Центральноукраїнський національний технічний університет
Центр заочної та дистанційної освіти
Кафедра кібербезпеки та програмного забезпечення

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

**ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА**
**за першим (бакалаврським) рівнем вищої освіти**
на тему
**“Програмне забезпечення системи кібербезпеки застосунків та даних захищеної файлової системи”**

Виконав здобувач вищої освіти
IV курсу, групи КБ-21СКЗ
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Лівітчук О.В.
«____» _____ 2023 р.

Керівник проекту
кандидат фізико-математичних наук, доцент
_____ Петренюк В.І.
«____» _____ 2023 р.
Рецензент _____
_____



м. Кропивницький

**Центральноукраїнський національний технічний університет**

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

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

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

*Лівітчуку Олексію Віталійовичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки застосунків та даних захищеної файлової системи*

2. Керівник роботи *Петренюк Володимир Ілліч, канд. фіз.-мат. наук, доцент*
( прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 17-02 від 5.01.2023 року

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

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи кібербезпеки застосунків та даних захищеної файлової системи*

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

*1. Призначення та область використання.*
*2.Перегляд аналогічних існуючих систем.*
*3. Опис і обґрунтування проектних рішень.*
*4. Етапи програмування системи.*
*5. Впровадження системи кібербезпеки в промислову експлуатацію.*
*6. Висновки*

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

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

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

## КАЛЕНДАРНИЙ ПЛАН

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

Дата видачі завдання
« 17 » січня 2023 р.

Підпис керівника
_____ Петренюк В.І.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2023 р.

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

# АНОТАЦІЯ

**Лівітчук О.В. Програмне забезпечення системи кібербезпеки застосунків та даних захищеної файлової системи. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2023.**

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки застосунків та даних захищеної файлової системи.

Метою розробки є програмне забезпечення системи кібербезпеки застосунків та даних захищеної файлової системи.

Результат роботи – програмна реалізація системи кібербезпеки застосунків та даних захищеної файлової системи.

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

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

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

Програму розроблено в середовищі Embarcadero Delphi.

**Ключові слова**: кібербезпека, захищена файлова система

# ABSTRACT

**Livitchuk O.V. Cyber security system software for secure file system applications and data. 125 Cyber security. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.**

In this final qualification work for the first (bachelor) level of higher education, software is developed, which is intended for the cyber security system of applications and data of a protected file system.

The purpose of the development is the software of the cyber security system of applications and data of the protected file system.

The result of the work is the software implementation of the cyber security system of applications and data of the protected file system.

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

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

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

The program was developed in the Embarcadero Delphi environment.

**Keywords:** cyber security, secure file system

**ЗМІСТ**





| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | ВКРБ-125.23.0054.00.00.ПЗ | |
| Вим | Арк. | № докум. | Підп. | Дата | | |
| Розроб. | | Лівітчук О.В. | | | Програмне забезпечення системи кібербезпеки застосунків та даних захищеної файлової системи | Літ. Б / Аркуш 1 / Аркушів 74 |
| Перев. | | Петренюк В.І. | | | | |
| Н.контр. | | Гермак В.С. | | | | ЦНТУ КБ-21СКЗ |
| Затв. | | Смірнов О.А. | | | | |

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

| | | |
|---|---|---|
| НЖМД | – | накопичувач на жорсткому магнітному диску |
| ПЗП | – | постійний запам’ятовуючий пристрій |
| ЦП | – | центральний процесор |
| DDF | – | data decryption field – поле дешифрування даних |
| DES | – | симетричний алгоритм шифрування |
| DRF | – | data recovery field – поле відновлення даних |
| EFS | – | система шифрування даних на диску |
| FEK | – | file encryption key – згенерований ключ певної довжини |
| FSRTL | – | бібліотека часу виконання файлової системи |
| IPSec | – | протокол захисту даних |
| LPC | – | шина |
| NTFS | – | файлова система |
| PKI | – | інфраструктура відкритих ключів |
| RSA | – | асиметричний алгоритм шифрування |
| WebDAV | – | протокол захисту даних |

# ВСТУП

**Актуальність теми**. Для забезпечення безпеки на комп'ютері часто встає питання про безпечні й прості в застосуванні продуктах шифрування файлів для Windows. Настільки ж часто виникає питання про способи, за допомогою яких можна перешкодити системним адміністраторам заглядати в конфіденційні файли компанії. Одним з найбільш удалих рішень поставлених питань є використання власної файлової системи із шифруванням (Encrypting File System, EFS) Windows. Encrypting File System (EFS) – система шифрування даних, що реалізує шифрування на рівні файлів в операційних системах Microsoft Windows NT (починаючи з Windows 2000 і вище). Дана система надає можливість «прозорого шифрування» даних, що зберігаються на розділах з файловою системою NTFS, для захисту потенційно конфіденційних даних від несанкціонованого доступу при фізичному доступі до комп'ютера й дисків.

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 шифрування для захисту файлів, а також шифрування засноване на парі відкритий/закритий ключ для захисту випадково згенерованого ключа шифрування для кожного файлу. За замовчуванням закритий ключ користувача захищений за допомогою шифрування користувальницьким паролем, і захищеність даних залежить від стійкості пароля користувача.

**Мета й завдання дослідження.** Метою роботи є програмне забезпечення системи кібербезпеки застосунків та даних захищеної файлової системи.

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

– Огляд існуючих систем застосунків та даних захищеної файлової системи.

– Дослідження системи кібербезпеки застосунків та даних захищеної файлової системи.

– Програмна реалізація системи кібербезпеки застосунків та даних захищеної файлової системи.

**Практична цінність отриманих результатів** полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі застосунків та даних захищеної файлової системи.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки застосунків та даних захищеної файлової системи, 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

Програмний продукт призначений для доступу до файлів і папок, зашифрованих засобами Encrypting File System (EFS). в Windows. Додаткові функції шифрування файлової системи (Encrypting File System, EFS) істотно підвищили безпеку Windows, забезпечивши додаткову гнучкість для корпоративних користувачів при розгортанні рішень безпеки, заснованих на шифруванні файлів з даними. Будь-який зловмисник, що має фізичний доступ до комп'ютера, може завантажити на ньому іншу ОС, обійти захист основної ОС і одержати доступ до конфіденційних даних. Шифрування конфіденційних файлів засобами EFS забезпечує додатковий захист. Дані зашифрованого файлу залишаться недоступними, навіть якщо атакуючий одержить повний доступ до середовища зберігання даних комп'ютера. Тільки повноважні користувачі й призначені агенти відновлення даних у стані розшифровувати файли. Користувачі з іншими обліковими записами, що володіють дозволами для файлу – навіть дозволом на передачу прав володіння (Take Ownership), не в змозі відкрити його. Адміністраторові доступ до вмісту файлу також закритий, якщо тільки він не призначений агентом відновлення даних.

Систему шифрування EFS можна виключати/включати в такий спосіб. У розділі реєстру: HKLM \Software \Microsoft \Windows NT \CurrentVersion \EFS key створити або відредагувати вже існуючий dword-параметр "EfsConfiguration". Привласнивши цьому параметру значення 1, ви виключите EFS, а привласнивши 0, включите назад. Для того щоб зміни ввійшли в силу, необхідно перезавантажити комп'ютер. Очевидно, що, відключивши EFS, ви втратите можливість не тільки шифрувати файли, але й розшифровувати раніше зашифровані. Як очевидно й те, що включення/відключення EFS впливає відразу

на всіх користувачів. Але на ключі користувачів (як відкриті, так і особисті) ця дія не зробить абсолютно ніякого впливу, тому після того як EFS буде включена знову, ніяких проблем з доступом до своїх файлів користувачі не випробують.

**Агент відновлення**

Користувач, якому виданий сертифікат відкритого ключа для відновлення користувальницьких даних, закодованих шифрованою файловою системою (EFS).

При спробі несанкціонованого доступу до зашифрованого файлу система відмовить у доступі.

**Архівування й відновлення зашифрованих файлів**

Архівні копії зашифрованих файлів також будуть зашифровані з використанням програми архівації для Windows 7. Після відновлення зашифровані дані залишаються в зашифрованому виді.

**Відновлення зашифрованих даних**

Відновлення даних має на увазі процес розшифровки файлу без закритого ключа користувача, що зашифрував файл.

Може знадобитися відновити дані за допомогою агента відновлення в наступних випадках.

– Користувач залишає організацію.

– Закритий ключ загублений користувачем.

– Отримано запит урядового закладу.

При відновленні файлу агентом відновлення виконуються наступні дії.

1. Архівація зашифрованих файлів.
2. Переміщення архівних копій у безпечну систему.
3. Імпорт сертифіката відновлення й закритого ключа.
4. Відновлення архівних файлів.
5. Розшифровка файлів за допомогою провідника Windows або команди EFS cipher.

**Налаштування політики відновлення**

Для визначення політики відновлення даних рядових серверів домену, автономних серверів або серверів робочих груп можна скористатися оснащенням «Групова політика». Сертифікат відновлення можна або запросити, або експортувати й потім імпортувати.

Можна делегувати керування політикою відновлення призначеному адміністраторові. Хоча число адміністраторів, які можуть відновлювати зашифровані дані, потрібно обмежувати, наявність декількох агентів відновлення дозволяє мати додаткове джерело, якщо відновлення необхідно.

Щоб створити резервну копію ключів відновлення, використовуваних за замовчуванням, на гнучкому диску:

1. Натисніть кнопку Пуск, виберіть команду Виконати, уведіть mmc і натисніть кнопку ОК.

2. У меню Консоль виберіть команду Додати/видалити оснащення й натисніть кнопку Додати.

3. У групі Додати ізольоване оснащення виберіть Сертифікати натисніть кнопку Додати.

4. Поставте перемикач у положення мого облікового запису користувача й натисніть Готово.

5. Клацніть Закрити, потім ОК.

6. Двічі клацніть Сертифікати – поточний користувач, двічі клацніть Особисті й двічі клацніть Сертифікати.

7. Виберіть сертифікат, що має текст Відновлення файлів у стовпці Призначення.

8. Клацніть сертифікат правою кнопкою, виберіть Всі завдання, а потім Експорт.

9. Додержуйтеся інструкцій майстра експорту сертифікатів для експорту сертифіката й зв'язаного закритого ключа у файл формату .pfx.

Примітки:

– Ця операція повинна бути виконана з обліковим записом агента відновлення, у якого є сертифікат відновлення й закритий ключ в особистих сховищах.

– Перед внесенням будь-яких змін у політику відновлення, використовувану за замовчуванням, перевірте безпеку ключів відновлення, використовуваних за замовчуванням. У домені ключі відновлення, використовувані за замовчуванням, зберігаються на першому контролері домену. Адміністратор домену за замовчуванням є агентом відновлення.

### 1.2 Область застосування

EFS базується на технології шифруванні з відкритим ключем і використовує архітектуру CryptoAPI. Стандартна (за замовчуванням) конфігурація EFS не вимагає адміністративного втручання: ви вправі виконувати шифрування файлів відразу ж після установки системи. EFS автоматично створює пари ключів шифрування й сертифікат користувача, якщо вони не були створені раніше. Як алгоритм шифрування EFS використовує DESX (E7anded Data Encryption Standard), 3DES (Triple-DES) або AES. Постачальники послуг криптографії підтримують два алгоритми: RSA Base і RSA Enhanced – для створення сертифікатів EFS і для шифрування симетричних ключів шифрування. Якщо зашифрувати папку, всі файли й підпапки в ній шифруються автоматично. Рекомендується шифрування саме на рівні папок, щоб у процесі роботи не з'являлися незашифровані тимчасові файли.

**EFS і NTFS**

Шифрована файлова система (EFS) захищає конфіденційні дані у файлах на томах NTFS. EFS – основна технологія шифрування й розшифровки файлів на томах NTFS. Відкривати файл і працювати з ним може тільки користувач, його що зашифрував. Це надзвичайно важливо для користувачів переносних комп'ютерів: навіть якщо зломщик одержить доступ до загубленого або

украденого комп'ютера, він не зможе відкрити зашифровані файли. В Windows 7 шифрована файлова система також підтримує автономні файли й папки (Offline Files and Folders). Зашифрований файл залишиться недоступним для перегляду у вихідному виді, навіть якщо атакуючий обійде системний захист, наприклад, завантаживши іншу ОС. EFS забезпечує стійке шифрування по стандартних алгоритмах і тісно інтегрована з NTFS. EFS в Windows 7 Professional надає нові можливості спільного використання зашифрованих файлів або відключення агентів відновлення даних, а також полегшує керування за допомогою групової політики й службових програм командного рядка.

**Як працює EFS**

EFS дозволяє зберегти конфіденційність інформації на комп'ютері в умовах, коли люди, що мають фізичний доступ до комп'ютера, можуть навмисно або ненавмисно скомпрометувати її. EFS надзвичайно зручна для забезпечення конфіденційності даних на мобільних комп'ютерах або на комп'ютерах, на яких працюють кілька користувачів, тобто таких системах, які можуть піддаватися атакам, що передбачають обхід обмежень списків ACL. У спільно використовуваній системі атакуючий звичайно одержує несанкціонований доступ, завантажуючи іншу ОС. Зловмисник також може захопити комп'ютер, вийняти жорсткий диск, помістити його на інший комп'ютер і одержати доступ до файлів. Однак якщо в нього немає ключа розшифровки, зашифрований засобами EFS файл буде виглядати як безглуздий набір символів. Оскільки EFS тісно інтегрована з NTFS, шифрування й розшифровка виконуються непомітно ("прозоро") для користувача. При відкритті файлу EFS автоматично розшифровує його в міру читання даних з диска, а при записі – шифрує дані при записі на диск. Працюючи із зашифрованим файлом, ви можете навіть не догадуватися, що він зашифрований (за умови, що у вас є відповідні права). У стандартній конфігурації EFS дозволяє зашифрувати файл прямо із Провідника Windows без якого-небудь втручання адміністратора. З погляду користувача шифрування файлу або папки – це призначення йому певного атрибута.

**Конфігурування EFS**

За замовчуванням система підтримує роботу EFS. Дозволяється шифрувати файли, для яких є дозвіл на зміну. Оскільки в EFS для шифрування файлів застосовується відкритий ключ, потрібно створити пари ключів відкритий/закритий і сертифікат з відкритим ключем шифрування. В EFS дозволені сертифікати, підписані самим власником, тому втручання адміністратора для нормальної роботи не потрібно. Якщо застосування EFS не відповідає вимогам організації або якщо є файли, які не можна шифрувати, існує багато способів відключити EFS або потрібним образом конфігурувати її. Для роботи з EFS всім користувачам потрібні сертифікати EFS. Якщо в організації немає інфраструктури відкритого ключа (Public Key Infrastructure, PKI), застосовуються підписані самим власником сертифікати, які автоматично створюються ОС. Сертифікати EFS звичайно випускають центри сертифікації. Якщо ви використовуєте EFS, обов'язково передбачите можливість відновлення даних при збої системи.

**Що дозволяється шифрувати**

На томах NTFS атрибут шифрування дозволяється призначати окремим файлам і папкам з файлами (або підпапками). Хоча папку з атрибутом шифрування й називають "зашифрованою", сама по собі вона не шифрується, і для установки атрибута пари ключів не потрібно. При встановленому атрибуті шифрування папки EFS автоматично шифрує: всі нові файли, створювані в папці; всі незашифровані файли, скопійовані або переміщені в папку; всі вкладені файли й підпапки (на особливу вимогу); автономні файли.

**Шифрування бази даних автономних файлів**

В Windows 7 можна шифрувати базу даних автономних файлів для локального захисту кешуємих документів від злодійства комп'ютера, а також забезпечення додаткової безпеки локально кешуємих даних. В Windows 2000 цієї функції не було – вона передбачає шифрування кешуємих файлів. Наприклад, ви вправі активно використовувати автономні файли, при цьому конфіденційність

даних забезпечується автоматично. Як адміністратор відділу технічної підтримки ви можете задіяти цю можливість, щоб убезпечити все локально кешуємі документи. Автономні файли – чудовий захист від втрати конфіденційних даних при захвату мобільного комп'ютера. Зазначена функція підтримує шифрування й розшифровку всієї автономної бази даних. Для конфігурування порядку шифрування автономних файлів потрібні адміністративні привілеї. Щоб зашифрувати автономні файли, відкрийте папку Мій комп'ютер (My Computer) і в меню Сервіс (Tools) виберіть команду Властивості папки (Folder Options), у вікні, що відкрилося, властивостей на вкладці Автономні файли (Offline Files) установите прапорець Шифрувати автономні файли для захисту даних (Encrypt Offline Files To Secure Data).

**Віддалені операції EFS на загальних файлах і Web-папках**

Можна шифрувати й розшифровувати файли, розташовані в Web-папках Web Distributed Authoring and Versioning (розподілена система зберігання файлів з доступом через Web), або WebDAV. В Web-папок багато переваг у порівнянні із загальними файлами, і Microsoft рекомендує максимально широко застосовувати їх для віддаленого зберігання шифрованих файлів. Web-папки вимагають менше уваги від адміністраторів і безпечніше, ніж загальні файли. Web-папки також забезпечують безпечне зберігання й доставку шифрованих файлів через Інтернет засобами стандартного протоколу HTTP. Щоб використовувати загальні файли для віддалених операцій EFS, потрібна доменне середовище Windows 2000 або більше пізніх версій Windows, тому що при шифруванні й розшифровці користувальницьких файлів EFS працює від імені користувача за допомогою протоколу делегування повноважень в Kerberos. Основна відмінність віддалених операцій EFS із загальними файлами й файлами в Web-папках – те, у якому місці ці операції виконуються. Якщо файли зберігаються в загальних файлах, всі операції EFS виконуються на комп'ютері, де розташований файл. Так, якщо ви підключилися до загального мережного файлу й намагаєтеся відкрити раніше зашифрований файл, він розшифровується на комп'ютері, де зберігається, а потім

передається відкритим текстом по мережі на ваш комп'ютер. При зберіганні файлу на Web-папках всі операції EFS виконуються на локальному комп'ютері. При підключенні до Web-папки й спроби відкрити зашифрований файл останній пересилається по мережі в зашифрованому виді на локальний комп'ютер і вже там розшифровується системою EFS. Вхідний і вихідний трафік Web-папок – це неопрацьовані дані, які, навіть перехоплені атакуючим, залишаються зашифрованими й зовсім для нього марні. Таке розходження у виконанні операцій EFS пояснює, чому загальні файли вимагають більших зусиль із боку адміністраторів, ніж Web-папки. EFS з Web-папками усуває необхідність у спеціалізованому ПЗ для безпечного спільного використання зашифрованих файлів користувачами й організаціями. Файл може зберігатися у вільному доступі на файлових серверах в інтрамережі або в Інтернеті й при цьому залишатися надійно захищеним засобами EFS.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки застосунків та даних захищеної файлової системи, 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

У рамках виконання бакалаврського проектування розглянемо існуючі програмні рішення захисту файлів на основі використання технології EFS.

**Advanced EFS Data Recovery**

Microsoft Encrypting File System (EFS) є складовою частиною операційних систем Microsoft Windows, що дозволяє захищати дані від несанкціонованого доступу навіть у тих випадках, коли зловмисник заволодів комп'ютером або накопичувачами із зашифрованими даними, що зберігаються на них.

Advanced EFS Data Recovery відновлює зашифровані файли й папки й працює у всіх версіях Windows 2000, XP, Windows Server 2003, Vista, Windows Server 2008, Windows 7. Відновлення можливо навіть у випадках, коли система ушкоджена, не завантажується або коли знищені деякі ключі шифрування.

Advanced EFS Data Recovery відновлює зашифровані дані, що стали недоступними внаслідок помилок адміністрування. Приклади таких помилок:

– видалення облікових записів користувачів;

– відсутність агентів відновлення (Data Recovery Agents) або їхнє неправильне конфігурування;

– некоректний перенос облікових записів в інший домен;

– перенос дисків із зашифрованими даними між комп'ютерами.

Advanced EFS Data Recovery є потужною утилітою відновлення даних, що допомагає одержати доступ до файлів у цілому ряді складних випадків:

– Диск із зашифрованими даними вставлений в інший комп'ютер.

– Видалено облікові записи або профілі користувачів.

– Некоректний (без обліку EFS) перенос облікового запису в інший домен.

– Системний адміністратор скинув пароль облікового запису.

– Ушкоджений диск або файлова система, операційна система не завантажується.

– Зроблений апгрейд комп'ютера або з операційна система Windows.

– Системний розділ відформатований, але зашифровані дані залишилися на іншому диску.

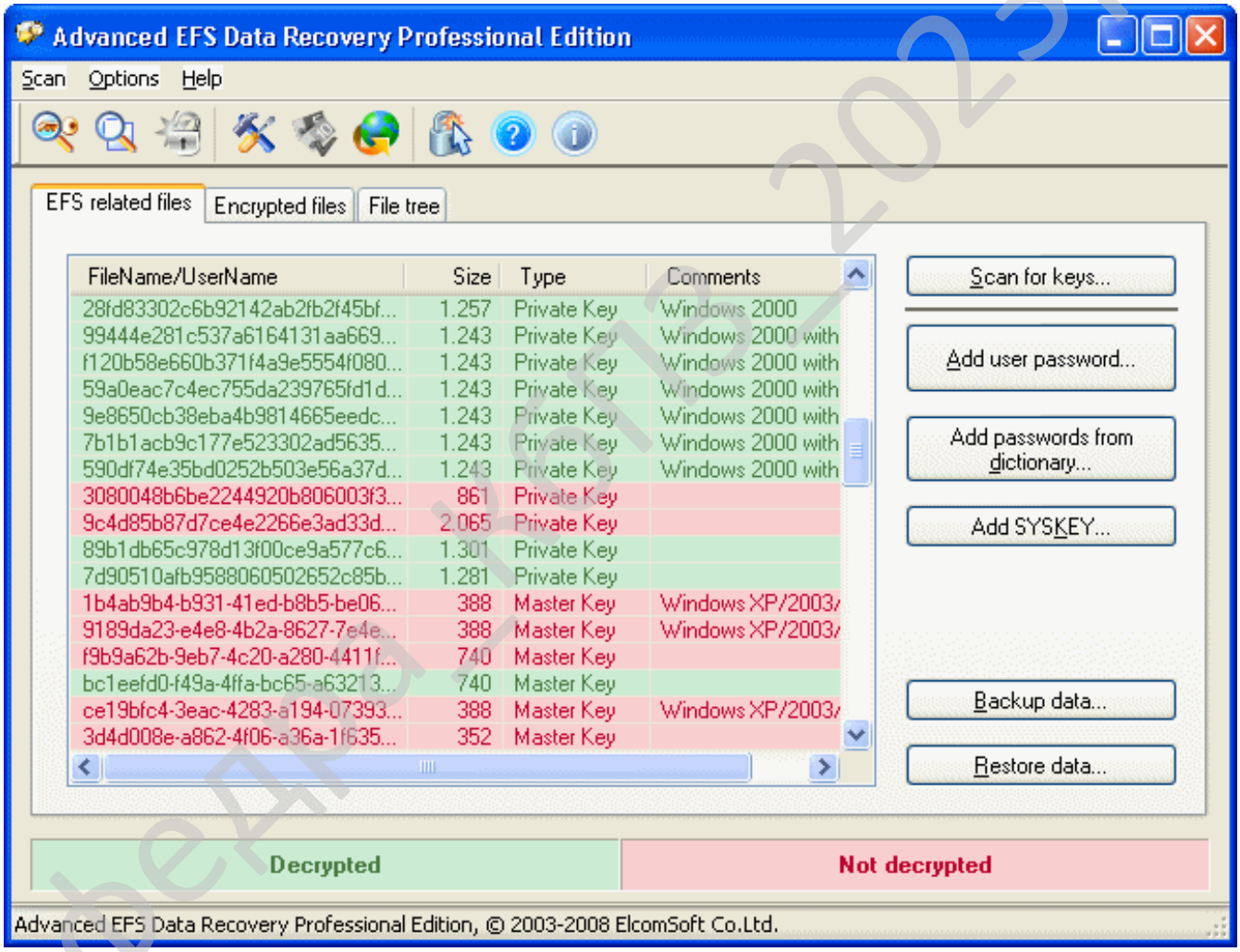


Рисунок 2.1 – Інтерфейс користувача Advanced EFS Data Recovery

EFS повністю прозора для користувачів і забезпечує шифрування й дешифрування на льоту з використанням сильних криптографічних алгоритмів.

Простота використання й повна прозорість для користувача можуть зіграти поганий жарт, оскільки користувач часто просто забуває, що на дисках зберігаються зашифрованні дані, і спокійно переустановлює Windows або встановлює диск із даними в новий комп'ютер.

Advanced EFS Data Recovery швидко й ефективно розшифровує дані, захищені засобами EFS. Скануючи диск сектор за сектором, Advanced EFS Data Recovery виявляє зашифровані файли й доступні ключі шифрування, після чого дешифрує виявлені файли. Прямий доступ до файлової системи дозволяє Advanced EFS Data Recovery відновлювати зашифровані файли в самих складних випадках – наприклад, коли немає можливості ввійти в систему, або коли деякі ключі шифрування ушкоджені або недоступні.

За допомогою Advanced EFS Data Recovery у переважній більшості випадків можливе відновлення зашифрованих EFS даних. У програмі повною мірою використовується EFS, що є присутнім в Windows 2000, що дозволяє істотно спростити відновлення файлів. Якщо відомо пароль облікового запису (або пароль, використовуваний раніше, перше ніж він був видалений системним адміністраторів) або Адміністратора, то доступ до зашифрованих файлів можна одержати практично миттєво.

Професійна редакція AEFSDR виявляє майстер- і особисті ключі у видалених файлах, сканує диск на рівні секторів і використовує шаблони для виявлення ключів шифрування, що дає можливість відновити дані у випадку переформатування дисків або переустановки Windows.

**Diskinternals EFS Recovery**

Цей інструмент може використовуватися для відновлення ушкоджених або видалених файлів і папок з дискових розділів, захищених засобами шифрування Windows Encrypted File System (EFS). Додаток також може виявитися корисним у випадку переносу диска із захищеними файлами на інший комп'ютер, що працює під керуванням тої ж або іншої версії Windows.

Файлова система EFS використовується в операційних системах Windows 7 і 2008 Server R1/R2, а також Windows 2000, XP, 2003 Server і Windows Vista, 7 для захисту конфіденційної ділової й персональної інформації. Diskinternals EFS Recovery повністю автоматизує процедуру відновлення даних з EFS-розділів будь-якого типу й гарантує успішне виявлення видалених файлів і папок на працездатних, ушкоджених і навіть більше недоступних дисках і розділах.

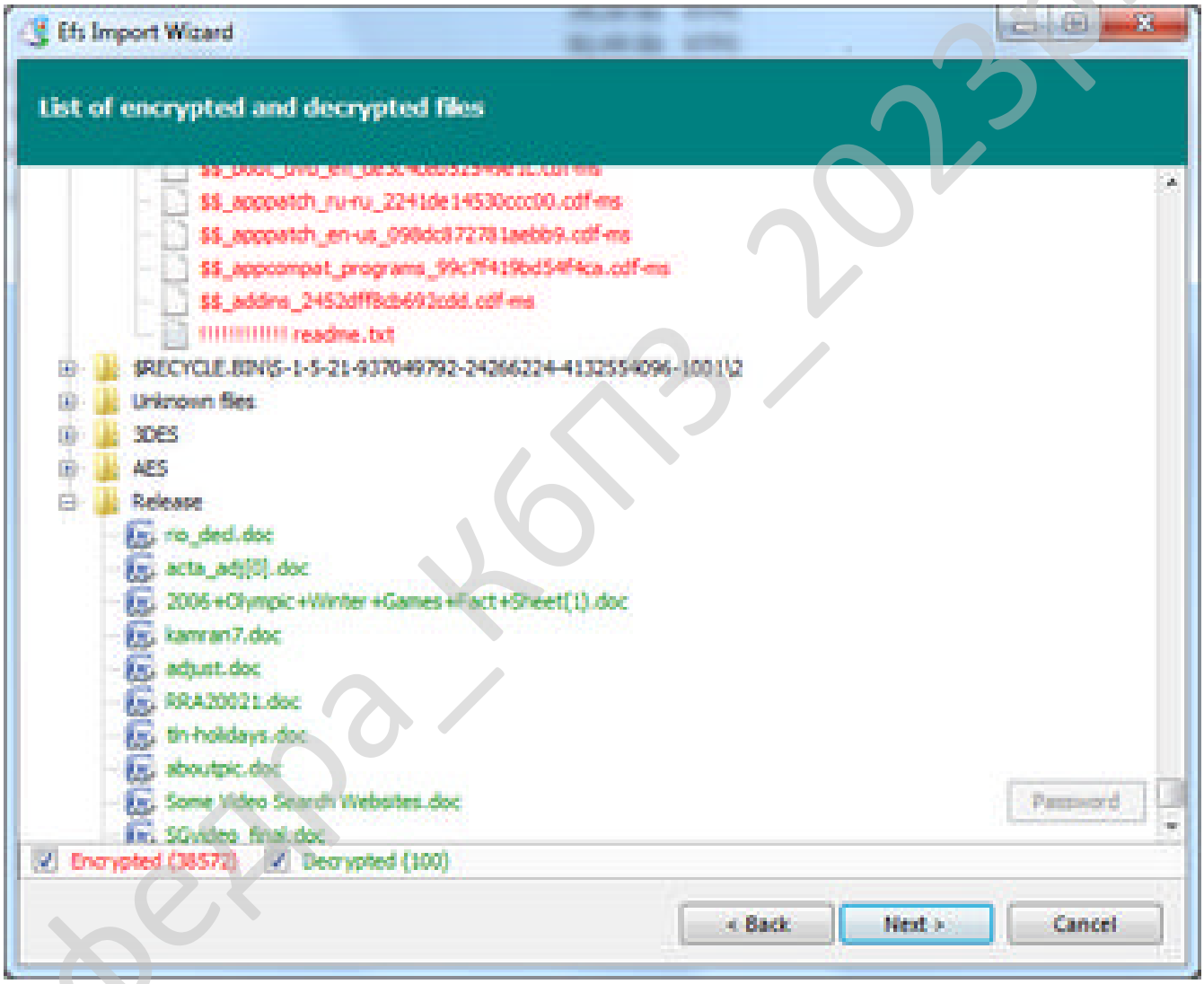


Рисунок 2.2 – Інтерфейс користувача EFS Recovery

Diskinternals EFS Recovery є першим комерційним продуктом, що підтримує патентовану технологію PowerSearch. Ця технологія допускає

відновлення файлів, більше не зареєстрованих у файловій системі й дозволяє вирішити поставлене завдання навіть у випадку серйозних механічних ушкоджень жорсткого диска, видалення розділу або повторного форматування носія. У ході сканування диска PowerSearch здатна виявляти файли 200 підтримуваних типів, у число яких входять розповсюджені формати документів, архіви ZIP і RAR, зображення, аудіо- і відеодані, а також інші дані.

Продукт пропонує функцію попереднього перегляду виявлених файлів. Завдяки цій можливості користувачі зможуть переглядати документи й зображення, відтворювати аудіо- і відеофайли й навіть вивчати вміст файлових архівів перед ухваленням рішення про їхнє відновлення. Перед тим як приступитися до відновлення, додаток попросить користувача ввести вірний пароль облікового запису. Якщо цей пароль у минулому був змінений, клієнтові досить буде згадати одне з використовуваних кодових слів.

**КРИПТО-ПРО EFS**

ПЗ "КРИПТО-ПРО EFS" предназначено для забезпечення захисту конфіденційної інформації при її зберіганні на ПЕОМ.

При використанні разом з "КРИПТО-ПРО CSP", "КРИПТО-ПРО EFS" забезпечує:

– конфіденційність інформації, що зберігається на комп'ютері шифруванням файлів файлової системи NTFS по алгоритму ДСТ 28147 89;

– контроль цілісності інформації, що зберігається на комп'ютері обчисленням імітовставки відповідно до ДСТ 28147 89;

– організацію спільного доступу до даних зашифрованого файлу обмеженій групі користувачів;

– організацію видаленої роботи із захищеними файлами, розташовуваними в Web-папках (Web Distributed Authoring and Versioning – розподілена система зберігання файлів з доступом через Web або WebDAV);

– можливість видаленої роботи із зашифрованими файлами користувачів, що використовують Terminal Services для видаленого входу в систему.

– відновлення даних у випадку видалення користувачів із системи, компрометації або втрати закритого ключа користувача.

КРИПТО-ПРО EFS повністю підтримує інтерфейс, запропонований Microsoft для файлової системи ОС Windows, що шифрує. Робота із захищеними файлами не вимагає змін у користувальницьких додатках, підтримуються інтерфейси для шифрування, розшифрування файлів і папок, редагування групи користувачів захищених файлів, пропонованих в ОС Windows.

Основні відмінності КРИПТО-ПРО EFS від убудованого в ОС Windows EFS:

– КРИПТО-ПРО EFS використовує російські стандарти криптографічного захисту даних.

– КРИПТО-ПРО EFS забезпечує контроль цілісності файлів.

– КРИПТО-ПРО EFS надає користувачеві інтерфейс для вибору поточного ключа шифрування, ключа, за замовчуванням використовуваного для шифрування файлів.

– КРИПТО-ПРО EFS дозволяє одночасно адмініструвати багато захищених файлів. Перевіряти цілісність багатьох файлів, робити одночасне перешифрування багатьох файлів.

Для проведення криптографічних операцій КРИПТО-ПРО EFS потрібно, щоб на робочій станції був установлений КРИПТО-ПРО CSP 3.0 або 3.6. Для виробітку сертифікатів можуть бути використані "КРИПТО-ПРО УЦ", MS CA. Користувач КРИПТО-ПРО EFS може зберігати свій секретний ключ на відокремлюваних ключових носіях – дискеті, флеш-диску, смарт-карті або USB-токені.

## 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 застосунків та даних захищеної файлової системи.

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

Файлова система EFS є стандартною файловою системою Windows, призначеною для шифрування файлів і папок. Вона є надбудовою над файловою системою NTFS, і дозволяє легко шифрувати й дешифрувати вміст окремих файлів.

Виконати шифрування файлу можна двома способами: або за допомогою програми командного рядка Cipher.exe, або за допомогою прапорця ШИФРУВАТИ ВМІСТ ДЛЯ ЗАХИСТУ ДАНИХ діалогу ДОДАТКОВІ АТРИБУТИ (відображається після натискання на кнопку ІНШІ..., розташовану на вкладці ЗАГАЛЬНІ діалогу ВЛАСТИВОСТІ ФАЙЛУ).

Дешифрація ж виконується автоматично при відкритті зашифрованого файлу користувачем, що цей файл шифрував, або входить у групу користувачів, які можуть відкрити даний зашифрований файл.

Довідка й підтримка: Основні відомості про механізм EFS можна одержати з наступних розділів довідки:

– 196e 3453-e553-4af 3-8220-bdee6e60148c. Використання смарт-карти для шифрування файлу.

– 21ad 2809-1f05-452 c-ae20-bca7994fed10. Зміна ключа, використовуваного для шифрування файлів і папок.

– 2bfc10 ba-0869-47 cd-9fb2-aee51a8a375a. Відновлення зашифрованих файлів і папок.

– 2fae 3470-742d-4902-95ab-f77cbafb12d4. У чому небезпека скидання пароля.

– 4121b78 c-9cb0-4715-93 ec-80ba841670a3. Резервне копіювання сертифіката шифрованої файлової системи (EFS).

– 42285c 67-e346-4523-a61 c-5649660c275b. Спільне використання зашифрованих файлів.

– 5a2b6b 98-9833-4d 73-967e-9293bd1a54e9. Шифрування й розшифровка папки або файлу.

– 62f 28747-c146-483c-b 378-ff3f3977f011. У чому різниця між шифруванням диска BitLocker і шифрованою файловою системою EFS.

– 67bc 0494-9a42-4c8 a-88fd-6fa4c9c4c498. Що робити, якщо загублено ключ шифрування файлу.

– 75d60d 1446-4106-bd 18-a0f58685371c. Шифрування: посилання, що рекомендуються.

– 90cdd1 fe-9cbb-4adc-bccf-7d613425e15e. Створення сертифіката відновлення для зашифрованих файлів.

**Етапи шифрування EFS**

Давайте докладніше розглянемо процес шифрування файлу за допомогою EFS.

**Етап 1. Завантаження профілю**

Для шифрування файлу необхідно, щоб профіль користувача був завантажений операційною системою, тому що саме на його основі виконується створення ключів.

При вході користувача в систему профіль завантажується завжди, однак, наприклад, при використанні програми командного рядка runas.exe профіль іншого користувача може й не завантажуватися. У цьому випадку профіль користувача буде завантажений вручну.

**Етап 2. Створення файлу журналу**

Якщо профіль користувача завантажений, тоді всі необхідні для шифрування файлу дані присутні, і починається процес шифрування.

Процес шифрування починається зі створення в каталозі System Volume Information файлу журналу шифрування. Він має ім'я формату efs.log, де X

визначає номер файлу, що був зашифрований у поточний сеанс шифрування процесом lsasrv.exe.

**Етап 3. Генерація FEK**

Починається процес генерації випадкового 128-бітного числа FEK.

Число FEK використовується для шифрування вмісту файлу по певному алгоритму. Після цього число FEK шифрується за допомогою алгоритму RSA, використовуючи при цьому відкритий користувальницький ключ, і отримане значення додається до самого зашифрованого файлу.

Після виконання цих операцій файл можна вважати зашифрованим. Тепер для відкриття файлу необхідно спочатку розшифрувати його вміст за допомогою FEK. А FEK, у свою чергу, потрібно взяти з файлу (зашифрованого) і розшифрувати за допомогою алгоритму RSA на основі користувальницьких ключів. Користувальницькі ж ключі можна одержати тільки після завантаження профілю того користувача, що зашифрував файл (або після завантаження профілю користувача, якому дозволений доступ до зашифрованого файлу).

Однак поки що тільки генерується число FEK, а користувальницькі ключі, необхідні для його шифрування, ще не отримані.

**Етап 4. Одержання відкритого й закритого користувальницького ключа**

Пари користувальницьких ключів (закритого і відкритого) генерується при першому процесі шифрування, ініційованим користувачем. Тому якщо користувач ніколи ще не виконував шифрування файлів, тоді відбувається генерація пари користувальницьких ключів.

Якщо ж користувальницькі ключі були згенеровані раніше, тоді операційна система бере дані, необхідні для створення сигнатури відкритого ключа, з параметра REG_BINARY типу CertificateHash, розташованого в гілці реєстру HKCU\Software\Microsoft\Windows NT\CurrentVersion\EFS\CurrentKeys. Після цього формується відкритий ключ.

Другий же ключ, закритий, береться з каталогу %userprofile%\AppData\Roaming\Microsoft\Crypto\RSA\«SID-користувача». Вміст даного каталогу зашифровано на основі симетричного ключа, називаного майстер-ключем користувача.

Майстер ключ користувача втримується в каталозі %userprofile%\AppData\Roaming\Microsoft\Protect\« SID-користувача». Даний ключ також зашифрований (за допомогою алгоритму 3DES і паролю від облікового запису користувача).

**Етап 5. Створення DDF**

Після одержання FEK і користувальницьких ключів, починається процес створення зв'язування ключів DDF (сукупності декількох DDF).

Один ключ DDF містить інформацію про один користувача, якому дозволено відкривати даний зашифрований файл, а також про його права доступу. Ключ DDF складається з наступної інформації: SID-користувача, ім'я контейнера, ім'я провайдеру, що зашифрував файл, хеш сертифіката EFS, використовуваний при розшифровці, зашифрований FEK для користувача.

**Етап 6. Створення DRF**

Наступним зв'язуванням ключів, що створюється для шифруємого файлу, є зв'язування ключів DRF.

Один ключ DRF містить інформацію про один агента відновлення, що може відкрити даний зашифрований файл. Інформація в ключі DRF аналогічна інформації ключа DDF.

Агентом відновлення називається обліковий запис користувача, що може відкрити будь-який зашифрований файл даної операційної системи. Як правило, агентом відновлення призначається адміністратор.

**7. Шифрування**

Всі дані, необхідні для шифрування файлу, створені. Тепер можна починати сам процес шифрування.

Процес шифрування починається зі створення резервної копії шифруємого файлу (який має ім'я efs0.tmp). Саме над резервною копією файлу буде виконуватися процес шифрування, після чого вміст резервного файлу буде скопійовано у вихідний файл, і резервний файл, разом з файлом журналу шифрування, буде видалений.

Алгоритм шифрування за допомогою FEK Отже, шифрування вмісту резервного файлу виконується на основі числа FEK за допомогою алгоритму AES.

В операційних системах сімейства Windows NT існує можливість змінити алгоритм, використовуваний при шифруванні файлу на 3DES, тим самим підсиливши його.

Для цього потрібно параметру DWORD-типу AlgorithmID, розташованому в гілці реєстру HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\EFS, привласнити значення 0x6603. У цьому випадку алгоритм 3DES буде використовуватися тільки службою EFS.

Також можна встановити використання алгоритму 3DES для шифрування, хешування й підписування (а не тільки для EFS). Для цього потрібно запустити оснащення РЕДАКТОР ОБ'ЄКТІВ ГРУПОВОЇ ПОЛІТИКИ (консоль gpedit.msc) і перейти до підрозділу КОНФІГУРАЦІЯ КОМП'ЮТЕРА / КОНФІГУРАЦІЯ WINDOWS / ПАРАМЕТРИ БЕЗПЕКИ / ЛОКАЛЬНІ ПОЛІТИКИ / ПАРАМЕТРИ БЕЗПЕКИ. У даному підрозділі потрібно встановити в положення ВКЛЮЧЕНЕ значення елемента СИСТЕМНА КРИПТОГРАФІЯ: ВИКОРИСТОВУВАТИ FIPS-СУМІСНІ АЛГОРИТМИ ДЛЯ ШИФРУВАННЯ, ХЕШУВАННЯ Й ПІДПИСУВАННЯ.

Шифрування FEK. Після того, як файл буде зашифрований, число FEK також шифрується. Для цього застосовується алгоритм, що шифрує число FEK на основі користувальницьких ключів.

В операційній системі Windows 7 також можна налаштувати параметри одержання доступу до закритих користувальницьких ключів.

Для цього потрібно запустити оснащення РЕДАКТОР ОБ'ЄКТІВ ГРУПОВОЇ ПОЛІТИКИ (консоль gpedit.msc) і перейти до підрозділу КОНФІГУРАЦІЯ КОМП'ЮТЕРА / КОНФІГУРАЦІЯ WINDOWS / ПАРАМЕТРИ БЕЗПЕКИ / ЛОКАЛЬНІ ПОЛІТИКИ / ПАРАМЕТРИ БЕЗПЕКИ. У даному підрозділі потрібно скористатися елементом СИСТЕМНА КРИПТОГРАФІЯ: ОБОВ'ЯЗКОВЕ ЗАСТОСУВАННЯ СИЛЬНОГО ЗАХИСТУ КЛЮЧІВ КОРИСТУВАЧІВ, ЩО ЗБЕРІГАЮТЬСЯ НА КОМП'ЮТЕРІ.

Значення параметра ForceKeyProtection:

– 0. Не потрібно введення даних користувачем при збереженні й використанні нових ключів.

– 1. Користувач одержує запит при першому використанні ключа.

– 2. Користувач повинен уводити пароль при кожному використанні ключа.

**Налаштування EFS**

Після того, як ми розглянули алгоритм шифрування файлів і папок за допомогою EFS, давайте розглянемо деякі налаштування цієї файлової системи для шифрування.

**Налаштування оснащення Редактор об'єктів групової політики**

Більшість налаштувань файлової системи EFS можна змінити за допомогою підрозділу КОНФІГУРАЦІЯ КОМП'ЮТЕРА / КОНФІГУРАЦІЯ WINDOWS / ПАРАМЕТРИ БЕЗПЕКИ / ПОЛІТИКИ ВІДКРИТОГО КЛЮЧА / ФАЙЛОВА СИСТЕМА, ЩО ШИФРУЄ (EFS) оснащення РЕДАКТОР ОБ'ЄКТІВ ГРУПОВОЇ ПОЛІТИКИ.

За допомогою команди ВЛАСТИВОСТІ контекстного меню даного підрозділу відображається діалог ВЛАСТИВОСТІ: ФАЙЛОВА СИСТЕМА, ЩО ШИФРУЄ (EFS). Елементи даного діалогу змінюють значення параметрів DWORD-типу гілки реєстру HKLM\SOFTWARE\Policies\Microsoft\Windows NT\CurrentVersion\EFS.

Більшість елементів діалогу змінюють біти бітової маски EfsOptions. Також елементи даного діалогу змінюють значення параметрів, представлених нижче.

Біти бітової маски EfsOptions:

– 0x00000001. Шифрувати вміст папки «Документи» користувача.

– 0x00000002. Створити кешуємий користувальницький ключ зі смарт-карты.

– 0x00000004. Дозволити EFS створювати сертифікати, що самозавіряють, коли центр сертифікації недоступний.

– 0x00000010. Минув час очікування кешу поля Очищати кеш ключів шифрування, коли.

– 0x00000020. Користувач заблокував робочу станцію поля Очищати кеш ключів шифрування, коли.

– 0x00000100. Вимагати смарт-карту для EFS.

– 0x00000400. Відображати повідомлення про архівацію ключа при створенні або зміні користувальницького ключа.

Гілка реєстру HKLM\SOFTWARE\Policies\Microsoft\Windows NT\CurrentVersion\EFS:

– EfsConfiguration. Якщо дорівнює 1, значення параметрів гілці HKLM\SOFTWARE\Policies\Microsoft\Windows NT\CurrentVersion\EFS застосовуватися не будуть.

– RsaKeyLength. Розмір ключа для сертифікатів, що самозавіряють.

– CacheTimeout. Час очікування кешу й визначає час (у хвилинах), після закінчення якого кеш ключів шифрування буде очищатися.

**Налаштування кольору**

За замовчуванням назви стислих файлів і папок відображаються синім кольором, а назви зашифрованих файлів і папок – зеленим.

Ці колірні налаштування можна змінити за допомогою параметрів REG_BINARY типу, розташованих у гілки реєстру

HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer. Дані параметри мають наступний формат: 0xR 0xG 0xB 00, де 0xR представляє червоний компонент кольору, 0xG зелену, а 0xB синю.

– AltColor. Визначає колір назви стислих файлів.

– AltEncryptionColor. Визначає колір назви зашифрованих файлів.

**Програми для роботи з EFS**

Крім стандартних можливостей оболонки Windows, у поставку Windows 7 входить набір програм для роботи з EFS. Нижче ми їх опишемо.

**Програма cipher.exe**

Крім шифрування файлів і папок за допомогою прапорця ШИФРУВАТИ ВМІСТ ДЛЯ ЗАХИСТУ ДАНИХ діалогу ДОДАТКОВІ АТРИБУТИ, для виконання шифрування EFS можна використовувати програму командного рядка cipher.exe.

Основних синтаксисів даної програми, використовуваних для роботи з можливостями шифрування Windows, п'ять:

– Cipher / E / S:«каталог» опції «файл або папка». Виконати шифрування файлу або папки, розташованих у каталозі, зазначеному в опції / S.

– Cipher / D / S:«каталог» опції «файл або папка». Виконати дешифрування файлу або папки, розташованих у каталозі, зазначеному в опції / S.

– Cipher / C / S:«каталог» опції «файл або папка». Відобразити стан (зашифрований чи ні) файлу або папки, розташованих у каталозі, зазначеному в опції / S.

– Cipher / ADDUSER [ / CERTHASH: «хеш» | / CERTFILE:«файл сертифіката»] / S:«каталог» опції «файл або папка». Додати користувача, якому дозволено відкривати даний зашифрований файл або папку.

– Cipher / REMOVEUSER / CERTHASH: «хеш» / S:«каталог» опції «файл або папка». Видаляє користувача, якому раніше було дозволено відкривати даний зашифрований файл або папку.

Додатково в описаних вище командах можна вказати опції / B і / H. Опція / B говорить про те, що програма повинна припинити свою роботу, якщо виникне помилка при шифруванні файлів (за замовчуванням проблемний файл пропускається). А за допомогою опції / H можна виконати шифрування схованих і системних файлів.

Також програма cipher.exe підтримує опції / f, / q, / I, / a (опції, доступні у версії програми cipher.exe для Windows XP), однак ці опції ніяк не впливають на роботу програми, тому що вона застосовує їх за замовчуванням. Саме тому ці опції не описані в довідці по програмі.

Крім цих опцій програма cipher.exe підтримує ще одну недокументовану опцію – / FLUSHCACHE.

Крім основних синтаксисів програми можна використовувати додаткові варіанти синтаксису.

Додаткові варіант використання програми cipher.exe:

– Cipher.exe. Відобразити стану файлів поточної папки.

– Cipher / K. Створити новий ключ шифрування.

– Cipher / R:«файл». Створити новий ключ шифрування й сертифікат агента відновлення (файли з розширеннями .PFX і .CER). Файл із розширенням .PFX буде запаролений паролем, що ви введете після уведення даної команди.

– Cipher / U / N. Відобразити список всіх зашифрованих файлів. Без опції / N також буде змінений ключ шифрування для знайдених файлів (якщо ви його тільки що змінили).

– Cipher / W:«папка». Видалити всю інформацію з папки.

– Cipher / R:« efs-файл» «архів». Архівувати сертифікат EFS і ключ шифрування.

– Cipher / Y. Відображає начерк поточного сертифіката EFS.

– Cipher / REKEY «файл або папка». Заново шифрує файли.

**Створення сертифіката шифрування**

Крім програми cipher.exe для створення сертифіката шифрування користувача можна використовувати майстер ФАЙЛОВА СИСТЕМА, ЩО ШИФРУЄ (EFS), викликуваний програмою rekeywiz.exe. Крім виклику даного майстра за допомогою програми rekeywiz.exe, для його відображення можна скористатися посиланням КЕРУВАННЯ СЕРТИФІКАТАМИ ШИФРУВАННЯ ФАЙЛІВ майстри ОБЛІКОВІ ЗАПИСИ КОРИСТУВАЧІВ.

За допомогою майстра ФАЙЛОВА СИСТЕМА, ЩО ШИФРУЄ (EFS) можна створити сертифікат шифрування й помістити його на жорсткий диск комп'ютера, на смарт-карту, або в центр сертифікації домену Active Directory.

## 3.2 Розробка структурної схеми

На рисунку 3.1 показана структурна схема EFS. Структурно EFS складається з наступних компонентів.

**Драйвер EFS**

Цей компонент розташований логічно на вершині NTFS. Він взаємодіє із сервісом EFS, одержує ключі шифрування файлів, 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).

**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 шифрування, 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 шифрування й дешифрування файлів при записі й зчитуванні файлів з диска.

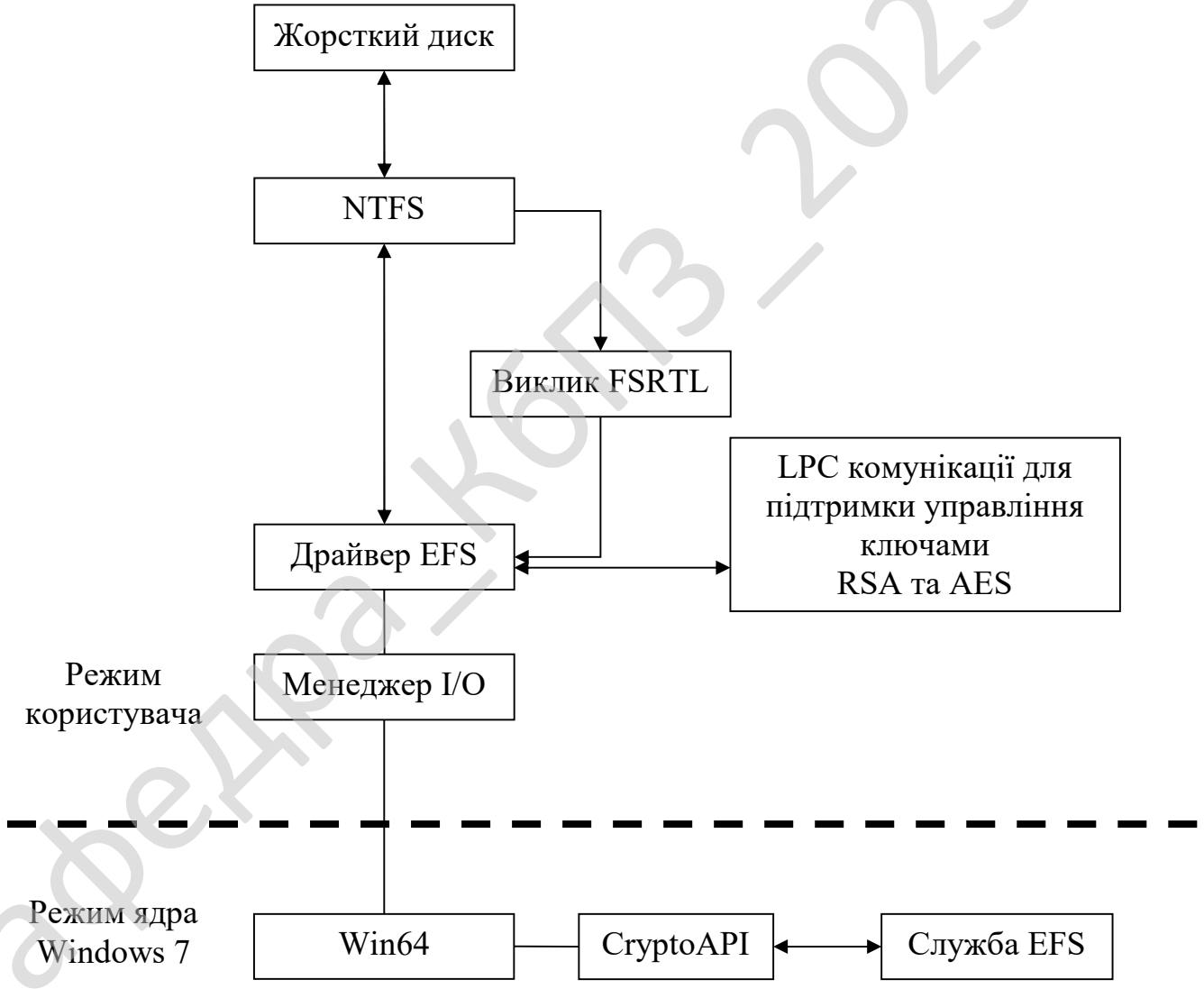


Рисунок 3.1 – Структурна схема EFS

**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 шифрування файлів і забезпечуючи генерацію DDF і DRF. Крім цього, служба EFS здійснює підтримку інтерфейсу Win32 API.

**Win32 API**

Забезпечує інтерфейс програмування для шифрування відкритих файлів, дешифрування й відновлення закритих файлів, прийому й передачі закритих файлів без їхньої попередньої розшифровки. Реалізований у вигляді стандартної системної бібліотеки advapi32.dll.

### 3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно. EFS використовує архітектуру Windows CryptoAPI. У її основі лежить технологія шифрування з відкритим ключем. Для шифрування кожного файлу випадковим образом генерується ключ шифрування файлу. При цьому для шифрування файлу може застосовуватися будь-який симетричний алгоритм шифрування. У сьогодення ж час в EFS використовується один алгоритм, це AES. Ключі шифрування EFS зберігаються в резидентному пулі пам'яті (сама EFS розташована в ядрі Windows 7), що виключає несанкціонований доступ до них через файл підкачування.

**Взаємодія з користувачем**

За замовчуванням EFS зконфігурована таким чином, що користувач може відразу почати використовувати шифрування файлів. Операція шифрування й зворотна підтримуються для файлів і каталогів. У тому випадку, якщо шифрується каталог, автоматично шифруються всі файли й підкаталоги цього каталогу. Необхідно відзначити, що якщо зашифрований файл переміщається або перейменовується із зашифрованого каталогу в незашифрований, то він однаково залишається зашифрованим. Операції шифрування/дешифрування можна виконати двома різними способами – використовуючи Windows Explorer або консольну утиліту Cipher.

Для того щоб зашифрувати каталог з Windows Explorer, користувачеві потрібно просто вибрати один або кілька каталогів і встановити прапорець шифрування у вікні розширених властивостей каталогу. Всі створювані пізніше файли й підкаталоги в цьому каталозі будуть також зашифровані. Таким чином, зашифрувати файл можна, просто скопіювавши (або перенісши) його в "зашифрований" каталог.

Зашифровані файли зберігаються на диску в зашифрованому виді. При читанні файлу дані автоматично розшифровуються, а при записі – автоматично

шифруються. Користувач може працювати із зашифрованими файлами так само, як і зі звичайними файлами, тобто відкривати й редагувати в текстовому редакторі Microsoft Word документи, редагувати рисунки в Adobe Photoshop або графічному редакторі Paint, і так далі.

Необхідно відзначити, що в жодному разі не можна шифрувати файли, які використовуються при запуску системи – у цей час особистий ключ користувача, за допомогою якого виробляється дешифрування, ще недоступний. Це може привести до неможливості запуску системи. В EFS передбачена простий захист від таких ситуацій: файли з атрибутом "системний" не шифруються. Однак будьте уважні: це може створити "діру" у системі безпеки! Перевіряйте, чи не встановлений атрибут файлу "системний" для того, щоб переконатися, що файл дійсно буде зашифрований.

Важливо також пам'ятати про те, що зашифровані файли не можуть бути стислі засобами Windows 7 і навпаки. Іншими словами, якщо каталог стислий, його вміст не може бути зашифровано, а якщо вміст каталогу зашифрований, то він не може бути стислий.

У тому випадку, якщо буде потрібно дешифрування даних, необхідно просто зняти прапорці шифрування в обраних каталогів в Windows Explorer, і файли й підкаталоги автоматично будуть дешифровані. Слід зазначити, що ця операція звичайно не потрібно, тому що EFS забезпечує "прозору" роботу із зашифрованими даними для користувача.

**Відновлення даних**

EFS забезпечує убудовану підтримку відновлення даних на той випадок, якщо буде потрібно їх розшифрувати, але, за якимись причинами, це не може бути виконано звичайним. За замовчуванням, EFS автоматично згенерує ключ відновлення, установить сертифікат доступу в обліковому записі адміністратора й збереже його при першому вході в систему. Таким чином, адміністратор стає так званим агентом відновлення, і зможе розшифрувати будь-який файл у системі. Зрозуміло, політикові відновлення даних можна змінити, і призначити як агент

відновлення спеціальної людини, відповідального за безпеку даних, або навіть кілька таких осіб.

EFS здійснює шифрування даних, використовуючи схему із загальним ключем. Дані шифруються швидким симетричним алгоритмом за допомогою ключа шифрування файлу FEK (file encryption key). FEK – це випадковим образом згенерований ключ певної довжини. Довжина ключа в північноамериканській версії EFS 128 біт, у міжнародній версії EFS використовується зменшена довжина ключа 40 або 56 біт.

FEK шифрується одним або декількома загальними ключами шифрування, у результаті чого виходить список зашифрованих ключів FEK. Список зашифрованих ключів FEK зберігається в спеціальному атрибуті EFS, що називається DDF (data decryption field – поле дешифрування даних). Інформація, за допомогою якої виробляється шифрування даних, жорстко пов'язана із цим файлом. Загальні ключі виділяються з пар користувальницьких ключів сертифіката X509 з додатковою можливістю використання "File encryption". Особисті ключі із цих пар використовуються при дешифруванні даних і FEK. Особиста частина ключів зберігається або на смарт-картах, або в іншому надійному місці (наприклад, у пам'яті, безпеку якої забезпечується за допомогою CryptoAPI).

FEK також шифрується за допомогою одного або декількох ключів відновлення (отриманих із сертифікатів X.509, записаних у політиці відновлення зашифрованих даних для даного комп'ютера, з додатковою можливістю "File recovery").

Як і в попередньому випадку, загальна частина ключа використовується для шифрування списку FEK. Список зашифрованих ключів FEK також зберігається разом з файлом у спеціальній області EFS, що називається DRF (data recovery field – поле відновлення даних). Для шифрування списку FEK в DRF використовується тільки загальна частина кожної пари ключів. Для нормального здійснення файлових операцій необхідні тільки загальні ключі відновлення.

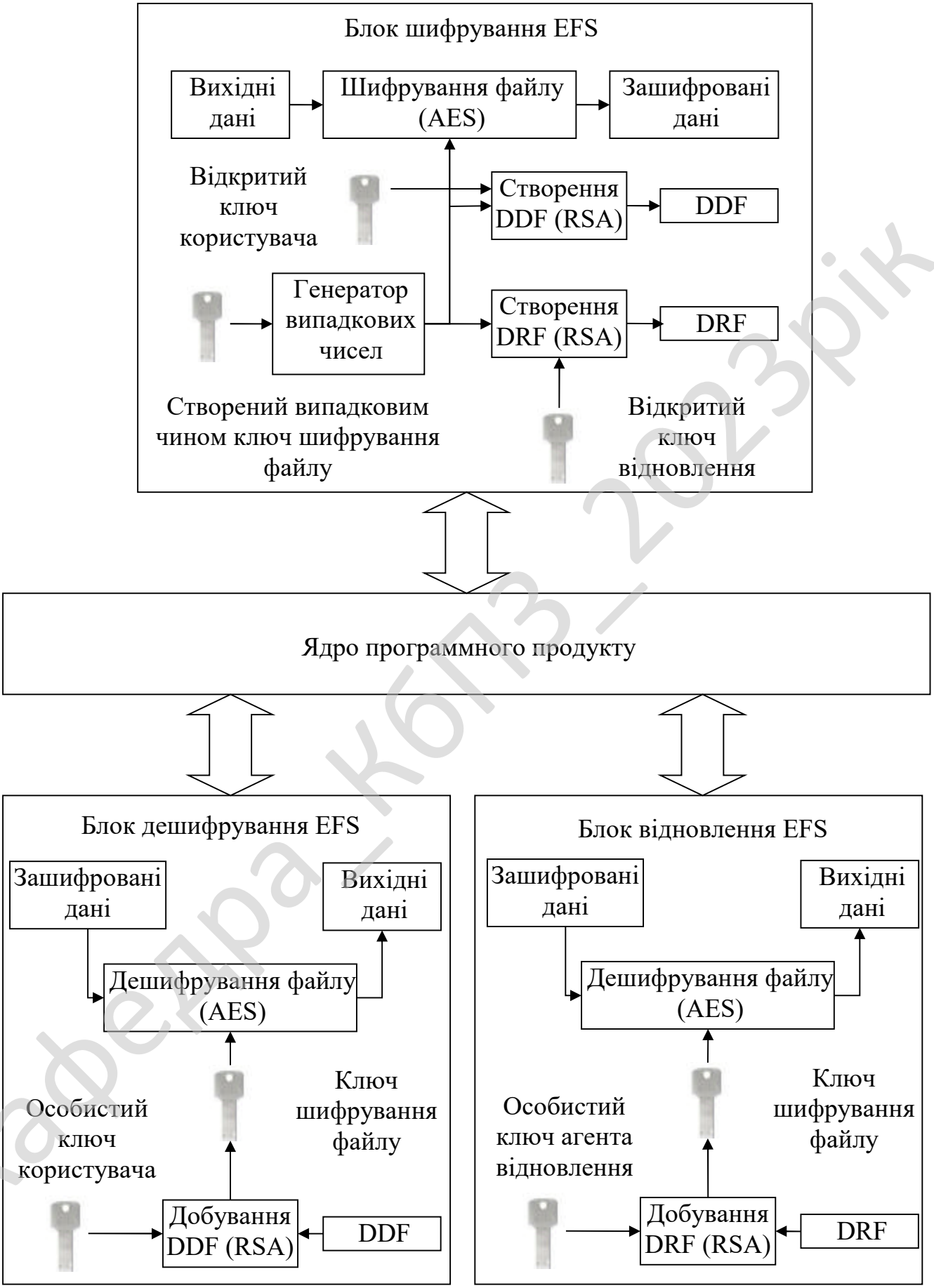


Рисунок 3.2 – Функціональна схема системи

Агенти відновлення можуть зберігати свої особисті ключі в безпечному місці поза системою (наприклад, на смарт-картах). На рисунку 3.2 наведені функціональні схеми процесів шифрування, дешифрування й відновлення даних.

**Процес шифрування**

Незашифрований файл користувача шифрується за допомогою випадково згенерованного ключа FEK. Цей ключ записується разом з файлом, файл дешифрується за допомогою загального ключа користувача (записаного в DDF), а також за допомогою загального ключа агента відновлення (записаного в DRF).

**Процес дешифрування**

Спочатку використовується особистий ключ користувача для дешифрації FEK – для цього використовується зашифрована версія FEK, що зберігається в DDF. Розшифрований FEK використовується для поблочного дешифрування файлу. Якщо у великому файлі блоки зчитуються не послідовно, то дешифруються тільки зчитувальні блоки. Файл при цьому залишається зашифрованим.

**Процес відновлення**

Цей процес аналогічний дешифруванню з тією різницею, що для дешифрування FEK використовується особистий ключ агента відновлення, а зашифрована версія FEK

Надомо опис алгоритмів шифрування, які використовуються у цих схемах. До них відносяться AES та RSA.

**RSA**

Алгоритм RSA являє собою блоковий алгоритм шифрування, де зашифруванні й незашифруванні дані є цілими між 0 і n-1 для деякого n.

Дані шифруються блоками, кожний блок розглядається як число, менше деякого числа n. Шифрування і дешифрування мають наступний вигляд для деякого незашифруванного блоку M та зашифруванного блоку C:

$$C = M^e \pmod n, \tag{3.1}$$

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

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 $\Phi(n)$: НСД $(\Phi(n), d) = 1$; $1 < d < \Phi(n)$.

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.

**Шифрування**

Незашифрований текст: $M < n$.

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

**Дешифрування**

Зашифрований текст: $C < n$.

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

**Обчислювальні аспекти**

Розглянемо складність обчислень в алгоритмі RSA при створенні ключів і при шифруванні/дешифруванні.

**Шифрування/дешифрування**

Як шифрування, так і дешифрування включають піднесення цілого числа в цілий ступінь по модулю n. При цьому проміжні значення будуть величезними.

Для того, щоб частково цього уникнути (тобто, щоб проміжні результати не виходили за границі n), використовується наступна властивість модульної арифметики:

$$(a \cdot b) \bmod n = (a \bmod n) \cdot (b \bmod n). \quad (3.3)$$

Також алгоритм можна оптимізувати за допомогою ефективного використання показника ступеня, тому що у випадку RSA показники ступеня дуже великі. Припустимо, що необхідно обчислити $x^{16}$. Прямий підхід вимагає 15 множень. Однак можна домогтися того ж кінцевого результату за допомогою тільки чотирьох множень, якщо використовувати квадрат кожного проміжного результату: $x^2$, $x^4$, $x^8$, $x^{16}$.

**Створення ключів**

Створення ключів включає наступні задачі:

1. Визначити два простих числа p та q.

2. Вибрати e й обчислити d.

Насамперед, розглянемо проблеми, пов'язані з вибором p та q. Тому що значення $n = p \cdot q$ буде відомо будь-якому потенційному зловмисникові, для запобігання розкриття p та q ці прості числа повинні бути обрані з досить великої множини, тобто p та q повинні бути великими числами. З іншого боку, метод, що використовується для пошуку великого простого числа, повинен бути досить ефективним. У наш час не відомі алгоритми, які створюють довільно великі прості числа. Процедура, що використовується для цього, вибирає випадкове непарне число з необхідного діапазону й перевіряє, чи є воно простим. Якщо число не є простим, то знову вибирається випадкове число доти, поки не буде знайдено просте.

Були розроблені різні тести для визначення того, чи є число простим. Це тести імовірнісні, тобто тест показує, що дане число ймовірно є простим. Незважаючи на це, вони можуть виконуватися таким чином, що зроблять імовірність близькою до 1. Якщо *n* "провалює" тест, то воно не є простим. Якщо *n* "пропускає" тест, то *n* може як бути, так і не бути простим. Якщо *n* пропускає

багато таких тестів, то можна з високим ступенем вірогідності сказати, що *n* є простим. Це досить довга процедура, але вона виконується відносно рідко: тільки при створенні нової пари (KU, KR).

**AES**

Для захисту розробленого програмного забезпечення запропоновано використати фіналіста конкурсу AES – шифр Rijndael. 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 шифрування 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).

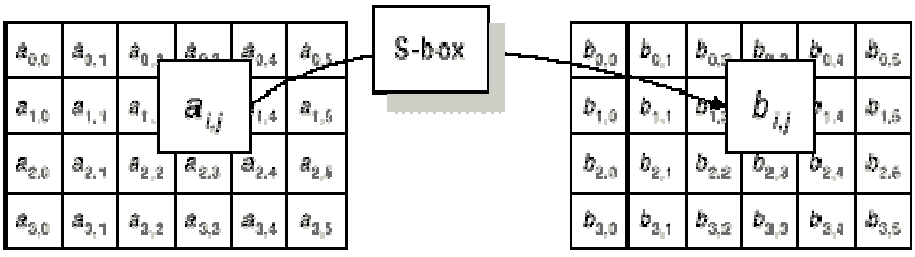


Рисунок 3.3 – Таблична підстановка 8x8 біт

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

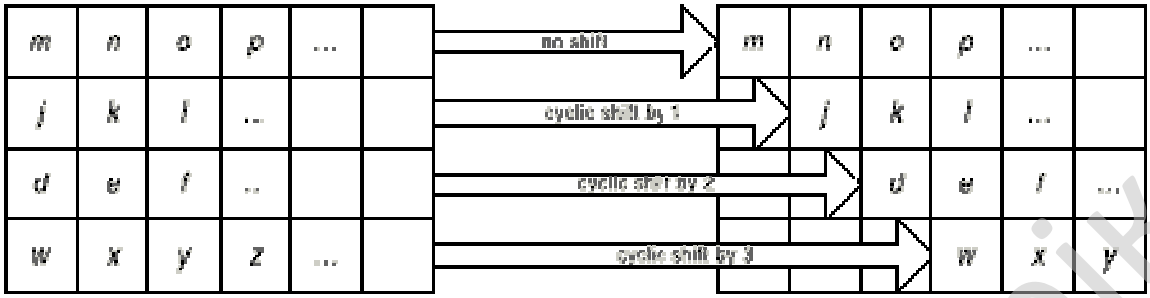


Рисунок 3.4 – Зрушення рядків у двовимірному масиві на різні зсуви

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

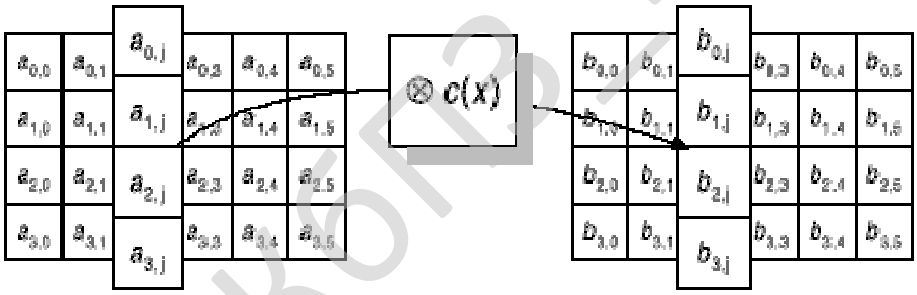


Рисунок 3.5 – Математичне перетворення, що перемішує дані усередині стовпця

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

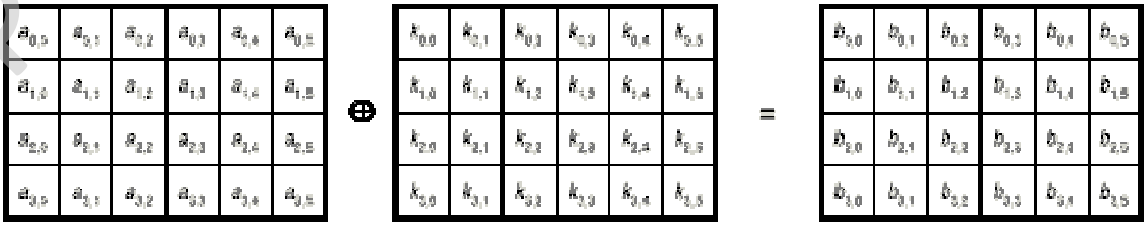

Рисунок 3.6 – Додавання матеріалу ключа операцією XOR

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 в системі є процес виведення головного вікна програми. Цей процес взаємодіє з наступними процесами:

– Процес створення ключів.

– Процес вибору диску.

Процес створення ключі взаємодіє з наступними процесами:

– Процес збереження ключів.

– Процес створення ключів AES.

– Процес створення відкритого та закритого ключів RSA.

Процес вибору дисків взаємодіє з процесом виведення списку файлів та папок.

Останній процес взаємодіє з наступними процесами:

– Процес копіювання, переміщення, створення та перейменування файлів та папок.

– Процес вибору файлів.

Процес вибору файлів взаємодіє, у свою чергу, з наступними процесами:

– Процес перегляду інформації про файл.

– Процес перегляду файлів.

– Процес шифрування файлів.

– Процес дешифрування файлів.

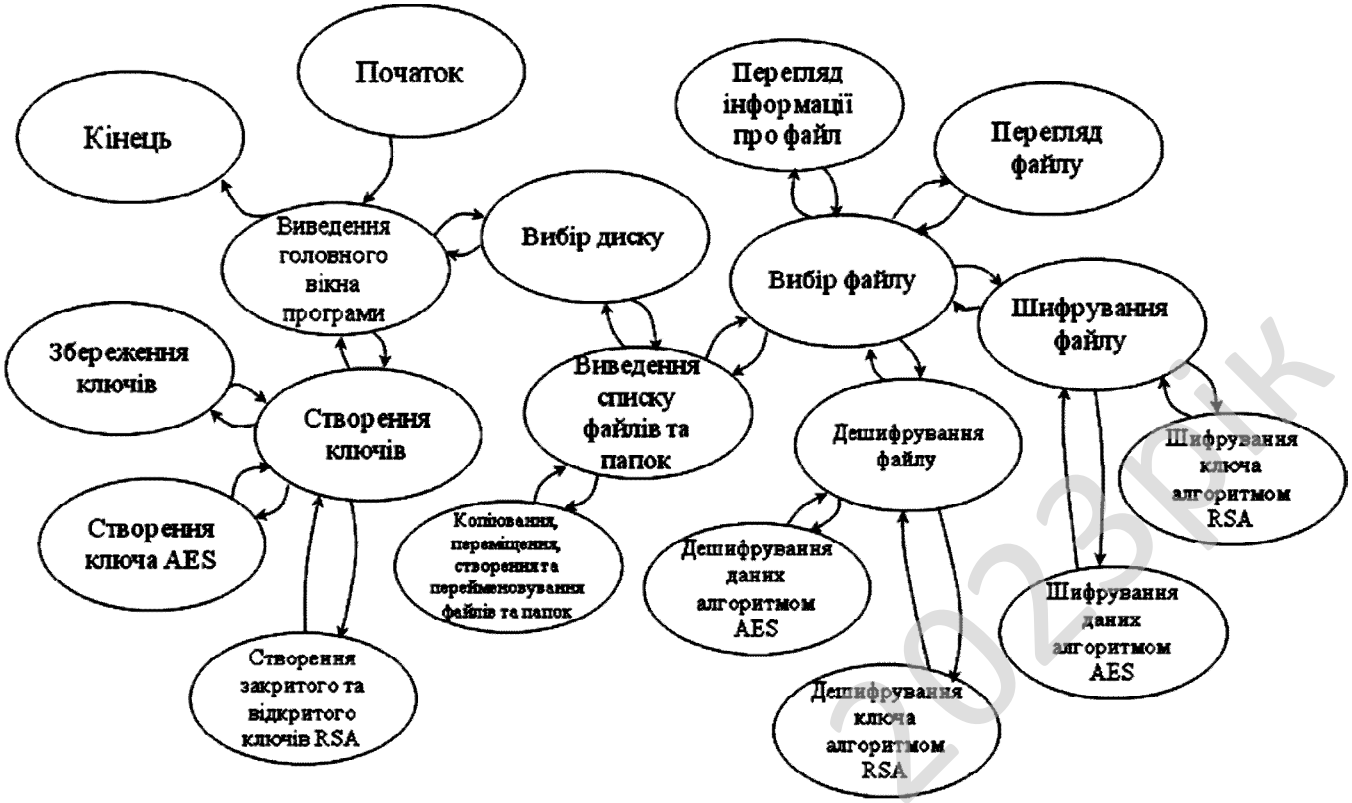


Рисунок 3.7 – Діаграма взаємодії процесів

Процес шифрування файлів взаємодіє з наступними процесами:

– Процес шифрування ключа алгоритмом RSA.

– Процес шифрування даних алгоритмом AES.

Процес дешифрування файлів взаємодіє з наступними процесами:

– Процес дешифрування ключа алгоритмом RSA.

– Процес дешифрування даних алгоритмом AES.

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

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

## 4.1 Блок-схеми та опис алгоритмів функціонування системи

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

Спершу відбувається виведення основного вікна програми. Після цього відбувається виведення дисків.

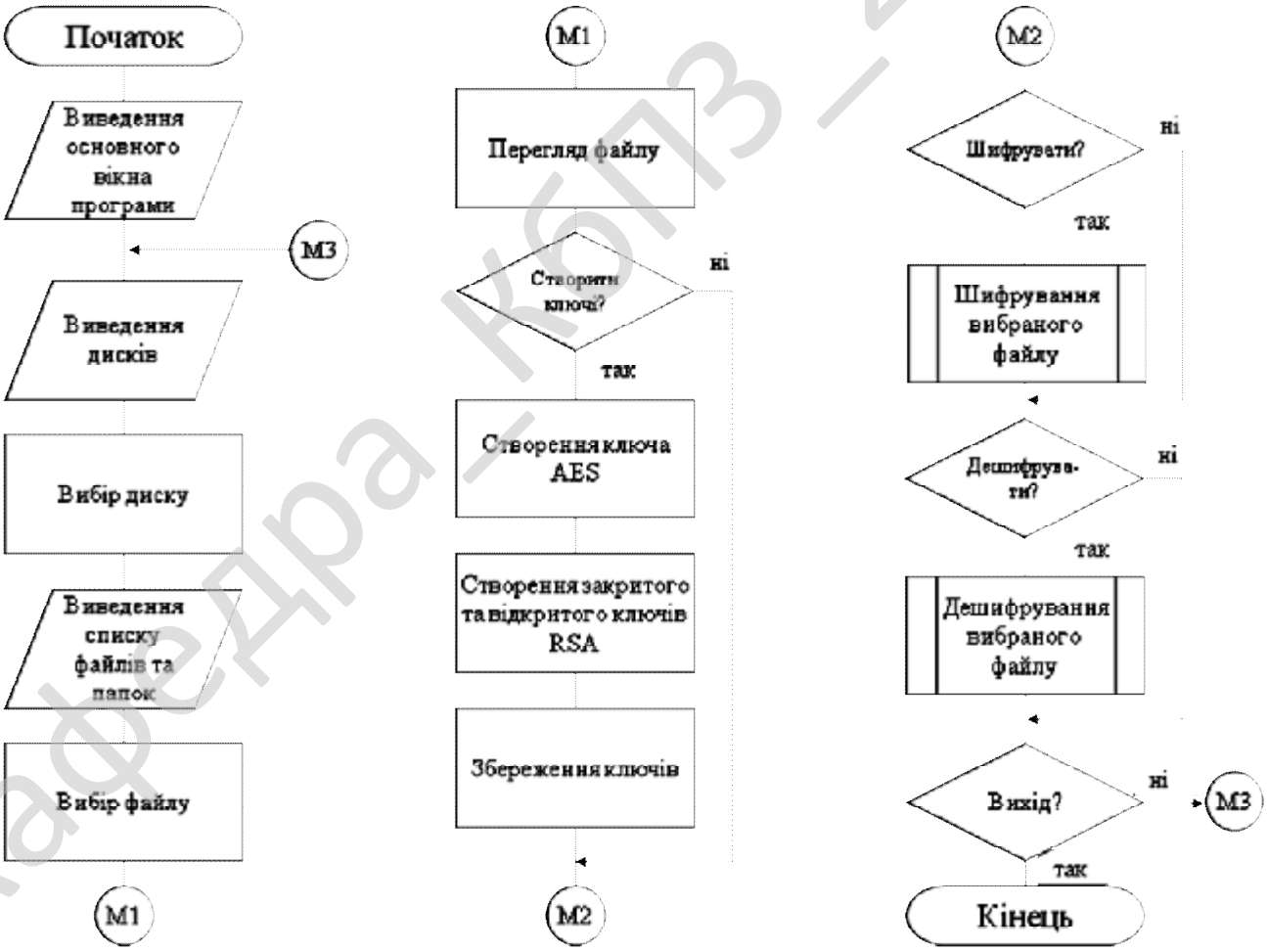


Рисунок 4.1 – Блок-схема роботи основної програми

Наступним кроком є обирання користувачем диску.

На екран виводиться список файлів та папок.

Наступним кроком є обирання файлу, який необхідно зашифрувати.

Після цього відбувається перегляд файлу, який потрібно зашифрувати.

Якщо необхідно створити ключ, то відбувається виконання наступних дій:

– Створення ключа AES.

– Створення відкритого та закритого ключів RSA.

– Збереження створених ключів.

Якщо необхідно шифрувати, то викликається підпрограма шифрування файлу алгоритмом AES.

Якщо необхідно дешифрувати, то викликається підпрограма дешифрування обраного файлу алгоритмом AES.

На рисунку 4.2 зображено підпрограму шифрування файлу алгоритмом AES.

Він працює наступним чином:

Спершу відбувається розбиття даних для шифрування на блоки.

Потім розраховуються раундові ключі з ключа шифрування AES.

Після цього ініціалізуються S-boxs.

Наступним кроком є вибір раундового ключа.

Раундів ключ трансформується.

Поки не оброблені усі блоки відбувається виконання наступних ітерацій:

– Підстановка байтів.

– Циклічний зсув рядків.

– Змішування даних у стовбцях.

– Вибір раундового ключа.

– Трансформація раундового ключа.

Останнім етапом є об’єднання блоків та формування зашифрованих даних.

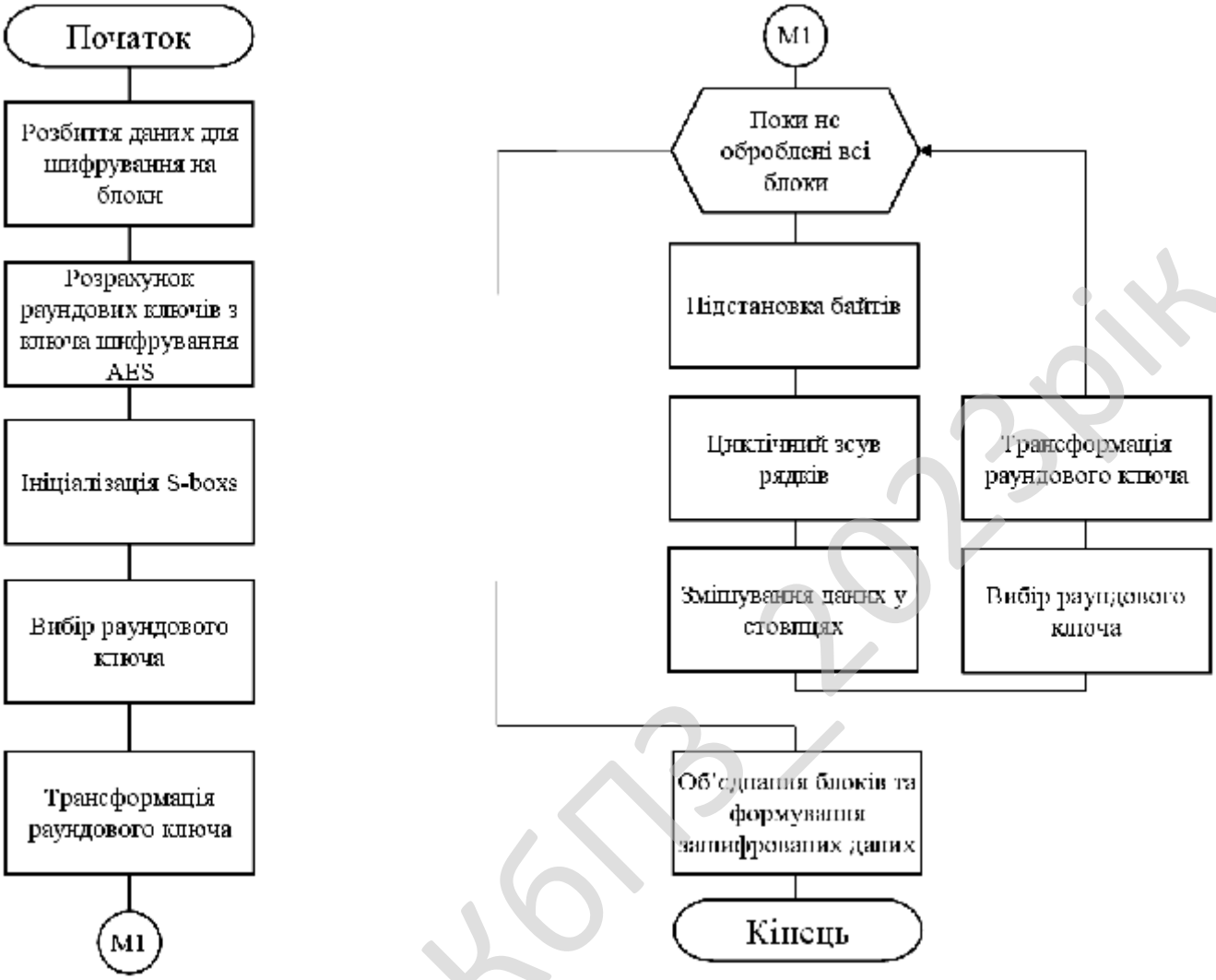


Рисунок 4.2 – Блок-схема роботи підпрограми шифрування алгоритмом AES

На рисунку 4.3 зображено підпрограму дешифрування файлу алгоритмом AES.

Він працює наступним чином:

Спершу відбувається розбиття даних для розшифрування на блоки.

Потім розраховуються раундові ключі з ключа шифрування AES.

Після цього ініціалізуються S-boxs.

Наступним кроком є вибір раундового ключа.

Раундів ключ трансформується.

Поки не оброблені усі блоки відбувається виконання наступних ітерацій:

– Зворотна підстановка байтів.

– Зворотний циклічний зсув рядків.

– Зворотне змішування даних у стовбцях.

– Вибір раундового ключа.

– Трансформація раундового ключа.

Останнім етапом є об’єднання блоків та формування розшифрованих даних.

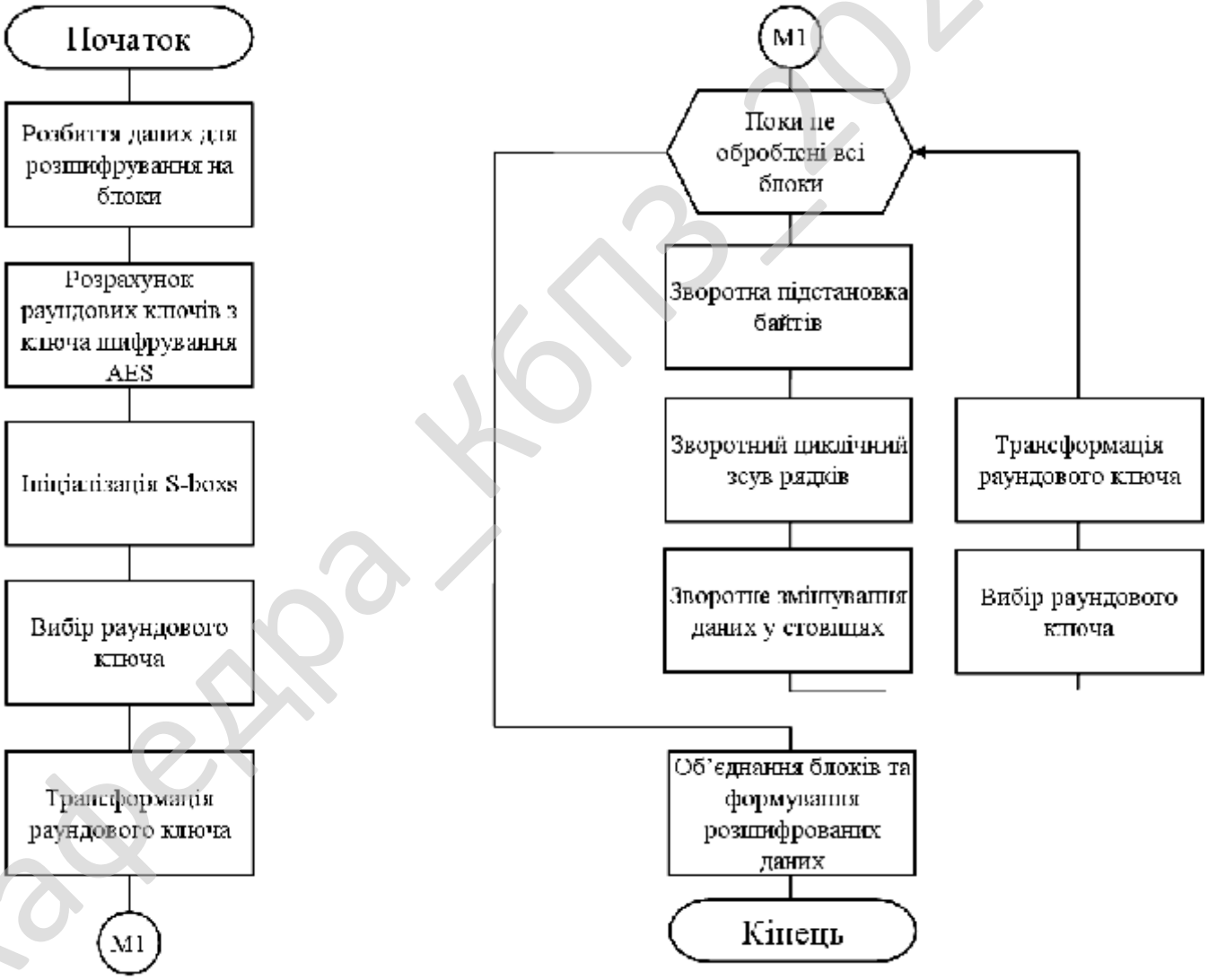


Рисунок 4.3 – Блок-схема роботи підпрограми дешифрування алгоритмом AES

Наведемо частину програмного коду генерації ключів RSA та шифрування ключа AES.

```
//Генерація ключів RSA
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;
   FGIntAEStroy(two);
   FGIntAEStroy(one);
   FGIntAEStroy(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;
```

```
    FGIntAEStroy(phi);
    FGIntAEStroy(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;
```

### Алгоритм шифрування RSA.

```
// 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);
      FGIntAEStroy(PGInt);
      tempstr3 := '';
      FGIntToBase2String(temp, tempstr3);
      While (length(tempstr3) Mod modbits) <> 0 Do tempstr3 := '0' + tempstr3;
```

```
      tempstr2 := tempstr2 + tempstr3;
      FGIntAEStroy(temp);
   End;
   While (tempstr2[1] = '0') And (length(tempstr2) > 1) Do delete(tempstr2, 1, 1);
   ConvertBase2To256(tempstr2, E);
   FGIntAEStroy(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);
      FGIntAEStroy(temp1);
      FGIntModInv(p, q, temp1);
      FGIntMul(p, temp1, ppinvq);
      FGIntAEStroy(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);
            FGIntAEStroy(temp1);
            FGIntAEStroy(temp2);
         End;
      End;
      FGIntAEStroy(EGInt);
      tempstr3 := '';
      FGIntToBase2String(temp, tempstr3);
      While (length(tempstr3) Mod (modbits - 1)) <> 0 Do tempstr3 := '0' + tempstr3;
      tempstr2 := tempstr2 + tempstr3;
      FGIntAEStroy(temp);
   End;
   If exp.Number = Nil Then
   Begin
      FGIntAEStroy(ppinvq);
      FGIntAEStroy(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);
   FGIntAEStroy(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);
      FGIntAEStroy(temp);
      FGIntModInv(q, p, temp);
      FGIntMul(q, temp, qqinvp);
      FGIntAEStroy(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);
      FGIntAEStroy(temp1);
      FGIntAEStroy(temp2);
      FGIntAEStroy(ppinvq);
      FGIntAEStroy(qqinvp);
   End;
   FGIntToBase256String(SGInt, S);
   FGIntAEStroy(MGInt);
   FGIntAEStroy(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);
   FGIntAEStroy(SGInt);
   FGIntAEStroy(MGInt);
End;
```

**У результаті виконання бакалаврського проектування можна зробити наступні висновки:**

– Розроблена система EFS надає користувачам можливість зашифровувати каталоги NTFS, використовуючи стійку, засновану на загальних ключах криптографічну схему, при цьому всі файли в закритих каталогах будуть зашифруванні. Шифрування окремих файлів підтримується, але не рекомендується через непередбачене поводження додатків.

– Розроблене програмне забезпечення системи EFS також підтримує шифрування віддалених файлів, доступ до яких здійснюється як до спільно використовуваних ресурсів. Якщо мають місце користувальницькі профілі для підключення, використовуються ключі й сертифікати віддалених профілів. В інших випадках генеруються локальні профілі й використовуються локальні ключі.

– Система EFS надає можливість встановити політику відновлення даних таким чином, що зашифруванні дані можуть бути відновлені за допомогою EFS, якщо це буде потрібно.

– Політика відновлення даних вбудована в загальну політику безпеки Windows. Контроль за дотриманням політики відновлення може бути делегований уповноваженим на це особам. Для кожного підрозділу організації може бути зконфігурована своя політика відновлення даних.

– Відновлення даних в EFS – закрита операція. У процесі відновлення розшифровуються дані, але не ключ користувача, за допомогою якого ці дані були зашифруванні.

– Робота із зашифруванними файлами в EFS не жадає від користувача яких-небудь спеціальних дій по шифруванню й дешифруванню даних. Дешифрування й шифрування відбуваються непомітно для користувача в процесі зчитування й запису даних на диск.

– Система EFS підтримує резервне копіювання й відновлення зашифруванних файлів без їхньої розшифровки.

– Система EFS вбудована в операційну систему таким чином, що витік інформації через файли підкачування неможливий, при цьому гарантується, що всі створювані копії будуть зашифруванні

– Передбачено численні запобіжні заходи для забезпечення безпеки відновлення даних, а також захист від витоку й втрати даних у випадку фатальних збоїв системи.

### 4.2 Захист розробленого програмного забезпечення

Tiny Encryption Algorithm (TEA) [1] – блочний алгоритм шифрування типу «Мережі Фейстеля». Алгоритм був розроблений на факультеті комп'ютерних наук Кембриджського університету Девідом Вілером (David Wheeler) і Роджером Нідгемом (Roger Needham) та вперше представлений в 1994 році [2] на симпозіумі зі швидкими алгоритмами шифрування в Льовені (Бельгія).

Шифр не патентований, широко використовується в ряді криптографічних додатків і широкому спектрі апаратного забезпечення, завдяки вкрай низькими вимогами до пам'яті й простоті реалізації. Алгоритм має як програмну реалізацію на різних мовах програмування, так і апаратну реалізацію на інтегральних схемах типу FPGA.

Захист розробленого програмного забезпечення буде відбуватися за допомогою алгоритму TEA, який заснований на бітових операцій з 64-бітним блоком, має 128-бітний ключ шифрування. Стандартна кількість раундів мережі Фейстеля біля 64 (32 циклу), однак, для досягнення найкращої продуктивності або шифрування, число циклів можна варіювати від 8 (16 раундів) до 64 (128 раундів). Мережа Фейстеля несиметрична через використання в якості операції накладення додавання по модулю 232.

Перевагами шифру є його простота в реалізації, невеликий розмір коду й досить висока швидкість виконання, а також можливість оптимізації виконання на стандартних 32-бітних процесорах, так як в якості основних операцій

використовуються операції виключна «АБО» (XOR), побітового зсуву й додавання по модулю 232. Оскільки алгоритм не використовує таблиць підстановки і раундова функція досить проста, алгоритму потрібно не менше 16 циклів (32 раундів) для досягнення ефективної дифузії, хоча повна дифузія досягається вже через 6 циклів (12 раундів).

Алгоритм має відмінну стійкість до лінійного криптоаналізу і досить гарну до диференціального криптоаналізу. Головним недоліком цього алгоритму шифрування є його вразливість до атак «на пов'язаних ключах» (англ. Related-key attack). Через простий розклад ключів кожен ключ має 3 еквівалентних ключа. Це означає, що ефективна довжина ключа складає всього 126 біт [3] [4], тому даний алгоритм не слід використовувати в якості геш-функції.

**Опис алгоритму**

Вихідний текст розбивається на блоки по 64 біта кожен. 128-бітний ключ K ділиться на чотири 32-бітних підключа K [0], K [1], K [2] і K [3]. На цьому підготовчий процес закінчується, після чого кожен 64-бітний блок шифрується протягом 32 циклів (64 раундів) за нижченаведеним алгоритмом. [5]

Припустимо, що на вхід n-го раунду ($1 \le n \le 64$) надходять права й ліва частини ($L_n$, $R_n$), тоді на виході n-го раунду будуть ліва й права частини ($L_{n+1}$, $R_{n+1}$), які обчислюються за такими правилами:

$$L_{n+1} = R_n.$$

Якщо $n = 2 * i - 1$ для $1 \le i \le 32$ (непарні раунди), то:

$$R_{n+1} = L_n (\{ [ R_n 4 ] K [0] \} \{ R_n i * \delta \} \{ [ R_n 5 ] K [1] \}).$$

Якщо $n = 2 * i$ для $1 \le i \le 32$ (парні раунди), то:

$$R_{n+1} = L_n (\{ [ R_n 4 ] K [2] \} \{ R_n i * \delta \} \{ [ R_n 5 ] K [3] \}).$$

Де

X Y – операція додавання чисел X і Y по модулю $2^{32}$.

X Y – побітове виключне АБО» (XOR) чисел X і Y, яке в мові програмування Сі позначається як X ^ Y

X Y і X Y – операції побітового зсуву числа X на Y біт вліво й вправо відповідно.

Константа δ була виведена з Золотого перерізу:

$$\delta = (-1) * 2^{31} = 2654435769 = 9E3779B9_h.$$

У кожному раунді константа множиться на номер циклу i. Це було зроблено для запобігання простих атак, заснованих на симетрії раундів.

Також очевидно, що в алгоритмі шифрування TEA немає як такого алгоритму розкладу ключів. Замість цього в непарних раундах використовуються підключі K [0] та [1], у парних – K [2] і [3].

Так як це блочний шифроалгоритм, де довжина блоку 64-біт, а довжина даних може бути не кратна 64-біт, значення всіх байтів, які доповнюють блок до кратності в 64-біт, встановлюється в 0x01.

# 5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено головне вікно програми. З нього ми бачимо, що інтерфейс користувача складається з наступних блоків:

– Блок випадаючих меню.

– Блок вибору папок.

– Блок вибору файлів у обраній папці.

– Клавіші швидкого доступу до функцій програми: додати пароль (ключ AES), згенерувати ключі RSA, шифрувати обраний файл, дешифрувати обраний файл.

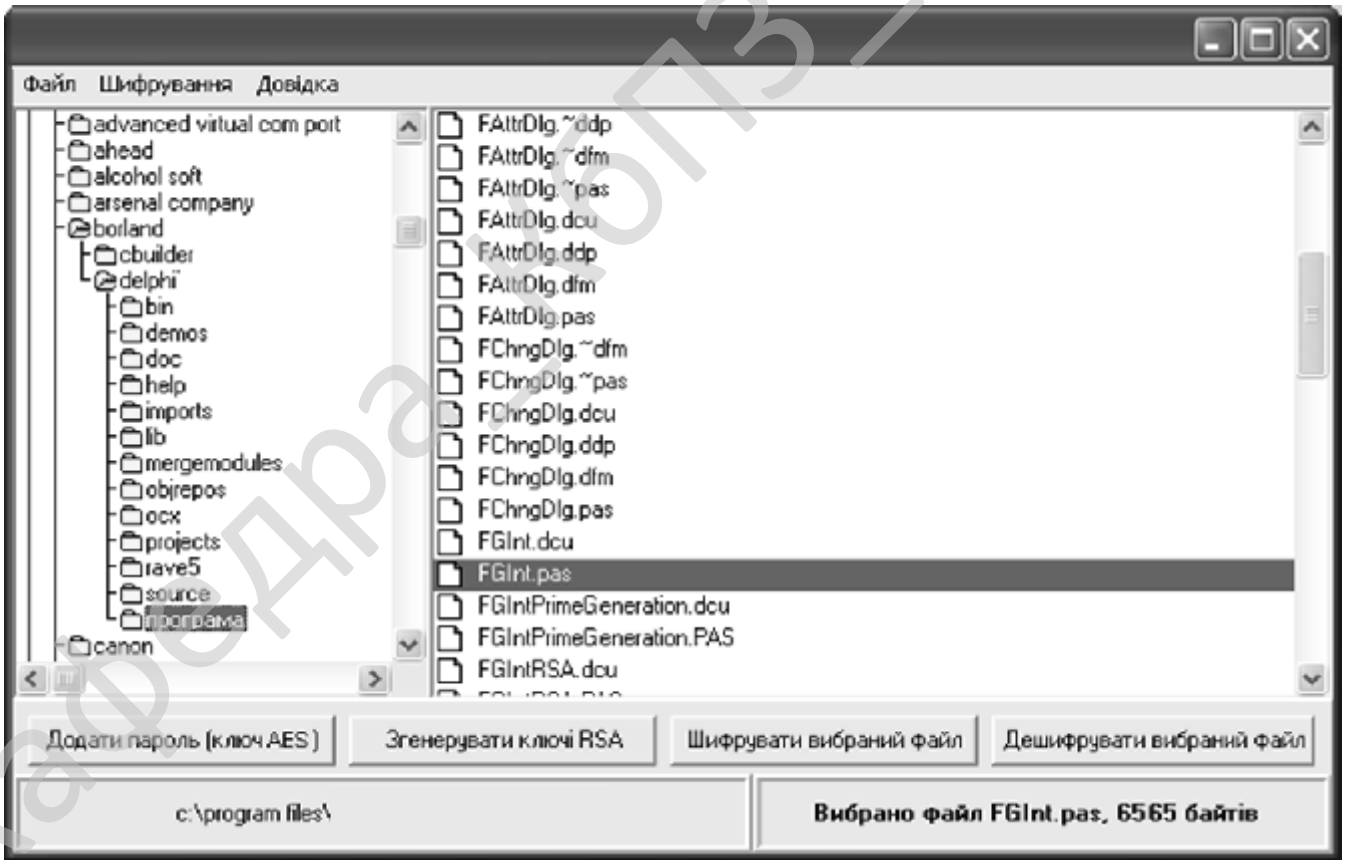


Рисунок 5.1 – Головне вікно програми

На рисунку 5.2 зображено вікно створення ключа AES.

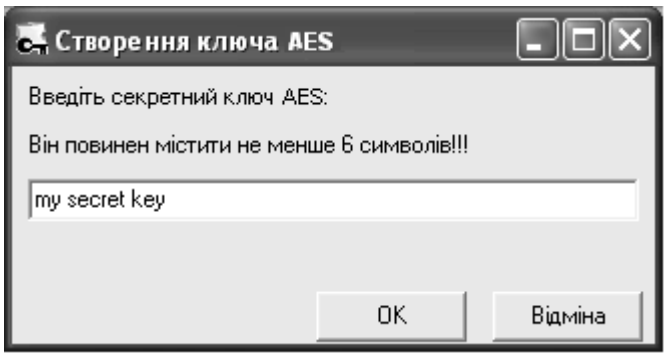


Рисунок 5.2 – Створення ключа AES

На рисунку 5.3 зображено вікно генерації ключів RSA.

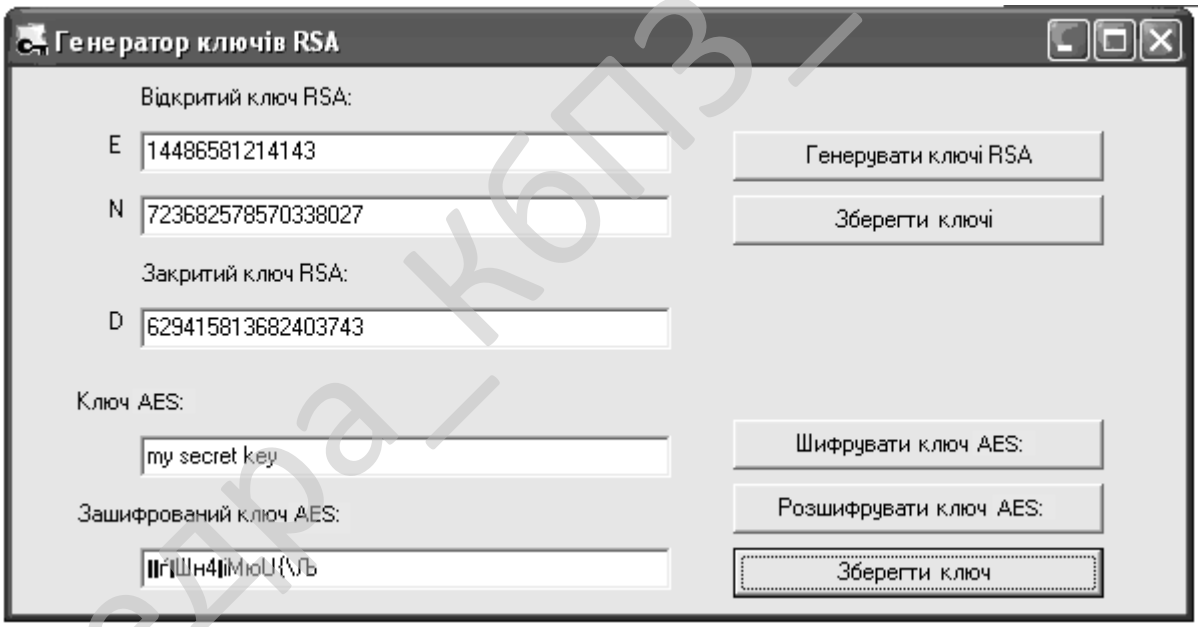


Рисунок 5.3 – Генерація ключів RSA

На рисунку 5.4 зображено вікно довідки, на якому зображено інформацію про розробника бакалаврського проекту, керівника бакалаврського проекту та місце розробки програмного забезпечення.

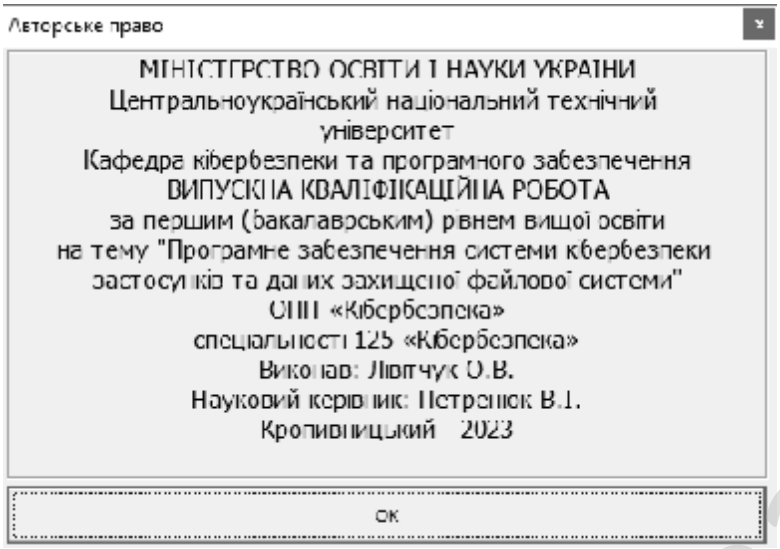


Рисунок 5.4 – Довідка

# 6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки застосунків та даних захищеної файлової системи.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем застосунків та даних захищеної файлової системи.

– Досліджена система застосунків та даних захищеної файлової системи.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки застосунків та даних захищеної файлової системи.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання застосунків та даних захищеної файлової системи.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об’єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Embarcadero Delphi. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи кібербезпеки застосунків та даних захищеної файлової системи. 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 TEA.

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

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» *CEUR Workshop Proceedings,* Volume 3187, 2022, pp. 1-12. **(Scopus).**

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

21. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407. (**Scopus)**.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

49. Смирнов А.А., Лысенко И.А., Информационная технология проектирования тестовых наборов на основе требований к программному обеспечению, Системи управління, навігації та зв'язку. – Випуск 4 (44). – Полтава: ПолтНТУ. – 2017. – С. 112-115.

50. Смірнов О.А., Мелешко Є.В., Хох В.Д., Дослідження методів аудиту систем управління інформаційною безпекою, Системи управління, навігації та зв'язку. – Випуск 1 (41). – Полтава: ПолтНТУ. – 2017. – С. 38-42.

Додаток А
(обов'язковий)

**Технічне завдання**

**Зміст**

## 1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки застосунків та даних захищеної файлової системи.

## 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. № 17-02 від 5.01.2023 року).

## 3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи кібербезпеки застосунків та даних захищеної файлової системи.

## 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 застосунків та даних захищеної файлової системи;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

### 5.3 Вимоги до функціональних характеристик

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

### 5.4 Вимоги до архітектури

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

### 5.5 Вимоги до надійності

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

### 5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

### 5.7 Вимоги до складу та параметрів технічних засобів

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

### 5.8 Вимоги до інформаційної і програмної сумісності

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

#### 5.8.1 Обладнання

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

#### 5.8.2 Мова програмування

Середовище Embarcadero 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 – 74 аркуша.

## 8 Етапи розробки

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

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

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

## 11 Порядок контролю та приймання

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

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

Додаток Б

(обов’язковий)

**Міністерство освіти і науки України**

**Центральноукраїнський національний технічний університет**

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Петренюк В.І.

***Програмне забезпечення системи кібербезпеки застосунків та даних захищеної файлової системи***

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 59

Літера: РП

Кропивницький – 2023 року

**Файл filmanex.dpr - основна програма**

```
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},// форма даних про автора
  FGInt in 'FGInt.pas',// бібліотека роботи з великими числами
  FGIntPrimeGeneration in 'FGIntPrimeGeneration.PAS',//формування великих чисел
  FGIntRSA in 'FGIntRSA.PAS',
  RSA_keys in 'RSA_keys.pas' {RSA_keys},//формування ключів RSA
  AES_key in 'AES_key.pas' {AES};

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TFMForm, FMForm);//створення головної форми
  Application.CreateForm(TFileAttrForm, FileAttrForm);// створення форми
атрибутів файлів
  Application.CreateForm(TChangeDlg, ChangeDlg);//створення форми зміни
параметрів
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(TRSA_keys, RSA_keys);// створення форми праці з RSA
ключами
  Application.CreateForm(TAES, AES);// створення форми праці з алгоритмом AES
  Application.Run; // форма запуску програми
end.
```

**Файл FmxUtils.pas – робота з файлами**

```
unit FmxUtils;

interface

uses SysUtils, Windows, Classes, Consts;

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

procedure CopyFile(const FileName, AEStName: string);
procedure MoveFile(const FileName, AEStName: 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
  SInvalidAESt = 'Вказаний каталог %s не існує';
  SFCantMove = 'Не можу перемістити файл %s';

//Копіювання файлу

procedure CopyFile(const FileName, AEStName: string);
var
  CopyBuffer: Pointer; { буфер для копіювання }
  BytesCopied: Longint;
  Source, AESt: Integer; { заголовки }
  Len: Integer;
  AEStination: TFileName; { розширене місце розташування }
const
  ChunkSize: Longint = 8192; { копіювання в 8К блок }
begin
  AEStination := ExpandFileName(AEStName); { розширення шляху призначення }
  if HasAttr(AEStination, faDirectory) then { якщо розширення є директорією... }
  begin
    Len :=  Length(AEStination);
    if AEStination[Len] = '\' then
      AEStination := AEStination + ExtractFileName(FileName) { ...clone file
name }
    else
      AEStination := AEStination + '\' + 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
      AESt := FileCreate(AEStination); { створюємо вихідний файл або
перезаписуємо }
      if AESt < 0 then raise EFCreateError.CreateFmt(SFCreateError,
[AEStination]);
      try
        repeat
          BytesCopied := FileRead(Source, CopyBuffer^, ChunkSize); { читаємо
блок }
          if BytesCopied > 0 then { якщо ми прочитали... }
            FileWrite(AESt, CopyBuffer^, BytesCopied); { ...записуємо блок }
        until BytesCopied < ChunkSize; { працюємо поки не закінчуються блоки }
```

```
      finally
        FileClose(AESt); { 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, AEStName: string);
var
  AEStination: string;
begin
  AEStination := ExpandFileName(AEStName); { розширюємо шлях призначення  }
  if not RenameFile(FileName, AEStination) then { переіменовуємо }
  begin
    if HasAttr(FileName, faReadOnly) then  { якщо тільки для читання... }
      raise EFCantMove.Create(Format(SFCantMove, [FileName])); { не в змозі
знищити }
      CopyFile(FileName, AEStination); { 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 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 (csAESigning 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('Видалити ' + 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,AESEncode(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,AESDecode(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
  TAES = class(TForm)
    EditAES: TEdit;
    Button1: TButton;
    Button2: TButton;
    Label1: TLabel;
    Label2: TLabel;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  AES: TAES;

implementation

{$R *.dfm}

procedure TAES.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}

//Генерація ключів RSA

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;
   FGIntAEStroy(two);
   FGIntAEStroy(one);
   FGIntAEStroy(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;
   FGIntAEStroy(phi);
   FGIntAEStroy(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.EditAES.Text;
end;

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 AESt:TBitString; Source:TBitString; NBits:Integer);
Function ConcatBits(Bits:Array Of TBitString):TBitString;

Function AESEncode(S,Key:String):TBitString;
Function AESDecode(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 AESt:TBitString; Source:TBitString; NBits:Integer);
Var
I:Integer;
Begin
SetLength(AESt,NBits);
For I:=0 To NBits-1 Do
  AESt[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 AESEncode(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 AESDecode(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 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);
      FGIntAEStroy(PGInt);
      tempstr3 := '';
      FGIntToBase2String(temp, tempstr3);
      While (length(tempstr3) Mod modbits) <> 0 Do tempstr3 := '0' + tempstr3;
      tempstr2 := tempstr2 + tempstr3;
      FGIntAEStroy(temp);
   End;

   While (tempstr2[1] = '0') And (length(tempstr2) > 1) Do delete(tempstr2, 1,
1);
   ConvertBase2To256(tempstr2, E);
   FGIntAEStroy(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);
      FGIntAEStroy(temp1);
      FGIntModInv(p, q, temp1);
      FGIntMul(p, temp1, ppinvq);
      FGIntAEStroy(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);
            FGIntAEStroy(temp1);
            FGIntAEStroy(temp2);
         End;
      End;
      FGIntAEStroy(EGInt);
      tempstr3 := '';
      FGIntToBase2String(temp, tempstr3);
      While (length(tempstr3) Mod (modbits - 1)) <> 0 Do tempstr3 := '0' +
tempstr3;
      tempstr2 := tempstr2 + tempstr3;
      FGIntAEStroy(temp);
   End;

   If exp.Number = Nil Then
   Begin
      FGIntAEStroy(ppinvq);
      FGIntAEStroy(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);
   FGIntAEStroy(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);
      FGIntAEStroy(temp);
      FGIntModInv(q, p, temp);
      FGIntMul(q, temp, qqinvp);
      FGIntAEStroy(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);
      FGIntAEStroy(temp1);
      FGIntAEStroy(temp2);
      FGIntAEStroy(ppinvq);
      FGIntAEStroy(qqinvp);
   End;
   FGIntToBase256String(SGInt, S);
   FGIntAEStroy(MGInt);
   FGIntAEStroy(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);
   FGIntAEStroy(SGInt);
   FGIntAEStroy(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 FGIntAEStroy(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;
   FGIntAEStroy(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 FGIntAEStroy(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;

   FGIntAEStroy(one);
   FGIntAEStroy(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;

   FGIntAEStroy(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);
         FGIntAEStroy(temp1);
         FGIntsub(FGInt2, MFGInt, temp1);
         FGIntCopy(temp1, MFGInt);
      End;
      If s2 = positive Then QFGInt.Sign := negative;
   End
   Else QFGInt.Sign := s2;
   FGIntAEStroy(one);
```

```
   FGIntAEStroy(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;

   FGIntAEStroy(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);
         FGIntAEStroy(temp1);
         FGIntsub(FGInt2, MFGInt, temp1);
         FGIntCopy(temp1, MFGInt);
      End;
      If s2 = positive Then QFGInt.Sign := negative;
   End
   Else QFGInt.Sign := s2;
   FGIntAEStroy(one);
   FGIntAEStroy(zero);
   FGIntAEStroy(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;

   FGIntAEStroy(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;
   FGIntAEStroy(one);
   FGIntAEStroy(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);
   FGIntAEStroy(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);
   FGIntAEStroy(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);
   FGIntAEStroy(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;
   FGIntAEStroy(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);
   FGIntAEStroy(temp);
   FGIntAdd(temp1, GInt, temp);
   FGIntAEStroy(temp1);
   MGInt.Number := copy(temp.Number, b - 1, temp.Number[0] - b + 2);
   MGInt.Sign := positive;
   MGInt.Number[0] := temp.Number[0] - b + 1;
   FGIntAEStroy(temp);
   If (head Shr 30) = 0 Then FGIntDivByIntBis(MGInt, head + 1, r)
   Else FGIntShiftRightBy31(MGInt);
   If FGIntCompareAbs(MGInt, base) <> St Then FGIntSubBis(MGInt, base);
   FGIntAEStroy(temp);
   FGIntAEStroy(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);
   FGIntAEStroy(temp2);
   FGIntMod(r, modb, res);
   FGIntMulMod(FGInt, res, modb, temp2);
   FGIntAEStroy(r);

   For i := length(S) Downto 1 Do
   Begin
      If S[i] = '1' Then
      Begin
         FGIntmul(res, temp2, temp3);
         FGIntAEStroy(res);
         FGIntMontgomeryMod(temp3, modb, baseinv, res, b, head);
         FGIntAEStroy(temp3);
      End;
      FGIntSquare(temp2, temp3);
      FGIntAEStroy(temp2);
      FGIntMontgomeryMod(temp3, modb, baseinv, temp2, b, head);
      FGIntAEStroy(temp3);
   End;
   FGIntAEStroy(temp2);
   FGIntMontgomeryMod(res, modb, baseinv, temp3, b, head);
   FGIntCopy(temp3, res);
   FGIntAEStroy(temp3);
   FGIntAEStroy(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);
            FGIntAEStroy(temp3);
         End;
         FGIntCopy(temp1, GCD);
         FGIntAEStroy(temp2);
         FGIntAEStroy(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);
   FGIntAEStroy(temp1);
   FGIntAEStroy(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);
   FGIntAEStroy(temp);
   FGIntAEStroy(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);
      FGIntAEStroy(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;

   FGIntAEStroy(zero);
   FGIntAEStroy(one);
   FGIntAEStroy(two);
   FGIntAEStroy(m);
   FGIntAEStroy(z);
   FGIntAEStroy(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);
         FGIntAEStroy(r1);
         r1 := r2;
         r2 := r3;

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

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

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

      FGIntAEStroy(ta);
      FGIntAEStroy(r1);
      FGIntAEStroy(r2);
      FGIntAEStroy(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
         FGIntAEStroy(r3);
         FGIntdivmod(r1, r2, temp, r3);
         FGIntCopy(r2, r1);
         FGIntCopy(r3, r2);

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

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

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

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

      FGIntAEStroy(tb);
      FGIntAEStroy(r1);
      FGIntAEStroy(r2);
   End;
   FGIntAEStroy(gcd);
   FGIntAEStroy(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
      FGIntAEStroy(temp1);
      L := 0;
   End
   Else
   Begin
      FGIntAEStroy(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);
            FGIntAEStroy(temp3);
            FGIntDivByInt(temp4, temp3, 8, i);
            If (temp3.Number[1] Mod 2) = 0 Then ok1 := false Else ok1 := true;
            FGIntAEStroy(temp3);
            FGIntAEStroy(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);
            FGIntAEStroy(temp3);
            FGIntAEStroy(temp4);
            FGIntDivByInt(temp5, temp3, 4, i);
            If (temp3.Number[1] Mod 2) = 0 Then ok2 := false Else ok2 := true;
            FGIntAEStroy(temp5);
            FGIntAEStroy(temp3);
```

```
            If ok2 = true Then L := L * (-1);
            FGIntMod(temp1, temp2, temp3);
            FGIntCopy(temp2, temp1);
            FGIntCopy(temp3, temp2);
         End;
      End;
      FGIntAEStroy(temp1);
      FGIntAEStroy(temp2);
   End;
   FGIntAEStroy(zero);
   FGIntAEStroy(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);
   FGIntAEStroy(temp);
   FGIntModInv(Square, Prime, temp1);

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

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

   FGIntCopy(r, SquareRoot);
   FGIntAEStroy(r);
   FGIntAEStroy(s);
   FGIntAEStroy(b);
   FGIntAEStroy(temp1);
   FGIntAEStroy(one);
   FGIntAEStroy(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;
   FGIntAEStroy(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 TFmAbout.FormCreate(Sender: TObject);
begin
Memo1.Clear;
Memo1.Lines.Add('БАКАЛАВРСЬКИЙ ПРОЕКТ');
Memo1.Lines.Add('');
Memo1.Lines.Add('на  тему:');
Memo1.Lines.Add('');
Memo1.Lines.Add('Програмне забезпечення системи кібербезпеки застосунків та
даних захищеної файлової системи');
Memo1.Lines.Add('');
Memo1.Lines.Add('Керівник:  Петренюк В.І.');
Memo1.Lines.Add('');
Memo1.Lines.Add('Розробив:  студент Лівітчук Олексій Віталійович');
Memo1.Lines.Add('           гр. КБ-21СКЗ');
Memo1.Lines.Add('');
Memo1.Lines.Add('м. Кропивницький 2023');
Memo1.Lines.Add('');
end;

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

end.
```