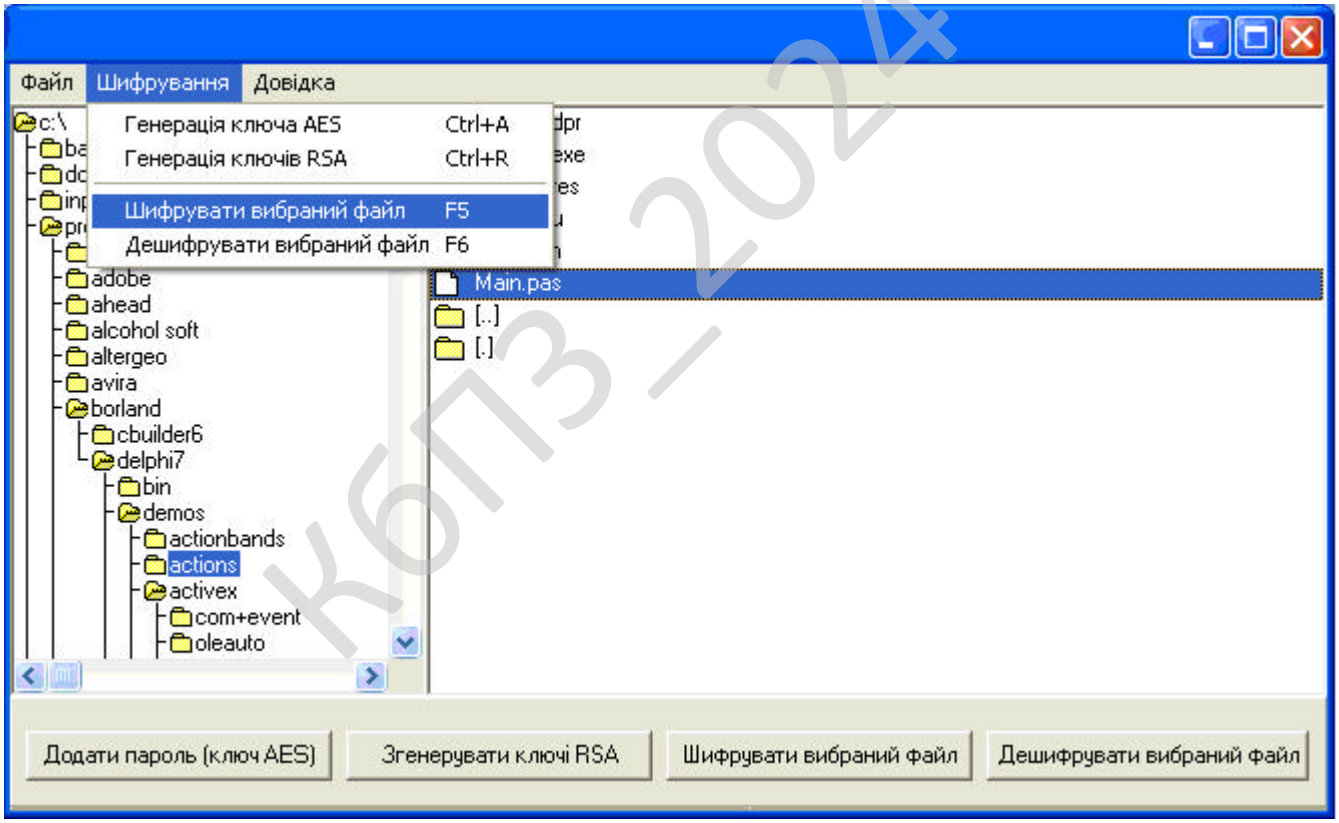

Рисунок 5.1 – Головне вікно програми

Блок меню складається із наступних елементів:

– Файл.

– Шифрування.

– Довідка.

За допомогою меню «Шифрування» можна виконати наступні операції:

– Генерація ключів AES (також операція доступна за допомогою швидких клавіш Ctrl+A).

– Генерація ключів RSA (клавіші Ctrl+R).

– Шифрувати вибраний файл (F5).

– Дешифрувати вибраний файл (F6).

Блок вікон відображення файлової структури містить:

– Вікно із деревом каталогів.

– Вікно із відображенням вмісту поточного каталогу.

– Панель вкладок для швидкого доступу до дисків.

За допомогою блоку кнопок швидкого доступу до елементів управління можливо виконати наступні операції:

– Додати пароль (ключ AES).

– Згенерувати ключі RSA.

– Шифрувати вибраний файл.

– Дешифрувати вибраний файл.

Блок вікон відображення інформації про вибраний об’єкт містить два вікна, в яких відображається інформація:

– повний шлях вибраного каталогу;

– вміст вибраного каталогу.

На рисунку 5.3 зображено вікно генерації ключів RSA. Це вікно містить наступні поля:

– поля E та N для відображення відкритого ключа RSA;

– поле D для відображення закритого ключа RSA;

– поле для ключа AES;

– поле для зашифрованого ключа AES.

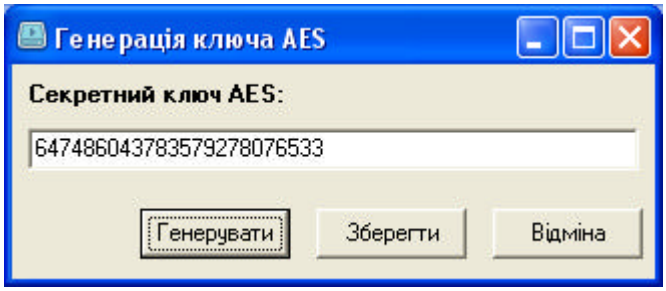

Рисунок 5.2 – Створення секретного ключа AES

На рисунку 5.3 зображено вікно генерації секретного ключа AES.

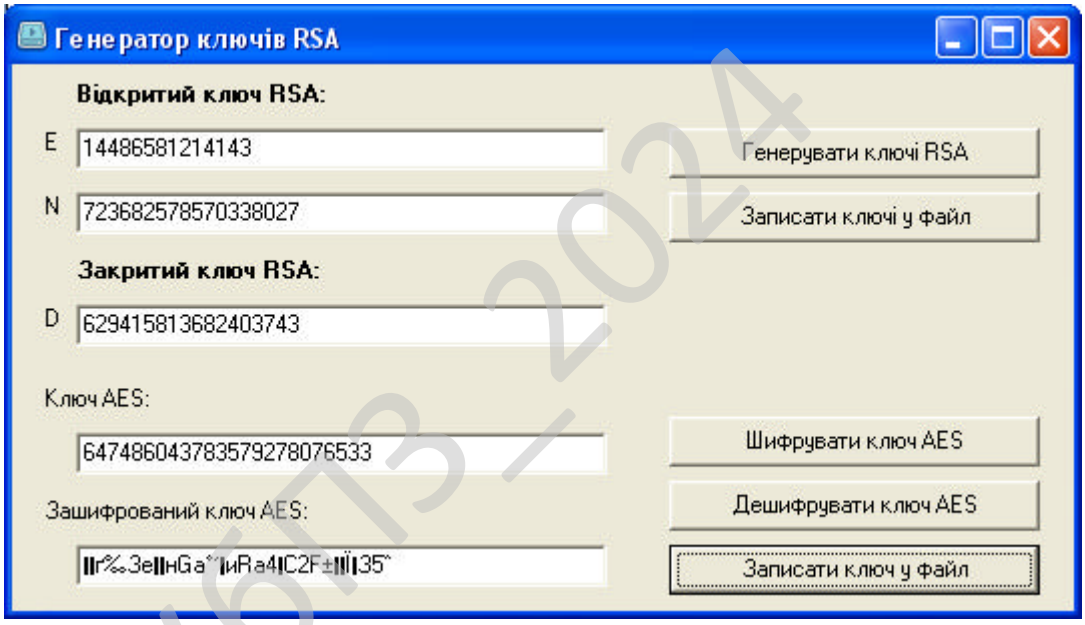

Рисунок 5.3 – Генерація ключів RSA

За допомогою кнопок у вікні генерації ключів RSA можливо здійснити наступні операції:

– Генерувати ключі RSA.

– Записати ключі у файл.

– Шифрувати ключ AES.

– Дешифрувати ключ AES.

– Записати ключ у файл.

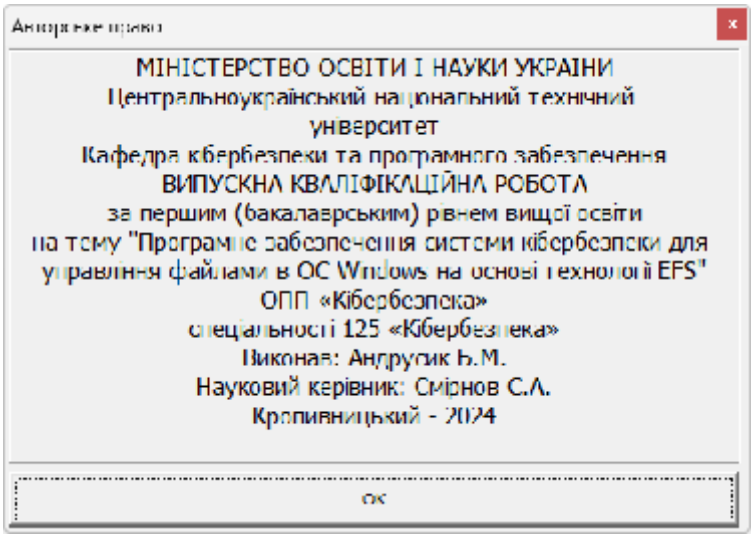

Рисунок 5.4 – Довідка

На рисунку 5.4 зображено вікно довідки про програму. В цьому вікні подається інформація про:

– тему бакалаврського проекту;

– керівника;

– розробника;

– дані про дату та місце розробки.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки для управління файлами в ОС Windows на основі технології EFS.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем для управління файлами в ОС Windows на основі технології EFS.

– Досліджена система для управління файлами в ОС Windows на основі технології EFS.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки для управління файлами в ОС Windows на основі технології EFS.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання для управління файлами в ОС Windows на основі технології EFS.

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 RAD Studio Delphi. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи кібербезпеки для управління файлами в ОС Windows на основі

технології EFS. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи кібербезпеки й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм Lucifer.

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. Vijay Kumar Velu. Mastering Kali Linux for Advanced Penetration Testing. Packt Publishing Ltd. 2022. 573 p.

2. Josh Armitage. Cloud Native Security Cookbook. O'Reilly Media. 2022. 516 p.

3. Massimo Bertaccini. Cryptography Algorithms. Packt Publishing. 2022. 358 p.

4. Alyssa Miller. Cybersecurity Career Guide. Manning Publications. 2022. 368 p.

5. Awais Rashid, Howard Chivers, George Danezis, Emil Lupu, Andrew Martin. CyBOK The Cyber Security Body of Knowledge. The National Cyber Security Centre. 2019. 854 p.

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

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

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

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

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

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

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

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

14. Kuznetsov, O., Kuznetsova, Y., Smirnov, O., Kostenko, O., Zvieriev, V. «Evaluating Hashing Algorithms in the Age of ASIC Resistance». *CEUR Workshop Proceedings*, 2023, 3628, pp. 93-105.

15. Kuznetsov O., Frontoni E., Kuznetsova Ye., Smirnov O., Chevardin V. «Achieving Enhanced Security in Biometric Authentication: A Rigorous Analysis of Code-Based Fuzzy Extractor». *CEUR Workshop Proceedings,* Volume 3624, 2023, pp. 330-339.

16. 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.

17. 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.

18. 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,

19. 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.

20. 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.

21. 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

22. 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.

23. 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.

24. 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.*

25. 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.

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

27. 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.

28. 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.

29. 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.

30. 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.

31. 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.

32. 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.

33. 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.

34. 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.

35. 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.

36. 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.

37. 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.

38. 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.

39. 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.

40. 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.

41. 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.

42. 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.

43. 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.

44. 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.

45. 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.

46. 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.

47. 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.

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

49. Козлов Я.О., Смірнова Т.В., Смірнов О.А. «Дослідження SIEM-систем для забезпечення кібербезпеки». *VII міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології" до 30-ти річчя*

кафедри кібербезпеки та програмного забезпечення, м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 26.

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

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

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

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

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

Додаток А
(обов’язковий)

Технічне завдання

Зміст

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

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки для управління файлами в ОС Windows на основі технології EFS.

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. № 135-02 від 01.04.2024 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи кібербезпеки для управління файлами в ОС Windows на основі технології EFS.

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 для управління файлами в ОС Windows на основі технології EFS;

– 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 RAD Studio Delphi.

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 – 77 аркушів.

8 Етапи розробки

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

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

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

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

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 4.06.2024 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Смірнов С.А.

Програмне забезпечення системи кібербезпеки для управління файлами в ОС Windows на основі технології EFS

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 59

Літера: РП

Кропивницький – 2024 року

Файл FmxUtils.pas – робота з файлами

```
unit FmxUtils;

interface

uses SysUtils, Windows, Classes, Consts;

type
  EInvalidDest = class(EStreamError);
  EFCantMove = class(EStreamError);

procedure CopyFile(const FileName, DestName: string);
procedure MoveFile(const FileName, DestName: string);
function GetFileSize(const FileName: string): LongInt;
function FileDateTime(const FileName: string): TDateTime;
function HasAttr(const FileName: string; Attr: Word): Boolean;
function ExecuteFile(const FileName, Params, DefaultDir: string;
  ShowCmd: Integer): THandle;

implementation

uses Forms, ShellAPI, RtlConsts;

const
  SInvalidDest = 'Вказаний каталог %s не існує';
  SFCantMove = 'Не можу перемістити файл %s';

procedure CopyFile(const FileName, DestName: string);
var
  CopyBuffer: Pointer; { буфер для копіювання }
  BytesCopied: Longint;
  Source, Dest: Integer; { заголовки }
  Len: Integer;
  Destination: TFileName; { розширене місце розташування }
const
  ChunkSize: Longint = 8192; { копіювання в 8K блок }
begin
  Destination := ExpandFileName(DestName); { розширення шляху призначення }
  if HasAttr(Destination, faDirectory) then { якщо розширення є директорією... }
  begin
    Len :=  Length(Destination);
    if Destination[Len] = '\' then
      Destination := Destination + ExtractFileName(FileName) { ...clone file
name }
    else
      Destination := Destination + '\' + ExtractFileName(FileName); {
...клонуємо ім’я файлу }
  end;
GetMem(CopyBuffer, ChunkSize); { виділяємо пам’ять під буфер }
  try
    Source := FileOpen(FileName, fmShareDenyWrite); { відкриваємо файл джерела}
    if Source < 0 then raise EFOpenError.CreateFmt(SFOpenError, [FileName]);
    try
      Dest := FileCreate(Destination); { створюємо вихідний файл або
перезаписуємо }
      if Dest < 0 then raise EFCreateError.CreateFmt(SFCreateError,
[Destination]);
      try
        repeat
          BytesCopied := FileRead(Source, CopyBuffer^, ChunkSize); { читаємо
блок }
          if BytesCopied > 0 then { якщо ми прочитали... }
            FileWrite(Dest, CopyBuffer^, BytesCopied); { ...записуємо блок }
        until BytesCopied < ChunkSize; { працюємо поки не закінчуються блоки }
      finally
```

```
        FileClose(Dest); { закриваємо файл розширення }
      end;
    finally
      FileClose(Source); { закриваємо файл джерела }
    end;
  finally
    FreeMem(CopyBuffer, ChunkSize); { звільнення буфера }
  end;
end;


{ Процедура переміщення файла }

procedure MoveFile(const FileName, DestName: string);
var
  Destination: string;
begin
  Destination := ExpandFileName(DestName); { розширюємо шлях призначення  }
  if not RenameFile(FileName, Destination) then { переіменовуємо }
  begin
    if HasAttr(FileName, faReadOnly) then  { якщо тільки для читання... }
      raise EFCantMove.Create(Format(SFCantMove, [FileName])); { не в змозі
знищити }
      CopyFile(FileName, Destination); { копіюємо в буфер...}
//      DeleteFile(FileName); { ...та знищуємо оригінал }
  end;
end;

{ GetFileSize функція, повертає розмір файлу }

function GetFileSize(const FileName: string): LongInt;
var
  SearchRec: TSearchRec;
begin
  try
    if FindFirst(ExpandFileName(FileName), faAnyFile, SearchRec) = 0 then
      Result := SearchRec.Size
    else Result := -1;
  finally
    SysUtils.FindClose(SearchRec);
  end;
end;

function FileDateTime(const FileName: string): System.TDateTime;
begin
  Result := FileDateToDateTime(FileAge(FileName));
end;

function HasAttr(const FileName: string; Attr: Word): Boolean;
var
 FileAttr: Integer;
begin
  FileAttr := FileGetAttr(FileName);
  if FileAttr = -1 then FileAttr := 0;
  Result := (FileAttr and Attr) = Attr;
end;

function ExecuteFile(const FileName, Params, DefaultDir: string;
  ShowCmd: Integer): THandle;
var
  zFileName, zParams, zDir: array[0..79] of Char;
begin
  Result := ShellExecute(Application.MainForm.Handle, nil,
    StrPCopy(zFileName, FileName), StrPCopy(zParams, Params),
    StrPCopy(zDir, DefaultDir), ShowCmd);
end;

end.
```

Файл filmanex.dpr основної програми

```
program FilManEx;

uses
  Forms,
  FMXWin in 'FMXWin.pas' {FMForm},
  FmxUtils in 'Fmxutils.pas',
  FAttrDlg in 'FAttrDlg.pas' {FileAttrForm},
  FChngDlg in 'FChngDlg.pas' {ChangeDlg},
  about in 'about.pas' {Form1},
  FGInt in 'FGInt.pas',
  FGIntPrimeGeneration in 'FGIntPrimeGeneration.PAS',
  FGIntRSA in 'FGIntRSA.PAS',
  RSA_keys in 'RSA_keys.pas' {RSA_keys},
  AES_key in 'AES_key.pas' {AES};

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TFMForm, FMForm);
  Application.CreateForm(TFileAttrForm, FileAttrForm);
  Application.CreateForm(TChangeDlg, ChangeDlg);
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(TRSA_keys, RSA_keys);
  Application.CreateForm(TDES, AES);
  Application.Run;
end.
```

Файл AES.pas – алгоритм шифрування AES

```
unit AES;

interface

Uses Windows, Classes, SysUtils, Math, Dialogs;

Type
  TBitString = Array of Boolean;
  PBitString = ^TBitString;

  TSplitKeyParts = record
    C:TBitString;
    D:TBitString;
    end;
  TSplitKey = Array[0..16]Of TSplitKeyParts;

  TConcatKey = Array[0..15]Of TBitString;

  TIPKeyParts = record
    L:TBitString;
    R:TBitString;
    end;
  TIPKey = Array[0..16]OF TIPKeyParts;

//матриці перестановок

Const
AES_PC1:Array[0..55] Of Byte = (57,49,41,33,25,17,9,
                                1,58,50,42,34,26,18,
                                10,2,59,51,43,35,27,
                                19,11,3,60,52,44,36,
                                63,55,47,39,31,23,15,
                                7,62,54,46,38,30,22,
                                14,6,61,53,45,37,29,
                                21,13,5,28,20,12,4);

AES_PC2:Array[0..47] Of Byte = (14,17,11,24,1,5,
                                3,28,15,6,21,10,
                                23,19,12,4,26,8,
                                16,7,27,20,13,2,
                                41,52,31,37,47,55,
                                30,40,51,45,33,48,
                                44,49,39,56,34,53,
                                46,42,50,36,29,32);

AES_IP:Array[0..63] Of Byte = (58,50,42,34,26,18,10,2,
                               60,52,44,36,28,20,12,4,
                               62,54,46,38,30,22,14,6,
                               64,56,48,40,32,24,16,8,
                               57,49,41,33,25,17,9,1,
                               59,51,43,35,27,19,11,3,
                               61,53,45,37,29,21,13,5,
                               63,55,47,39,31,23,15,7);

AES_E:Array[0..47] Of Byte = (32,1,2,3,4,5,
                              4,5,6,7,8,9,
                              8,9,10,11,12,13,
                              12,13,14,15,16,17,
                              16,17,18,19,20,21,
                              20,21,22,23,24,25,
                              24,25,26,27,28,29,
                              28,29,30,31,32,1);

// S-блоки
```

```
S_BOXES:Array[0..7,0..3,0..15]Of Byte = (
((14,04,13,01,02,15,11,08,03,10,06,12,05,09,00,07),
  (00,15,07,04,14,02,13,01,10,06,12,11,09,05,03,08),
  (04,01,14,08,13,06,02,11,15,12,09,07,03,10,05,00),
  (15,12,08,02,04,09,01,07,05,11,03,14,10,00,06,13)),

((15,01,08,14,06,11,03,04,09,07,02,13,12,00,05,10),
   (03,13,04,07,15,02,08,14,12,00,01,10,06,09,11,05),
   (00,14,07,11,10,04,13,01,05,08,12,06,09,03,02,15),
   (13,08,10,01,03,15,04,02,11,06,07,12,00,05,14,09)),

((10,00,09,14,06,03,15,05,01,13,12,07,11,04,02,08),
   (13,07,00,09,03,04,06,10,02,08,05,14,12,11,15,01),
   (13,06,04,09,08,15,03,00,11,01,02,12,05,10,14,07),
   (01,10,13,00,06,09,08,07,04,15,14,03,11,05,02,12)),

((07,13,14,03,00,06,09,10,01,02,08,05,11,12,04,15),
   (13,08,11,05,06,15,00,03,04,07,02,12,01,10,14,09),
   (10,06,09,00,12,11,07,13,15,01,03,14,05,02,08,04),
   (13,15,00,06,10,01,13,08,09,04,05,11,12,07,02,14)),

((02,12,04,01,07,10,11,06,08,05,03,15,13,00,14,09),
   (14,11,02,12,04,07,13,01,05,00,15,10,03,08,09,06),
   (04,02,01,11,10,13,07,08,15,09,12,05,06,03,00,14),
   (11,08,12,07,01,14,02,13,06,15,00,09,10,04,05,03)),

((12,01,10,15,09,02,06,08,00,13,03,04,14,07,05,11),
   (10,15,04,02,07,12,09,05,06,01,13,14,00,11,03,08),
   (09,14,15,05,02,08,12,03,07,00,04,10,01,13,11,06),
   (04,03,02,12,09,05,15,10,11,14,01,04,06,00,08,13)),

((04,11,02,14,15,00,08,13,03,12,09,07,05,10,06,01),
   (13,00,11,07,04,09,01,10,14,03,05,12,02,15,08,06),
   (01,04,11,13,12,03,07,14,10,15,06,08,00,05,09,02),
   (06,11,13,08,01,04,10,07,09,05,00,15,14,02,03,12)),

((13,02,08,04,06,15,11,01,10,09,03,14,05,00,12,07),
   (01,15,13,08,10,03,07,04,12,05,06,11,00,14,09,02),
   (07,11,04,01,09,12,14,02,00,06,10,13,15,03,05,08),
   (02,01,14,07,04,10,08,13,15,12,09,00,03,05,06,11))
);

AES_P:Array[0..31] Of Byte = (16,7,20,21,
                              29,12,28,17,
                              1,15,23,26,
                              5,18,31,10,
                              2,8,24,14,
                              32,27,3,9,
                              19,13,30,6,
                              22,11,4,25);

AES_REVERSE_IP:Array[0..63] Of Byte = (40,8,48,16,56,24,64,32,
            39,7,47,15,55,23,63,31,
            38,6,46,14,54,22,62,30,
            37,5,45,13,53,21,61,29,
            36,4,44,12,52,20,60,28,
            35,3,43,11,51,19,59,27,
            34,2,42,10,50,18,58,26,
            33,1,41,9,49,17,57,25);

AES_LSH:Array[0..15] Of Byte = (1,1,2,2,2,2,2,2,1,2,2,2,2,2,2,1);

Function BinToInt(S:TBitString):Integer;
Function IntToBin(N:Integer;Precision:Integer=8):TBitString;

Function BinToStr(Bits:TBitString):String;
Function StrToBin(S:String):TBitString;
```

```
Function AnsiStrToBin(S:String; Zeroes:Boolean=True):TBitString;
Function BinToAnsiStr(Bits:TBitString):String;

Procedure CopyBits(Var Dest:TBitString; Source:TBitString; NBits:Integer);
Function ConcatBits(Bits:Array Of TBitString):TBitString;

Function DESEncode(S,Key:String):TBitString;
Function DESDecode(S,Key:String):TBitString;

Function GetPermutedKey(Key:TBitString):TBitString;
Function GetPermutedKey2(Key:TBitString):TBitString;

Function GetSplitKey(Key:TBitString):TSplitKey;
Function GetConcatKey(Key:TSplitKey):TConcatKey;
Function GetIPKey(M:TBitString; ConcatKey:TConcatKey):TIPKey;
Function GetF(R,K:TBitString):TBitString;
Function GetSBox(Index:Integer; T:TBitString):TBitString;
Function GetReverseIP(RL:TBitString):TBitString;
Procedure ReverseSubKeys(Var Keys:TConcatKey);

implementation

Function ConcatBits(Bits:Array Of TBitString):TBitString;
Var
I,C:Integer;
Begin
SetLength(Result,0);
For C:=0 To Length(Bits)-1 Do
  Begin
  SetLength(Result,Length(Result)+Length(Bits[C]));
  For I:=0 To Length(Bits[C])-1 Do
    Result[Length(Result)-Length(Bits[C])+I]:=Bits[C][I];
  End;
End;

Procedure CopyBits(Var Dest:TBitString; Source:TBitString; NBits:Integer);
Var
I:Integer;
Begin
SetLength(Dest,NBits);
For I:=0 To NBits-1 Do
  Dest[I]:=Source[I];
End;

Function BinToInt(S: TBitString): Integer;
Var
L,I:Integer;
Begin
Result:=0;
L:=Length(S);
IF L=0 Then
  Raise EConvertError.Create(спеціальна бітова строка довжиною 0 біт');
For I:=L-1 DownTo 0 Do
  Result:=Result+Ord(S[I])*Trunc(Power(2,L-I-1));
End;

Function IntToBin(N:Integer; Precision:Integer):TBitString;
Var
BitList:TList;
Bit:PBoolean;
Begin
SetLength(Result,0);
BitList:=TList.Create;
While N>0 Do
  Begin
  New(Bit);
  Bit^:=Boolean(N mod 2);
  BitList.Insert(0,Bit);
  N:=N div 2;
```

```
  End;
While BitList.Count<Precision Do
  Begin
  New(Bit);
  Bit^:=False;
  BitList.Insert(0,Bit);
  End;
For N:=0 To BitList.Count-1 Do
  Begin
  SetLength(Result,N+1);
  Bit:=BitList.Items[N];
  Result[N]:=Bit^;
  Dispose(Bit);
  End;
BitList.Free;
end;

Function AnsiStrToBin(S: String; Zeroes:Boolean):TBitString;
Var
Temp,B:TBitString;
L,I,J:Integer;
Begin
L:=0;
SetLength(Result,L);
SetLength(Temp,L);
SetLength(B,0);
For I:=1 To Length(S) Do
  Begin
  B:=IntToBin(Ord(S[I]));
  L:=L+Length(B);
  SetLength(Temp,L);
  For J:=0 To Length(B)-1 Do
    Temp[Length(Temp)-Length(B)+J]:=B[J];
  End;
Result:=Temp;
End;

Function BinToStr(Bits:TBitString):String;
Var
I,L:Integer;
Begin
Result:='';
L:=Length(Bits);
IF L=0 Then
  Raise EConvertError.Create('Спеціальна бітова строка довжиною 0 біт');
For I:=0 To L-1 Do
  IF Bits[I] Then Result:=Result+'1'
  Else Result:=Result+'0';
End;

Function StrToBin(S:String):TBitString;
Var
I:Integer;
Begin
SetLength(Result,0);
For I:=1 To Length(S) Do
  Begin
  IF (S[I]<>'1')And(S[I]<>'0') Then
    Raise EConvertError.Create(S+' некоректна бінарна строка');
  SetLength(Result,I);
  Result[I-1]:=Boolean(StrToInt(S[I]));
  End;
End;

Function BinToAnsiStr(Bits:TBitString):String;
Var
I:Integer;
B:TBitString;
Begin
```

```
Result:='';
SetLength(B,8);
I:=0;
While I<=Length(Bits)-8 Do
  Begin
  CopyMemory(B,Ptr(Integer(Bits)+I),8);
  Result:=Result+Char(BinToInt(B));
  Inc(I,8);
  End;
End;

Function GetPermutedKey(Key:TBitString):TBitString;
Var
I:Integer;
Begin
SetLength(Result,Length(AES_PC1));
For I:=0 To Length(AES_PC1)-1 Do
  Result[I]:=Key[AES_PC1[I]-1];
End;

Function GetPermutedKey2(Key:TBitString):TBitString;
Var
I:Integer;
Begin
SetLength(Result,Length(AES_PC2));
For I:=0 To Length(AES_PC2)-1 Do
  Result[I]:=Key[AES_PC2[I]-1];
End;

Function GetSplitKey(Key:TBitString):TSplitKey;
  Function LeftShift(Key:TBitString; N:Integer):TBitString;
  Var
  I,J:Integer;
  Temp:TBitString;
  Begin
  SetLength(Result,28);
  SetLength(Temp,28);
  For I:=0 To 27 Do
    Temp[I]:=Key[I];
  For J:=1 To N Do
    Begin
    For I:=1 To 27 Do
      Result[I-1]:=Temp[I];
    Result[27]:=Temp[0];
    For I:=0 To 27 Do
      Temp[I]:=Result[I];
    End;
  End;
Var
I,J:Integer;
Begin
For J:=1 To 16 Do
  Begin
  SetLength(Result[J].C,28);
  SetLength(Result[J].D,28);
  End;
CopyBits(Result[0].C,Key,28);
CopyBits(Result[0].D,TBitString(Integer(Key)+28),28);
For I:=1 To 16 Do
  Begin
  Result[I].C:=LeftShift(Result[I-1].C,AES_LSH[I-1]);
  Result[I].D:=LeftShift(Result[I-1].D,AES_LSH[I-1]);
  End;
End;

Function GetConcatKey(Key:TSplitKey):TConcatKey;
Var
I:Integer;
Temp:TBitString;
```

```
Begin
For I:=0 To 15 Do
  Begin
  SetLength(Result[I],56);
  Temp:=ConcatBits([Key[I+1].C,Key[I+1].D]);
  Result[I]:=GetPermutedKey2(Temp);
  End;
End;

Function GetIPKey(M:TBitString; ConcatKey:TConcatKey):TIPKey;
Var
I,J:Integer;
IP, F:TBitString;
Begin
For I:=0 To 16 Do
  Begin
  SetLength(Result[I].L,32);
  SetLength(Result[I].R,32);
  End;

SetLength(IP,64);
For I:=0 To Length(AES_IP)-1 Do
  IP[I]:=M[AES_IP[I]-1];

For I:=0 To 31 Do
  Result[0].L[I]:=IP[I];
For I:=32 To 63 Do
  Result[0].R[I-32]:=IP[I];

For I:=1 To 16 Do
  Begin
  Result[I].L:=Result[I-1].R;
  F:=GetF(Result[I-1].R,ConcatKey[I-1]);
  For J:=0 To 31 Do
    Result[I].R[J]:=Result[I-1].L[J] XOR F[J];
  End;
End;

Function GetF(R,K:TBitString):TBitString;
Var
I,J:Integer;
S,E,KE,F,T:TBitString;
Begin
SetLength(E,48);
For I:=0 To 47 Do
  E[I]:=R[AES_E[I]-1];

SetLength(KE,48);
For I:=0 To 47 Do
  KE[I]:=K[I] XOR E[I];

SetLength(T,6);
SetLength(F,0);
SetLength(S,4);
I:=0;
While I<48 Do
  Begin
  For J:=0 To 6 Do
    T[J]:=KE[J+I];
  S:=GetSBox(I div 6,T);
  F:=ConcatBits([F,S]);
  I:=I+6;
  End;
SetLength(Result,32);
For I:=0 To 31 Do
  Result[I]:=F[AES_P[I]-1];
End;

Function GetSBox(Index:Integer; T:TBitString):TBitString;
```

```
Var
Val,Row,Col:Integer;
Temp:TBitString;
Begin
SetLength(Result,4);
SetLength(Temp,2);
Temp[0]:=T[0];
Temp[1]:=T[5];
Row:=BinToInt(Temp);
SetLength(Temp,4);
CopyBits(Temp,TBitString(@T[1]),4);
Col:=BinToInt(Temp);
Val:=S_BOXES[Index,Row,Col];
SetLength(Result,4);
Result:=IntToBin(Val,4);
End;

Function GetReverseIP(RL:TBitString):TBitString;
Var
I:Integer;
Begin
SetLength(Result,64);
For I:=0 To Length(AES_REVERSE_IP)-1 Do
  Result[I]:=RL[AES_REVERSE_IP[I]-1];
End;

Procedure ReverseSubKeys(Var Keys:TConcatKey);
Var
I,L:Integer;
T:TBitString;
Begin
SetLength(T,48);
L:=Length(Keys);
For I:=0 To (L-1)Div 2 Do
  Begin
  T:=Keys[I];
  Keys[I]:=Keys[(L-I)-1];
  Keys[(L-I)-1]:=T;
  End;
End;

Function DESEncode(S,Key:String):TBitString;
Var
I:Integer;
K:TBitString;
M:TBitString;
RL:TBitString;
Kplus:TBitString;
SplitKey:TSplitKey;
ConcatKey:TConcatKey;
IPKey:TIPKey;
Begin
K:=AnsiStrToBin(Key);
Kplus:=GetPermutedKey(K);
SplitKey:=GetSplitKey(Kplus);
ConcatKey:=GetConcatKey(SplitKey);
M:=AnsiStrToBin(S);
IPKey:=GetIPKey(M,ConcatKey);
SetLength(RL,64);
For I:=0 To 31 Do
  Begin
  RL[I]:=IPKey[16].R[I];
  RL[I+32]:=IPKey[16].L[I];
  End;
RL:=GetReverseIP(RL);
Result:=RL;
End;

Function DESDecode(S,Key:String):TBitString;
```

```
Var
I:Integer;
K:TBitString;
M:TBitString;
RL:TBitString;
Kplus:TBitString;
SplitKey:TSplitKey;
ConcatKey:TConcatKey;
IPKey:TIPKey;
Begin
K:=AnsiStrToBin(Key);
Kplus:=GetPermutedKey(K);
SplitKey:=GetSplitKey(Kplus);
ConcatKey:=GetConcatKey(SplitKey);
ReverseSubKeys(ConcatKey);
M:=AnsiStrToBin(S);
IPKey:=GetIPKey(M,ConcatKey);
SetLength(RL,64);
For I:=0 To 31 Do
  Begin
  RL[I]:=IPKey[16].R[I];
  RL[I+32]:=IPKey[16].L[I];
  End;
RL:=GetReverseIP(RL);
Result:=RL;
End; end.
```

Файл FMXWin.pas – вивід дерева файлів та каталогів, шифрування, дешифрування

```
unit FMXWin;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, FileCtrl, Grids, Outline, DirOutln, Tabs, ExtCtrls, Menus, about,
RSA_keys, AES_key;

type
  TFMForm = class(TForm)
    StatusBar: TPanel;
    DirectoryPanel: TPanel;
    FilePanel: TPanel;
    DriveTabSet: TTabSet;
    DirectoryOutline: TDirectoryOutline;
    FileList: TFileListBox;
    MainMenu1: TMainMenu;
    File1: TMenuItem;
    Open1: TMenuItem;
    Move1: TMenuItem;
    Copy1: TMenuItem;
    Delete1: TMenuItem;
    Rename1: TMenuItem;
    Properties1: TMenuItem;
    N1: TMenuItem;
    Exit1: TMenuItem;
    Floppy: TImage;
    Fixed: TImage;
    Network: TImage;
    CDRom: TImage;
    RamDisk: TImage;
    N2: TMenuItem;
    N3: TMenuItem;
    N4: TMenuItem;
    N5: TMenuItem;
    Panel1: TPanel;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    RSA1: TMenuItem;
    N6: TMenuItem;
    N7: TMenuItem;
    N8: TMenuItem;
    procedure Exit1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure DirectoryOutlineChange(Sender: TObject);
    procedure FileListChange(Sender: TObject);
    procedure DriveTabSetMeasureTab(Sender: TObject; Index: Integer;
      var TabWidth: Integer);
    procedure DriveTabSetDrawTab(Sender: TObject; TabCanvas: TCanvas;
      R: TRect; Index: Integer; Selected: Boolean);
    procedure File1Click(Sender: TObject);
    procedure Delete1Click(Sender: TObject);
    procedure Properties1Click(Sender: TObject);
    procedure FileChange(Sender: TObject);
    procedure Open1Click(Sender: TObject);
    procedure FileListMouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure DirectoryOutlineDragOver(Sender, Source: TObject; X,
      Y: Integer; State: TDragState; var Accept: Boolean);
    procedure DirectoryOutlineDragDrop(Sender, Source: TObject; X,
      Y: Integer);
    procedure FileListEndDrag(Sender, Target: TObject; X, Y: Integer);
```

```
    procedure DriveTabSetChange(Sender: TObject; NewTab: Integer;
      var AllowChange: Boolean);
    procedure N4Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);

  private
    procedure ConfirmChange(const ACaption, FromFile, ToFile: string);
  public
    { Public declarations }
  end;

var
  FMForm: TFMForm;

implementation

uses FmxUtils, FAttrDlg, FChngDlg;

{$R *.dfm}

procedure TFMForm.Exit1Click(Sender: TObject);
begin
  Close;
end;

procedure TFMForm.FormCreate(Sender: TObject);
var
  Drive: Char;
  AddedIndex: Integer;
begin
//Вибір диску
  for Drive := 'a' to 'z' do
  begin
    case GetDriveType(PChar(Drive + ':\')) of
      DRIVE_REMOVABLE:
        AddedIndex := DriveTabSet.Tabs.AddObject(Drive, Floppy.Picture.Graphic);
      DRIVE_FIXED:
        AddedIndex := DriveTabSet.Tabs.AddObject(Drive, Fixed.Picture.Graphic);
      DRIVE_CDROM:
        AddedIndex := DriveTabSet.Tabs.AddObject(Drive, CDRom.Picture.Graphic);
      DRIVE_RAMDISK:
        AddedIndex := DriveTabSet.Tabs.AddObject(Drive,
RamDisk.Picture.Graphic);
      DRIVE_REMOTE:
        AddedIndex := DriveTabSet.Tabs.AddObject(Drive,
Network.Picture.Graphic);
    else
      AddedIndex := 0;
    end;
    if UpCase(Drive) = FileList.Drive then
      DriveTabSet.TabIndex := AddedIndex;
  end;
end;

procedure TFMForm.DriveTabSetChange(Sender: TObject; NewTab: Integer;
  var AllowChange: Boolean);
begin
  if not (csDesigning in ComponentState) then
  begin
    AllowChange := True;
    try
      with DriveTabSet do
        DirectoryOutline.Drive := Tabs[NewTab][1];
    except
      on EInOutError do
      begin
```

```
        AllowChange := False;
        with DriveTabSet do
          DirectoryOutline.Drive := Tabs[TabIndex][1];
        raise;
      end;
    end;
  end;
end;

procedure TFMForm.DirectoryOutlineChange(Sender: TObject);
begin
  FileList.Directory := DirectoryOutline.Directory;
  DirectoryPanel.Caption := DirectoryOutline.Directory;
end;

procedure TFMForm.FileListChange(Sender: TObject);
var
  TheFileName: string;
begin
  with FileList do
  begin
    if ItemIndex >= 0 then
    begin
      TheFileName := Items[ItemIndex];
      FilePanel.Caption := Format('Вибрано файл %s, %d байтів', [TheFileName,
GetFileSize(TheFileName)]);
    end
    else FilePanel.Caption := '';
  end;
end;

procedure TFMForm.DriveTabSetMeasureTab(Sender: TObject; Index: Integer;
  var TabWidth: Integer);
var
  BitmapWidth: Integer;
begin
  BitmapWidth := TBitmap(DriveTabSet.Tabs.Objects[Index]).Width;
  Inc(TabWidth, 2 + BitmapWidth);
end;

procedure TFMForm.DriveTabSetDrawTab(Sender: TObject; TabCanvas: TCanvas;
  R: TRect; Index: Integer; Selected: Boolean);
var
  Bitmap: TBitmap;
begin
  Bitmap := TBitmap(DriveTabSet.Tabs.Objects[Index]);
  with TabCanvas do
  begin
    Draw(R.Left, R.Top + 4, Bitmap);
    TextOut(R.Left + 2 + Bitmap.Width, R.Top + 2, DriveTabSet.Tabs[Index]);
  end;
end;

procedure TFMForm.File1Click(Sender: TObject);
var
  FileSelected: Boolean;
begin
  FileSelected := FileList.ItemIndex >= 0;
  Open1.Enabled := FileSelected;
  Delete1.Enabled := FileSelected;
  Copy1.Enabled := FileSelected;
  Move1.Enabled := FileSelected;
  Rename1.Enabled := FileSelected;
  Properties1.Enabled := FileSelected;
end;

procedure TFMForm.Delete1Click(Sender: TObject);
begin
  with FileList do
```

```
    if MessageDlg('Delete ' + FileName + '?', mtConfirmation,
      [mbYes, mbNo], 0) = mrYes then
      if DeleteFile(FileName) then Update;
end;

procedure TFMForm.Properties1Click(Sender: TObject);
var
  Attributes, NewAttributes: Word;
begin
  with FileAttrForm do
  begin
    FileDirName.Caption := FileList.Items[FileList.ItemIndex];
    FilePathName.Caption := FileList.Directory;
    ChangeDate.Caption := DateTimeToStr(FileDateTime(FileList.FileName));
    Attributes := FileGetAttr(FileDirName.Caption);
    ReadOnly.Checked := (Attributes and faReadOnly) = faReadOnly;
    Archive.Checked := (Attributes and faArchive) = faArchive;
    System.Checked := (Attributes and faSysFile) = faSysFile;
    Hidden.Checked := (Attributes and faHidden) = faHidden;
    if ShowModal <> mrCancel then
    begin
      NewAttributes := Attributes;
      if ReadOnly.Checked then NewAttributes := NewAttributes or faReadOnly
      else NewAttributes := NewAttributes and not faReadOnly;
      if Archive.Checked then NewAttributes := NewAttributes or faArchive
      else NewAttributes := NewAttributes and not faArchive;
      if System.Checked then NewAttributes := NewAttributes or faSysFile
      else NewAttributes := NewAttributes and not faSysFile;
      if Hidden.Checked then NewAttributes := NewAttributes or faHidden
      else NewAttributes := NewAttributes and not faHidden;
      if NewAttributes <> Attributes then
        FileSetAttr(FileDirName.Caption, NewAttributes);
    end;
  end;
end;

procedure TFMForm.ConfirmChange(const ACaption, FromFile, ToFile: string);
begin
  if MessageDlg(Format('%s %s to %s?', [ACaption, FromFile, ToFile]),
    mtConfirmation, [mbYes, mbNo], 0) = mrYes then
  begin
    if ACaption = 'Move' then
      MoveFile(FromFile, ToFile)
    else if ACaption = 'Copy' then
      CopyFile(FromFile, ToFile)
    else if ACaption = 'Rename' then
      RenameFile(FromFile, ToFile);
    FileList.Update;
  end;
end;

procedure TFMForm.FileChange(Sender: TObject);
begin
  with ChangeDlg do
  begin
    if Sender = Move1 then Caption := 'Move'
    else if Sender = Copy1 then Caption := 'Copy'
    else if Sender = Rename1 then Caption := 'Rename'
    else Exit;
    CurrentDir.Caption := DirectoryOutline.Directory;
    FromFileName.Text := FileList.FileName;
    ToFileName.Text := '';
    if (ShowModal <> mrCancel) and (ToFileName.Text <> '') then
      ConfirmChange(Caption, FromFileName.Text, ToFileName.Text);
  end;
end;

procedure TFMForm.Open1Click(Sender: TObject);
begin
```

```
  with FileList do
  begin
    if HasAttr(FileName, faDirectory) then
      DirectoryOutline.Directory := FileName
    else ExecuteFile(FileName, '', Directory, SW_SHOW);
  end;
end;

procedure TFMForm.FileListMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  if Button = mbLeft then
  with Sender as TFileListBox do
  begin
    if ItemAtPos(Point(X, Y), True) >= 0 then
      BeginDrag(False);
  end;
end;

procedure TFMForm.DirectoryOutlineDragOver(Sender, Source: TObject; X,
  Y: Integer; State: TDragState; var Accept: Boolean);
begin
  Accept := (Source is TFileListBox) and (DirectoryOutline.GetItem(X, Y) > 0);
end;

procedure TFMForm.DirectoryOutlineDragDrop(Sender, Source: TObject; X,
  Y: Integer);
begin
  if Source is TFileListBox then
    with DirectoryOutline do
      ConfirmChange('Move', FileList.FileName, Items[GetItem(X, Y)].FullPath);
end;

procedure TFMForm.FileListEndDrag(Sender, Target: TObject; X, Y: Integer);
begin
  if Target <> nil then FileList.Update;
end;

procedure TFMForm.N4Click(Sender: TObject);
begin
Form1.Show;
end;

procedure TFMForm.Button2Click(Sender: TObject);
begin
RSA_keys.Show;
end;

procedure TFMForm.Button1Click(Sender: TObject);
begin
AES.Show;
end;

procedure TFMForm.Button3Click(Sender: TObject);
I:Integer;
S:String;
Begin
SetLength(Data,0);
I:=1;
While I<=Length(fileX) Do
  Begin
  S:=Copy(fileX,I,8);
  Data:=ConcatBits([Data,DESEncode(S, fileX)]);
  I:=I+8;
  End;
fileY:=BinToAnsiStr(Data);
MessageBox(Handle,   'Файл зашифровано!',   'EFS',MB_OK    );
end;
```

```
procedure TFMForm.Button4Click(Sender: TObject);
I:Integer;
Begin
SetLength(Data,0);
I:=1;
While I<=Length(fileY) Do
  Begin
Data:=ConcatBits([Data,DESDecode(Copy(fileY,I,8), fileX)]);
  I:=I+8;
  End;
fileX:=BinToAnsiStr(Data);
MessageBox(Handle,   'Файл розшифровано!',   'EFS',MB_OK   );
end;

end.
```

Файл AES_key.pas – введення користувачем секретного ключа AES

```
unit AES_key;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls;

type
  TDES = class(TForm)
    EditDES: TEdit;
    Button1: TButton;
    Button2: TButton;
    Label1: TLabel;
    Label2: TLabel;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  AES: TDES;

implementation

{$R *.dfm}

procedure TDES.Button1Click(Sender: TObject);
begin
RSA_keys.Edit1.Text:=Edit1.Text;
AES.Close;
end;

end.
```

Файл RSA_keys.pas – генерація ключів RSA та шифрування ключа AES

```
unit RSA_keys;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, FGInt, FGIntPrimeGeneration, FGIntRSA, StdCtrls, AES_key;

type
  TRSA_keys = class(TForm)
    Button1: TButton;
    Label3: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    Edit3: TEdit;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    Edit_E: TEdit;
    Edit_N: TEdit;
    Edit_D: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label4: TLabel;
    Button5: TButton;
    Edit1: TEdit;
    Label122: TLabel;
    Label12: TLabel;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  RSA_keys: TRSA_keys;
   n, e, d, dp, dq, p, q, phi, one, two, gcd, temp, nilgint : TFGInt;
   test, signature : String;
   ok : boolean;
   st: string;
   tx1: text;
   tx2: text;
   tx3: text;
implementation



{$R *.dfm}

procedure TRSA_keys.Button1Click(Sender: TObject);
begin

// Генерація p та q
```

```
   Base256StringToFGInt('3557', p);
   Base256StringToFGInt('2579', q);
   PrimeSearch(p);
   PrimeSearch(q);
   FGIntToBase256String(p, st);
   FGIntToBase256String(q, st);


 // Обчислення N

   FGIntMul(p, q, n);
   p.Number[1] := p.Number[1] - 1;
   q.Number[1] := q.Number[1] - 1;
   FGIntMul(p, q, phi);
   FGIntToBase10String(n, st);
   Edit_N.Text:=st;


// Обчислення E – ключ шифрування

//   Base10StringToFGInt('65537', e); // непарне число
//   Base10StringToFGInt('8171921', e);
   Base10StringToFGInt('14486581214143', e);
   Base10StringToFGInt('1', one);
   Base10StringToFGInt('2', two);
   FGIntGCD(phi, e, gcd);
   While FGIntCompareAbs(gcd, one) <> Eq Do
   Begin
      FGIntadd(e, two, temp);
      FGIntCopy(temp, e);
      FGIntGCD(phi, e, gcd);
   End;
   FGIntDestroy(two);
   FGIntDestroy(one);
   FGIntDestroy(gcd);

// Обчислення D – ключ дешифрування

   FGIntModInv(e, phi, d);
   FGIntModInv(e, p, dp);
   FGIntModInv(e, q, dq);
   p.Number[1] := p.Number[1] + 1;
   q.Number[1] := q.Number[1] + 1;
   FGIntDestroy(phi);
   FGIntDestroy(nilgint);


   FGIntToBase10String(e, st);
   Edit_E.Text:=st;
   FGIntToBase10String(d, st);
   Edit_D.Text:=st;


end;

//дешифрування ключа AES

procedure TRSA_keys.Button2Click(Sender: TObject);
var prom: string;
begin
   test := Edit3.Text;
   RSAEncrypt(test, e, n, test);
   Edit1.Text:=test;
end;

//запис ключів RSA у файли

procedure TRSA_keys.Button3Click(Sender: TObject);
begin
```

```
  AssignFile(tx1,'OpenKey.keys');
  AssignFile(tx2,'CloseKey.keys');

  ReWrite(tx1); CloseFile(tx1);
  ReWrite(tx2); CloseFile(tx2);
  Append(tx1);
  Append(tx2);
  FGIntToBase256String(n, st);
  ConvertBase256to64(st, st);
  WriteLn(tx1,st);
  WriteLn(tx2,st);
  FGIntToBase256String(e, st);
  ConvertBase256to64(st, st);
  WriteLn(tx1,st);
  FGIntToBase256String(d, st);
  ConvertBase256to64(st, st);
  WriteLn(tx2,st);
  CloseFile(tx1);
  CloseFile(tx2);

end;

//шифрування ключа AES


procedure TRSA_keys.Button4Click(Sender: TObject);
begin
 RSADecrypt(test, d, n, Nilgint, Nilgint, Nilgint, Nilgint, test);

       Edit3.Text:=test;

end;


//Запис ключа AES у файл

procedure TRSA_keys.Button5Click(Sender: TObject);
begin
AssignFile(tx3,'SecretKey.keys');
  ReWrite(tx3); CloseFile(tx3);
  Append(tx3);
  WriteLn(tx3,Edit1.Text);
  CloseFile(tx3);
end;

procedure TRSA_keys.FormCreate(Sender: TObject);
begin
Edit1.Text:=AES.EditDES.Text;
end;

end.
```

Файл FGIntRSA.pas – алгоритм шифрування RSA

```
Unit FGIntRSA;

Interface

Uses Windows, SysUtils, Controls, FGInt;

Procedure RSAEncrypt(P : String; Var exp, modb : TFGInt; Var E : String);
Procedure RSADecrypt(E : String; Var exp, modb, d_p, d_q, p, q : TFGInt; Var D :
String);
Procedure RSASign(M : String; Var d, n, dp, dq, p, q : TFGInt; Var S : String);
Procedure RSAVerify(M, S : String; Var e, n : TFGInt; Var valid : boolean);


Implementation


{$H+}



// Шифруємо рядок алгоритмом RSA, P^exp mod modb = E

Procedure RSAEncrypt(P : String; Var exp, modb : TFGInt; Var E : String);
Var
   i, j, modbits : longint;
   PGInt, temp, zero : TFGInt;
   tempstr1, tempstr2, tempstr3 : String;
Begin
   Base2StringToFGInt('0', zero);
   FGIntToBase2String(modb, tempstr1);
   modbits := length(tempstr1);
   convertBase256to2(P, tempstr1);
   tempstr1 := '111' + tempstr1;
   j := modbits - 1;
   While (length(tempstr1) Mod j) <> 0 Do tempstr1 := '0' + tempstr1;

   j := length(tempstr1) Div (modbits - 1);
   tempstr2 := '';
   For i := 1 To j Do
   Begin
      tempstr3 := copy(tempstr1, 1, modbits - 1);
      While (copy(tempstr3, 1, 1) = '0') And (length(tempstr3) > 1) Do
delete(tempstr3, 1, 1);
      Base2StringToFGInt(tempstr3, PGInt);
      delete(tempstr1, 1, modbits - 1);
      If tempstr3 = '0' Then FGIntCopy(zero, temp) Else
FGIntMontgomeryModExp(PGInt, exp, modb, temp);
      FGIntDestroy(PGInt);
      tempstr3 := '';
      FGIntToBase2String(temp, tempstr3);
      While (length(tempstr3) Mod modbits) <> 0 Do tempstr3 := '0' + tempstr3;
      tempstr2 := tempstr2 + tempstr3;
      FGIntdestroy(temp);
   End;

   While (tempstr2[1] = '0') And (length(tempstr2) > 1) Do delete(tempstr2, 1,
1);
   ConvertBase2To256(tempstr2, E);
   FGIntDestroy(zero);
End;


Дешифруємо рядок алгоритмом RSA, E^exp mod modb = D
// modb = p*q, d_p*e mod (p-1) = 1
// d_q*e mod (q-1)
```

```
Procedure RSADecrypt(E : String; Var exp, modb, d_p, d_q, p, q : TFGInt; Var D :
String);
Var
   i, j, modbits : longint;
   EGInt, temp, temp1, temp2, temp3, ppinvq, qqinvp, zero : TFGInt;
   tempstr1, tempstr2, tempstr3 : String;
Begin
   Base2StringToFGInt('0', zero);
   FGIntToBase2String(modb, tempstr1);
   modbits := length(tempstr1);
   convertBase256to2(E, tempstr1);
   While copy(tempstr1, 1, 1) = '0' Do delete(tempstr1, 1, 1);
   While (length(tempstr1) Mod modbits) <> 0 Do tempstr1 := '0' + tempstr1;
   If exp.Number = Nil Then
   Begin
      FGIntModInv(q, p, temp1);
      FGIntMul(q, temp1, qqinvp);
      FGIntDestroy(temp1);
      FGIntModInv(p, q, temp1);
      FGIntMul(p, temp1, ppinvq);
      FGIntDestroy(temp1);
   End;

   j := length(tempstr1) Div modbits;
   tempstr2 := '';
   For i := 1 To j Do
   Begin
      tempstr3 := copy(tempstr1, 1, modbits);
      While (copy(tempstr3, 1, 1) = '0') And (length(tempstr3) > 1) Do
delete(tempstr3, 1, 1);
      Base2StringToFGInt(tempstr3, EGInt);
      delete(tempstr1, 1, modbits);
      If tempstr3 = '0' Then FGIntCopy(zero, temp) Else
      Begin
         If exp.Number <> Nil Then FGIntMontgomeryModExp(EGInt, exp, modb, temp)
Else
         Begin
            FGIntMontgomeryModExp(EGInt, d_p, p, temp1);
            FGIntMul(temp1, qqinvp, temp3);
            FGIntCopy(temp3, temp1);
            FGIntMontgomeryModExp(EGInt, d_q, q, temp2);
            FGIntMul(temp2, ppinvq, temp3);
            FGIntCopy(temp3, temp2);
            FGIntAddMod(temp1, temp2, modb, temp);
            FGIntDestroy(temp1);
            FGIntDestroy(temp2);
         End;
      End;
      FGIntDestroy(EGInt);
      tempstr3 := '';
      FGIntToBase2String(temp, tempstr3);
      While (length(tempstr3) Mod (modbits - 1)) <> 0 Do tempstr3 := '0' +
tempstr3;
      tempstr2 := tempstr2 + tempstr3;
      FGIntdestroy(temp);
   End;

   If exp.Number = Nil Then
   Begin
      FGIntDestroy(ppinvq);
      FGIntDestroy(qqinvp);
   End;
   While (Not (copy(tempstr2, 1, 3) = '111')) And (length(tempstr2) > 3) Do
delete(tempstr2, 1, 1);
   delete(tempstr2, 1, 3);
   ConvertBase2To256(tempstr2, D);
   FGIntDestroy(zero);
End;
```

```
// підписуємо рядок RSA, M^d mod n = S
// n = p*q, dp*e mod (p-1) = 1
// dq*e mod (q-1)


Procedure RSASign(M : String; Var d, n, dp, dq, p, q : TFGInt; Var S : String);
Var
   MGInt, SGInt, temp, temp1, temp2, temp3, ppinvq, qqinvp : TFGInt;
Begin
   Base256StringToFGInt(M, MGInt);
   If d.Number <> Nil Then FGIntMontgomeryModExp(MGInt, d, n, SGInt) Else
   Begin
      FGIntModInv(p, q, temp);
      FGIntMul(p, temp, ppinvq);
      FGIntDestroy(temp);
      FGIntModInv(q, p, temp);
      FGIntMul(q, temp, qqinvp);
      FGIntDestroy(temp);
      FGIntMontgomeryModExp(MGInt, dp, p, temp1);
      FGIntMul(temp1, qqinvp, temp2);
      FGIntCopy(temp2, temp1);
      FGIntMontgomeryModExp(MGInt, dq, q, temp2);
      FGIntMul(temp2, ppinvq, temp3);
      FGIntCopy(temp3, temp2);
      FGIntAddMod(temp1, temp2, n, SGInt);
      FGIntDestroy(temp1);
      FGIntDestroy(temp2);
      FGIntDestroy(ppinvq);
      FGIntDestroy(qqinvp);
   End;
   FGIntToBase256String(SGInt, S);
   FGIntDestroy(MGInt);
   FGIntDestroy(SGInt);
End;


// Перевіряємо правильність алгоритму  RSA,
// Якщо M = S^e mod n тоді ok:=true інакше ok:=false

Procedure RSAVerify(M, S : String; Var e, n : TFGInt; Var valid : boolean);
Var
   MGInt, SGInt, temp : TFGInt;
Begin
   Base256StringToFGInt(S, SGInt);
   Base256StringToFGInt(M, MGInt);
   FGIntMod(MGInt, n, temp);
   FGIntCopy(temp, MGInt);
   FGIntMontgomeryModExp(SGInt, e, n, temp);
   FGIntCopy(temp, SGInt);
   valid := (FGIntCompareAbs(SGInt, MGInt) = Eq);
   FGIntDestroy(SGInt);
   FGIntDestroy(MGInt);
End;

End.
```

Файл FGInt.pas – робота з великими числами для алгоритму RSA

```
Unit FGInt;

Interface

Uses Windows, SysUtils, Controls, Math;

Type
   TCompare = (Lt, St, Eq, Er);
   TSign = (negative, positive);
   TFGInt = Record
      Sign : TSign;
      Number : Array Of int64;
   End;

Procedure zeronetochar8(Var g : char; Const x : String);
Procedure zeronetochar6(Var g : integer; Const x : String);
Procedure initialize8(Var trans : Array Of String);
Procedure initialize6(Var trans : Array Of String);
Procedure initialize6PGP(Var trans : Array Of String);
Procedure ConvertBase256to64(Const str256 : String; Var str64 : String);
Procedure ConvertBase64to256(Const str64 : String; Var str256 : String);
Procedure ConvertBase256to2(Const str256 : String; Var str2 : String);
Procedure ConvertBase64to2(Const str64 : String; Var str2 : String);
Procedure ConvertBase2to256(str2 : String; Var str256 : String);
Procedure ConvertBase2to64(str2 : String; Var str64 : String);
Procedure ConvertBase256StringToHexString(Str256 : String; Var HexStr : String);
Procedure ConvertHexStringToBase256String(HexStr : String; Var Str256 : String);
Procedure PGPConvertBase256to64(Var str256, str64 : String);
Procedure PGPConvertBase64to256(str64 : String; Var str256 : String);
Procedure PGPConvertBase64to2(str64 : String; Var str2 : String);
Procedure Base10StringToFGInt(Base10 : String; Var FGInt : TFGInt);
Procedure FGIntToBase10String(Const FGInt : TFGInt; Var Base10 : String);
Procedure FGIntDestroy(Var FGInt : TFGInt);
Function FGIntCompareAbs(Const FGInt1, FGInt2 : TFGInt) : TCompare;
Procedure FGIntAdd(Const FGInt1, FGInt2 : TFGInt; Var Sum : TFGInt);
Procedure FGIntChangeSign(Var FGInt : TFGInt);
Procedure FGIntSub(Var FGInt1, FGInt2, dif : TFGInt);
Procedure FGIntMulByInt(Const FGInt : TFGInt; Var res : TFGInt; by : int64);
Procedure FGIntMulByIntbis(Var FGInt : TFGInt; by : int64);
Procedure FGIntDivByInt(Const FGInt : TFGInt; Var res : TFGInt; by : int64; Var
modres : int64);
Procedure FGIntDivByIntBis(Var FGInt : TFGInt; by : int64; Var modres : int64);
Procedure FGIntModByInt(Const FGInt : TFGInt; by : int64; Var modres : int64);
Procedure FGIntAbs(Var FGInt : TFGInt);
Procedure FGIntCopy(Const FGInt1 : TFGInt; Var FGInt2 : TFGInt);
Procedure FGIntShiftLeft(Var FGInt : TFGInt);
Procedure FGIntShiftRight(Var FGInt : TFGInt);
Procedure FGIntShiftRightBy31(Var FGInt : TFGInt);
Procedure FGIntAddBis(Var FGInt1 : TFGInt; Const FGInt2 : TFGInt);
Procedure FGIntSubBis(Var FGInt1 : TFGInt; Const FGInt2 : TFGInt);
Procedure FGIntMul(Const FGInt1, FGInt2 : TFGInt; Var Prod : TFGInt);
Procedure FGIntSquare(Const FGInt : TFGInt; Var Square : TFGInt);
Procedure FGIntToBase2String(Const FGInt : TFGInt; Var S : String);
Procedure Base2StringToFGInt(S : String; Var FGInt : TFGInt);
Procedure FGIntToBase256String(Const FGInt : TFGInt; Var str256 : String);
Procedure Base256StringToFGInt(str256 : String; Var FGInt : TFGInt);
Procedure PGPMPIToFGInt(PGPMPI : String; Var FGInt : TFGInt);
Procedure FGIntToPGPMPI(FGInt : TFGInt; Var PGPMPI : String);
Procedure FGIntExp(Const FGInt, exp : TFGInt; Var res : TFGInt);
Procedure FGIntFac(Const FGInt : TFGInt; Var res : TFGInt);
Procedure FGIntShiftLeftBy31(Var FGInt : TFGInt);
Procedure FGIntDivMod(Var FGInt1, FGInt2, QFGInt, MFGInt : TFGInt);
Procedure FGIntDiv(Var FGInt1, FGInt2, QFGInt : TFGInt);
Procedure FGIntMod(Var FGInt1, FGInt2, MFGInt : TFGInt);
Procedure FGIntSquareMod(Var FGInt, Modb, FGIntSM : TFGInt);
```

```
Procedure FGIntAddMod(Var FGInt1, FGInt2, base, FGIntres : TFGInt);
Procedure FGIntMulMod(Var FGInt1, FGInt2, base, FGIntres : TFGInt);
Procedure FGIntModExp(Var FGInt, exp, modb, res : TFGInt);
Procedure FGIntModBis(Const FGInt : TFGInt; Var FGIntOut : TFGInt; b : longint;
head : int64);
Procedure FGIntMulModBis(Const FGInt1, FGInt2 : TFGInt; Var Prod : TFGInt; b :
longint; head : int64);
Procedure FGIntMontgomeryMod(Const GInt, base, baseInv : TFGInt; Var MGInt :
TFGInt; b : longint; head : int64);
Procedure FGIntMontgomeryModExp(Var FGInt, exp, modb, res : TFGInt);
Procedure FGIntGCD(Const FGInt1, FGInt2 : TFGInt; Var GCD : TFGInt);
Procedure FGIntLCM(Const FGInt1, FGInt2 : TFGInt; Var LCM : TFGInt);
Procedure FGIntTrialDiv9999(Const FGInt : TFGInt; Var ok : boolean);
Procedure FGIntRandom1(Var Seed, RandomFGInt : TFGInt);
Procedure FGIntRabinMiller(Var FGIntp : TFGInt; nrtest : integer; Var ok :
boolean);
Procedure FGIntBezoutBachet(Var FGInt1, FGInt2, a, b : TFGInt);
Procedure FGIntModInv(Const FGInt1, base : TFGInt; Var Inverse : TFGInt);
Procedure FGIntPrimetest(Var FGIntp : TFGInt; nrRMtests : integer; Var ok :
boolean);
Procedure FGIntLegendreSymbol(Var a, p : TFGInt; Var L : integer);
Procedure FGIntSquareRootModP(Square, Prime : TFGInt; Var SquareRoot : TFGInt);



Implementation

Var
   primes : Array[1..1228] Of integer =
      (3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71,
73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127,
      131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199,
211, 223, 227, 229, 233, 239, 241, 251,
      257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347,
349, 353, 359, 367, 373, 379, 383, 389,
      397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479,
487, 491, 499, 503, 509, 521, 523, 541,
      547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631,
641, 643, 647, 653, 659, 661, 673, 677,
      683, 691, 701, 709, 719, 727, 733, 739, 743, 751, 757, 761, 769, 773, 787,
797, 809, 811, 821, 823, 827, 829, 839,
      853, 857, 859, 863, 877, 881, 883, 887, 907, 911, 919, 929, 937, 941, 947,
953, 967, 971, 977, 983, 991, 997, 1009,
      1013, 1019, 1021, 1031, 1033, 1039, 1049, 1051, 1061, 1063, 1069, 1087,
1091, 1093, 1097, 1103, 1109, 1117, 1123,
      1129, 1151, 1153, 1163, 1171, 1181, 1187, 1193, 1201, 1213, 1217, 1223,
1229, 1231, 1237, 1249, 1259, 1277, 1279,
      1283, 1289, 1291, 1297, 1301, 1303, 1307, 1319, 1321, 1327, 1361, 1367,
1373, 1381, 1399, 1409, 1423, 1427, 1429,
      1433, 1439, 1447, 1451, 1453, 1459, 1471, 1481, 1483, 1487, 1489, 1493,
1499, 1511, 1523, 1531, 1543, 1549, 1553,
      1559, 1567, 1571, 1579, 1583, 1597, 1601, 1607, 1609, 1613, 1619, 1621,
1627, 1637, 1657, 1663, 1667, 1669, 1693,
      1697, 1699, 1709, 1721, 1723, 1733, 1741, 1747, 1753, 1759, 1777, 1783,
1787, 1789, 1801, 1811, 1823, 1831, 1847,
      1861, 1867, 1871, 1873, 1877, 1879, 1889, 1901, 1907, 1913, 1931, 1933,
1949, 1951, 1973, 1979, 1987, 1993, 1997,
      1999, 2003, 2011, 2017, 2027, 2029, 2039, 2053, 2063, 2069, 2081, 2083,
2087, 2089, 2099, 2111, 2113, 2129, 2131,
      2137, 2141, 2143, 2153, 2161, 2179, 2203, 2207, 2213, 2221, 2237, 2239,
2243, 2251, 2267, 2269, 2273, 2281, 2287,
      2293, 2297, 2309, 2311, 2333, 2339, 2341, 2347, 2351, 2357, 2371, 2377,
2381, 2383, 2389, 2393, 2399, 2411, 2417,
      2423, 2437, 2441, 2447, 2459, 2467, 2473, 2477, 2503, 2521, 2531, 2539,
2543, 2549, 2551, 2557, 2579, 2591, 2593,
      2609, 2617, 2621, 2633, 2647, 2657, 2659, 2663, 2671, 2677, 2683, 2687,
2689, 2693, 2699, 2707, 2711, 2713, 2719,
      2729, 2731, 2741, 2749, 2753, 2767, 2777, 2789, 2791, 2797, 2801, 2803,
2819, 2833, 2837, 2843, 2851, 2857, 2861,
```

2879, 2887, 2897, 2903, 2909, 2917, 2927, 2939, 2953, 2957, 2963, 2969,
2971, 2999, 3001, 3011, 3019, 3023, 3037,
3041, 3049, 3061, 3067, 3079, 3083, 3089, 3109, 3119, 3121, 3137, 3163,
3167, 3169, 3181, 3187, 3191, 3203, 3209,
3217, 3221, 3229, 3251, 3253, 3257, 3259, 3271, 3299, 3301, 3307, 3313,
3319, 3323, 3329, 3331, 3343, 3347, 3359,
3361, 3371, 3373, 3389, 3391, 3407, 3413, 3433, 3449, 3457, 3461, 3463,
3467, 3469, 3491, 3499, 3511, 3517, 3527,
3529, 3533, 3539, 3541, 3547, 3557, 3559, 3571, 3581, 3583, 3593, 3607,
3613, 3617, 3623, 3631, 3637, 3643, 3659,
3671, 3673, 3677, 3691, 3697, 3701, 3709, 3719, 3727, 3733, 3739, 3761,
3767, 3769, 3779, 3793, 3797, 3803, 3821,
3823, 3833, 3847, 3851, 3853, 3863, 3877, 3881, 3889, 3907, 3911, 3917,
3919, 3923, 3929, 3931, 3943, 3947, 3967,
3989, 4001, 4003, 4007, 4013, 4019, 4021, 4027, 4049, 4051, 4057, 4073,
4079, 4091, 4093, 4099, 4111, 4127, 4129,
4133, 4139, 4153, 4157, 4159, 4177, 4201, 4211, 4217, 4219, 4229, 4231,
4241, 4243, 4253, 4259, 4261, 4271, 4273,
4283, 4289, 4297, 4327, 4337, 4339, 4349, 4357, 4363, 4373, 4391, 4397,
4409, 4421, 4423, 4441, 4447, 4451, 4457,
4463, 4481, 4483, 4493, 4507, 4513, 4517, 4519, 4523, 4547, 4549, 4561,
4567, 4583, 4591, 4597, 4603, 4621, 4637,
4639, 4643, 4649, 4651, 4657, 4663, 4673, 4679, 4691, 4703, 4721, 4723,
4729, 4733, 4751, 4759, 4783, 4787, 4789,
4793, 4799, 4801, 4813, 4817, 4831, 4861, 4871, 4877, 4889, 4903, 4909,
4919, 4931, 4933, 4937, 4943, 4951, 4957,
4967, 4969, 4973, 4987, 4993, 4999, 5003, 5009, 5011, 5021, 5023, 5039,
5051, 5059, 5077, 5081, 5087, 5099, 5101,
5107, 5113, 5119, 5147, 5153, 5167, 5171, 5179, 5189, 5197, 5209, 5227,
5231, 5233, 5237, 5261, 5273, 5279, 5281,
5297, 5303, 5309, 5323, 5333, 5347, 5351, 5381, 5387, 5393, 5399, 5407,
5413, 5417, 5419, 5431, 5437, 5441, 5443,
5449, 5471, 5477, 5479, 5483, 5501, 5503, 5507, 5519, 5521, 5527, 5531,
5557, 5563, 5569, 5573, 5581, 5591, 5623,
5639, 5641, 5647, 5651, 5653, 5657, 5659, 5669, 5683, 5689, 5693, 5701,
5711, 5717, 5737, 5741, 5743, 5749, 5779,
5783, 5791, 5801, 5807, 5813, 5821, 5827, 5839, 5843, 5849, 5851, 5857,
5861, 5867, 5869, 5879, 5881, 5897, 5903,
5923, 5927, 5939, 5953, 5981, 5987, 6007, 6011, 6029, 6037, 6043, 6047,
6053, 6067, 6073, 6079, 6089, 6091, 6101,
6113, 6121, 6131, 6133, 6143, 6151, 6163, 6173, 6197, 6199, 6203, 6211,
6217, 6221, 6229, 6247, 6257, 6263, 6269,
6271, 6277, 6287, 6299, 6301, 6311, 6317, 6323, 6329, 6337, 6343, 6353,
6359, 6361, 6367, 6373, 6379, 6389, 6397,
6421, 6427, 6449, 6451, 6469, 6473, 6481, 6491, 6521, 6529, 6547, 6551,
6553, 6563, 6569, 6571, 6577, 6581, 6599,
6607, 6619, 6637, 6653, 6659, 6661, 6673, 6679, 6689, 6691, 6701, 6703,
6709, 6719, 6733, 6737, 6761, 6763, 6779,
6781, 6791, 6793, 6803, 6823, 6827, 6829, 6833, 6841, 6857, 6863, 6869,
6871, 6883, 6899, 6907, 6911, 6917, 6947,
6949, 6959, 6961, 6967, 6971, 6977, 6983, 6991, 6997, 7001, 7013, 7019,
7027, 7039, 7043, 7057, 7069, 7079, 7103,
7109, 7121, 7127, 7129, 7151, 7159, 7177, 7187, 7193, 7207, 7211, 7213,
7219, 7229, 7237, 7243, 7247, 7253, 7283,
7297, 7307, 7309, 7321, 7331, 7333, 7349, 7351, 7369, 7393, 7411, 7417,
7433, 7451, 7457, 7459, 7477, 7481, 7487,
7489, 7499, 7507, 7517, 7523, 7529, 7537, 7541, 7547, 7549, 7559, 7561,
7573, 7577, 7583, 7589, 7591, 7603, 7607,
7621, 7639, 7643, 7649, 7669, 7673, 7681, 7687, 7691, 7699, 7703, 7717,
7723, 7727, 7741, 7753, 7757, 7759, 7789,
7793, 7817, 7823, 7829, 7841, 7853, 7867, 7873, 7877, 7879, 7883, 7901,
7907, 7919, 7927, 7933, 7937, 7949, 7951,
7963, 7993, 8009, 8011, 8017, 8039, 8053, 8059, 8069, 8081, 8087, 8089,
8093, 8101, 8111, 8117, 8123, 8147, 8161,
8167, 8171, 8179, 8191, 8209, 8219, 8221, 8231, 8233, 8237, 8243, 8263,
8269, 8273, 8287, 8291, 8293, 8297, 8311,
8317, 8329, 8353, 8363, 8369, 8377, 8387, 8389, 8419, 8423, 8429, 8431,
8443, 8447, 8461, 8467, 8501, 8513, 8521,

```
      8527, 8537, 8539, 8543, 8563, 8573, 8581, 8597, 8599, 8609, 8623, 8627,
8629, 8641, 8647, 8663, 8669, 8677, 8681,
      8689, 8693, 8699, 8707, 8713, 8719, 8731, 8737, 8741, 8747, 8753, 8761,
8779, 8783, 8803, 8807, 8819, 8821, 8831,
      8837, 8839, 8849, 8861, 8863, 8867, 8887, 8893, 8923, 8929, 8933, 8941,
8951, 8963, 8969, 8971, 8999, 9001, 9007,
      9011, 9013, 9029, 9041, 9043, 9049, 9059, 9067, 9091, 9103, 9109, 9127,
9133, 9137, 9151, 9157, 9161, 9173, 9181,
      9187, 9199, 9203, 9209, 9221, 9227, 9239, 9241, 9257, 9277, 9281, 9283,
9293, 9311, 9319, 9323, 9337, 9341, 9343,
      9349, 9371, 9377, 9391, 9397, 9403, 9413, 9419, 9421, 9431, 9433, 9437,
9439, 9461, 9463, 9467, 9473, 9479, 9491,
      9497, 9511, 9521, 9533, 9539, 9547, 9551, 9587, 9601, 9613, 9619, 9623,
9629, 9631, 9643, 9649, 9661, 9677, 9679,
      9689, 9697, 9719, 9721, 9733, 9739, 9743, 9749, 9767, 9769, 9781, 9787,
9791, 9803, 9811, 9817, 9829, 9833, 9839,
      9851, 9857, 9859, 9871, 9883, 9887, 9901, 9907, 9923, 9929, 9931, 9941,
9949, 9967, 9973);
   chr64 : Array[1..64] Of char = ('a', 'A', 'b', 'B', 'c', 'C', 'd', 'D', 'e',
'E', 'f', 'F',
      'g', 'G', 'h', 'H', 'i', 'I', 'j', 'J', 'k', 'K', 'l', 'L', 'm', 'M', 'n',
'N', 'o', 'O', 'p',
      'P', 'q', 'Q', 'r', 'R', 's', 'S', 't', 'T', 'u', 'U', 'v', 'V', 'w', 'W',
'x', 'X', 'y', 'Y',
      'z', 'Z', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '+', '=');
   PGPchr64 : Array[1..64] Of char = ('A', 'B', 'C', 'D', 'E', 'F', 'G', 'H',
'I', 'J', 'K', 'L', 'M',
      'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', 'a', 'b',
'c', 'd', 'e', 'f',
      'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u',
'v', 'w', 'x', 'y',
      'z', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '+', '/');


{$H+}


Procedure zeronetochar8(Var g : char; Const x : String);
Var
   i : Integer;
   b : byte;
Begin
   b := 0;
   For i := 1 To 8 Do
   Begin
      If copy(x, i, 1) = '1' Then
         b := b Or (1 Shl (8 - I));
   End;
   g := chr(b);
End;


Procedure zeronetochar6(Var g : integer; Const x : String);
Var
   I : Integer;
Begin
   G := 0;
   For I := 1 To Length(X) Do
   Begin
      If I > 6 Then
         Break;
      If X[I] <> '0' Then
         G := G Or (1 Shl (6 - I));
   End;
   Inc(G);
End;


Procedure initialize8(Var trans : Array Of String);
```

```
Var
   c1, c2, c3, c4, c5, c6, c7, c8 : integer;
   x : String;
   g : char;
Begin
   For c1 := 0 To 1 Do
      For c2 := 0 To 1 Do
         For c3 := 0 To 1 Do
            For c4 := 0 To 1 Do
               For c5 := 0 To 1 Do
                  For c6 := 0 To 1 Do
                     For c7 := 0 To 1 Do
                        For c8 := 0 To 1 Do
                        Begin
                           x := '';
                           x := inttostr(c1) + inttostr(c2) + inttostr(c3) +
inttostr(c4) + inttostr(c5) + inttostr(c6) + inttostr(c7) + inttostr(c8);
                           zeronetochar8(g, x);
                           trans[ord(g)] := x;
                        End;
End;


Procedure initialize6(Var trans : Array Of String);
Var
   c1, c2, c3, c4, c5, c6 : integer;
   x : String;
   g : integer;
Begin
   For c1 := 0 To 1 Do
      For c2 := 0 To 1 Do
         For c3 := 0 To 1 Do
            For c4 := 0 To 1 Do
               For c5 := 0 To 1 Do
                  For c6 := 0 To 1 Do
                  Begin
                     x := '';
                     x := inttostr(c1) + inttostr(c2) + inttostr(c3) +
inttostr(c4) + inttostr(c5) + inttostr(c6);
                     zeronetochar6(g, x);
                     trans[ord(chr64[g])] := x;
                  End;
End;

Procedure initialize6PGP(Var trans : Array Of String);
Var
   c1, c2, c3, c4, c5, c6 : integer;
   x : String;
   g : integer;
Begin
   For c1 := 0 To 1 Do
      For c2 := 0 To 1 Do
         For c3 := 0 To 1 Do
            For c4 := 0 To 1 Do
               For c5 := 0 To 1 Do
                  For c6 := 0 To 1 Do
                  Begin
                     x := '';
                     x := inttostr(c1) + inttostr(c2) + inttostr(c3) +
inttostr(c4) + inttostr(c5) + inttostr(c6);
                     zeronetochar6(g, x);
                     trans[ord(PGPchr64[g])] := x;
                  End;
End;


// перетворюємо строки довжиною 256 біт у 64 біта

Procedure ConvertBase256to64(Const str256 : String; Var str64 : String);
```

```
Var
   temp : String;
   trans : Array[0..255] Of String;
   i, len6 : longint;
   g : integer;
Begin
   initialize8(trans);
   temp := '';
   For i := 1 To length(str256) Do temp := temp + trans[ord(str256[i])];
   While (length(temp) Mod 6) <> 0 Do temp := temp + '0';
   len6 := length(temp) Div 6;
   str64 := '';
   For i := 1 To len6 Do
   Begin
      zeronetochar6(g, copy(temp, 1, 6));
      str64 := str64 + chr64[g];
      delete(temp, 1, 6);
   End;
End;

// перетворюємо строки довжиною 64 біт у 256 біта

Procedure ConvertBase64to256(Const str64 : String; Var str256 : String);
Var
   temp : String;
   trans : Array[0..255] Of String;
   i, len8 : longint;
   g : char;
Begin
   initialize6(trans);
   temp := '';
   For i := 1 To length(str64) Do temp := temp + trans[ord(str64[i])];
   str256 := '';
   len8 := length(temp) Div 8;
   For i := 1 To len8 Do
   Begin
      zeronetochar8(g, copy(temp, 1, 8));
      str256 := str256 + g;
      delete(temp, 1, 8);
   End;
End;


// перетворюємо строки довжиною 256 біт у 2 біта

Procedure ConvertBase256to2(Const str256 : String; Var str2 : String);
Var
   trans : Array[0..255] Of String;
   i : longint;
Begin
   str2 := '';
   initialize8(trans);
   For i := 1 To length(str256) Do str2 := str2 + trans[ord(str256[i])];
End;

// перетворюємо строки довжиною 64 біт у 2 біта

Procedure ConvertBase64to2(Const str64 : String; Var str2 : String);
Var
   trans : Array[0..255] Of String;
   i : longint;
Begin
   str2 := '';
   initialize6(trans);
   For i := 1 To length(str64) Do str2 := str2 + trans[ord(str64[i])];
End;

// перетворюємо строки довжиною 2 біт у 256 біт
```

```
Procedure ConvertBase2to256(str2 : String; Var str256 : String);
Var
   i, len8 : longint;
   g : char;
Begin
   str256 := '';
   While (length(str2) Mod 8) <> 0 Do str2 := '0' + str2;
   len8 := length(str2) Div 8;
   For i := 1 To len8 Do
   Begin
      zeronetochar8(g, copy(str2, 1, 8));
      str256 := str256 + g;
      delete(str2, 1, 8);
   End;
End;

// перетворюємо строки довжиною 2 біта у 64 біта

Procedure ConvertBase2to64(str2 : String; Var str64 : String);
Var
   i, len6 : longint;
   g : integer;
Begin
   str64 := '';
   While (length(str2) Mod 6) <> 0 Do str2 := '0' + str2;
   len6 := length(str2) Div 6;
   For i := 1 To len6 Do
   Begin
      zeronetochar6(g, copy(str2, 1, 6));
      str64 := str64 + chr64[g];
      delete(str2, 1, 6);
   End;
End;


// перетворюємо строки довжиною 256 біт у 16 біта

Procedure ConvertBase256StringToHexString(Str256 : String; Var HexStr : String);
Var
   i : longint;
   b : byte;
Begin
   HexStr := '';
   For i := 1 To length(str256) Do
   Begin
      b := ord(str256[i]);
      If (b Shr 4) < 10 Then HexStr := HexStr + chr(48 + (b Shr 4))
      Else HexStr := HexStr + chr(55 + (b Shr 4));
      If (b And 15) < 10 Then HexStr := HexStr + chr(48 + (b And 15))
      Else HexStr := HexStr + chr(55 + (b And 15));
   End;
End;




// перетворюємо строки довжиною 16 біт у 256 біт

Procedure ConvertHexStringToBase256String(HexStr : String; Var Str256 : String);
Var
   i : longint;
   b, h1, h2 : byte;
Begin
   Str256 := '';
   For i := 1 To (length(Hexstr) Div 2) Do
   Begin
      h2 := ord(HexStr[2 * i]);
      h1 := ord(HexStr[2 * i - 1]);
      If h1 < 58 Then b := ((h1 - 48) Shl 4) Else b := ((h1 - 55) Shl 4);
```

```
      If h2 < 58 Then b := (b Or (h2 - 48)) Else b := (b Or (h2 - 55));
      Str256 := Str256 + chr(b);
   End;
End;


// перетворюємо строки довжиною 256 біт у 64 біта для PGP

Procedure PGPConvertBase256to64(Var str256, str64 : String);
Var
   temp, x, a : String;
   i, len6 : longint;
   g : integer;
   trans : Array[0..255] Of String;
Begin
   initialize8(trans);
   temp := '';
   For i := 1 To length(str256) Do temp := temp + trans[ord(str256[i])];
   If (length(temp) Mod 6) = 0 Then a := '' Else
      If (length(temp) Mod 6) = 4 Then
      Begin
         temp := temp + '00';
         a := '='
      End
      Else
      Begin
         temp := temp + '0000';
         a := '=='
      End;
   str64 := '';
   len6 := length(temp) Div 6;
   For i := 1 To len6 Do
   Begin
      x := copy(temp, 1, 6);
      zeronetochar6(g, x);
      str64 := str64 + PGPchr64[g];
      delete(temp, 1, 6);
   End;
   str64 := str64 + a;
End;

// перетворюємо строки довжиною 64 біт у 256 біт для PGP

Procedure PGPConvertBase64to256(str64 : String; Var str256 : String);
Var
   temp, x : String;
   i, j, len8 : longint;
   g : char;
   trans : Array[0..255] Of String;
Begin
   initialize6PGP(trans);
   temp := '';
   str256 := '';
   If str64[length(str64) - 1] = '=' Then j := 2 Else
      If str64[length(str64)] = '=' Then j := 1 Else j := 0;
   For i := 1 To (length(str64) - j) Do temp := temp + trans[ord(str64[i])];
   If j <> 0 Then delete(temp, length(temp) - 2 * j + 1, 2 * j);
   len8 := length(temp) Div 8;
   For i := 1 To len8 Do
   Begin
      x := copy(temp, 1, 8);
      zeronetochar8(g, x);
      str256 := str256 + g;
      delete(temp, 1, 8);
   End;
End;

// перетворюємо строки довжиною 64 біт у 2 біта для PGP
```

```
Procedure PGPConvertBase64to2(str64 : String; Var str2 : String);
Var
   i, j : longint;
   trans : Array[0..255] Of String;
Begin
   str2 := '';
   initialize6(trans);
   If str64[length(str64) - 1] = '=' Then j := 2 Else
      If str64[length(str64)] = '=' Then j := 1 Else j := 0;
   For i := 1 To (length(str64) - j) Do str2 := str2 + trans[ord(str64[i])];
   delete(str2, length(str2) - 2 * j + 1, 2 * j);
End;


// перетворюємо строки довжиною 10 біт у  FGInt

Procedure Base10StringToFGInt(Base10 : String; Var FGInt : TFGInt);
Var
   i, size : longint;
   j : int64;
   S : String;
   sign : TSign;

   Procedure GIntDivByIntBis1(Var GInt : TFGInt; by : int64; Var modres :
int64);
   Var
      i, size : longint;
      rest : int64;
   Begin
      size := GInt.Number[0];
      modres := 0;
      For i := size Downto 1 Do
      Begin
         modres := modres * 1000000000;
         rest := modres + GInt.Number[i];
         GInt.Number[i] := rest Div by;
         modres := rest Mod by;
      End;
      While (GInt.Number[size] = 0) And (size > 1) Do size := size - 1;
      If size <> GInt.Number[0] Then
      Begin
         SetLength(GInt.Number, size + 1);
         GInt.Number[0] := size;
      End;
   End;

Begin
   While (Not (Base10[1] In ['-', '0'..'9'])) And (length(Base10) > 1) Do
      delete(Base10, 1, 1);
   If copy(Base10, 1, 1) = '-' Then
   Begin
      Sign := negative;
      delete(Base10, 1, 1);
   End
   Else Sign := positive;
   While (length(Base10) > 1) And (copy(Base10, 1, 1) = '0') Do delete(Base10,
1, 1);
   size := length(Base10) Div 9;
   If (length(Base10) Mod 9) <> 0 Then size := size + 1;
   SetLength(FGInt.Number, size + 1);
   FGInt.Number[0] := size;
   For i := 1 To size - 1 Do
   Begin
      FGInt.Number[i] := StrToInt(copy(Base10, length(Base10) - 8, 9));
      delete(Base10, length(Base10) - 8, 9);
   End;
   FGInt.Number[size] := StrToInt(Base10);

   S := '';
```

```
   While (FGInt.Number[0] <> 1) Or (FGInt.Number[1] <> 0) Do
   Begin
      GIntDivByIntBis1(FGInt, 2, j);
      S := inttostr(j) + S;
   End;
   S := '0' + S;
   FGIntDestroy(FGInt);
   Base2StringToFGInt(S, FGInt);
   FGInt.Sign := sign;
End;


// перетворюємо  FGInt в рядок 10 біт

Procedure FGIntToBase10String(Const FGInt : TFGInt; Var Base10 : String);
Var
   S : String;
   j : int64;
   temp : TFGInt;
Begin
   FGIntCopy(FGInt, temp);
   Base10 := '';
   While (temp.Number[0] > 1) Or (temp.Number[1] > 0) Do
   Begin
      FGIntDivByIntBis(temp, 1000000000, j);
      S := IntToStr(j);
      While Length(S) < 9 Do S := '0' + S;
      Base10 := S + Base10;
   End;
   Base10 := '0' + Base10;
   While (length(Base10) > 1) And (Base10[1] = '0') Do delete(Base10, 1, 1);
   If FGInt.Sign = negative Then Base10 := '-' + Base10;
End;



// Видаляємо FGInt для звільнення пам'яті

Procedure FGIntDestroy(Var FGInt : TFGInt);
Begin
   FGInt.Number := Nil;
End;



Function FGIntCompareAbs(Const FGInt1, FGInt2 : TFGInt) : TCompare;
Var
   size1, size2, i : longint;
Begin
   FGIntCompareAbs := Er;
   size1 := FGInt1.Number[0];
   size2 := FGInt2.Number[0];
   If size1 > size2 Then FGIntCompareAbs := Lt Else
      If size1 < size2 Then FGIntCompareAbs := St Else
      Begin
         i := size2;
         While (FGInt1.Number[i] = FGInt2.Number[i]) And (i > 1) Do i := i - 1;
         If FGInt1.Number[i] = FGInt2.Number[i] Then FGIntCompareAbs := Eq Else
            If FGInt1.Number[i] < FGInt2.Number[i] Then FGIntCompareAbs := St
Else
               If FGInt1.Number[i] > FGInt2.Number[i] Then FGIntCompareAbs :=
Lt;
      End;
End;


// Додаємо 2 FGInts, FGInt1 + FGInt2 = Sum

Procedure FGIntAdd(Const FGInt1, FGInt2 : TFGInt; Var Sum : TFGInt);
```

```
Var
   i, size1, size2, size : longint;
   rest : integer;
   Trest : int64;
Begin
   size1 := FGInt1.Number[0];
   size2 := FGInt2.Number[0];
   If size1 < size2 Then FGIntAdd(FGInt2, FGInt1, Sum) Else
   Begin
      If FGInt1.Sign = FGInt2.Sign Then
      Begin
         Sum.Sign := FGInt1.Sign;
         SetLength(Sum.Number, size1 + 2);
         rest := 0;
         For i := 1 To size2 Do
         Begin
            Trest := FGInt1.Number[i] + FGInt2.Number[i] + rest;
            Sum.Number[i] := Trest And 2147483647;
            rest := Trest Shr 31;
         End;
         For i := (size2 + 1) To size1 Do
         Begin
            Trest := FGInt1.Number[i] + rest;
            Sum.Number[i] := Trest And 2147483647;
            rest := Trest Shr 31;
         End;
         size := size1 + 1;
         Sum.Number[0] := size;
         Sum.Number[size] := rest;
         While (Sum.Number[size] = 0) And (size > 1) Do size := size - 1;
         If Sum.Number[0] > size Then SetLength(Sum.Number, size + 1);
         Sum.Number[0] := size;
      End
      Else
      Begin
         If FGIntCompareAbs(FGInt2, FGInt1) = Lt Then FGIntAdd(FGInt2, FGInt1,
Sum)
         Else
         Begin
            SetLength(Sum.Number, size1 + 1);
            rest := 0;
            For i := 1 To size2 Do
            Begin
               Trest := 2147483648 + FGInt1.Number[i] - FGInt2.Number[i] + rest;
               Sum.Number[i] := Trest And 2147483647;
               If (Trest > 2147483647) Then rest := 0 Else rest := -1;
            End;
            For i := (size2 + 1) To size1 Do
            Begin
               Trest := 2147483648 + FGInt1.Number[i] + rest;
               Sum.Number[i] := Trest And 2147483647;
               If (Trest > 2147483647) Then rest := 0 Else rest := -1;
            End;
            size := size1;
            While (Sum.Number[size] = 0) And (size > 1) Do size := size - 1;
            If size < size1 Then
            Begin
               SetLength(Sum.Number, size + 1);
            End;
            Sum.Number[0] := size;
            Sum.Sign := FGInt1.Sign;
         End;
      End;
   End;
End;


Procedure FGIntChangeSign(Var FGInt : TFGInt);
Begin
```

```
   If FGInt.Sign = negative Then FGInt.Sign := positive Else FGInt.Sign :=
negative;
End;


// Віднімаємо 2 FGInts, FGInt1 - FGInt2 = dif

Procedure FGIntSub(Var FGInt1, FGInt2, dif : TFGInt);
Begin
   FGIntChangeSign(FGInt2);
   FGIntAdd(FGInt1, FGInt2, dif);
   FGIntChangeSign(FGInt2);
End;


// Перемножуємоа FGInt з цілим, FGInt * by = res, by < 1000000000

Procedure FGIntMulByInt(Const FGInt : TFGInt; Var res : TFGInt; by : int64);
Var
   i, size : longint;
   Trest, rest : int64;
Begin
   size := FGInt.Number[0];
   SetLength(res.Number, size + 2);
   rest := 0;
   For i := 1 To size Do
   Begin
      Trest := FGInt.Number[i] * by + rest;
      res.Number[i] := Trest And 2147483647;
      rest := Trest Shr 31;
   End;
   If rest <> 0 Then
   Begin
      size := size + 1;
      Res.Number[size] := rest;
   End
   Else SetLength(Res.Number, size + 1);
   Res.Number[0] := size;
   Res.Sign := FGInt.Sign;
End;


// перемножуємо FGInt з цілим, FGInt * by = res, by < 1000000000

Procedure FGIntMulByIntbis(Var FGInt : TFGInt; by : int64);
Var
   i, size : longint;
   Trest, rest : int64;
Begin
   size := FGInt.Number[0];
   SetLength(FGInt.Number, size + 2);
   rest := 0;
   For i := 1 To size Do
   Begin
      Trest := FGInt.Number[i] * by + rest;
      FGInt.Number[i] := Trest And 2147483647;
      rest := Trest Shr 31;
   End;
   If rest <> 0 Then
   Begin
      size := size + 1;
      FGInt.Number[size] := rest;
   End
   Else SetLength(FGInt.Number, size + 1);
   FGInt.Number[0] := size;
End;


// ділимо FGInt на ціле, FGInt = res * by + modres
```

```
Procedure FGIntDivByInt(Const FGInt : TFGInt; Var res : TFGInt; by : int64; Var
modres : int64);
Var
   i, size : longint;
   rest : int64;
Begin
   size := FGInt.Number[0];
   SetLength(res.Number, size + 1);
   modres := 0;
   For i := size Downto 1 Do
   Begin
      modres := modres Shl 31;
      rest := modres Or FGInt.Number[i];
      res.Number[i] := rest Div by;
      modres := rest Mod by;
   End;
   While (res.Number[size] = 0) And (size > 1) Do size := size - 1;
   SetLength(res.Number, size + 1);
   res.Number[0] := size;
   Res.Sign := FGInt.Sign;
End;


// ділимо FGInt на ціле, FGInt = FGInt * by + modres

Procedure FGIntDivByIntBis(Var FGInt : TFGInt; by : int64; Var modres : int64);
Var
   i, size : longint;
   rest : int64;
Begin
   size := FGInt.Number[0];
   modres := 0;
   For i := size Downto 1 Do
   Begin
      modres := modres Shl 31;
      rest := modres Or FGInt.Number[i];
      FGInt.Number[i] := rest Div by;
      modres := rest Mod by;
   End;
   While (FGInt.Number[size] = 0) And (size > 1) Do size := size - 1;
   If size <> FGInt.Number[0] Then
   Begin
      SetLength(FGInt.Number, size + 1);
      FGInt.Number[0] := size;
   End;
End;


// Беремо FGInt по модулю цілого числа, FGInt mod by = modres

Procedure FGIntModByInt(Const FGInt : TFGInt; by : int64; Var modres : int64);
Var
   i, size : longint;
   rest : int64;
Begin
   size := FGInt.Number[0];
   modres := 0;
   For i := size Downto 1 Do
   Begin
      modres := modres Shl 31;
      rest := modres + FGInt.Number[i];
      modres := rest Mod by;
   End;
End;


// повертаємо FGInt по модулю
```

```
Procedure FGIntAbs(Var FGInt : TFGInt);
Begin
   FGInt.Sign := positive;
End;


// Копіюємо FGInt1 в FGInt2

Procedure FGIntCopy(Const FGInt1 : TFGInt; Var FGInt2 : TFGInt);
Begin
   FGInt2.Sign := FGInt1.Sign;
   FGInt2.Number := Nil;
   FGInt2.Number := Copy(FGInt1.Number, 0, FGInt1.Number[0] + 1);
End;


// Зрушуємо FGInt уліво на 2 біта FGInt = FGInt * 2

Procedure FGIntShiftLeft(Var FGInt : TFGInt);
Var
   l, m : int64;
   i, size : longint;
Begin
   size := FGInt.Number[0];
   l := 0;
   For i := 1 To Size Do
   Begin
      m := FGInt.Number[i] Shr 30;
      FGInt.Number[i] := ((FGInt.Number[i] Shl 1) Or l) And 2147483647;
      l := m;
   End;
   If l <> 0 Then
   Begin
      setlength(FGInt.Number, size + 2);
      FGInt.Number[size + 1] := l;
      FGInt.Number[0] := size + 1;
   End;
End;


// Зрушуємо FGInt вправо на 2 біта, FGInt = FGInt div 2

Procedure FGIntShiftRight(Var FGInt : TFGInt);
Var
   l, m : int64;
   i, size : longint;
Begin
   size := FGInt.Number[0];
   l := 0;
   For i := size Downto 1 Do
   Begin
      m := FGInt.Number[i] And 1;
      FGInt.Number[i] := (FGInt.Number[i] Shr 1) Or l;
      l := m Shl 30;
   End;
   If (FGInt.Number[size] = 0) And (size > 1) Then
   Begin
      setlength(FGInt.Number, size);
      FGInt.Number[0] := size - 1;
   End;
End;


// FGInt = FGInt / 2147483648

Procedure FGIntShiftRightBy31(Var FGInt : TFGInt);
Var
   size : longint;
Begin
```

```
   size := FGInt.Number[0];
   If size > 1 Then
   Begin
      FGInt.Number := Copy(FGInt.Number, 1, Size);
      FGInt.Number[0] := size - 1;
   End
   Else FGInt.Number[1] := 0;
End;


// FGInt1 = FGInt1 + FGInt2, FGInt1 > FGInt2

Procedure FGIntAddBis(Var FGInt1 : TFGInt; Const FGInt2 : TFGInt);
Var
   i, size1, size2 : longint;
   rest : integer;
   Trest : int64;
Begin
   size1 := FGInt1.Number[0];
   size2 := FGInt2.Number[0];
   rest := 0;
   For i := 1 To size2 Do
   Begin
      Trest := FGInt1.Number[i] + FGInt2.Number[i] + rest;
      rest := Trest Shr 31;
      FGInt1.Number[i] := Trest And 2147483647;
   End;
   For i := size2 + 1 To size1 Do
   Begin
      Trest := FGInt1.Number[i] + rest;
      rest := Trest Shr 31;
      FGInt1.Number[i] := Trest And 2147483647;
   End;
   If rest <> 0 Then
   Begin
      SetLength(FGInt1.Number, size1 + 2);
      FGInt1.Number[0] := size1 + 1;
      FGInt1.Number[size1 + 1] := rest;
   End;
End;


// FGInt1 = FGInt1 - FGInt2, використовується тільки якщо 0 < FGInt2 < FGInt1

Procedure FGIntSubBis(Var FGInt1 : TFGInt; Const FGInt2 : TFGInt);
Var
   i, size1, size2 : longint;
   rest : integer;
   Trest : int64;
Begin
   size1 := FGInt1.Number[0];
   size2 := FGInt2.Number[0];
   rest := 0;
   For i := 1 To size2 Do
   Begin
      Trest := 2147483648 + FGInt1.Number[i] - FGInt2.Number[i] + rest;
      If (Trest > 2147483647) Then rest := 0 Else rest := -1;
      FGInt1.Number[i] := Trest And 2147483647;
   End;
   For i := size2 + 1 To size1 Do
   Begin
      Trest := 2147483648 + FGInt1.Number[i] + rest;
      If (Trest > 2147483647) Then rest := 0 Else rest := -1;
      FGInt1.Number[i] := Trest And 2147483647;
   End;
   i := size1;
   While (FGInt1.Number[i] = 0) And (i > 1) Do i := i - 1;
   If i < size1 Then
   Begin
```

```
      SetLength(FGInt1.Number, i + 1);
      FGInt1.Number[0] := i;
   End;
End;


// Перемножуємо 2 FGInts, FGInt1 * FGInt2 = Prod

Procedure FGIntMul(Const FGInt1, FGInt2 : TFGInt; Var Prod : TFGInt);
Var
   i, j, size, size1, size2 : longint;
   rest, Trest : int64;
Begin
   size1 := FGInt1.Number[0];
   size2 := FGInt2.Number[0];
   size := size1 + size2;
   SetLength(Prod.Number, size + 1);
   For i := 1 To size Do Prod.Number[i] := 0;

   For i := 1 To size2 Do
   Begin
      rest := 0;
      For j := 1 To size1 Do
      Begin
         Trest := Prod.Number[j + i - 1] + FGInt1.Number[j] * FGInt2.Number[i] +
rest;
         Prod.Number[j + i - 1] := Trest And 2147483647;
         rest := Trest Shr 31;
      End;
      Prod.Number[i + size1] := rest;
   End;

   Prod.Number[0] := size;
   While (Prod.Number[size] = 0) And (size > 1) Do size := size - 1;
   If size < Prod.Number[0] Then
   Begin
      SetLength(Prod.Number, size + 1);
      Prod.Number[0] := size;
   End;
   If FGInt1.Sign = FGInt2.Sign Then Prod.Sign := Positive Else prod.Sign :=
negative;
End;


// Підводимо в квадрат FGInt, FGIntI = Square

Procedure FGIntSquare(Const FGInt : TFGInt; Var Square : TFGInt);
Var
   size, size1, i, j : longint;
   rest, Trest : int64;
Begin
   size1 := FGInt.Number[0];
   size := 2 * size1;
   SetLength(Square.Number, size + 1);
   Square.Number[0] := size;
   For i := 1 To size Do Square.Number[i] := 0;
   For i := 1 To size1 Do
   Begin
      Trest := Square.Number[2 * i - 1] + FGInt.Number[i] * FGInt.Number[i];
      Square.Number[2 * i - 1] := Trest And 2147483647;
      rest := Trest Shr 31;
      For j := i + 1 To size1 Do
      Begin
         Trest := Square.Number[i + j - 1] + 2 * FGInt.Number[i] *
FGInt.Number[j] + rest;
         Square.Number[i + j - 1] := Trest And 2147483647;
         rest := Trest Shr 31;
      End;
      Square.Number[i + size1] := rest;
```

```
   End;
   Square.Sign := positive;
   While (Square.Number[size] = 0) And (size > 1) Do size := size - 1;
   If size < 2 * size1 Then
   Begin
      SetLength(Square.Number, size + 1);
      Square.Number[0] := size;
   End;
End;


// Перетворюємо FGInt у бінарну рядок

Procedure FGIntToBase2String(Const FGInt : TFGInt; Var S : String);
Var
   i : longint;
   j : integer;
Begin
   S := '';
   For i := 1 To FGInt.Number[0] Do
   Begin
      For j := 0 To 30 Do S := inttostr(1 And (FGInt.Number[i] Shr j)) + S;
   End;
   While (length(S) > 1) And (S[1] = '0') Do delete(S, 1, 1);
   If S = '' Then S := '0';
End;


Procedure Base2StringToFGInt(S : String; Var FGInt : TFGInt);
Var
   i, j, size : longint;
Begin
   while (S[1]='0') and (length(S)>1) do delete(S,1,1);
   size := length(S) Div 31;
   If (length(S) Mod 31) <> 0 Then size := size + 1;
   SetLength(FGInt.Number, size + 1);
   FGInt.Number[0] := size;
   j := 1;
   FGInt.Number[j] := 0;
   i := 0;
   While length(S) > 0 Do
   Begin
      If S[length(S)] = '1' Then
         FGInt.Number[j] := FGInt.Number[j] Or (1 Shl i);
      i := i + 1;
      If i = 31 Then
      Begin
         i := 0;
         j := j + 1;
         If j <= size Then FGInt.Number[j] := 0;
      End;
      delete(S, length(S), 1);
   End;
   FGInt.Sign := positive;
End;


// перетворюємо FGInt у рядок 256 біт

Procedure FGIntToBase256String(Const FGInt : TFGInt; Var str256 : String);
Var
   temp1 : String;
   i, len8 : longint;
   g : char;
Begin
   FGIntToBase2String(FGInt, temp1);
   While (length(temp1) Mod 8) <> 0 Do temp1 := '0' + temp1;
   len8 := length(temp1) Div 8;
   str256 := '';
```

```
   For i := 1 To len8 Do
   Begin
      zeronetochar8(g, copy(temp1, 1, 8));
      str256 := str256 + g;
      delete(temp1, 1, 8);
   End;
End;


Procedure Base256StringToFGInt(str256 : String; Var FGInt : TFGInt);
Var
   temp1 : String;
   i : longint;
   trans : Array[0..255] Of String;
Begin
   temp1 := '';
   initialize8(trans);
   For i := 1 To length(str256) Do temp1 := temp1 + trans[ord(str256[i])];
   While (temp1[1] = '0') And (temp1 <> '0') Do delete(temp1, 1, 1);
   Base2StringToFGInt(temp1, FGInt);
End;

// Перетворюємо MPI (Multiple Precision Integer, для PGP) у FGInt

Procedure PGPMPIToFGInt(PGPMPI : String; Var FGInt : TFGInt);
Var
   temp : String;
Begin
   temp := PGPMPI;
   delete(temp, 1, 2);
   Base256StringToFGInt(temp, FGInt);
End;


Procedure FGIntToPGPMPI(FGInt : TFGInt; Var PGPMPI : String);
Var
   len, i : word;
   c : char;
   b : byte;
Begin
   FGIntToBase256String(FGInt, PGPMPI);
   len := length(PGPMPI) * 8;
   c := PGPMPI[1];
   For i := 7 Downto 0 Do If (ord(c) Shr i) = 0 Then len := len - 1 Else break;
   b := len Mod 256;
   PGPMPI := chr(b) + PGPMPI;
   b := len Div 256;
   PGPMPI := chr(b) + PGPMPI;
End;


// Піднімаємо у ступінь FGInt, FGInt^exp = res

Procedure FGIntExp(Const FGInt, exp : TFGInt; Var res : TFGInt);
Var
   temp2, temp3 : TFGInt;
   S : String;
   i : longint;
Begin
   FGIntToBase2String(exp, S);
   If S[length(S)] = '0' Then Base10StringToFGInt('1', res) Else
FGIntCopy(FGInt, res);
   FGIntCopy(FGInt, temp2);
   If length(S) > 1 Then
      For i := (length(S) - 1) Downto 1 Do
      Begin
         FGIntSquare(temp2, temp3);
         FGIntCopy(temp3, temp2);
         If S[i] = '1' Then
```

```
         Begin
            FGIntMul(res, temp2, temp3);
            FGIntCopy(temp3, res);
         End;
      End;
End;


// Розраховуємо FGInt! = FGInt * (FGInt - 1) * (FGInt - 2) * ... * 3 * 2 * 1

Procedure FGIntFac(Const FGInt : TFGInt; Var res : TFGInt);
Var
   one, temp, temp1 : TFGInt;
Begin
   FGIntCopy(FGInt, temp);
   Base10StringToFGInt('1', res);
   Base10StringToFGInt('1', one);

   While Not (FGIntCompareAbs(temp, one) = Eq) Do
   Begin
      FGIntMul(temp, res, temp1);
      FGIntCopy(temp1, res);
      FGIntSubBis(temp, one);
   End;

   FGIntDestroy(one);
   FGIntDestroy(temp);
End;


// FGInt = FGInt * 2147483648

Procedure FGIntShiftLeftBy31(Var FGInt : TFGInt);
Var
   f1, f2 : int64;
   i, size : longint;
Begin
   size := FGInt.Number[0];
   SetLength(FGInt.Number, size + 2);
   f1 := 0;
   For i := 1 To (size + 1) Do
   Begin
      f2 := FGInt.Number[i];
      FGInt.Number[i] := f1;
      f1 := f2;
   End;
   FGInt.Number[0] := size + 1;
End;


// Ділимо 2 FGInts, FGInt1 = FGInt2 * QFGInt + MFGInt, MFGInt позитивне

Procedure FGIntDivMod(Var FGInt1, FGInt2, QFGInt, MFGInt : TFGInt);
Var
   one, zero, temp1, temp2 : TFGInt;
   s1, s2 : TSign;
   i, j, s, t : longint;
Begin
   s1 := FGInt1.Sign;
   s2 := FGInt2.Sign;
   FGIntAbs(FGInt1);
   FGIntAbs(FGInt2);
   FGIntCopy(FGInt1, MFGInt);
   FGIntCopy(FGInt2, temp1);

   If FGIntCompareAbs(FGInt1, FGInt2) <> St Then
   Begin
      s := FGInt1.Number[0] - FGInt2.Number[0];
      setlength(QFGInt.Number, s + 2);
```

```
      QFGInt.Number[0] := s + 1;
      For t := 1 To s Do
      Begin
         FGIntShiftLeftBy31(temp1);
         QFGInt.Number[t] := 0;
      End;
      j := s + 1;
      QFGInt.Number[j] := 0;
      While FGIntCompareAbs(MFGInt, FGInt2) <> St Do
      Begin
         While FGIntCompareAbs(MFGInt, temp1) <> St Do
         Begin
            If MFGInt.Number[0] > temp1.Number[0] Then
               i := (2147483648 * MFGInt.Number[MFGInt.Number[0]] +
MFGInt.Number[MFGInt.Number[0] - 1]) Div (temp1.Number[temp1.Number[0]] + 1)
            Else i := MFGInt.Number[MFGInt.Number[0]] Div
(temp1.Number[temp1.Number[0]] + 1);
            If (i <> 0) Then
            Begin
               FGIntCopy(temp1, temp2);
               FGIntMulByIntBis(temp2, i);
               FGIntSubBis(MFGInt, temp2);
               QFGInt.Number[j] := QFGInt.Number[j] + i;
               If FGIntCompareAbs(MFGInt, temp2) <> St Then
               Begin
                  QFGInt.Number[j] := QFGInt.Number[j] + i;
                  FGIntSubBis(MFGInt, temp2);
               End;
            End Else
            Begin
               QFGInt.Number[j] := QFGInt.Number[j] + 1;
               FGIntSubBis(MFGInt, temp1);
            End;
         End;
         If MFGInt.Number[0] <= temp1.Number[0] Then
            If FGIntCompareAbs(temp1, FGInt2) <> Eq Then
            Begin
               FGIntShiftRightBy31(temp1);
               j := j - 1;
            End;
      End;
   End
   Else Base10StringToFGInt('0', QFGInt);
   s := QFGInt.Number[0];
   While (s > 1) And (QFGInt.Number[s] = 0) Do s := s - 1;
   If s < QFGInt.Number[0] Then
   Begin
      setlength(QFGInt.Number, s + 1);
      QFGInt.Number[0] := s;
   End;
   QFGInt.Sign := positive;

   FGIntDestroy(temp1);
   Base10StringToFGInt('0', zero);
   Base10StringToFGInt('1', one);
   If s1 = negative Then
   Begin
      If FGIntCompareAbs(MFGInt, zero) <> Eq Then
      Begin
         FGIntadd(QFGInt, one, temp1);
         FGIntCopy(temp1, QFGInt);
         FGIntDestroy(temp1);
         FGIntsub(FGInt2, MFGInt, temp1);
         FGIntCopy(temp1, MFGInt);
      End;
      If s2 = positive Then QFGInt.Sign := negative;
   End
   Else QFGInt.Sign := s2;
   FGIntDestroy(one);
```

```
   FGIntDestroy(zero);

   FGInt1.Sign := s1;
   FGInt2.Sign := s2;
End;


// Розраховуємо MFGInt

Procedure FGIntDiv(Var FGInt1, FGInt2, QFGInt : TFGInt);
Var
   one, zero, temp1, temp2, MFGInt : TFGInt;
   s1, s2 : TSign;
   i, j, s, t : longint;
Begin
   s1 := FGInt1.Sign;
   s2 := FGInt2.Sign;
   FGIntAbs(FGInt1);
   FGIntAbs(FGInt2);
   FGIntCopy(FGInt1, MFGInt);
   FGIntCopy(FGInt2, temp1);

   If FGIntCompareAbs(FGInt1, FGInt2) <> St Then
   Begin
      s := FGInt1.Number[0] - FGInt2.Number[0];
      setlength(QFGInt.Number, s + 2);
      QFGInt.Number[0] := s + 1;
      For t := 1 To s Do
      Begin
         FGIntShiftLeftBy31(temp1);
         QFGInt.Number[t] := 0;
      End;
      j := s + 1;
      QFGInt.Number[j] := 0;
      While FGIntCompareAbs(MFGInt, FGInt2) <> St Do
      Begin
         While FGIntCompareAbs(MFGInt, temp1) <> St Do
         Begin
            If MFGInt.Number[0] > temp1.Number[0] Then
               i := (2147483648 * MFGInt.Number[MFGInt.Number[0]] +
MFGInt.Number[MFGInt.Number[0] - 1]) Div (temp1.Number[temp1.Number[0]] + 1)
            Else i := MFGInt.Number[MFGInt.Number[0]] Div
(temp1.Number[temp1.Number[0]] + 1);
            If (i <> 0) Then
            Begin
               FGIntCopy(temp1, temp2);
               FGIntMulByIntBis(temp2, i);
               FGIntSubBis(MFGInt, temp2);
               QFGInt.Number[j] := QFGInt.Number[j] + i;
               If FGIntCompareAbs(MFGInt, temp2) <> St Then
               Begin
                  QFGInt.Number[j] := QFGInt.Number[j] + i;
                  FGIntSubBis(MFGInt, temp2);
               End;
            End Else
            Begin
               QFGInt.Number[j] := QFGInt.Number[j] + 1;
               FGIntSubBis(MFGInt, temp1);
            End;
         End;
         If MFGInt.Number[0] <= temp1.Number[0] Then
            If FGIntCompareAbs(temp1, FGInt2) <> Eq Then
            Begin
               FGIntShiftRightBy31(temp1);
               j := j - 1;
            End;
      End;
   End
   Else Base10StringToFGInt('0', QFGInt);
```

```
   s := QFGInt.Number[0];
   While (s > 1) And (QFGInt.Number[s] = 0) Do s := s - 1;
   If s < QFGInt.Number[0] Then
   Begin
      setlength(QFGInt.Number, s + 1);
      QFGInt.Number[0] := s;
   End;
   QFGInt.Sign := positive;

   FGIntDestroy(temp1);
   Base10StringToFGInt('0', zero);
   Base10StringToFGInt('1', one);
   If s1 = negative Then
   Begin
      If FGIntCompareAbs(MFGInt, zero) <> Eq Then
      Begin
         FGIntadd(QFGInt, one, temp1);
         FGIntCopy(temp1, QFGInt);
         FGIntDestroy(temp1);
         FGIntsub(FGInt2, MFGInt, temp1);
         FGIntCopy(temp1, MFGInt);
      End;
      If s2 = positive Then QFGInt.Sign := negative;
   End
   Else QFGInt.Sign := s2;
   FGIntDestroy(one);
   FGIntDestroy(zero);
   FGIntDestroy(MFGInt);

   FGInt1.Sign := s1;
   FGInt2.Sign := s2;
End;


// MFGInt = FGInt1 mod FGInt2

Procedure FGIntMod(Var FGInt1, FGInt2, MFGInt : TFGInt);
Var
   one, zero, temp1, temp2 : TFGInt;
   s1, s2 : TSign;
   i : int64;
   s, t : longint;
Begin
   s1 := FGInt1.Sign;
   s2 := FGInt2.Sign;
   FGIntAbs(FGInt1);
   FGIntAbs(FGInt2);
   FGIntCopy(FGInt1, MFGInt);
   FGIntCopy(FGInt2, temp1);

   If FGIntCompareAbs(FGInt1, FGInt2) <> St Then
   Begin
      s := FGInt1.Number[0] - FGInt2.Number[0];
      For t := 1 To s Do FGIntShiftLeftBy31(temp1);
      While FGIntCompareAbs(MFGInt, FGInt2) <> St Do
      Begin
         While FGIntCompareAbs(MFGInt, temp1) <> St Do
         Begin
            If MFGInt.Number[0] > temp1.Number[0] Then
               i := (2147483648 * MFGInt.Number[MFGInt.Number[0]] +
MFGInt.Number[MFGInt.Number[0] - 1]) Div (temp1.Number[temp1.Number[0]] + 1)
            Else i := MFGInt.Number[MFGInt.Number[0]] Div
(temp1.Number[temp1.Number[0]] + 1);
            If (i <> 0) Then
            Begin
               FGIntCopy(temp1, temp2);
               FGIntMulByIntBis(temp2, i);
               FGIntSubBis(MFGInt, temp2);
```

```
               If FGIntCompareAbs(MFGInt, temp2) <> St Then FGIntSubBis(MFGInt,
temp2);
            End Else FGIntSubBis(MFGInt, temp1);
//         If FGIntCompareAbs(MFGInt, temp1) <> St Then
FGIntSubBis(MFGInt,temp1);
         End;
         If MFGInt.Number[0] <= temp1.Number[0] Then
            If FGIntCompareAbs(temp1, FGInt2) <> Eq Then
FGIntShiftRightBy31(temp1);
      End;
   End;

   FGIntDestroy(temp1);
   Base10StringToFGInt('0', zero);
   Base10StringToFGInt('1', one);
   If s1 = negative Then
   Begin
      If FGIntCompareAbs(MFGInt, zero) <> Eq Then
      Begin
         FGIntSub(FGInt2, MFGInt, temp1);
         FGIntCopy(temp1, MFGInt);
      End;
   End;
   FGIntDestroy(one);
   FGIntDestroy(zero);

   FGInt1.Sign := s1;
   FGInt2.Sign := s2;
End;


// підводимо у квадрат FGInt за модулем  Modb, FGInt^2 mod Modb = FGIntSM

Procedure FGIntSquareMod(Var FGInt, Modb, FGIntSM : TFGInt);
Var
   temp : TFGInt;
Begin
   FGIntSquare(FGInt, temp);
   FGIntMod(temp, Modb, FGIntSM);
   FGIntDestroy(temp);
End;


// Додаємо 2 FGInts за модулем, (FGInt1 + FGInt2) mod base = FGIntres

Procedure FGIntAddMod(Var FGInt1, FGInt2, base, FGIntres : TFGInt);
Var
   temp : TFGInt;
Begin
   FGIntadd(FGInt1, FGInt2, temp);
   FGIntMod(temp, base, FGIntres);
   FGIntDestroy(temp);
End;


// Перемножуємо 2 FGInts за модулем, (FGInt1 * FGInt2) mod base = FGIntres

Procedure FGIntMulMod(Var FGInt1, FGInt2, base, FGIntres : TFGInt);
Var
   temp : TFGInt;
Begin
   FGIntMul(FGInt1, FGInt2, temp);
   FGIntMod(temp, base, FGIntres);
   FGIntDestroy(temp);
End;


// Підводимо у ступінь 2 FGInts за модулем, (FGInt1 ^ FGInt2) mod modb = res
```

```
Procedure FGIntModExp(Var FGInt, exp, modb, res : TFGInt);
Var
   temp2, temp3 : TFGInt;
   i : longint;
   S : String;
Begin
   If (Modb.Number[1] Mod 2) = 1 Then
   Begin
      FGIntMontgomeryModExp(FGInt, exp, modb, res);
      exit;
   End;
   FGIntToBase2String(exp, S);
   Base10StringToFGInt('1', res);
   FGIntcopy(FGInt, temp2);

   For i := length(S) Downto 1 Do
   Begin
      If S[i] = '1' Then
      Begin
         FGIntmulMod(res, temp2, modb, temp3);
         FGIntCopy(temp3, res);
      End;
      FGIntSquareMod(temp2, Modb, temp3);
      FGIntCopy(temp3, temp2);
   End;
   FGIntDestroy(temp2);
End;


// Процедура підводення у ступень за Монтгомері

Procedure FGIntModBis(Const FGInt : TFGInt; Var FGIntOut : TFGInt; b : longint;
head : int64);
Var
   i : longint;
Begin
   If b <= FGInt.Number[0] Then
   Begin
      FGIntOut.Number := Copy(FGInt.Number, 0, b + 1);
      FGIntOut.Number[b] := FGIntOut.Number[b] And head;
      i := b;
      While (FGIntOut.Number[i] = 0) And (i > 1) Do i := i - 1;
      If i < b Then SetLength(FGIntOut.Number, i + 1);
      FGIntOut.Number[0] := i;
      FGIntOut.Sign := positive;
   End Else FGIntCopy(FGInt, FGIntOut);
End;


Procedure FGIntMulModBis(Const FGInt1, FGInt2 : TFGInt; Var Prod : TFGInt; b :
longint; head : int64);
Var
   i, j, size, size1, size2, t : longint;
   rest, Trest : int64;
Begin
   size1 := FGInt1.Number[0];
   size2 := FGInt2.Number[0];
   size := min(b, size1 + size2);
   SetLength(Prod.Number, size + 1);
   For i := 1 To size Do Prod.Number[i] := 0;

   For i := 1 To size2 Do
   Begin
      rest := 0;
      t := min(size1, b - i + 1);
      For j := 1 To t Do
      Begin
         Trest := Prod.Number[j + i - 1] + FGInt1.Number[j] * FGInt2.Number[i] +
rest;
```

```
         Prod.Number[j + i - 1] := Trest And 2147483647;
         rest := Trest Shr 31;
      End;
      If (i + size1) <= b Then Prod.Number[i + size1] := rest;
   End;

   Prod.Number[0] := size;
   If size = b Then Prod.Number[b] := Prod.Number[b] And head;
   While (Prod.Number[size] = 0) And (size > 1) Do size := size - 1;
   If size < Prod.Number[0] Then
   Begin
      SetLength(Prod.Number, size + 1);
      Prod.Number[0] := size;
   End;
   If FGInt1.Sign = FGInt2.Sign Then Prod.Sign := Positive Else prod.Sign :=
negative;
End;


Procedure FGIntMontgomeryMod(Const GInt, base, baseInv : TFGInt; Var MGInt :
TFGInt; b : longint; head : int64);
Var
   m, temp, temp1 : TFGInt;
   r : int64;
Begin
   FGIntModBis(GInt, temp, b, head);
   FGIntMulModBis(temp, baseInv, m, b, head);
   FGIntMul(m, base, temp1);
   FGIntDestroy(temp);
   FGIntAdd(temp1, GInt, temp);
   FGIntDestroy(temp1);
   MGInt.Number := copy(temp.Number, b - 1, temp.Number[0] - b + 2);
   MGInt.Sign := positive;
   MGInt.Number[0] := temp.Number[0] - b + 1;
   FGIntDestroy(temp);
   If (head Shr 30) = 0 Then FGIntDivByIntBis(MGInt, head + 1, r)
   Else FGIntShiftRightBy31(MGInt);
   If FGIntCompareAbs(MGInt, base) <> St Then FGIntSubBis(MGInt, base);
   FGIntDestroy(temp);
   FGIntDestroy(m);
End;


Procedure FGIntMontgomeryModExp(Var FGInt, exp, modb, res : TFGInt);
Var
   temp2, temp3, baseInv, r : TFGInt;
   i, j, t, b : longint;
   S : String;
   head : int64;
Begin
   FGIntToBase2String(exp, S);
   t := modb.Number[0];
   b := t;

   If (modb.Number[t] Shr 30) = 1 Then t := t + 1;
   setlength(r.Number, t + 1);
   r.Number[0] := t;
   r.Sign := positive;
   For i := 1 To t Do r.Number[i] := 0;
   If t = modb.Number[0] Then
   Begin
      head := 2147483647;
      For j := 29 Downto 0 Do
      Begin
         head := head Shr 1;
         If (modb.Number[t] Shr j) = 1 Then
         Begin
            r.Number[t] := 1 Shl (j + 1);
            break;
```

```
         End;
      End;
   End
   Else
   Begin
      r.Number[t] := 1;
      head := 2147483647;
   End;

   FGIntModInv(modb, r, temp2);
   If temp2.Sign = negative Then FGIntCopy(temp2, BaseInv)
   Else
   Begin
      FGIntCopy(r, BaseInv);
      FGIntSubBis(BaseInv, temp2);
   End;
//   FGIntBezoutBachet(r, modb, temp2, BaseInv);
   FGIntAbs(BaseInv);
   FGIntDestroy(temp2);
   FGIntMod(r, modb, res);
   FGIntMulMod(FGInt, res, modb, temp2);
   FGIntDestroy(r);

   For i := length(S) Downto 1 Do
   Begin
      If S[i] = '1' Then
      Begin
         FGIntmul(res, temp2, temp3);
         FGIntDestroy(res);
         FGIntMontgomeryMod(temp3, modb, baseinv, res, b, head);
         FGIntDestroy(temp3);
      End;
      FGIntSquare(temp2, temp3);
      FGIntDestroy(temp2);
      FGIntMontgomeryMod(temp3, modb, baseinv, temp2, b, head);
      FGIntDestroy(temp3);
   End;
   FGIntDestroy(temp2);
   FGIntMontgomeryMod(res, modb, baseinv, temp3, b, head);
   FGIntCopy(temp3, res);
   FGIntDestroy(temp3);
   FGIntDestroy(baseinv);
End;


// розраховуємо найбільший загальний дільник 2 FGInts

Procedure FGIntGCD(Const FGInt1, FGInt2 : TFGInt; Var GCD : TFGInt);
Var
   k : TCompare;
   zero, temp1, temp2, temp3 : TFGInt;
Begin
   k := FGIntCompareAbs(FGInt1, FGInt2);
   If (k = Eq) Then FGIntCopy(FGInt1, GCD) Else
      If (k = St) Then FGIntGCD(FGInt2, FGInt1, GCD) Else
      Begin
         Base10StringToFGInt('0', zero);
         FGIntCopy(FGInt1, temp1);
         FGIntCopy(FGInt2, temp2);
         While (temp2.Number[0] <> 1) Or (temp2.Number[1] <> 0) Do
         Begin
            FGIntMod(temp1, temp2, temp3);
            FGIntCopy(temp2, temp1);
            FGIntCopy(temp3, temp2);
            FGIntDestroy(temp3);
         End;
         FGIntCopy(temp1, GCD);
         FGIntDestroy(temp2);
         FGIntDestroy(zero);
```

```
      End;
End;


// розраховуємо найменше загальне кратне 2 FGInts

Procedure FGIntLCM(Const FGInt1, FGInt2 : TFGInt; Var LCM : TFGInt);
Var
   temp1, temp2 : TFGInt;
Begin
   FGIntGCD(FGInt1, FGInt2, temp1);
   FGIntmul(FGInt1, FGInt2, temp2);
   FGIntdiv(temp2, temp1, LCM);
   FGIntDestroy(temp1);
   FGIntDestroy(temp2);
End;


// Знаходження взаємо простого FGInt до 9999 та зупинка коли знайдено таке
число, повертає ok=false

Procedure FGIntTrialDiv9999(Const FGInt : TFGInt; Var ok : boolean);
Var
   j : int64;
   i : integer;
Begin
   If ((FGInt.Number[1] Mod 2) = 0) Then ok := false
   Else
   Begin
      i := 0;
      ok := true;
      While ok And (i < 1228) Do
      Begin
         i := i + 1;
         FGIntmodbyint(FGInt, primes[i], j);
         If j = 0 Then ok := false;
      End;
   End;
End;


// Генератор випадкових чисел

Procedure FGIntRandom1(Var Seed, RandomFGInt : TFGInt);
Var
   temp, base : TFGInt;
Begin
   Base10StringToFGInt('281474976710656', base);
   Base10StringToFGInt('44485709377909', temp);
   FGIntMulMod(seed, temp, base, RandomFGInt);
   FGIntDestroy(temp);
   FGIntDestroy(base);
End;


// Тест на простоту числа FGIntp методом Рабіна-Мілера, повертає ok=true якщо
FGIntp пройшло тест

Procedure FGIntRabinMiller(Var FGIntp : TFGInt; nrtest : integer; Var ok :
boolean);
Var
   j, b, i : int64;
   m, z, temp1, temp2, temp3, zero, one, two, pmin1 : TFGInt;
   ok1, ok2 : boolean;
Begin
   randomize;
   j := 0;
   Base10StringToFGInt('0', zero);
   Base10StringToFGInt('1', one);
```

```
   Base10StringToFGInt('2', two);
   FGIntsub(FGIntp, one, temp1);
   FGIntsub(FGIntp, one, pmin1);

   b := 0;
   While (temp1.Number[1] Mod 2) = 0 Do
   Begin
      b := b + 1;
      FGIntShiftRight(temp1);
   End;
   m := temp1;

   i := 0;
   ok := true;
   Randomize;
   While (i < nrtest) And ok Do
   Begin
      i := i + 1;
      Base10StringToFGInt(inttostr(Primes[Random(1227) + 1]), temp2);
      FGIntMontGomeryModExp(temp2, m, FGIntp, z);
      FGIntDestroy(temp2);
      ok1 := (FGIntCompareAbs(z, one) = Eq);
      ok2 := (FGIntCompareAbs(z, pmin1) = Eq);
      If Not (ok1 Or ok2) Then
      Begin

         While (ok And (j < b)) Do
         Begin
            If (j > 0) And ok1 Then ok := false
            Else
            Begin
               j := j + 1;
               If (j < b) And (Not ok2) Then
               Begin
                  FGIntSquaremod(z, FGIntp, temp3);
                  FGIntCopy(temp3, z);
                  ok1 := (FGIntCompareAbs(z, one) = Eq);
                  ok2 := (FGIntCompareAbs(z, pmin1) = Eq);
                  If ok2 Then j := b;
               End
               Else If (Not ok2) And (j >= b) Then ok := false;
            End;
         End;

      End
   End;

   FGIntDestroy(zero);
   FGIntDestroy(one);
   FGIntDestroy(two);
   FGIntDestroy(m);
   FGIntDestroy(z);
   FGIntDestroy(pmin1);
End;


// Розраховуємо коефіцієнти з теореми Безу, FGInt1 * a + FGInt2 * b =
GCD(FGInt1, FGInt2)

Procedure FGIntBezoutBachet(Var FGInt1, FGInt2, a, b : TFGInt);
Var
   zero, r1, r2, r3, ta, gcd, temp, temp1, temp2 : TFGInt;
Begin
   If FGIntCompareAbs(FGInt1, FGInt2) <> St Then
   Begin
      FGIntcopy(FGInt1, r1);
      FGIntcopy(FGInt2, r2);
      Base10StringToFGInt('0', zero);
      Base10StringToFGInt('1', a);
```

```
      Base10StringToFGInt('0', ta);

      Repeat
         FGIntdivmod(r1, r2, temp, r3);
         FGIntDestroy(r1);
         r1 := r2;
         r2 := r3;

         FGIntmul(ta, temp, temp1);
         FGIntsub(a, temp1, temp2);
         FGIntCopy(ta, a);
         FGIntCopy(temp2, ta);
         FGIntDestroy(temp1);

         FGIntDestroy(temp);
      Until FGIntCompareAbs(r3, zero) = Eq;

      FGIntGCD(FGInt1, FGInt2, gcd);
      FGIntmul(a, FGInt1, temp1);
      FGIntsub(gcd, temp1, temp2);
      FGIntDestroy(temp1);
      FGIntdiv(temp2, FGInt2, b);
      FGIntDestroy(temp2);

      FGIntDestroy(ta);
      FGIntDestroy(r1);
      FGIntDestroy(r2);
      FGIntDestroy(gcd);
   End
   Else FGIntBezoutBachet(FGInt2, FGInt1, b, a);
End;


// Знахродимо мультпликативне зворотнє FGInt у кінцевому кільці позитивного
порядку

Procedure FGIntModInv(Const FGInt1, base : TFGInt; Var Inverse : TFGInt);
Var
   zero, one, r1, r2, r3, tb, gcd, temp, temp1, temp2 : TFGInt;
Begin
   Base10StringToFGInt('1', one);
   FGIntGCD(FGInt1, base, gcd);
   If FGIntCompareAbs(one, gcd) = Eq Then
   Begin
      FGIntcopy(base, r1);
      FGIntcopy(FGInt1, r2);
      Base10StringToFGInt('0', zero);
      Base10StringToFGInt('0', inverse);
      Base10StringToFGInt('1', tb);

      Repeat
         FGIntDestroy(r3);
         FGIntdivmod(r1, r2, temp, r3);
         FGIntCopy(r2, r1);
         FGIntCopy(r3, r2);

         FGIntmul(tb, temp, temp1);
         FGIntsub(inverse, temp1, temp2);
         FGIntDestroy(inverse);
         FGIntDestroy(temp1);
         FGIntCopy(tb, inverse);
         FGIntCopy(temp2, tb);

         FGIntDestroy(temp);
      Until FGIntCompareAbs(r3, zero) = Eq;

      If inverse.Sign = negative Then
      Begin
         FGIntadd(base, inverse, temp);
```

```
         FGIntCopy(temp, inverse);
      End;

      FGIntDestroy(tb);
      FGIntDestroy(r1);
      FGIntDestroy(r2);
   End;
   FGIntDestroy(gcd);
   FGIntDestroy(one);
End;


// Простй комбінований тест на простоту FGIntp

Procedure FGIntPrimetest(Var FGIntp : TFGInt; nrRMtests : integer; Var ok :
boolean);
Begin
   FGIntTrialdiv9999(FGIntp, ok);
   If ok Then FGIntRabinMiller(FGIntp, nrRMtests, ok);
End;


//Розрахунок символу Лагранжа

Procedure FGIntLegendreSymbol(Var a, p : TFGInt; Var L : integer);
Var
   temp1, temp2, temp3, temp4, temp5, zero, one : TFGInt;
   i : int64;
   ok1, ok2 : boolean;
Begin
   Base10StringToFGInt('0', zero);
   Base10StringToFGInt('1', one);
   FGIntMod(a, p, temp1);
   If FGIntCompareAbs(zero, temp1) = Eq Then
   Begin
      FGIntDestroy(temp1);
      L := 0;
   End
   Else
   Begin
      FGIntDestroy(temp1);
      FGIntCopy(p, temp1);
      FGIntCopy(a, temp2);
      L := 1;
      While FGIntCompareAbs(temp2, one) <> Eq Do
      Begin
         If (temp2.Number[1] Mod 2) = 0 Then
         Begin
            FGIntSquare(temp1, temp3);
            FGIntSub(temp3, one, temp4);
            FGIntDestroy(temp3);
            FGIntDivByInt(temp4, temp3, 8, i);
            If (temp3.Number[1] Mod 2) = 0 Then ok1 := false Else ok1 := true;
            FGIntDestroy(temp3);
            FGIntDestroy(temp4);
            If ok1 = true Then L := L * (-1);
            FGIntDivByIntBis(temp2, 2, i);
         End
         Else
         Begin
            FGIntSub(temp1, one, temp3);
            FGIntSub(temp2, one, temp4);
            FGIntMul(temp3, temp4, temp5);
            FGIntDestroy(temp3);
            FGIntDestroy(temp4);
            FGIntDivByInt(temp5, temp3, 4, i);
            If (temp3.Number[1] Mod 2) = 0 Then ok2 := false Else ok2 := true;
            FGIntDestroy(temp5);
            FGIntDestroy(temp3);
```

```
            If ok2 = true Then L := L * (-1);
            FGIntMod(temp1, temp2, temp3);
            FGIntCopy(temp2, temp1);
            FGIntCopy(temp3, temp2);
         End;
      End;
      FGIntDestroy(temp1);
      FGIntDestroy(temp2);
   End;
   FGIntDestroy(zero);
   FGIntDestroy(one);
End;


// Розрахунок квадратичного корня за модулем простого числа
// SquareRoot^2 mod Prime = Square

Procedure FGIntSquareRootModP(Square, Prime : TFGInt; Var SquareRoot : TFGInt);
Var
   one, n, b, s, r, temp, temp1, temp2, temp3 : TFGInt;
   a, L, i, j : longint;
Begin
   Base2StringToFGInt('1', one);
   Base2StringToFGInt('10', n);
   a := 0;
   FGIntLegendreSymbol(n, Prime, L);
   While L <> -1 Do
   Begin
      FGIntAddBis(n, one);
      FGIntLegendreSymbol(n, Prime, L);
   End;
   FGIntCopy(Prime, s);
   s.Number[1] := s.Number[1] - 1;
   While (s.Number[1] Mod 2) = 0 Do
   Begin
      FGIntShiftRight(s);
      a := a + 1;
   End;
   FGIntMontgomeryModExp(n, s, Prime, b);
   FGIntAdd(s, one, temp);
   FGIntShiftRight(temp);
   FGIntMontgomeryModExp(Square, temp, Prime, r);
   FGIntDestroy(temp);
   FGIntModInv(Square, Prime, temp1);

   For i := 0 To (a - 2) Do
   Begin
      FGIntSquareMod(r, Prime, temp2);
      FGIntMulMod(temp1, temp2, Prime, temp);
      FGIntDestroy(temp2);
      For j := 1 To (a - i - 2) Do
      Begin
         FGIntSquareMod(temp, Prime, temp2);
         FGIntDestroy(temp);
         FGIntCopy(temp2, temp);
         FGIntDestroy(temp2);
      End;
      If FGIntCompareAbs(temp, one) <> Eq Then
      Begin
         FGIntMulMod(r, b, Prime, temp3);
         FGIntDestroy(r);
         FGIntCopy(temp3, r);
         FGIntDestroy(temp3);
      End;
      FGIntDestroy(temp);
      FGIntDestroy(temp2);
      If i = (a - 2) Then break;
      FGIntSquareMod(b, Prime, temp3);
      FGIntDestroy(b);
```

```
      FGIntCopy(temp3, b);
      FGIntDestroy(temp3);
   End;

   FGIntCopy(r, SquareRoot);
   FGIntDestroy(r);
   FGIntDestroy(s);
   FGIntDestroy(b);
   FGIntDestroy(temp1);
   FGIntDestroy(one);
   FGIntDestroy(n);
End;


End.
```

Файл FGIntPrimeGeneration.pas – генерація випадкових простих чисел для алгоритму RSA

```
Unit FGIntPrimeGeneration;

Interface

Uses Windows, SysUtils, Controls, FGInt;

Procedure PrimeSearch(Var GInt : TFGInt);

Implementation


{$H+}


// Відбувається пошаговий пошук простого числа починаючи з GInt,
// якщо воно знайдено то зберігається у GInt

Procedure PrimeSearch(Var GInt : TFGInt);
Var
   temp, two : TFGInt;
   ok : Boolean;
Begin
   If (GInt.Number[1] Mod 2) = 0 Then GInt.Number[1] := GInt.Number[1] + 1;
   Base10StringToFGInt('2', two);
   ok := false;
   While Not ok Do
   Begin
      FGIntAdd(GInt, two, temp);
      FGIntCopy(temp, GInt);
      FGIntPrimeTest(GInt, 4, ok);
   End;
   FGIntDestroy(two);
End;

End.
```

Файл about.pas – довідка

```
unit about;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, StdCtrls, jpeg;

type
  TForm1 = class(TForm)
    Image1: TImage;
    Memo1: TMemo;
    Panel1: TPanel;
    procedure Panel1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.Panel1Click(Sender: TObject);
begin
Form1.Close;
end;

end.
```