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 другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи управління файлами в ОС Windows на основі технології BitLocker Drive Encryption”

КбПЗ_2023

Виконав здобувач вищої освіти
ІІ курсу, групи КН-22МЗ
ОПП «Комп’ютерні науки»
спеціальності 122 «Комп’ютерні науки»
_____ Пісаренко Д.О.
«____» _____ 2023 р.

Керівник проекту
кандидат технічних наук
_____ Лисенко І.А.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Центр *Заочної та дистанційної освіти*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _12 _____*"Інформаційні технології"*_____

Спеціальність 122 *"Комп'ютерні науки"*

Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерні науки"*_

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

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Пісаренко Діані Олександрівні
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи управління файлами в ОС Windows на основі технології BitLocker Drive Encryption*

2. Керівник роботи *Лисенко Ірина Анатоліївна, канд. техн. наук*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 37-13 від 04.08.2023 року

3. Строк подання студентом роботи до захисту *10.12.2023 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи управління файлами в ОС Windows на основі технології BitLocker Drive Encryption*

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

1. Призначення та область використання.	*6. Наукова новизна.*
2.Перегляд аналогічних існуючих систем.	*7. Економічна ефективність розробленої програми.*
3. Опис і обґрунтування проектних рішень.	*8. Заходи з охорони праці та техніки безпеки.*
4. Етапи програмування системи.	*9. Висновки.*
5. Впровадження системи в промислову експлуатацію	

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

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Показники економічної ефективності	*1 аркуш*

КОПІЯ 2023

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В.	05.10.2023	14.11.2023
Охорона праці	Оришака О.В.	06.10.2023	16.11.2023

7. Дата видачі завдання « 6 » вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2023 р.	
3.	Розробка моделі компонента	20.10.2023 р.	
4.	Розробка структур даних	25.10.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2023 р.	
6.	Програмування алгоритмів	10.11.2023 р.	
7.	Розрахунок економічної ефективності	13.11.2023 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2023 р.	
9.	Оформлення ПЗ	17.11.2023 р.	
10.	Попередній захист роботи	10.12.2023 р.	

КОПЗ_2023

Дата видачі завдання
« 6 » вересня 2023 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2023 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Пісаренко Д.О. Дослідження та програмна реалізація системи управління файлами в ОС Windows на основі технології BitLocker Drive Encryption. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи управління файлами в ОС Windows на основі технології BitLocker Drive Encryption.

Метою розробки є дослідження та програмна реалізація системи управління файлами в ОС Windows на основі технології BitLocker Drive Encryption.

Об'єктом дослідження є процес управління файлами в ОС Windows на основі технології BitLocker Drive Encryption.

Предметом дослідження є методи управління файлами в ОС Windows на основі технології BitLocker Drive Encryption.

Методи дослідження базуються на методах захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи управління файлами в ОС Windows на основі технології BitLocker Drive Encryption.

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 Delphi 10.4.

Ключові слова: комп'ютерні науки, BitLocker Drive Encryption

КОПЗ-2023

ABSTRACT

Pisarenko D.O. Research and software implementation of a file management system in the Windows OS based on BitLocker Drive Encryption technology. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this graduation thesis for the second (master's) level of higher education, software is developed, which is intended for the file management system in the Windows OS based on the BitLocker Drive Encryption technology.

The purpose of the development is the research and software implementation of the file management system in the Windows OS based on the BitLocker Drive Encryption technology.

The object of the study is the file management process in the Windows OS based on the BitLocker Drive Encryption technology.

The subject of the study is file management methods in the Windows OS based on BitLocker Drive Encryption technology.

Research methods are based on information protection methods, mathematical statistics methods, and software development methods.

The result of the work is a software implementation of the file management system in the Windows OS based on the BitLocker Drive Encryption technology.

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

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

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

The program was developed in the Delphi 10.4 environment.

Keywords: computer science, BitLocker Drive Encryption

КОПЗ-2023

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 3
ВСТУП..... 4
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 7
1.1 Призначення системи..... 7
1.2 Область застосування..... 8
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 10
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти..... 10
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування..... 14
2.3 Розгорнута постановка завдання 20
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 22
3.1 Опис функціонування системи 22
3.2 Розробка структурної схеми..... 34
3.3 Розробка функціональної схеми 45
3.4 Розробка діаграми процесів..... 51
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ..... 53
4.1 Розробка блок-схем та опис алгоритмів функціонування системи..... 53
4.2 Захист розробленого програмного забезпечення..... 64
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ..... 65
6 НАУКОВА НОВИЗНА 67

КОПЗ_2023

7 ЕКОНОМІЧНА ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ..... 68
7.1 Техніко економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти..... 68
7.2 Розрахунок трудомісткості розробки програмної продукції..... 70
7.3 Визначення чисельності виконавців і планового фонду зарплати..... 72
7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника..... 77
7.5 Визначення собівартості розробки та ціни програмної продукції..... 81
7.6 Визначення об’єму капітальних вкладень та експлуатаційних витрат у споживача програмної продукції..... 84
7.7 Визначення експлуатаційних витрат..... 85
7.8 Визначення економічної ефективності програмної продукції..... 86
7.9 Висновок..... 88
8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ..... 89
8.1 Вступ..... 89
8.2 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста..... 90
8.3 Розробка заходів з умов поліпшення охорони праці..... 93
8.4 Розрахункова частина..... 94
8.5 Список використаних джерел інформації..... 96
9 ОСНОВНІ ВИСНОВКИ..... 99
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ..... 101

КОПЗ – 2023

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

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

КбПЗ-2023

ВСТУП

Актуальність теми. Шифрування – це спосіб підвищення безпеки повідомлення або файлу, шляхом перетворення його вмісту таким чином, щоб його змогла прочитати лише та особа, яка має ключ шифрування для розшифрування файлу. Наприклад під час купівлі товарів або послуг через Інтернет відомості про транзакцію (такі як ваша адреса, номер телефону та кредитної картки) зазвичай шифруються з метою посилення безпеки. Використовуйте шифрування, коли потрібен надійний захист даних. Щоб зашифрувати диск, на який інстальовано Windows, на комп'ютері мають бути два розділи: системний розділ (який містить файли, потрібні для запуску комп'ютера) та розділ операційної системи (на якому міститься Windows). Розділ операційної системи буде зашифровано, а системний розділ залишиться незашифрованим, отже комп'ютер можна буде запускати.

У попередніх версіях Windows можна створити ці розділи вручну. У поточних версіях Windows розділи створюються автоматично. Якщо на комп'ютері немає системного розділу, майстер BitLocker створить його, використавши 200 МБ доступного дискового простору. Системному розділу не присвоюватиметься буква диска і він не відображатиметься у папці «Комп'ютер».

За допомогою програми шифрування дисків BitLocker можна захистити файли на всьому диску. BitLocker може допомогти запобігти доступу хакерів до системних файлів, через які вони дізнаються ваші паролі, або запобігти доступу до диска шляхом його фізичного вилучення з ПК та встановлення в інший ПК. При цьому ви можете входити до системи Windows і використовувати файли, як робите це завжди.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи управління файлами в ОС Windows на основі технології BitLocker Drive Encryption.

КПЗ – 2023

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

– Огляд існуючих систем управління файлами в ОС Windows на основі технології BitLocker Drive Encryption.

– Дослідження системи управління файлами в ОС Windows на основі технології BitLocker Drive Encryption.

– Програмна реалізація системи управління файлами в ОС Windows на основі технології BitLocker Drive Encryption.

Об'єктом дослідження є процес управління файлами в ОС Windows на основі технології BitLocker Drive Encryption.

Предметом дослідження є методи управління файлами в ОС Windows на основі технології BitLocker Drive Encryption.

Методи дослідження базуються на методах захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод управління файлами в ОС Windows на основі технології BitLocker Drive Encryption.

– Розроблено вітчизняний продукт управління файлами в ОС Windows на основі технології BitLocker Drive Encryption, який має більш широкі можливості, на відміну від існуючих аналогів.

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

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

КОПЗ-2023

Робота апробована на LVII Науково-технічній конференції здобувачів вищої освіти «Наука – виробництву», 2023, основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №14.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи управління файлами в ОС Windows на основі технології BitLocker Drive Encryption, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

КбПЗ_2023

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Захист файлів за допомогою програми шифрування дисків BitLocker

Вмикаючи BitLocker вперше, обов’язково створіть ключ відновлення. Інакше можна назавжди втратити доступ до своїх файлів.

Якщо ви користуєтеся невізуальним екраном, екрани BitLocker, що відображаються до появи екрана привітання, озвучені не будуть. Це такі екрани, як екран вводу PIN-коду BitLocker або екран відновлення BitLocker .

Відкрийте засіб BitLocker Drive Encryption, для чого проведіть пальцем від правого краю екрану і торкніться кнопки Пошук (або, якщо використовуєте мишу, наведіть вказівник на правий верхній кут екрана, проведіть униз і натисніть кнопку Пошук), у поле пошуку введіть BitLocker, виберіть пункт Настройки, а потім BitLocker Drive Encryption.

Натисніть кнопку Увімкнути BitLocker. Можливо, буде запропоновано ввести пароль адміністратора або підтвердити вибір.

Відкриється діалогове вікно настроювання програми BitLocker Drive Encryption.

Дотримуйтеся вказівок.

Відключення або тимчасове вимкнення програми BitLocker

Можна тимчасово вимкнути програму BitLocker, приміром, коли потрібно інсталювати нове ПЗ, яке BitLocker може блокувати, а потім відновити його роботу, коли цей процес буде завершено. Можна також зовсім відключити BitLocker, унаслідок чого диск буде дешифровано і захист BitLocker буде повністю знято.

Відкрийте засіб BitLocker Drive Encryption, для чого проведіть пальцем від правого краю екрану і торкніться кнопки Пошук (або, якщо використовуєте

КбПЗ-2023

мишу, наведіть вказівник на правий верхній кут екрана, проведіть униз і натисніть кнопку Пошук), у поле пошуку введіть BitLocker, виберіть пункт Настройки, а потім BitLocker Drive Encryption.

Виконайте одну з нижченаведених дій.

Щоб тимчасово вимкнути BitLocker, виберіть пункт Призупинити захист. Можливо, буде запропоновано ввести пароль адміністратора або підтвердити вибір. Коли все буде готово, натисніть кнопку Так.

Щоб відключити BitLocker і дешифрувати диск, виберіть посилання Вимкнути BitLocker. Можливо, буде запропоновано ввести пароль адміністратора або підтвердити вибір. Коли все буде готово, натисніть кнопку Вимкнути BitLocker.

1.2 Область застосування

Нові файли, які ви додаєте на диск, де використовується BitLocker, автоматично шифруються. Та якщо скопіювати ці файли на інший диск або на інший ПК, їх буде автоматично дешифровано.

Програма BitLocker може зашифрувати диск, на якому інстальовано систему Windows (диск операційної системи), а також незнімні диски даних (такі як внутрішні жорсткі диски). Засіб BitLocker To Go можна використовувати для покращення захисту всіх файлів, збережених на знімному диску (наприклад, на зовнішньому жорсткому диску або у флеш-пам’яті USB).

Під час запуску ПК BitLocker перевіряє його на наявність умов, що можуть свідчити про ризики для безпеки (наприклад, про зміну програмного забезпечення BIOS, яке запускає операційну систему під час ввімкнення ПК, або файлів запуску). Якщо виявлено потенційну загрозу безпеці, засіб BitLocker блокує диск операційної системи та вимагає спеціальний ключ відновлення BitLocker для його розблокування.

КОПЗ-2023

Можна вибрати спосіб розблокування системного диска під час ввімкнення ПК: за допомогою ключа запуску або PIN-коду. Додаткові відомості про ключі запуску та PIN-коди, а також про принцип їхньої дії див. в розділі Відмінність між паролем власника модуля TPM, паролем відновлення, ключем відновлення, паролем, PIN-кодом, розширеним PIN-кодом і ключем запуску. Докладніше про модуль TPM див. у розділі Переваги модуля TPM.

Можна вибрати, як потрібно розблокувати зашифрований диск даних: за допомогою пароля чи смарт-картки. Для жорстких дисків із даними також можна встановити налаштування автоматичного розблокування в разі розблокування ПК; для цього потрібно, щоб диск операційної системи було захищено програмою BitLocker. Для знімних дисків із даними, зашифрованих за допомогою BitLocker To Go, можна встановити налаштування автоматичного розблокування під час входу на ПК.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи управління файлами в ОС Windows на основі технології BitLocker Drive Encryption, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

КбПЗ-2023

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Вільно розповсюджувана програма TrueCrypt з відкритим вихідним кодом була популярна протягом 10 років завдяки свій незалежності від основних вендорів. Творці програми привселюдно невідомі. Серед найвідоміших користувачів програми можна виділити Едварда Сноудена й експерта по безпеці Брюса Шнайера. Утиліта дозволяє перетворити флеш-накопичувач або жорсткий диск у захищене зашифроване сховище, у якому конфіденційна інформація схована від сторонніх очей.

Таємничі розроблювачі утиліти оголосили про закриття проекту в середовище 28 травня, пояснивши, що використання TrueCrypt небезпечно. «УВАГА: Використовувати TrueCrypt небезпечно, так як програма може містити неусунуті уразливості» – таке повідомлення можна побачити на сторінці продукту на порталі SourceForge. Далі треба ще один обіг: «Ви повинні перенести всі дані , зашифровані в TrueCrypt на зашифровані диски або образи віртуальних дисків, підтримувані на вашій платформі».

Спочатку з'являлися припущення, що сайт програми був зламаний кіберзлочинцями, але тепер стає ясно, що це не обман. Сайт SourceForge зараз пропонує оновлену версію TrueCrypt (яка має цифровий підпис розроблювачів), під час установки якої пропонується перейти на BitLocker або інший альтернативний інструмент.

КПЗ – 2023

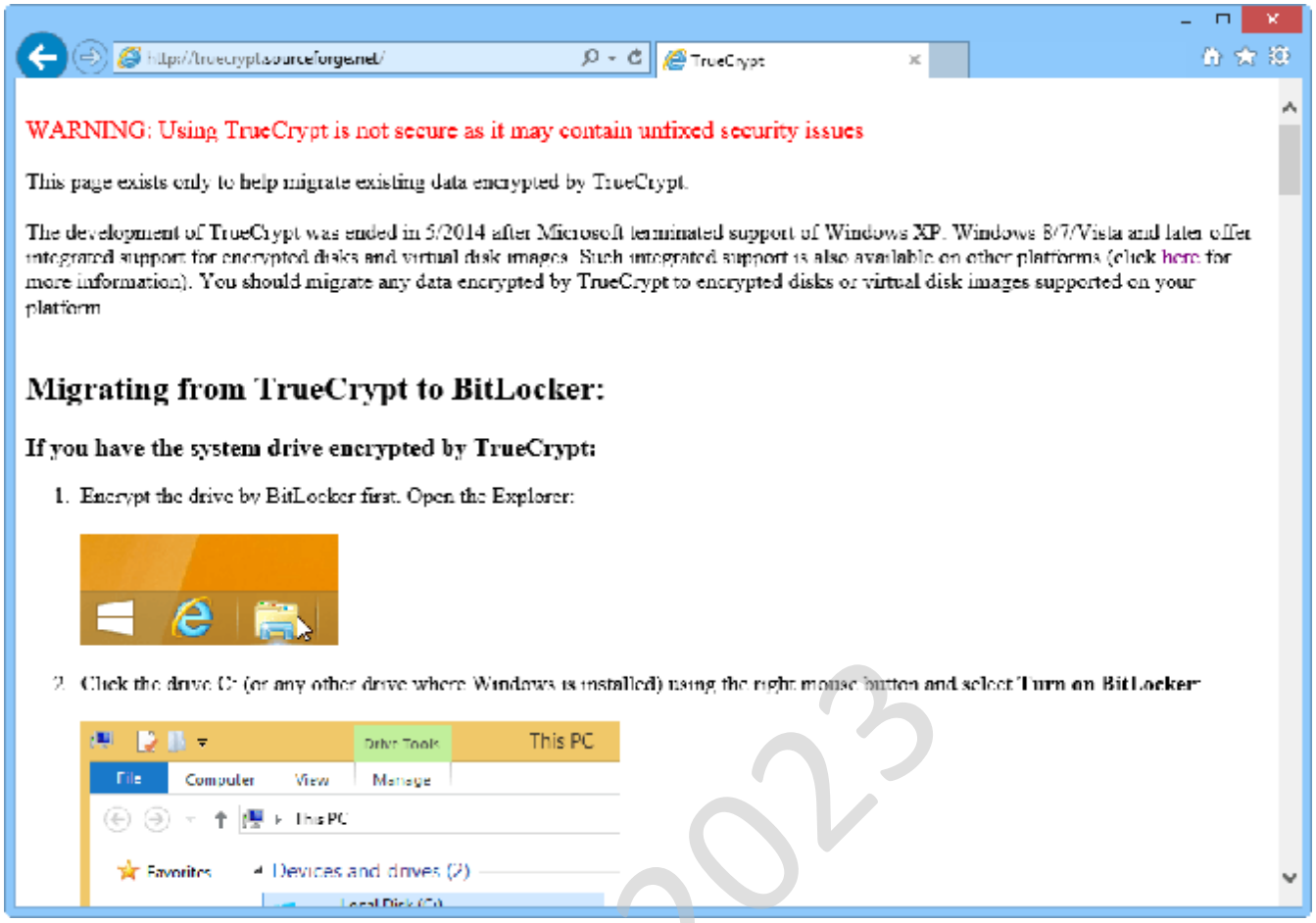


Рисунок 2.1 – Сторінка TrueCrypt

Сайт і спливаюче оповіщення в самій програмі містить інструкції з переносу файлів, зашифрованих TrueCrypt на сервіс BitLocker від Microsoft, що поставляється разом з ОС Windows 10/11.

TrueCrypt дозволяє дешифрувати файли, але не дозволяє створювати нові зашифровані розділи.

Самою очевидною альтернативою програмі є BitLocker, але є й інші варіанти. Шнайер поділився, що він вертається до використання PGPDisk від Symantec. Symantec Drive Encrpytion (110 доларів за одну користувальницьку ліцензію) використовує добре відомий і перевірений метод шифрування PGP.

КБПЗ_2023

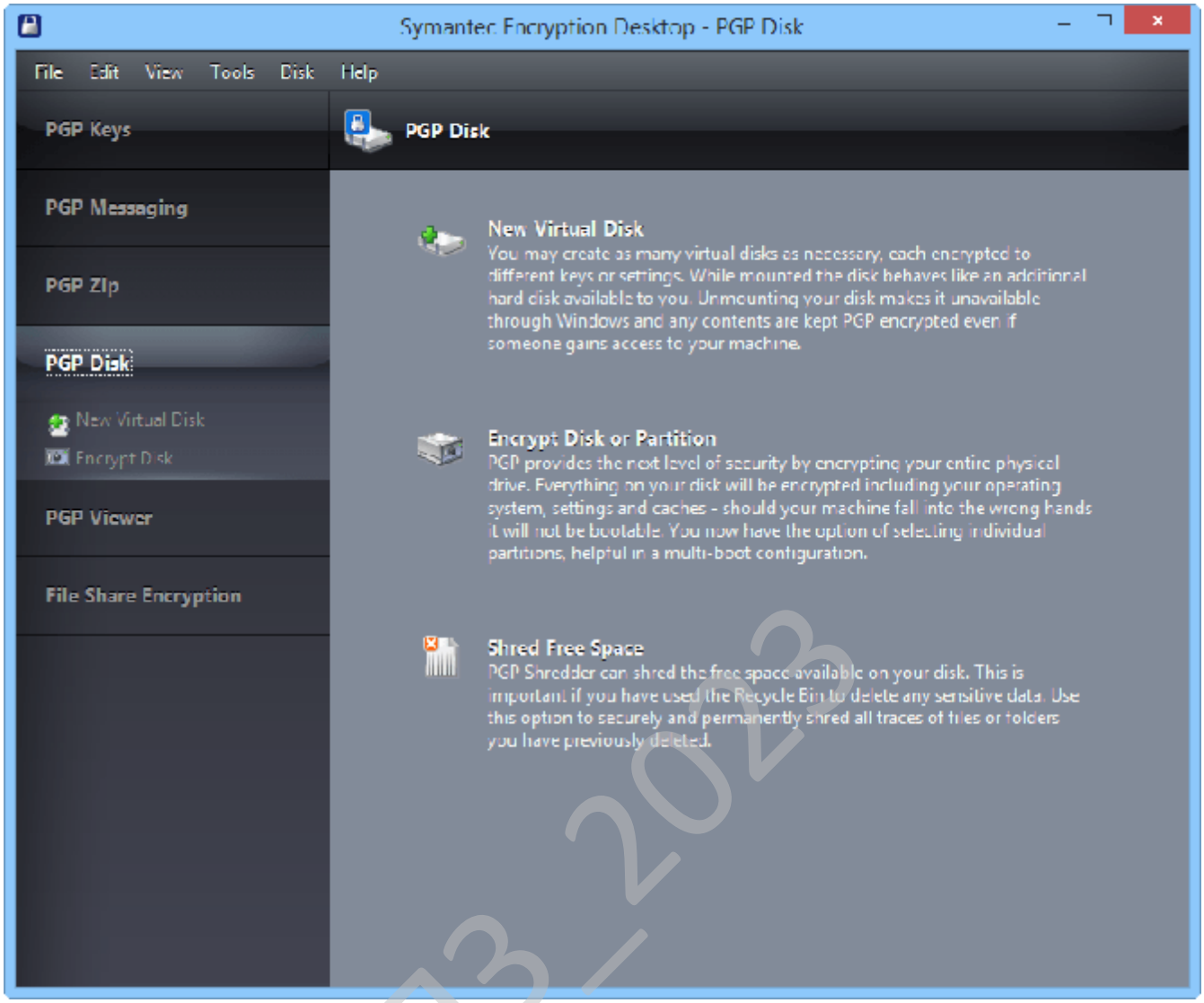


Рисунок 2.2 – Інтерфейс користувача PGPDisk

Існують і інші безкоштовні альтернативи для Windows, наприклад DiskCryptor. Дослідник по комп'ютерній безпеці, відомий як The Grugq торік склав цілий список альтернатив TrueCrypt, що актуальний і донині.

Йоханнес Ульріх, науковий керівник технологічного інституту SANS користувачам Mac OS X рекомендує звернути увагу на FileVault 2, що убудований в OS X і більше пізні ОС даного сімейства. FileVault використовує 128-бітне шифрування XTS-AES, що застосовується в агентстві національної безпеки США (NSA). На думку Ульріха користувачі Linux повинні дотримуватися убудованого системного інструмента Linux Unified Key Setup

КБПЗ_2023

(LUKS). Якщо Ви використовуєте Ubuntu, то установник цієї ОС уже дозволяє включити повне шифрування дисків із самого початку.

Проте, користувачам знадобляться інші додатки для шифрування переносних носіїв, які використовуються на комп'ютерах з різними ОС. Ульріх сказав, що в цьому випадку на розум приходить PGP/GnuPG.

Німецька компанія Steganos пропонує скористатися старою версією своєї утиліти шифрування Steganos Safe (актуальна версія на даний момент – 15, а пропонується скористатися 14 версією), що поширюється безкоштовно.

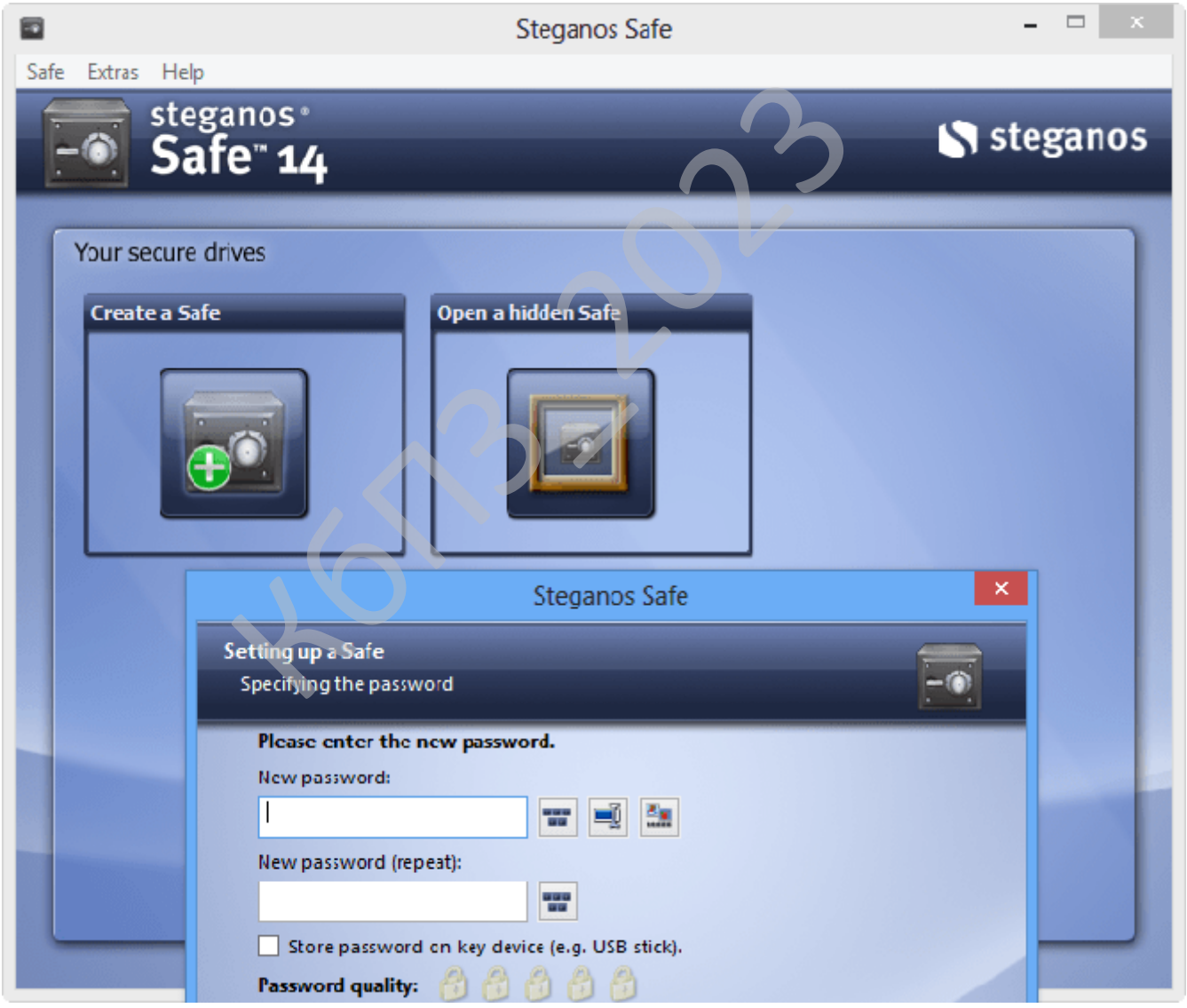


Рисунок 2.3 – Інтерфейс користувача Steganos

Той факт, що TrueCrypt може мати уразливості в безпеці викликає серйозні побоювання, особливо з огляду на, що аудит програми не виявив подібних проблем. Користувачі програми нагромадили 70 000 доларів для проведення аудита після слухів про те, що агентство національної безпеки США можуть декодувати значні обсяги зашифрованих даних. Перший етап дослідження, у якому аналізувався завантажник TrueCrypt був проведений минулого місяця. Аудит не виявив ні бекдорів, ні навмисних уразливостей. Наступна фаза дослідження, у якій повинні були перевірятися використовувані методи криптографії була запланована на це літо.

Грін був одним з експертів, що беруть участь в аудиті. Він розповів, що не мав ніякої попередньої інформації про те, що розроблювачі планую закрити проект. Грін розповів: «Останнє що я чув від розроблювачів TrueCrypt: «Ми з нетерпінням чекаємо результати 2 фази випробування. Дякуємо за ваші старання!». Потрібно відзначити, що аудит продовжиться, як було заплановано, незважаючи на зупинку проекту TrueCrypt.

Щоб додати масла у вогонь відзначимо, що 19 травня TrueCrypt була вилучена із захищеної системи Tails (улюбленої системи Сноудена). Причина до кінця не ясна, але використовувати програму явно не треба.

КПЗ-2023

2.2 Обґрунтування вибору засобів для побудови системи 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.

КОПЗ-2023

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

КОПЗ_2023

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++

КОПЗ – 2023

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.

КОПЗ – 2023

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.

КОПЗ-2023

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 другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи управління файлами в ОС Windows на основі технології BitLocker Drive Encryption.

В процесі розробки випускної кваліфікаційної роботи за другим (магістерським) 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 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

КОПЗ_2023

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 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.

КбПЗ_2023

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

BitLocker Drive Encryption – пропрієтарна технологія, що є частиною операційних систем Microsoft Windows 10/11 Ultimate/Enterprise, Windows 7 Ultimate/Enterprise, Windows Server 2008 R2 і Windows 8. BitLocker дозволяє захищати дані шляхом повного шифрування диска (ів) (логічних, з Windows 7 – і карт SD та USB-флешек) (у термінології Microsoft – тому(ів)). Підтримуються наступні алгоритми шифрування:

– AES 128.

– AES 128 с Elephant diffuser (використовується за замовчуванням).

– AES 256.

– AES 256 с Elephant diffuser.

Сам ключ може зберігатися в TPM або на USB-пристрої, або ж на комп'ютері. У випадку з TPM при завантаженні комп'ютера ключ може бути отриманий з нього відразу, або тільки після автентифікації за допомогою USB-ключа або уведення PIN-коду користувачем. Таким чином, можливі наступні комбінації для доступу:

– TPM.

– TPM + PIN.

– TPM + PIN + USB-ключ.

– TPM + USB-ключ.

– USB-ключ (даний режим вимагає активації через групові політики).

– Пароль (даний режим доступний починаючи з Windows 8, а також вимагає активації через групові політики).

КОПЗ-2023

Принципи роботи

BitLocker шифрує том, а не фізичний диск. Том може займати частину диска, а може містити в собі масив з декількох дисків. Для роботи BitLocker'у у випадку шифрування системного диска буде потрібно два NTFS-томи, один для ОС і один для завантажувальної частини. Останній повинен бути не менш 1,5 Гб, і не буде зашифрований. Починаючи з Windows Vista SP1 з'явилася можливість шифрувати несистемні томи. Після створення розділів необхідно ініціалізувати TPM-модуль у ПК, де він є, і активувати BitLocker. В Windows 7 з'явився BitLocker To Go, що дозволяє шифрувати змінні носії, а також знижені вимоги для завантажувальної частини, для неї досить 100 Мб. При установці Windows 7 на порожній диск завантажувальний розділ створюється автоматично.

Механізми розшифровки і їхньої уразливості

Існує три механізми перевірки дійсності, які можна використовувати для реалізації Bitlocker шифрування:

– Прозорий режим роботи: Цей режим використовує можливості апаратного забезпечення Trusted Platform Module (TPM) для надання прозорої роботи користувачів. Користувачі входять на комп’ютер з операційною системою Windows, як звичайно. Ключ, використовуваний для шифрування диска, закодований у чип TPM і він може бути виданий тільки в коді завантажника ОС (якщо завантажувальні файли, показуються як не змінені). Цей режим уразливий для нападу при холодному завантаженні, так як дозволяє виключити комп'ютер і завантажитися зловмисникові.

– Режим перевірки дійсності користувача: Цей режим припускає, що користувач пройшов деяку автентифікацію в передзавантажувальне середовище у вигляді попереднього уведення PIN-коду. Цей режим уразливий при використанні буткіт-атак.

– Режим USB-ключа: Користувач повинен вставити пристрій USB у комп'ютер, що містить ключ запуску, щоб мати можливість завантаження в захищену операційну систему. Зверніть увагу, що для цього режиму необхідно,

КОПЗ_2023

щоб BIOS на комп'ютері підтримував читання пристроїв USB у завантажувальному середовищі. Цей режим також уразливий до буткіт-нападів.

Використання в інших операційних системах

Існує неофіційна утиліта dislocker [1] для операційних систем GNU/Linux і Mac OS X, що являє собою інструмент для читання й запису томів, зашифрованих через BitLocker.

Для надання доступу до зашифрованого тому утиліта dislocker використовує FUSE-драйвер, а створювати нові зашифровані томи дана утиліта не вміє. [2]

Trusted Platform Module

В обчислювальній техніці Trusted Platform Module (TPM) – назва специфікації, що описує криптопроцесор, у якому зберігаються криптографічні ключі для захисту інформації, а також узагальнене найменування реалізацій зазначеної специфікації, наприклад, у вигляді «чипа TPM» або «пристрою безпеки TPM» (Dell). Раніше називався «чипом Фріца» (колишній сенатор Ернест «Фріц» Холлінгс відомий своєю гарячою підтримкою системи захисту авторських прав на цифрову інформацію, DRM). Специфікація TPM розроблена некомерційною організацією "Trusted Computing Group" (англ.). Поточна версія специфікації TPM – 1.2 ревізія 116, видання 3 березня 2011.

У січні 1999 року була створена робоча група виробничих компаній «Альянс комп'ютерних платформ, що довіряються,» (англ. Trusted Computing Platform Alliance, TCPA) з метою розвитку механізмів безпеки й довіри в комп'ютерних платформах. Спочатку в TCPA входили провідні розроблювачі апаратного й програмного забезпечення – HP, Compaq (у цей час підрозділ HP), IBM, Intel, Microsoft. [1]

У жовтні 1999 року була анонсована проектна специфікація й відкрита можливість іншим компаніям приєднатися до альянсу. У серпні 2000 року була випущена для обговорення попередня публічна версія специфікації. Специфікація

КОПЗ-2023

TCPA версії 1.0 була опублікована в лютому 2001 року, у ній були визначені основні вимоги до TPM з погляду виробника електронних пристроїв. [2]

Потім була створена робоча група по створенню TPM, що переглянула загальну специфікацію з погляду практичного застосування модуля, що довіряється, (TPM). У серпні 2001 року була випущена специфікація версії 1.1 і створена робоча група по проектуванню платформи ПК, на яку встановлюється довіряється модуль, що. [2]

У квітні 2003 року була організована некомерційна організація «Trusted Computer Group» (TCG), що стала спадкоємцем TCPA і продовжила працювати над розвитком уже випущених специфікацій. На додаток до вже створених робочих груп по проектуванню TPM і платформи ПК були створені групи по розробці специфікацій для мобільних пристроїв, ПК-клієнтів, серверів, запам'ятовувальних пристроїв, інфраструктури обчислень, що довіряються, програмного забезпечення (англ. Trusted Software Stack, TSS) і мережного з'єднання, що довіряється. У листопаді 2003 року була опублікована специфікація TPM версії 1.2, остання версія з істотними змінами, у якій по суті описана функціональність TPM. [2]

Короткий огляд

Trusted Platform Module (TPM), що містить у собі криптопроцесор, забезпечує засобу безпечного створення ключів шифрування, здатних обмежити використання ключів (як для підпису, так і для шифрування/дешифрування) з тим же ступенем неповторюваності, що й генератор випадкових чисел. Також цей модуль має наступні можливості: вилучену атестацію, прив'язку й надійне захищене зберігання. Вилучена атестація створює зв'язок апаратних засобів, завантаження системи й конфігурації хоста (ОС комп'ютера), дозволяючи третій особі (начебто цифрового магазина музики) перевіряти, щоб програмне забезпечення або музика, завантажена з магазина, не були змінені або скопійовані користувачем (див. DRM). Криптопроцесор шифрує дані таким способом, що вони можуть бути розшифровані тільки на комп'ютері, де були зашифровані, під

КОПЗ_2023

керуванням того ж самого програмного забезпечення. Прив'язка шифрує дані, використовуючи ключ підтвердження TPM – унікальний ключ RSA, записаний у чип у процесі його виробництва, або інший ключ, якому довіряють. [3]

Модуль TPM може використовуватися, щоб підтвердити дійсність апаратних засобів. Так як кожний чип TPM унікальний для специфічного пристрою, це уможливлює однозначне встановлення дійсності платформи. Наприклад, щоб перевірити, що система, до якої здійснюється доступ – очікувана система.

Архітектура TPM

У специфікації TCG описаний мінімальний набір алгоритмів і протоколів, яким повинен задовольняти чип TPM. Крім того, виробником можуть бути реалізовані додаткові алгоритми й протоколи (які, зрозуміло, повинні бути описані виробником у відповідній документації).

В архітектурі чипа реалізовані наступні захисні механізми:

– захищене керування пам'яттю;

– шифрування шини й даних;

– тестування режимів блокування;

– активне екранування.

У чип імплементовані алгоритми асиметричної криптографії, що забезпечують високий рівень захисту. Деякі елементи логічного дизайну чипа є нестандартними з погляду типових методів проектування інтегральних схем (IC). Застосовуються й спеціальні прийоми проектування IC: "заплутування" топології шарів IC, що ускладнює аналіз функцій елементів мікросхеми. Ряд технологічних особливостей чипів безпеки спеціально не розголошується компаніями-виробниками, щоб зменшити ймовірність злому навіть у тому випадку, коли для цього застосовуються сучасні методи аналізу функціонування мікросхем і дороге устаткування. [4]

КПЗ_2023

Ввод/Вивід (англ. I/O)

Цей компонент управляє потоком інформації із шини й управляє повідомленнями між відповідними компонентами TPM. I/O компонент уводить у дію політикові доступу, пов'язану з функціями TPM. Правила доступу визначаються прапорами доступу, що зберігаються в блоці Opt-In енергонезалежної пам'яті. [5]

Криптографічний процесор

Здійснює криптографічні операції усередині TPM. Ці операції містять у собі:

– Генерація асиметричних ключів (RSA).

– Асиметричне шифрування/дешифрування (RSA).

– Хешування (SHA-1).

– Генерація випадкових чисел.

TPM використовує ці можливості для генерації випадкових послідовностей, генерації асиметричних ключів, цифрового підпису й конфіденційності збережених даних. Також TPM підтримує симетричне шифрування для внутрішніх потреб. Всі збережені ключі по силі повинні відповідати ключу RSA довжиною 2048 біт. [5]

Енергонезалежна пам'ять (англ. Non-Volatile Storage)

Використовується для зберігання ключа підтвердження, кореневого ключа (англ. Storage Root Key, SRK), авторизаційних даних, різних прапорів доступу й блоку Opt-In. Обсяг цього типу пам'яті обмежений (1280 байт). [6]

Ключ підтвердження (англ. Endorsement Key, EK)

EK – ключ RSA розміром 2048 біт, що ідентифікує чип, а також весь пристрій, фундаментальний компонент TPM. Відкрита частина називається PUBEK, закрита – PRIVEK. Відповідно до політики безпеки PRIVEK не повинен бути доступний поза чипом, він ніколи не використовується для генерування підписів. PUBEK зберігається в сертифікаті, використовується тільки для встановлення власника TPM і в процесі генерації AIK. EK генерується до того, як

КБПЗ-2023

кінцевий користувач одержить платформу. Стандарт дозволяє змінити цей ключ, через що використання TPM може бути обмеженим. [5]

Ключі підтвердження дійсності (англ. Attestation Identity Keys, AIK)

AIK – ключ RSA довжиною 2048 біт, використовуваний тільки для підписів, для шифрування не використовується. TPM може згенерувати необмежена кількість AIK, ці ключі повинні бути постійними, але рекомендується зберігати AIK у вигляді блоків у постійній зовнішній пам'яті, а не усередині енергонезалежної пам'яті TPM. У відповідності зі специфікацією передбачається, що виробники забезпечать досить місця для багатьох блоків AIK, які будуть одночасно завантажуватися в енергонезалежну пам'ять TPM. Перехід AIK від одного TPM до іншого заборонений. [6]

Регістри конфігурації платформи (Platform Configuration Registers, PCR)

PCR – це унікальні ознаки TPM, у яких у зашифрованому виді втримується вся інформація про цілісність метрик системи, починаючи із завантаження BIOS до завершення роботи системи. Інформація, що втримується в PCR, формує корінь довіри для вимірів (RTM). Можуть зберігатися як в енергонезалежної, так і в енергонезалежній пам'яті. Ці регістри скидаються при старті й при перезавантаженні системи. Специфікація пропонує мінімальну кількість регістрів (16), кожний регістр містить 20 біт інформації. Регістри 0-7 зарезервовані для потреб TPM. Регістри 8-15 доступні для використання операційною системою й додатками. [5] Зміни значень PCR необоротні і їхні значення не можна записати прямо, їх можна тільки розширити новими значеннями, які залежать від попередніх. Всі зміни значень PCR записуються в балку змін, що зберігається в енергонезалежній пам'яті. [6]

Генератор випадкових чисел (англ. Random Number Generator, RNG)

Використовується для генерації ключів і випадків у сигнатурах (підписах). [6] TPM повинен бути здатним забезпечити 32 випадкових біта на кожний виклик. RNG чипа складається з наступних компонентів.

КОПЗ-2023

Джерело ентропії й колектор

Джерело ентропії – процес (або процеси), що забезпечують ентропію. Такими джерелами можуть бути шум, лічильник тактів процесора й інші події. Колектор ентропії – процес, що збирає ентропію, видаляє зсув, вирівнює вихідні дані. Ентропія повинна передаватися тільки регістру стану.

Регістр стану

Реалізація регістра стану може використовувати 2 регістри: енергонезалежний і незалежний. При старті TPM завантажує енергонезалежний регістр із енергонезалежного. Будь-яка наступна зміна регістра стану від джерела ентропії або від функції, що змішує, впливає на енергонезалежний регістр. При вимиканні TPM записує поточне значення регістра стану в енергонезалежний регістр (таке відновлення може відбуватися й у будь-який інший час). Причиною такої реалізації є прагнення реалізувати енергонезалежний регістр на флеш-пам'яті, кількість запису в яку обмежено. TPM повинен забезпечити відсутність експорту регістра стану.

Функція, що змішує

Бере значення з регістра стану й видає вихідні дані RNG. Кожне використання функції, що змішує, повинне змінювати регістр стану.

При втраті живлення відбувається скидання RNG. Будь-які вихідні дані RNG для TPM повинні бути захищені.

Блок SHA-1 (англ. SHA-1 Engine)

Використовується для обчислення сигнатур (підписів), створення блоків ключів і інших цілей загального призначення. Хеш-інтерфейси доступні поза TPM. Це дозволяє оточенню мати доступ до хеш-функції.

Генератор ключів RSA (англ. RSA Key Generator)

Створює пари ключів RSA. TCG не визначає вимог вчасно генерації ключів. [6]

КОПЗ_2023

Пристрій RSA (англ. RSA Engine)

Використовується для цифрових підписів і шифрування. Немає обмежень на реалізацію алгоритму RSA. Довжина ключа, що рекомендується мінімально – 2048 біт. [5] Виробники можуть використовувати китайську теорему про залишки або будь-який інший метод. Значення відкритої експоненти повинне бути $2^{16} + 1$.

Компонент Opt-In

Цей компонент відповідає за стан TPM і статус володіння користувачем TPM. За це відповідають три групи змінних: TPM включена/відключений (у відключеному стані всі операції блокуються), TPM активований/деактивований (у деактивованому стані можливий виконання операцій, напр. зміна власника), користувач пройшов/не пройшов автентифікацію як власник модуля. Дана інформація зберігається у вигляді прапорів. [5]

Довірена платформа (англ. The trusted Platform)

Ідея довіреної платформи або платформи, який можна довіряти (її очікуване поводження завжди збігається з реальним), заснована на понятті «корінь довіри» (англ. Root of Trust) – набір компонентів, яким потрібно довіряти. Повний набір корінь довіри має мінімальну функціональність, необхідну для опису платформи, що впливає на доручення цій платформі. Є три корені довіри: корінь довіри для вимірів (RTM), корінь довіри для зберігання (RTS) і корінь довіри для повідомлень (RTR). [3] RTM – обчислювальний механізм, що робить надійні виміри цілісності платформи. RTS – обчислювальний механізм, здатний зберігати хеші значень цілісності. RTR – механізм, що надійно повідомляє про збережену в RTS інформації. Дані вимірів описують властивості й характеристики вимірюваних компонентів. Хеші цих вимірів – «знімок» стану комп'ютера. Їхнє зберігання здійснюється функціональністю RTS і RTR. Порівнюючи хеш обмірюваних значень із хешом довіреного стану платформи, можна судити про цілісність системи. [7]

КБПЗ-2023

Можливі застосування

Автентифікація

TPM являє собою токен (Security token) автентифікації наступного покоління. Криптопроцесор підтримує автентифікацію й користувача, і комп'ютера, забезпечуючи доступ до мережі тільки авторизованим користувачам і комп'ютерам. [TCG 3] Це може використовуватися, наприклад, при захисті електронної пошти, заснованої на шифруванні або для підписання цифрових сертифікатів, прив'язаних до TPM. Відмова від паролів і використання TPM дозволяють створити більше сильні моделі автентифікації для провідного, бездротового й VPN доступу. [8]

Захист даних від крадіжки

Це є основним призначенням «захищеного контейнера». Пристрої, що самошифруються, реалізовані на основі специфікацій Trusted Computing Group, роблять доступними убудоване шифрування й контроль доступу до даних. Такі пристрої забезпечують повне шифрування диска, захищаючи дані при втраті або крадіжці комп'ютера. [TCG 4]

Переваги:

– Поліпшення продуктивності.

– Апаратне шифрування дозволяє оперувати з усім діапазоном даних без втрат продуктивності.

– Посилення безпеки.

– Шифрування завжди включене. Крім того, ключі генеруються усередині пристрою й ніколи не залишають його.

– Низькі витрати використання.

– Не потрібні модифікації операційної системи, додатків і т.д. Для шифрування не використовуються ресурси центрального процесора. [8].

– Більші перспективи має зв'язування TPM+Bitlocker.Таке рішення дозволяє прозоро від ПЗ шифрувати весь диск. [7]

КОПЗ_2023

Керування доступом до мережі (NAC)

TPM може підтверджувати дійсність комп'ютера й навіть його працездатність ще до одержання доступу до мережі й, якщо необхідно, поміщати комп'ютер у карантин. [9]

Захист ПЗ від зміни

Сертифікація програмного коду забезпечить захист ігор від читерства, а програми, що вимагають особливої обережності, на зразок банківських і поштових клієнтів, – від навмисної модифікації. [8] Відразу ж буде припинене додавання «троянського коня» у встановлюваному додатку.

Захист від копіювання

Захист від копіювання заснований на такому ланцюжку: програма має сертифікат, що забезпечує їй (і тільки їй) доступ до ключа розшифровки (який також зберігається в TPM). Це дає захист від копіювання, що неможливо обійти програмними засобами. [8]

Реалізація

Хоча специфікація припускає як апаратну, так і програмну реалізації системи TPM, забезпечення належного рівня безпеки, установленого в загальній специфікації, на сьогоднішній день можливо тільки при апаратній реалізації. [2]. Апаратна реалізація у вигляді чипа TPM була вперше випущена в 2005 році. [10] На сьогоднішній день чипом TPM оснащено більше 500.000.000 комп'ютерів. У майбутньому TPM зможе встановлюватися на такі пристрої, як мобільні телефони, пристрої уведення й зберігання інформації. Мікроконтролери TPM на сьогоднішній день виробляються й застосовуються багатьма компаніями.

Проблема «довіри»

Trusted Platform Module критикується деякими [11] IT-фахівцями за назву. Довіра (англ. trust) завжди повинне бути обопільним, у той час як розроблювачі TPM користувачеві не довіряють, що приводить до обмеження волі. На думку окремих IT-фахівців, [8] [12] для довірених обчислень більше підходить назва "віроломні обчислення" (англ. treacherous computing), оскільки наявність модуля гарантує зворотне – систематичний вихід комп'ютера з підпорядкування.

КПЗ-2023

Фактично, комп'ютер перестає функціонувати як комп'ютер загального призначення, оскільки будь-яка операція може зажадати явного дозволу власника комп'ютера. [12]

Втрата «володіння» комп'ютером

Власник комп'ютера більше не може робити з ним усе, що завгодно, оскільки передає частина своїх прав виробникам програмного забезпечення. Зокрема, TPM може заважати (через помилки в ПЗ або навмисного рішення розроблювачів):

– переносити дані на інший комп'ютер;

– вільно вибирати програмне забезпечення для свого комп'ютера;

– обробляти наявні дані будь-якими доступними програмами. [8]

Втрата анонімності

Комп'ютер, обладнаний TPM, має унікальний ідентифікатор, що залишається в чипі. Ідентифікатор відомий виробникові програмного забезпечення і його неможливо змінити. Це ставить під погрозу одне із природних переваг Інтернету – анонімність [8]. На даний момент, якщо на комп'ютері немає троянських програм, у програмному забезпеченні немає явних помилок, а cookie вилучені, єдиним ідентифікатором користувача залишається IP-Адреса й заголовки HTTP. Поряд з підвищенням безпеки, наявність модуля TPM може мати негативний ефект на волю слова, що особливо актуально для країн, що розвиваються. Щоб зрозуміти, до чого може привести віддалено що читається й незмінний ідентифікатор комп'ютера, досить згадати аналогічну проблему з ідентифікаційним номером процесора Pentium III.

Потенційна погроза вільної конкуренції

Програма, що стала лідером галузі (як AutoCAD, Microsoft Word або Adobe Photoshop), може встановити шифрування на свої файли, унеможливлюючи доступ до цих файлів за допомогою програм інших виробників, створюючи, таким чином, потенційну погрозу вільної конкуренції на ринку прикладного ПЗ. [8]

КОПЗ_2023

Проблеми несправності модуля TPM

У випадку несправності модуля TPM контейнери, захищені їм, стають недоступними, а дані, що перебувають у них – невідновними. Для повної гарантії відновлення даних у випадку псування модуля TPM необхідно здійснювати складну процедуру резервного копіювання. Для забезпечення таємності система резервного копіювання (backup) також повинна мати власні TPM модулі.

Зломи

На конференції по комп'ютерній безпеці Black Hat 2010 було оголошено про злом чипа Infineon SLE66 CL PE, виготовленого по специфікації TPM. [13] Даний чип використовується в комп'ютерах, устаткуванні супутникового зв'язку й ігрових приставок. Для злому використовувався електронний мікроскоп (вартістю біля $70000). Оболонка чипа була розчинена кислотою, для перехоплення команд були використані дрібні голки. В Infineon затверджують, що вони знали про можливість фізичного злому чипа. Борчерт (Borchert), віце-президент компанії, завірив, що дороге устаткування й технічна складність злому не представляє небезпеки для переважної більшості користувачів чипів.

3.2 Розробка структурної схеми

Разом з мікросхемою TPM (Trusted Platform Module), що встановлюється на материнській платі, BitLocker шифрує весь жорсткий диск комп'ютера. Причому при цьому використовується класична модель двофакторної автентифікації (пароль, збережений у TPM, і PIN-код для доступу до мікросхеми). Така схема досить стійка до злому, хоча, у майбутньому безсумнівно з'являться атаки на PIN-код шляхом простого перебору (метод «грубої сили», brute force).

Принцип дії BitLocker™ Drive Encryption досить простий. Дане програмне забезпечення являє собою механізм шифрування по алгоритму Advanced Encryption Standard (AES) з 128 або 256-розрядним ключем. Це ПЗ інтегроване з

КПЗ_2023

мікросхемою TPM версії 1.2. Разом з тим, дане ПЗ можна використовувати й на комп'ютерах, не оснащених TPM, однак для доступу до системи необхідно застосувати USB-пам'ять або пароль відновлення. Але в такому випадку ступінь захищеності буде залежати насамперед від самого користувача. Саме такий варіант використання BitLocker(без TPM) ми й розглянемо.

Установка BitLocker Для установки BitLocker нам необхідно розмітити жорсткий диск відповідно до запропонованих вимог, а саме:

1. Створити новий Primary розділ довжиною в 1.5 Gb.

2. Призначити цей розділ активним.

3. Створити інший Primary розділ з місця, що залишилося, на жорсткому диску.

4. Відформатувати обидва розділи використовуючи NTFS.

5. Інсталювати Windows 10/11 на більший з розділів.

Для цього необхідно:

1. Завантажити комп'ютер, використовуючи інсталяційний DVD Windows 10/11.

2. На екрані установки вибрати ваша мова інсталяції.

3. Далі, вибрати System Recovery Options.

4. У вікні System Recovery Options переконатися, що не обрано операційну систему. Для цього вибрати порожню область у списку операційних систем і потім нажати Next.

5. Далі вибрати Command Promt і використовувати утиліту diskpart для створення роздягнула. Для цього в командному рядку набрати diskpart.

6. Вибрати disk 0 (select disk0).

7. Використовувати команду clean для видалення існуючих розділів на диску.

8. Create partition primary size=1500. Створити перший розділ на диску й призначити його primary.

9. Assign letter=D призначити даному розділу букву D.

КБПЗ-2023

10. Active призначити даний розділ активним.

11. Assign letter=C призначити даному розділу букву C.

12. List volume переглянути всі розділи на диску..

13. Нажати Exit для виходу з утиліти diskpart.

14. Відформатувати розділ З format c: /y /q /fs:NTFS.

15. Відформатувати розділ D format d: /y /q /fs:NTFS.

16. Exit.

17. У вікні System Recovery Options нажати Alt+F4 для повернення в головне вікно інсталяції.

18. Установити Windows 10/11 на більший з розділів.

Однак якщо ви вважаєте, що на цьому етап підготовки до установки закінчений, те повинен помітити, що ми тільки підійшли до середини.

Для установки BitLocker на комп'ютер, не обладнаний TPM, або обладнаний TPM із версією нижче 1.2 вам необхідно ще зробити деякі дії за допомогою групових політик.

1. Необхідно вибрати Start і набрати gpedit.msc у рядку Start Search, а потім нажати Enter.

2. В Group Policy Object Editor виберіть Local Computer Policy-Administrative Templates-Windows Components-BitLocker Drive Encryption.

3. Двічі кликніть на Control Panel Setup: Enable Advanced Startup Options.

4. Виберіть Enabled, потім позначте Allow BitLocker without a compatible TPM.

5. Закрийте Group Policy Object Editor для застосування змін групової політики натисніть Start, і введіть gpupdate.exe /force у рядку Start Search. Почекайте застосування групових політик.

6. Для того щоб відкрити BitLocker Drive Encryption, включите Turn On BitLocker.

КБПЗ_2023

7. На сторінці Set BitLocker Startup Preferences, виберіть Require Startup USB Key at every startup. Дана опція доступна лише при відсутності TPM. USB-ключ повинен бути вставлений щораз коли ви завантажуєте комп'ютер.

8. У діалозі Save your Startup Key укажіть як місце зберігання вашого ключа ваш USB-флеш диск і натисніть Save.

9. На сторінці Save the recovery password виконаєте наступні дії:

– Save the password on a USB drive Збережете ваш пароль відновлення в текстовий файл на USB флеш-диск.

– Save the password in a folder Збережете ваш пароль відновлення в текстовий файл на інший пристрій, мережне чи ні.

– Print the password роздрукуйте пароль відновлення.

Пароль відновлення повинен зберігатися в захищеному місці щоб уникнути його компрометації.

10. На сторінці Encrypt the selected disk volume page, необхідно відзначити вибрати Run BitLocker System Check і потім нажати Continue. Після цього необхідно перезавантажити комп'ютер, щоб перевірити що ваш комп'ютер сполучимо з BitLocker, а потім почати процес шифрування.

11. Якщо перевірка пройшла нормально, то ви побачите рядок стану Encryption in Progress. Після закінчення цієї процедури ваш диск буде зашифрований.

Відновлення даних, зашифрованих за допомогою BitLocker

Відновлення даних може знадобитися в наступних ситуаціях:

– Помилка читання TPM.

– Завантажувальний запис модифікований.

– TPM виключений і комп'ютер виключений.

– Уміст TPM стертий і комп'ютер виключений.

Якщо комп'ютер заблокований, процес відновлення буде досить простий, так як операційна система вже запущена. Ви можете відновити пароль або записавши його з іншого носія, на якому ви попередньо його зберегли на USB

КБПЗ-2023

флеш диск або використовуючи функціональні клавіші F 1-F10, де F 1-F9 відповідають цифрам 1-9, а F10 – 0.

Процес відновлення включає два кроки:

– Тестування відновлюваних даних.

– Відновлення доступу до даних, зашифрованим за допомогою BitLocker Drive Encryption.

Тестування відновлюваних даних

Коли ви перезапускаєте комп'ютер, вам буде потрібно пароль відновлення, так як конфігурація запуску змінилася, з тих пор як ви зашифрували диск.

1. Натисніть Start – All Programs – Accessories – Run.

2. Для запуску TPM Management Console наберіть tpm.msc у вікні, що відкрилося.

3. У рядку Actions виберіть Turn TPM Off.

4. Якщо буде потреба, уведіть PIN-код TPM.

5. Закрийте всі вікна.

6. Якщо USB флеш диск, що містить ваш пароль відновлення, підключений до комп'ютера, використовуйте Safely Remove Hardware для безпечного видалення диска з комп'ютера.

7. Потім виключите комп'ютер.

Відновлення доступу до даних, зашифрованим за допомогою BitLocker Drive Encryption:

1. Включите комп'ютер.

2. Якщо комп'ютер заблокований, з'явиться вікно BitLocker Drive Encryption Recovery Console.

3. Вам буде запропоновано вставити USB флеш-диск, що містить пароль відновлення:

– Якщо у вас є USB флеш-диск, що містить пароль відновлення, вставте його й натисніть Esc. Ваш комп'ютер буде автоматично перезавантажений і вам не буде потрібно вводити пароль відновлення вручну.

КБПЗ – 2023

– Якщо у вас немає USB флеш-диска, що містить пароль відновлення, натисніть Enter.

– Вам буде запропоновано ввести пароль відновлення вручну.

– Якщо ви знаєте пароль відновлення, уведіть його вручну й натисніть Enter.

– Якщо ви не знаєте пароля відновлення, натисніть Enter двічі й виключите комп'ютер.

– Якщо Ви зберегли ваш пароль відновлення у файлі, що перебуває в папці, на іншому комп'ютері або на змінному носії, ви зможете використовувати інший комп'ютер для прочитання файлу, що містить пароль. Для того, щоб довідатися ім'я файлу, що містить пароль, запишіть ідентифікатор Пароля на дисплеї пульта відновлення на блокованому комп'ютері. Файл, що містить ключ відновлення, використовує цей ідентифікатор, як ім'я файлу.

Вимикання BitLocker Drive Encryption

Дана процедура однакова як для комп'ютерів, обладнаних TPM, так і для комп'ютерів без TPM.

При вимиканні BitLocker, ви можете вибрати або тимчасово відключити BitLocker або розшифрувати весь диск. Відключення BitLocker дозволяє замінити TPM або провести відновлення операційної системи. Якщо ж ви вирішите розшифрувати весь диск, то для повторного зашифровування ви повинні будете повторно генерувати нові ключі й повторити процес шифрування знову.

Для вимикання BitLocker ви повинні зробити наступні дії:

1. Start – Control Panel – Security – BitLocker Drive Encryption.

2. На сторінці BitLocker Drive Encryption, знайдіть том, на якому ви хочете відключити BitLocker Drive Encryption і виберіть Turn Off BitLocker Drive Encryption.

3. У діалоговому вікні What level of decryption do you want виберіть Disable BitLocker Drive Encryption або Decrypt the volume.

КОПЗ – 2023

4. По закінченні цієї процедури, ви або відключите BitLocker Drive Encryption або розшифруєте том.

Використання BitLocker Як показало невелике дослідження, процедура використання BitLocker досить прозора, однак саме використання породжує багато питань, починаючи від стандартно-поліцейських, типу, а що буде якщо цю процедуру візьмуть на озброєння злочинці, адже тоді поліція не зможе прочитати вміст їхніх жорстких дисків і закінчуючи більше серйозними, по зберіганню ключів шифрування.

Зберігання ключів з використанням TPM

Якщо зберігання ключів шифрування використовуючи TPM і PIN-код питань не викликає (фактично ми маємо двофакторну автентифікацію), то от у випадку використання зберігання ключів у TPM без PIN-коду виникає питання. Ми завантажуємо комп'ютер, що нічого в нас не запитує. Тобто у випадку влучення вашого комп'ютера в руки зловмисника, він зможе скористатися вашим комп'ютером практично так само, як якби він взагалі не був зашифрований. Єдине, від чого при такому варіанті зберігання ключів захищає шифрування, так це від несанкціонованого прочитання даних у випадку здачі в ремонт жорсткого диска (без самого комп'ютера) і від компрометації інформації у випадку крадіжки жорсткого диска (без самого комп'ютера). Зрозуміло, що підвищеною безпекою такий режим назвати досить складно.

Зберігання ключів на USB-диску В цьому випадку ми маємо USB-диск, відчужуваний від власника, а так як він потрібний лише на момент старту системи, то на користувача накладається ще більше обов'язків по зберіганню самого пристрою.

У випадку, якщо користувач у ході діалогу зберіг запропонований пароль на USB-диск, то на диску ми маємо два файли. Один, схований, з розширенням BEK, а другий – текстовий файл.

КОПІЯ 2023

Уміст текстового файлу:

```
The recovery password is used to recover the data on a BitLocker protected drive.
Recovery Password:
521345-324918-713900
To verify that this is the correct recovery password compare these tags with tags presented on the recovery screen.
Drive Label: VLAD-PC C: 06.02.2007.
Password ID: {980CBD6 B-0A2 C-4187-9A1 C-BC232450654A}.
```

На жаль, але більшість користувачів не подумає про те, що даний текстовий файл необхідно просто видалити з USB-диска, попередньо його роздрукувавши.

USB-диск нам потрібний лише на етапі старту системи, отже, можливий сценарій при якому користувач залишає флеш-диск у комп'ютері й іде, припустимо, курити. Що заважає його сусідові взяти цю флешку, переписати необхідні файли (або роздрукувати потім текстовий файл) і скористатися цими знаннями зі злим наміром? Ніхто! Більше того, так як логування такого процесу не ведеться (адже немає ні ім'я ні PIN-коду), те довести факт незаконного використання буде досить складно. Мені можуть заперечити, що мати тільки код BitLocker – це ще досить мало. Згодний. Однак так як його використання вселяє користувачеві почуття помилкової захищеності, те цілком імовірно, що в такому випадку користувач буде менш строгий до вибору пароля.

Так, ви праві, це всього лише допущення. Однак ці допущення мають право на життя, отже, їх не можна скидати з рахунку.

Використовувати BitLocker звичайно ж потрібно. Однак, на мій погляд, найбільш прийнятним варіантом його використання є застосування даного ПЗ на комп'ютерах, обладнаних TPM версії не нижче 1.2 у сполученні з PIN-кодом.

Структурна схема системи зображена на рисунку 3.1. Робота системи відбувається наступним чином. BitLocker Drive Encryption працює, управляючи кожним файлом з метою збереження конфіденційності, за допомогою алгоритму симетричного управління файлами з метою збереження конфіденційності, що залежить від версії операційної системи й настроювань. При цьому

КОПЗ_2023

використовується випадково згенерований ключ для кожного файлу, називаний File Encryption Key (FEK), вибір симетричного управління файлами з метою збереження конфіденційності, на даному етапі пояснюється його швидкістю й більшою надійністю стосовно асиметричного управління файлами з метою збереження конфіденційності.

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

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

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

КБПЗ-2023

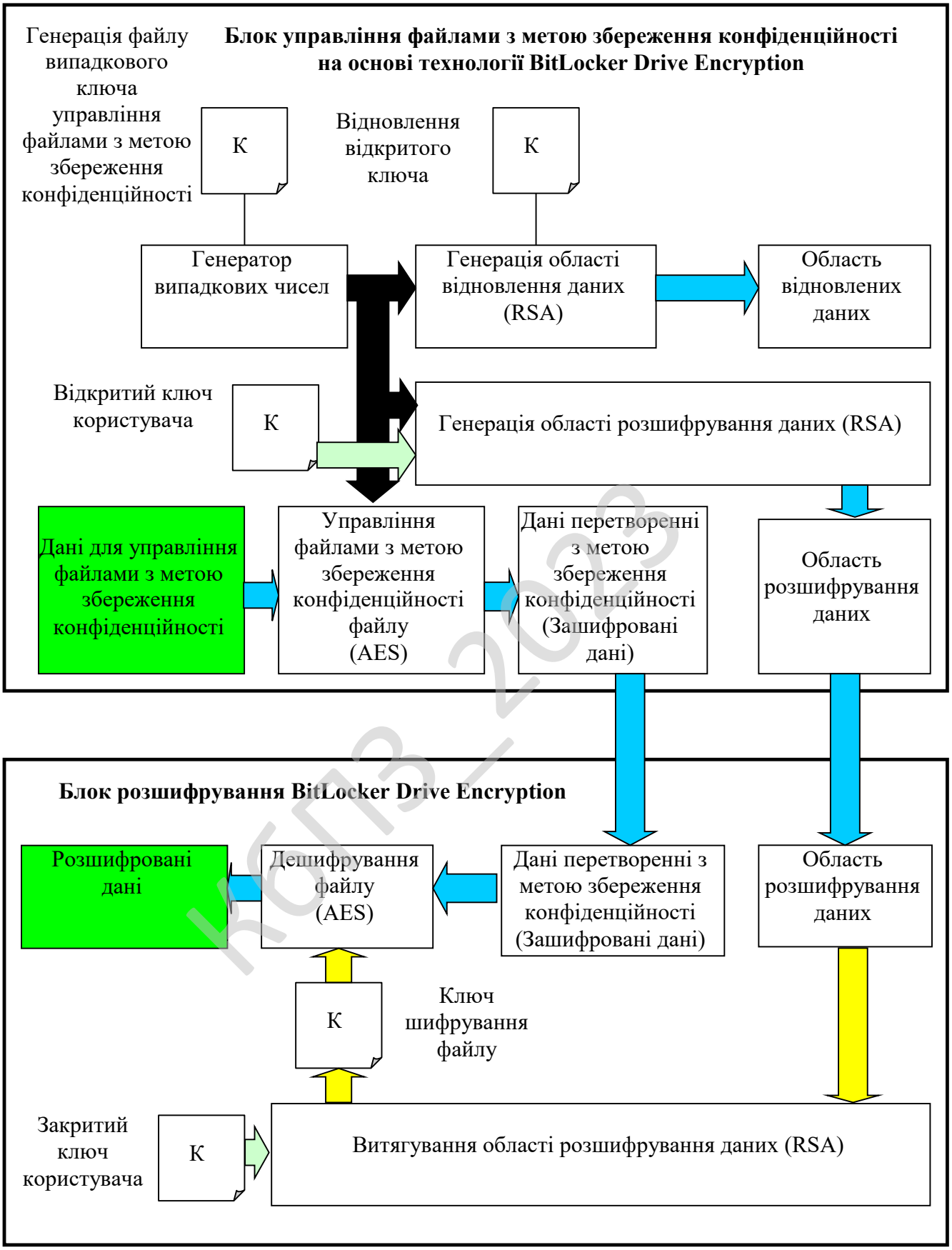


Рисунок 3.1 – Структурна схема системи

КПЗ 2023

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

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

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

Шифрувати можна як окремі файли, так і цілі папки, при цьому якщо шифрується папка вже утримуюча файли, тобто можливість вибору, шифрувати тільки папку або папку й вкладені файли. Управління файлами з метою збереження конфіденційності, папки не означає що інші користувачі не зможуть переглядати вміст папки – вони лише не зможуть відкривати файли, управління

КБПЗ_2023

якими реалізовано з метою збереження конфіденційності. Всі нові файли, збережені в зашифрованій папці або скопійовані в неї будуть автоматично зашифровані. Шифрувати папки більш зручно, і крім того безпечно, оскільки BitLocker Drive Encryption має схему відновлення після аварійного збою (наприклад якщо під час операції управління файлами з метою збереження конфіденційності, відбулася критична помилка) яка передбачає створення незашифрованої архівної копії вихідного файлу – при успішному завершенні операції управління файлами з метою збереження конфіденційності, архівна копія віддаляється, але може бути відновлена спеціальними програмами відновлення віддалених даних, що створює потенційну погрозу інформаційної безпеки. А при збереженні файлу в зашифрованій папці управління файлами з метою збереження конфіденційності, відбувається без створення такої резервної копії. Якщо все-таки шифрувалися одиночні файли, тобто можливість перезаписати кластери, що залишилися після зміни або видалення файлів на томах NTFS випадковими значеннями – для цього може бути використана команда cipher /w: шлях запущена з командного рядка або програми сторонніх розроблювачів.

3.3 Розробка функціональної схеми

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

Win64 API

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

КОПЗ_2023

Бібліотека часу виконання BitLocker Drive Encryption (FSRTL)

FSRTL – це модуль усередині драйвера BitLocker Drive Encryption, що здійснює зовнішні виклики NTFS для виконання різних операцій файлової системи, таких як читання, запис, відкриття зашифрованих файлів і каталогів, а також операцій управління файлами з метою збереження конфіденційності, дешифрування, відновлення даних при записі на диск і читанні з диска. Незважаючи на те, що драйвер BitLocker Drive Encryption 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 BitLocker Drive Encryption (DDF і DRF) і передачу обчислених в BitLocker Drive Encryption ключів FEK у бібліотеку FSRTL, тому що ці ключі повинні встановлюватися в контексті відкриття файлу. Такий контекст відкриття файлу дозволяє потім здійснювати непомітне управління файлами з метою збереження конфіденційності, й дешифрування при записі й зчитуванні файлів з диска.

Драйвер BitLocker Drive Encryption

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

Служба BitLocker Drive Encryption

Служба BitLocker Drive Encryption 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 BitLocker Drive Encryption.

КОПЗ_2023

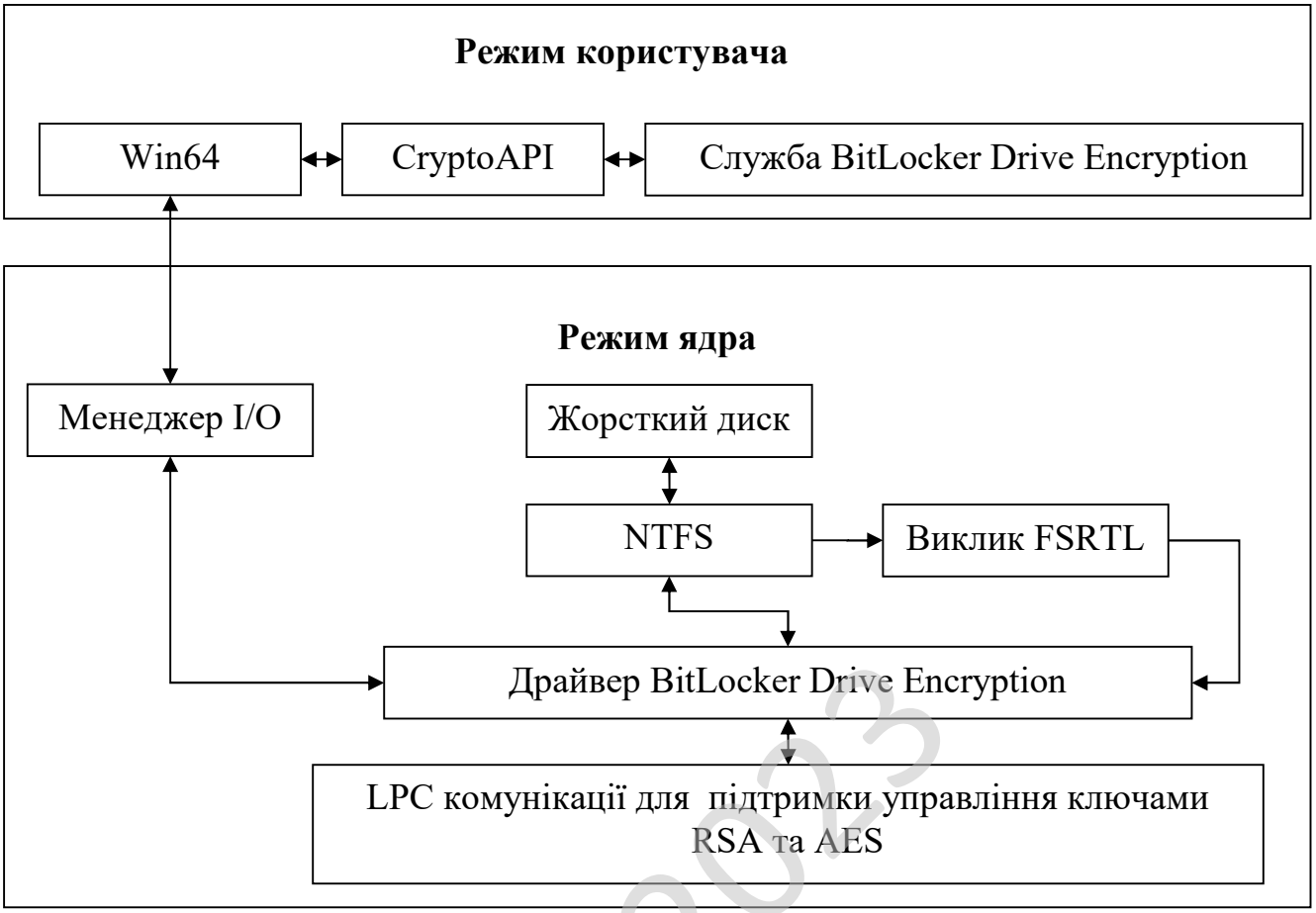


Рисунок 3.2 – Функціональна схема системи

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

КПЗ – 2023

для керування периферійними контролерами, які підключалися до шини ISA.

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

RSA

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

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

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

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

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

1. Можливість знайти значення e, d і n такі, що $M^{ed} = M \bmod n$ для всіх $M < n$.
2. Відносна легкість обчислення M^e й C^d для всіх значень $M < n$.
3. Неможливість визначити d, знаючи e та n.

Створення ключів

Вибрати прості p та q.

Обчислити $n = p \cdot q$.

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

Вибрати d взаємно просте з $\Phi(n)$: 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

КПЗ – 2023

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

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

Управління файлами з метою збереження конфіденційності,

Незашифрований текст: M < n.

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

Дешифрування

Зашифрований текст: C < n.

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

AES

Він є нетрадиційним блоковим шифром, оскільки не використовує мережу Фейштеля для криптоперетворень. Алгоритм представляє кожний блок кодуємих даних у вигляді двовимірного масиву байт розміром 4x4, 4x6 або 4x8 залежно від установленої довжини блоку. Далі на відповідних етапах перетворення відбуваються або над незалежними стовпцями, або над незалежними рядками, або взагалі над окремими байтами в таблиці.

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

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

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

КПЗ – 2023

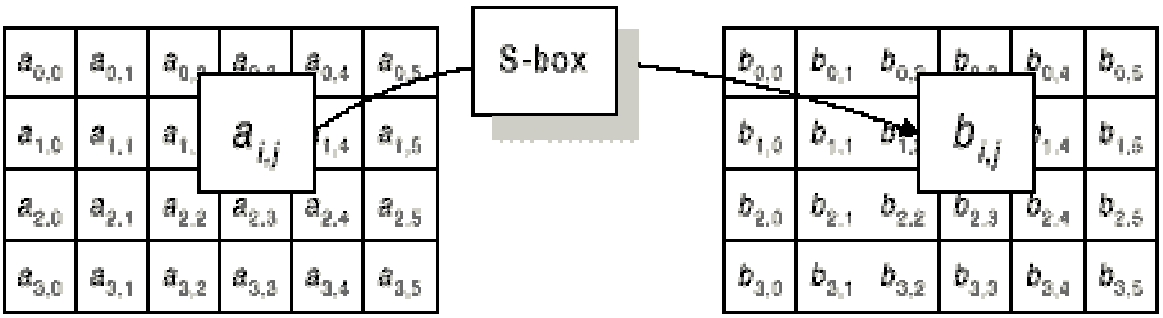


Рисунок 3.3 – Таблична підстановка 8x8 біт

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

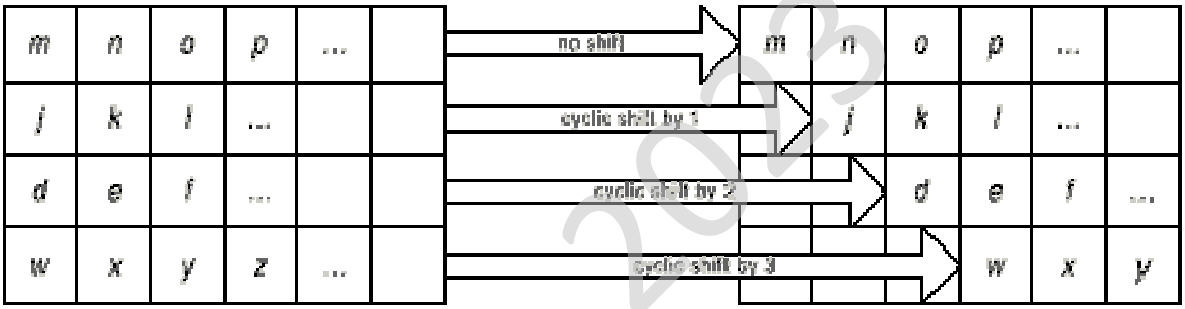


Рисунок 3.4 – Зрушення рядків у двовимірному масиві на різні зсуви

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

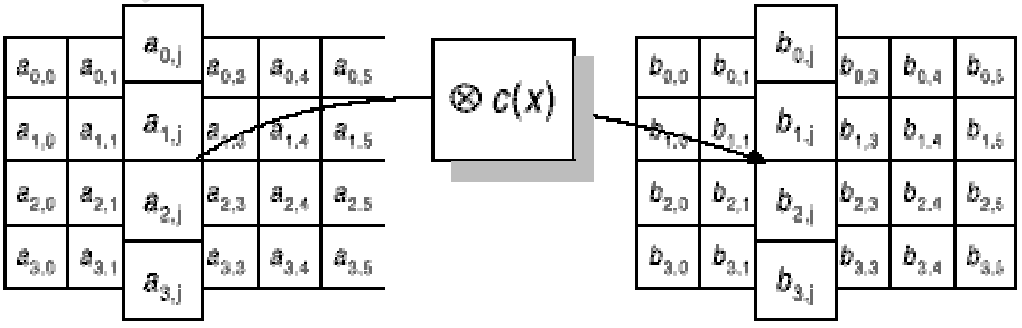


Рисунок 3.5 – Математичне перетворення, що перемішує дані усередині стовпця

КбПЗ_2023

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

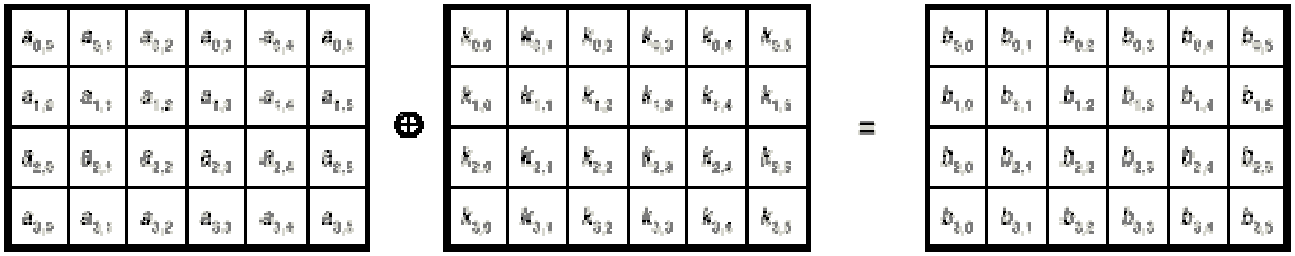


Рисунок 3.6 – Додавання матеріалу ключа операцією XOR

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 процесів розробленої системи зображена на рисунку 3.7 Після початку роботи розробленого ПЗ ми потрапляємо до головного блоку системи звідки через ланку дій відбувається наступне:

– Інтерфейс ПЗ.

– Моніторинг дискової підсистеми.

– Перегляд файлу.

– Вибір файлу.

– Перегляд властивостей файлу.

– Копіювання, переміщення, перейменування файлів.

– Шифрування вибраного файлу.

– Вибір ключів шифрування.

– Дешифрування вибраного файлу.

– Ключі шифрування підсистеми BitLocker Drive Encryption.

КПЗ_2023

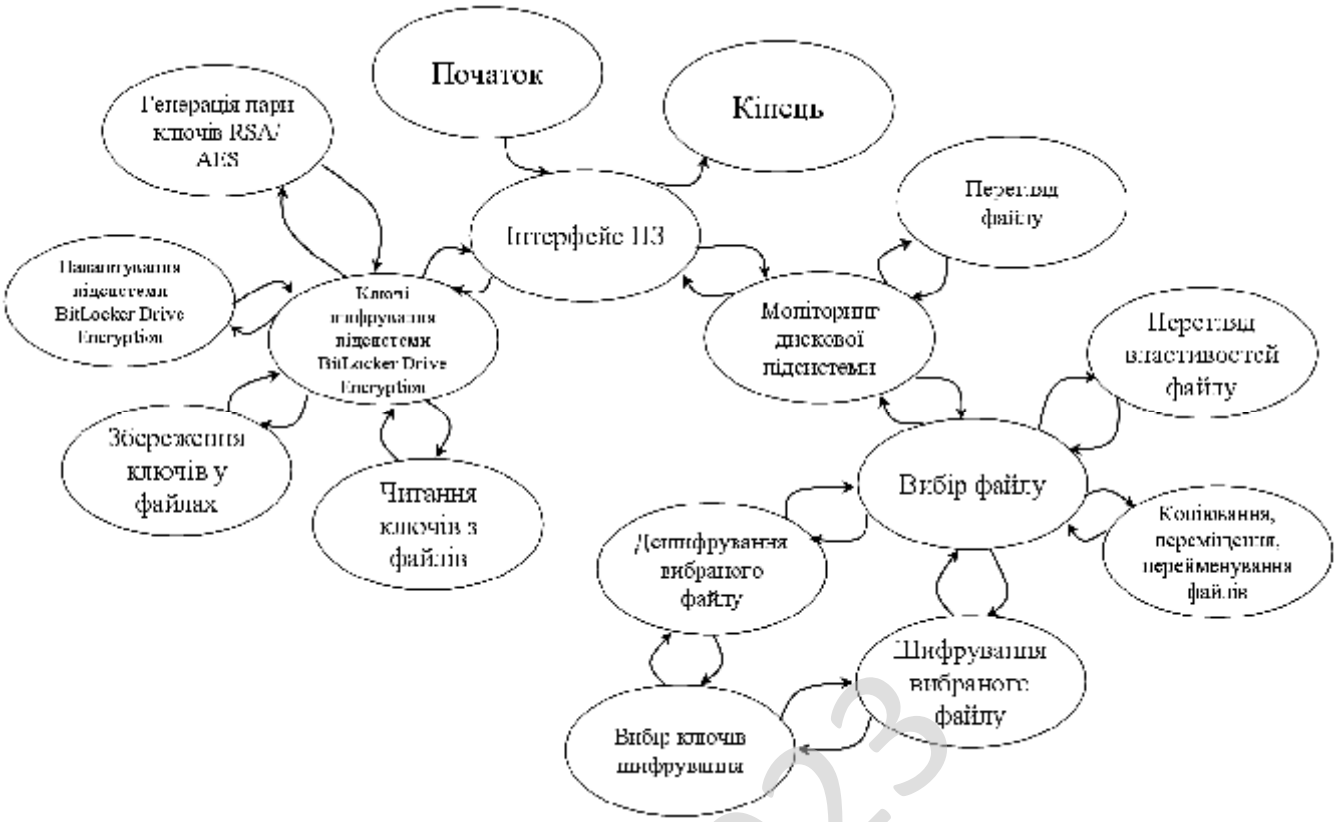


Рисунок 3.7– Діаграма взаємодії процесів

– Читання ключів з файлів.

– Збереження ключів у файлах.

– Налаштування підсистеми BitLocker Drive Encryption.

– Генерація пари ключів RSA/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.

КОПЗ_2023

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

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

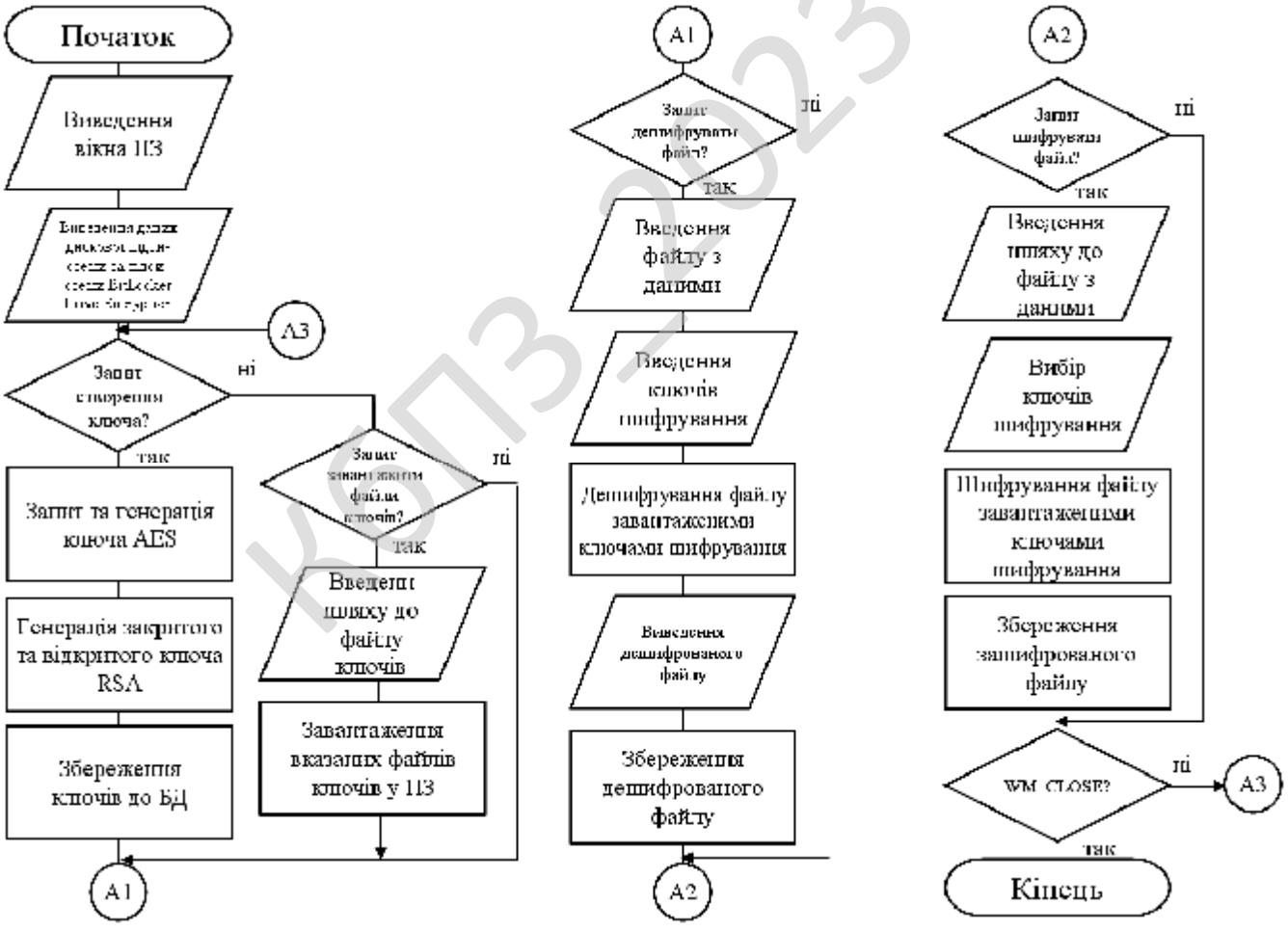
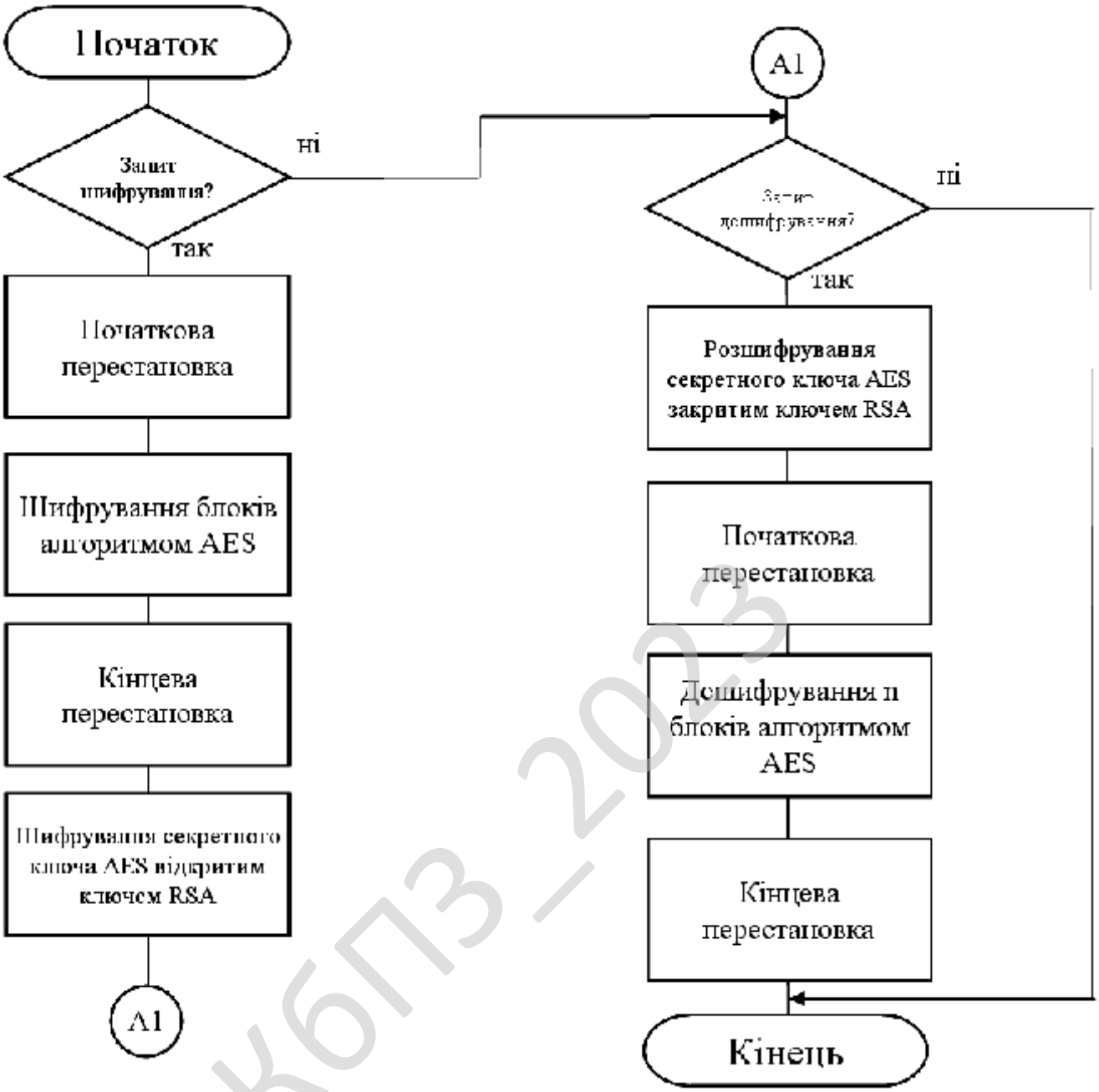


Рисунок 4.1 – Блок схема основної програми

КПЗ-2023



КбПЗ_2023

Рисунок 4.2 – Блок схема підпрограми

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

функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Опис алгоритмів функціонування системи

Розглянемо вихідний код реалізації шифрування алгоритмом RSA.

```
Procedure RSA_E(P : String; Var exp, modb : TFGInt; Var E : String);
Procedure RSA_D(E : String; Var exp, modb, d_p, d_q, p, q : TFGInt; Var D :
String);
Procedure RSA_S(M : String; Var d, n, dp, dq, p, q : TFGInt; Var S : String);
Procedure RSA_V(M, S : String; Var e, n : TFGInt; Var valid : boolean);
Implementation
{$H+}

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

КОПЗ_2023

```
   Begin
      tempstr3 := copy(tempstr1, 1, modbits);
      While (copy(tempstr3, 1, 1) = '0') And (length(tempstr3) > 1) Do
delete(tempstr3, 1, 1);
      Base2StringToFGInt(tempstr3, EGInt);
      delete(tempstr1, 1, modbits);
      If tempstr3 = '0' Then FGIntCopy(zero, temp) Else
      Begin
         If exp.Number <> Nil Then FGIntMontgomeryModExp(EGInt, exp, modb, temp)
Else
         Begin
            FGIntMontgomeryModExp(EGInt, d_p, p, temp1);
            FGIntMul(temp1, qqinvp, temp3);
            FGIntCopy(temp3, temp1);
            FGIntMontgomeryModExp(EGInt, d_q, q, temp2);
            FGIntMul(temp2, ppinvq, temp3);
            FGIntCopy(temp3, temp2);
            FGIntAddMod(temp1, temp2, modb, temp);
            FGIntDestroy(temp1);
            FGIntDestroy(temp2);
         End;
      End;
      FGIntDestroy(EGInt);
      tempstr3 := '';
      FGIntToBase2String(temp, tempstr3);
      While (length(tempstr3) Mod (modbits - 1)) <> 0 Do tempstr3 := '0' +
tempstr3;
      tempstr2 := tempstr2 + tempstr3;
      FGIntdestroy(temp);
   End;
   If exp.Number = Nil Then
   Begin
      FGIntDestroy(ppinvq);
      FGIntDestroy(qqinvp);
   End;
   While (Not (copy(tempstr2, 1, 3) = '111')) And (length(tempstr2) > 3) Do
delete(tempstr2, 1, 1);
   delete(tempstr2, 1, 3);
   ConvertBase2To256(tempstr2, D);
   FGIntDestroy(zero);
End;
// підписуємо рядок RSA, M^d mod n = S
```

КОПЗ_2023

```
// n = p*q, dp*e mod (p-1) = 1
// dq*e mod (q-1)
Procedure RSA_S(M : String; Var d, n, dp, dq, p, q : TFGInt; Var S : String);
Var
   MGInt, SGInt, temp, temp1, temp2, temp3, ppinvq, qqinvp : TFGInt;
Begin
   Base256StringToFGInt(M, MGInt);
   If d.Number <> Nil Then FGIntMontgomeryModExp(MGInt, d, n, SGInt) Else
   Begin
      FGIntModInv(p, q, temp);
      FGIntMul(p, temp, ppinvq);
      FGIntDestroy(temp);
      FGIntModInv(q, p, temp);
      FGIntMul(q, temp, qqinvp);
      FGIntDestroy(temp);
      FGIntMontgomeryModExp(MGInt, dp, p, temp1);
      FGIntMul(temp1, qqinvp, temp2);
      FGIntCopy(temp2, temp1);
      FGIntMontgomeryModExp(MGInt, dq, q, temp2);
      FGIntMul(temp2, ppinvq, temp3);
      FGIntCopy(temp3, temp2);
      FGIntAddMod(temp1, temp2, n, SGInt);
      FGIntDestroy(temp1);
      FGIntDestroy(temp2);
      FGIntDestroy(ppinvq);
      FGIntDestroy(qqinvp);
   End;
   FGIntToBase256String(SGInt, S);
   FGIntDestroy(MGInt);
   FGIntDestroy(SGInt);
End;
// Шифруємо рядок алгоритмом RSA, P^exp mod modb = E
Procedure RSA_E(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);
```

КбПЗ_2023

```
   tempstr1 := '111' + tempstr1;
   j := modbits - 1;
   While (length(tempstr1) Mod j) <> 0 Do tempstr1 := '0' + tempstr1;
   j := length(tempstr1) Div (modbits - 1);
   tempstr2 := '';
   For i := 1 To j Do
   Begin
      tempstr3 := copy(tempstr1, 1, modbits - 1);
      While (copy(tempstr3, 1, 1) = '0') And (length(tempstr3) > 1) Do
delete(tempstr3, 1, 1);
      Base2StringToFGInt(tempstr3, PGInt);
      delete(tempstr1, 1, modbits - 1);
      If tempstr3 = '0' Then FGIntCopy(zero, temp) Else
FGIntMontgomeryModExp(PGInt, exp, modb, temp);
      FGIntDestroy(PGInt);
      tempstr3 := '';
      FGIntToBase2String(temp, tempstr3);
      While (length(tempstr3) Mod modbits) <> 0 Do tempstr3 := '0' + tempstr3;
      tempstr2 := tempstr2 + tempstr3;
      FGIntdestroy(temp);
   End;
   While (tempstr2[1] = '0') And (length(tempstr2) > 1) Do delete(tempstr2, 1, 1);
   ConvertBase2To256(tempstr2, E);
   FGIntDestroy(zero);
End;
// Перевіряємо правильність алгоритму  RSA,
// Якщо M = S^e mod n тоді ok:=true інакше ok:=false
Procedure RSA_V(M, S : String; Var e, n : TFGInt; Var valid : boolean);
Var
   MGInt, SGInt, temp : TFGInt;
Begin
   Base256StringToFGInt(S, SGInt);
   Base256StringToFGInt(M, MGInt);
   FGIntMod(MGInt, n, temp);
   FGIntCopy(temp, MGInt);
   FGIntMontgomeryModExp(SGInt, e, n, temp);
   FGIntCopy(temp, SGInt);
   valid := (FGIntCompareAbs(SGInt, MGInt) = Eq);
   FGIntDestroy(SGInt);
   FGIntDestroy(MGInt);
End;
```

КОПЗ_2023

Розглянемо вихідний код, за допомогою якого відбувається шифрування/дешифрування алгоритмом AES.

```
Type
  TBitString = Array of Boolean;
  PBitString = ^TBitString;
  TSplitKeyParts = record
    C:TBitString;
    D:TBitString;
    end;
  TSplitKey = Array[0..16]Of TSplitKeyParts;
  TConcatKey = Array[0..15]Of TBitString;
  TIPKeyParts = record
    L:TBitString;
    R:TBitString;
    end;
  TIPKey = Array[0..16]OF TIPKeyParts;
//функції  перестановок
Function BinToInt(S:TBitString):Integer;
Function IntToBin(N:Integer;Precision:Integer=8):TBitString;
Function BinToStr(Bits:TBitString):String;
Function StrToBin(S:String):TBitString;
Function AnsiStrToBin(S:String; Zeroes:Boolean=True):TBitString;
Function BinToAnsiStr(Bits:TBitString):String;
Procedure CopyBits(Var Dest:TBitString; Source:TBitString; NBits:Integer);
Function ConcatBits(Bits:Array Of TBitString):TBitString;
Function DESEncode(S,Key:String):TBitString;
Function DESDecode(S,Key:String):TBitString;
Function GetPermutedKey(Key:TBitString):TBitString;
Function GetPermutedKey2(Key:TBitString):TBitString;
Function GetSplitKey(Key:TBitString):TSplitKey;
Function GetConcatKey(Key:TSplitKey):TConcatKey;
Function GetIPKey(M:TBitString; ConcatKey:TConcatKey):TIPKey;
Function GetF(R,K:TBitString):TBitString;
Function GetSBox(Index:Integer; T:TBitString):TBitString;
Function GetReverseIP(RL:TBitString):TBitString;
Procedure ReverseSubKeys(Var Keys:TConcatKey);
implementation
Function ConcatBits(Bits:Array Of TBitString):TBitString;
Var
I,C:Integer;
Begin
```

КПІ 2023

```
SetLength(Result,0);
For C:=0 To Length(Bits)-1 Do
  Begin
  SetLength(Result,Length(Result)+Length(Bits[C]));
  For I:=0 To Length(Bits[C])-1 Do
    Result[Length(Result)-Length(Bits[C])+I]:=Bits[C][I];
  End;
End;
Procedure CopyBits(Var Dest:TBitString; Source:TBitString; NBits:Integer);
Var
I:Integer;
Begin
SetLength(Dest,NBits);
For I:=0 To NBits-1 Do
  Dest[I]:=Source[I];
End;
Function BinToInt(S: TBitString): Integer;
Var
L,I:Integer;
Begin
Result:=0;
L:=Length(S);
IF L=0 Then
  Raise EConvertError.Create(спеціальна бітова строка довжиною 0 біт');
For I:=L-1 DownTo 0 Do
  Result:=Result+Ord(S[I])*Trunc(Power(2,L-I-1));
End;
Function IntToBin(N:Integer; Precision:Integer):TBitString;
Var
BitList:TList;
Bit:PBoolean;
Begin
SetLength(Result,0);
BitList:=TList.Create;
While N>0 Do
  Begin
  New(Bit);
  Bit^:=Boolean(N mod 2);
  BitList.Insert(0,Bit);
  N:=N div 2;
  End;
While BitList.Count<Precision Do
```

КОПЗ – 2023

```
  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
```

КбПЗ_2023

```
  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
```

КОПЗ_2023

```
  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);
```

КбПЗ_2023

```
  Temp:=ConcatBits([Key[I+1].C,Key[I+1].D]);
  Result[I]:=GetPermutedKey2(Temp);
  End;
End;
end.
```

4.2 Захист розробленого програмного забезпечення

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

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

Як вхід процесу шифрування використовується 64-бітовий блок відкритого тексту. Спочатку блок даних підлягає операції XOR з 64 бітами ключа. Потім блок даних розщеплюється на ліву і праву половини. Об'єднання лівої і правої половин за допомогою XOR утворює нову праву половину. Ліва половина і нова права половина проходять через N етапів (спочатку 4). На кожному етапі половина об'єднується за допомогою функції F[1] з 16 бітами ключа і за допомогою XOR – з лівою половиною, створюючи нову праву половину. Вихідна права половина (на початок етапу) стає новою лівою половиною. Після N етапів (ліва і права половини не переставляти після N-го етапу) ліва половина знову об'єднується з допомогою XOR з правою половиною, утворюючи нову праву половину, потім ліва і права об'єднуються разом в 64-бітове ціле. Блок даних об'єднується за допомогою XOR з іншими 64 бітами ключа і алгоритм завершується.

КБПВ_2023

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ яке зображено на рисунку 5.1. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні розділи:

– Розділ меню який складається із наступних елементів: Файл; Шифрування; Довідка.

– Розділ відображення файлової структури: Вікно із деревом каталогів; Вікно із відображенням вмісту поточного каталогу; Панель вкладок для швидкого доступу до дисків.

– Розділ кнопок швидкого доступу до елементів керування: Генерація ключів AES; Генерація ключів RSA; Шифрувати вибраний файл; Дешифрувати вибраний файл.

– Розділ вікон відображення інформації про вибраний об’єкт.

КБПЗ – 2023

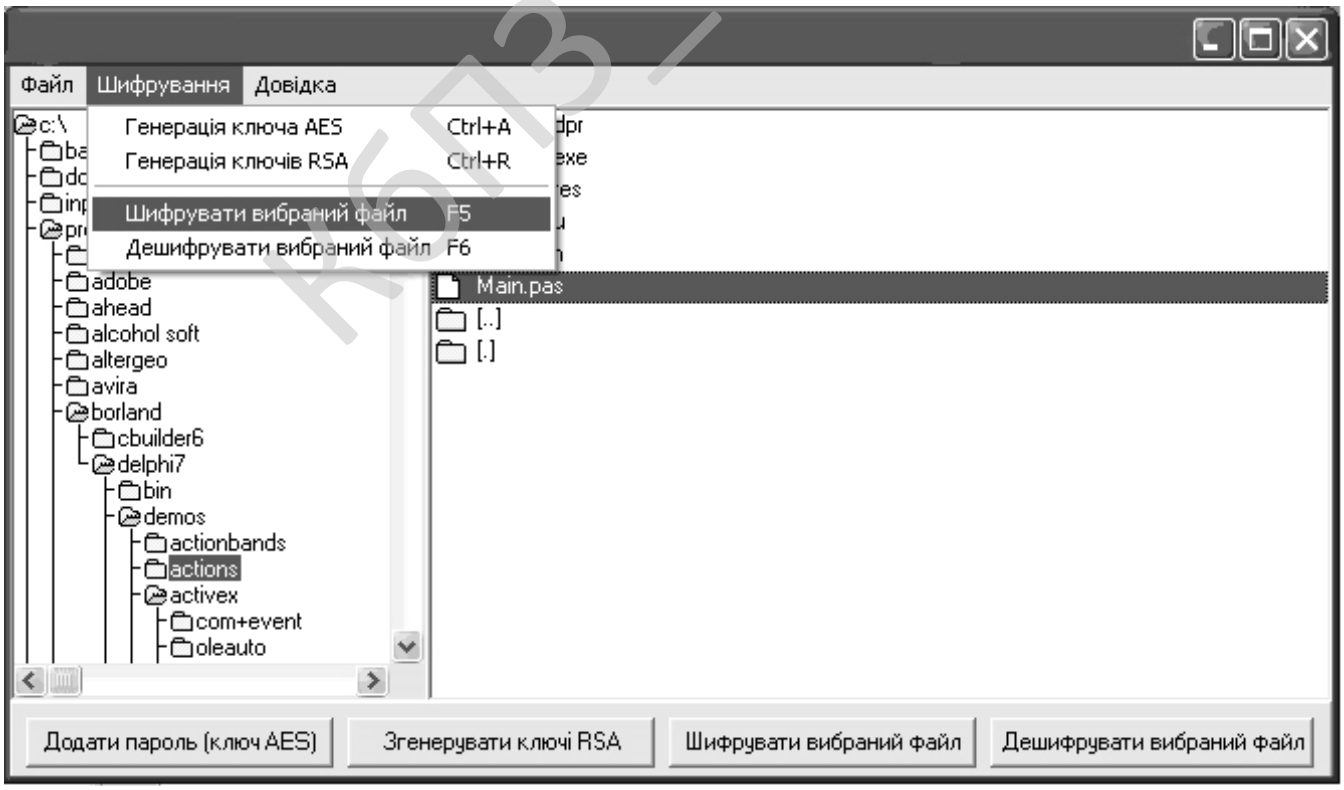


Рисунок 5.1 – Головне вікно ПЗ

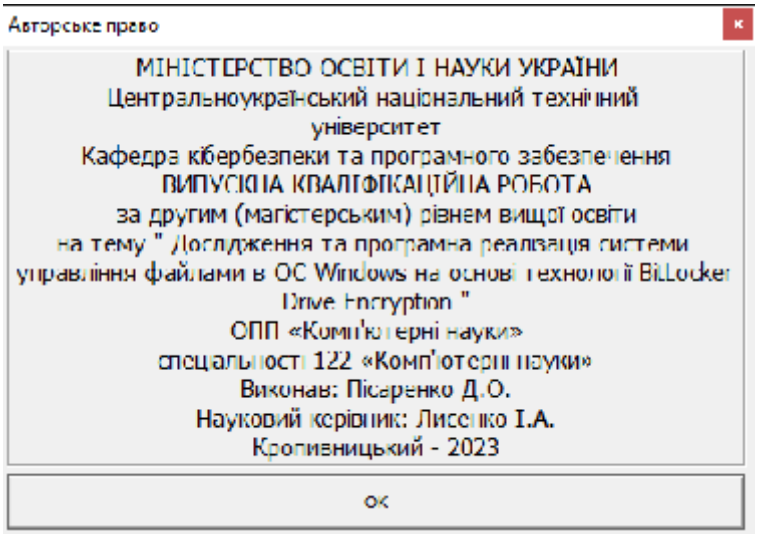


Рисунок 5.2 – Авторське право

Обрано умови розповсюдження – proprietary software. Програмне забезпечення, на яке зберігаються як немайнові, так і майнові авторські права. Отримавши або придбавши таке програмне забезпечення, користувач отримує обмежені права користування ним: може бути заборонено або закрито доступ до коду (вивчення), внесення змін, тиражування, розповсюдження та перепродаж. Програмне забезпечення вважається власницьким, якщо наявне хоча б одне з перелічених обмежень.

Власницьке програмне забезпечення та комерційне програмне забезпечення не є синонімами – власницьким може бути і безплатне (тобто, некомерційне) програмне забезпечення.

На противагу власницькому ПЗ існує вільне програмне забезпечення, автори і власники якого дозволяють вивчати, модифікувати і поширювати свій продукт. Саме визначення власницького програмного забезпечення виникло в результаті діяльності громадського руху вільного програмного забезпечення (представленого Фондом вільного програмного забезпечення та іншими організаціями) і осмислення умов свободи користування програмами.

КПЗ_2023

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи управління файлами в ОС Windows на основі технології BitLocker Drive Encryption.

Метою розробки є дослідження та програмна реалізація системи управління файлами в ОС Windows на основі технології BitLocker Drive Encryption.

Об'єктом дослідження є процес управління файлами в ОС Windows на основі технології BitLocker Drive Encryption.

Предметом дослідження є методи управління файлами в ОС Windows на основі технології BitLocker Drive Encryption.

Методи дослідження базуються на методах захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод управління файлами в ОС Windows на основі технології BitLocker Drive Encryption.

– Розроблено вітчизняний продукт управління файлами в ОС Windows на основі технології BitLocker Drive Encryption, який має більш широкі можливості, на відміну від існуючих аналогів.

КОПЗ-2023

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 60 днів (три місяці).

В магістерській роботі було проведено дослідження та виконана програмна реалізація системи управління файлами в ОС Windows на основі технології BitLocker Drive Encryption.

Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликі системні потреби;

б) незалежність від встановлених на комп’ютері баз даних;

в) зручність у користуванні та надійність.

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт.	N	1
2. Кількість екземплярів програм, шт.	Ne	19
3. Запланований термін розробки, днів	Fpq	60 (3 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2

КОПЗ_2023

Продовження таблиці 7.1

1	2	3
7. Кількість макетів вхідної інформації	–	3
8. Кількість форм вихідної інформації.	–	4
9. Мова програмування (1-6)	–	2
10. Попередній досвід (1-6)	–	3
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	–	2
13. Рівень спрацьованості колективу (1-6)	–	2
14. Ступінь вимірності процесів (1-6)	–	3
15. Необхідна надійність програмного забезпечення (1-6)	–	2
16. Розмір бази даних (порівняно з розміром програми) (1-6)	–	2
17. Складність кінцевого програмного продукту (1-6)	–	2
18. Необхідний рівень забезпечення повторного використання (1-6)	–	2
19. Документованість відповідно до планованого життєвого циклу (1-6)	–	2
20. Вимоги до швидкодії ПП (1-6)	–	2
21. Обмеження на розміри основного сховища даних (1-6)	–	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	–	2
23. Професійний рівень аналітиків (1-6)	–	2
24. Професійний рівень програмістів (1-6)	–	2
25. Постійність складу команди розробників (1-6)	–	2
26. Досвід розробки додатків (1-6)	–	2
27. Досвід роботи з обчислювальною платформою (1-6)	–	2

КПІ_2023

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	19000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	50
38. Ставка податку на додану вартість, %	Ндв	20

КОПЗ - 2023

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначаємо по типовим нормам часу приведеним в додатках МВ. Стадія РП є найбільш тривалою і трудомісткою, що робить значний вплив на інші стадії проекту.

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

$$T_{ном} = A \, \text{Size}^B, \qquad (7.1)$$

де: A – коефіцієнт Боема, A = 2,45;

Size – загальний об'єм відлагодженого програмного коду, тис. рядків;

B – показник ступеня, що визначається співвідношенням:

$$B = 1{,}01 + 0{,}001\sum W_i, \qquad (7.2)$$

де: *Wi* – сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

$$B = 1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}027.$$

$$T_{ном} = 2{,}45 \cdot 2{,}7^{1{,}026} = 6{,}78 \text{ люд-міс.}$$

Визначаємо уточнені (з урахуванням приведених в МВ додатку 3 сімнадцяти додаткових коефіцієнтів) трудовитрати, люд-міс.:

$$T_{уточн} = T_{ном}\, ПВ_j, \qquad (7.3)$$

де: $ПВ_j$ – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн} = 6{,}78 \cdot (0{,}88 \cdot 0{,}93 \cdot 0{,}88 \cdot 0{,}91 \cdot 0{,}95 \cdot 1 \cdot 1 \cdot 0{,}87 \cdot 1{,}22 \cdot 1{,}16 \cdot 1{,}1 \cdot 1{,}1 \cdot$$
$$\cdot 1{,}12 \cdot 1{,}1 \cdot 1 \cdot 1{,}1 \cdot 1{,}1) = 9{,}37 \text{ люд-міс.}$$

Ці коефіцієнти дозволяють диференційовано оцінювати результати роботи програмістів, беручи до уваги швидкодію програми, використання різноманітних обчислювальних платформ і інструментів розробки, взаємодію декількох серверів, вимоги до об'ємів баз даних і ін.

Визначаємо підсумкові трудовитрати по стадії робочий проект, люд-дні:

$$T_{РП} = 0{,}3 C T_{уточн}^{0{,}33+0{,}2(B-1{,}01)} S, \qquad (7.4)$$

де: *C* – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4). Для мови програмування Delphi він складає 2,66;

S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПЗ згідно встановленим вимогам. Вибираємо в межах (25…350)%. Приймаємо *S* =100 %

$$T_{РП} = 0{,}3 \cdot 2{,}66 \cdot 9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)} \cdot 100 = 168 \text{ люд/день.}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.2.

КОПЗ–2023

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	9	Д7
Робочий проект	168	Ф 7.1-7.4
Впровадження	13	Д13
Всього	209	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Чисельність ставок інженерів-програмістів для розробки програмного забезпечення визначається за формулою:

$$Ч = \frac{T_{пз} N}{F_{рq} - H_{ев}}, \quad (7.5)$$

де: $F_{рq}$ – плановий фонд робочого часу одного спеціаліста, днів;

$T_{пз}$ – трудомісткість розробки програмного забезпечення люд-дні.

$$Ч = \frac{209 \cdot 1}{60 - 5} = 3,8 \text{ ставки.}$$

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року.

Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3.

КПЗ_2023

Таблиця 7.3 – Затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за розрахунковий період

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	385	12	4620	77
Монітор	160	12	1920	32
Клавіатура	140	12	1680	28
Маніпулятор «мишка»	30	12	360	6
Принтер матричний	185	1	185	3
Принтер лазерний	355	2	710	12
Принтер струминний	300	1	300	5
Сканер	155	2	310	5
Концентратор-маршрутизатор	155	2	310	5
Кабельні господарства ЛОМ на 1 м.п.	2,5	100	250	4
Кабельне господарство електромережі	48	50	2400	40
Копіювальний апарат	285	2	570	10
Усього за рік:			$З_ч$	227

КбПЗ 2023

Час на профілактику обладнання в загальному балансі робочого часу інженерів-електронщиків неповинен складати більше 10%.

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф_{др}^{с} = \frac{З_ч \cdot n_{міс}}{1,2}, \quad (7.6)$$

$$Ф_{др}^{с} = \frac{227 \cdot 3}{1,2} = 567,5 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф^{с}_{др}}{F_{др} \cdot T_{зм}}, \quad (7.7)$$

$$Ч_{ел} = 567,5/(60 \cdot 8) = 1,2 \text{ ставки.}$$

Для забезпечення нормального технічного обслуговування засобів ТО та мереж, необхідно прийняти найбільше ціле значення розрахункової чисельності інженерів-електронщиків.

Чисельність інженерів-системотехніків, адміністраторів мережі, дизайнерів WEB вузлів, системних програмістів (аналітиків), бухгалтерів-економістів визначається за потребою в залежності від функціональних обов'язків. Після визначення чисельності персоналу складається штатний розклад.

Таблиця 7.4 – Розрахунок чисельності штатного персоналу сектору системного та адміністративного обслуговування засобів ОТ та комп’ютерних мереж

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 2019, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	2	0,5
	Налаштування і конфігурування базової станції безпровідного зв’язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв’язку між віддаленими об’єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв’язку клієнтів до мережі Інтернет	1	
Всього		4	

КПЗ_2023

Продовження таблиці 7.4

Посада	Вид роботи	Час	К-ть штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	1	0,25
	Підтримка постійних клієнтів	0,5	
	Оформлення договорів, ведення тендерів	0,25	
	Контроль взаєморозрахунків з постачальниками	0,25	
Всього		2	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	1	0,25
	Створення графічних і стилістичних елементів сайту	0,5	
	Оформлення банерів і промо-сторінок	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,25
	Верстка друкованих видань	0,5	
	Додрукова підготовка макетів	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	

КОПЗ_2023

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	1	13000	39000
Продакт-менеджер	0,25	12000	9000
Інженер-програміст	3,8	12000	136800
Інженер-електронщик	1,2	11500	41400
Інженер-системотехнік	0,25	11500	8625
Адміністратор мережі	0,5	11500	17250
Системний програміст	0,25	11500	8625
Дизайнер WEB	0,25	12000	9000
Інженер-верстальник	0,25	11700	8775
Бухгалтер-економіст	0,5	12500	18750
Всього за період розробки	$R_{сп}$ = 8,25	-	$Ф_{роб}$ = 297225

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рд}}, \quad (7.8)$$

де: $Ф_{роб}$ – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{297225}{8,25 \cdot 60} = 600 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість будівель визначається з урахуванням кількості робочих місць виконавців, питомої площі на одне робоче місце, та вартості одного квадратного метра виробничої площі:

КбПЗ_2023

$$Б_{уд} = R^1_{сп} S_у Ц_{пл}, \quad (7.9)$$

де: $R^1_{сп}$ – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць;

$S_у$ – питома площа на одне робоче місце, $м^2$;

$Ц_{пл}$ – вартість одного квадратного метра площі, грн.

Згідно даних ТОВ науково-дослідницького консалтингового підприємства «Пекторaль» (м. Кіровоград, вул. Глинки 16) ціна одного квадратного метра площі новобудови, вік якої не перевищує 25 років, по місту складає 800…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 25 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно 8 $м^2$. З урахуванням цього:

$$Б_{уд} = 8 \cdot 8 \cdot 20000 = 1280000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 128000 грн.

Балансова вартість інвентарю розраховується за нормою 3500 грн. на одне робоче місце. Тобто:

$$І_{нв} = R^1_{сп} \cdot Ц_м, \quad (7.10)$$

де: $Ц_м$ – ціна меблів для одного робочого місця, грн.

$$І_{нв} = 8 \cdot 3500 = 28000 \text{ грн.}$$

Балансова вартість обчислювальної техніки визначається по оптовим цінам постачальника з врахуванням витрат на транспортування.

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу фірми Комп'ютерторг за 15.10.23 – джерело

http://computorg.ua/ru/price.html

КбПЗ – 2023

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10947
Системний блок		7347
Процесор	Intel Celeron G5905 (BX80701G5905) 1200, 2 ядра, 2 потоки, 3.5 GHz, TDP - 58 Вт, 14nm box	-
Системна плата	GIGABYTE H470M H 1200, DDR4 64 ГБ, 3200 MHz, LAN - 1 Гбит/с, D-Sub (VGA), HDMI, 1 x M.2 2280, 4 x Sata 6.0 Gb/s, Micro-ATX	-
Відеокарта	Radeon R7 350 2Gb Afox (AFR7350-2048D5H4-V3) PCI-Express 3.0, 2 ГБ, GDDR5, 128 Bit	-
Жорсткий диск	SSD 2.5" 256GB Mibrand, 3D TLC NAND, 2.5", SATA III (6Gb/s)	-
Оперативна пам’ять	DDR4 8GB 2666 MHZ GOODRAM (GR2666D464L19S/8G)	-
DVD-привод	-	-
Корпус	ATX Middle Tower FOXCONN Pro, 3GTLA-489, PSU 350W(FSP Brand: ATX-350PNR, 12cm), black, (front bezel – black+light silver; body material – 0.6mm), 80mm fan (rear), 2xUSB2.0/AUDIO/MIC, Air Duct, Tool-less chassis design,Thermally Advantaged Chassis	-

КОПЗ-2023

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Кулер	–	–
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-B, int. 3.5", 1*USB2.0+AUDIO+1394, multi: All Type Cards, black	220
інше	Клавіатура, мишка	Подару-нок
Монітор	22" TFT, ASUS VW223D (5ms, 300/3000: 1, 170/160, D-SUB, Wide)	3600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп’ютери	8	10947	8757,6	96333,6
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	-	-	-	0
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	114885,1

КПІ 2023

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межах до 10% від оптової ціни.

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400
Група 4			
3. Обчислювальна техніка	114885	-	-
Всього по групі	114885	50	57442,5
Група 5, 6			
4. Вимірювальні пристрої	5190	25	1297,5
5. Транспортні засоби	97500	20	19500
6. Господарський інвентар	28000	25	7000
Всього по групі	130690	-	27797,5
7. Нематеріальні активи	19000	10	1900
Разом	$К_р$ = 1672575		$А_р$ = 157540

КПІ – 2023

Примітка: вартість автомобіля взята по даним з автосалону автотрейдінг, вкладки автобазар, джерело http: //www.auto-trading.com.ua/sale/lot20772.html, складає 2635 USD, що враховуючи курс 37 складає 97500 грн.

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

$$З_о = \frac{З_{сд} \cdot Т_{пз}}{N_е}, \quad (7.11)$$

де: $N_е$ – кількість екземплярів програм, шт.

$$З_о = 600 \cdot 209/19 = 6600 \text{ грн.}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних та суспільних обов'язків) на рівні 10%:

$$З_д = З_о \cdot Н_q \cdot 0{,}01, \quad (7.12)$$

де: $Н_q$ – норматив додаткової зарплати, %.

$$З_д = 6600 \cdot 10 \cdot 0{,}01 = 660 \text{ грн.}$$

Відрахування на соціальні потреби за нормативом $Н_с = 22\%$ від суми основної та додаткової зарплати:

$$С_{оц} = 0{,}01 \cdot Н_с (З_о + З_о), \quad (7.13)$$

де: $Н_с$ – відрахування на соціальні потреби, %.

$$С_{оц} = 0{,}01 \cdot 22(6600+660) = 2686 \text{ грн.}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом $Н_г = 15\%$ від основної зарплати:

$$Г_{осп} = З_о \cdot Н_г \cdot 0{,}01, \quad (7.14)$$

де: $Н_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 6600 \cdot 15 \cdot 0{,}01 = 990 \text{ грн.}$$

Визначимо витрати на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

КбПЗ_2023

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.15)$$

де: $З_{М1}$ – вартість паперу, грн.; $З_{М2}$ – вартість запам'ятовуючих пристроїв, грн.; $З_{М3}$ – вартість фарби, картриджів, тонеру, грн.; $N_е$ – кількість екземплярів програм, шт.

Згідно норм приймаємо 0,5 пачки паперу на три місяці розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$ = 210 грн., визначаємо вартість паперу за період розробки $N_м$ = 3 міс:

$$З_{М1} = Ц_п \cdot N_м. \quad (7.16)$$

$$З_{М1} = 210\ 0{,}5 = 105 \text{ грн.}$$

Згідно норм до вартості запам'ятовуючих пристроїв входить вартість CD дисків в кількості, що дорівнює двом екземплярам програм та одного DVD диска для збереження резервної копії програми:

$$З_{М2} = \sum Ц_д\ , \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR TDK 700Mb, 80Min, 52x Cake box – 28 грн./шт., DVD-R LG 4,7Gb, 16x speed Cake box – 51 грн./шт.

$$З_{М2} = 28 \cdot 2 + 51 = 107 \text{ грн.}$$

Згідно виданих викладачем норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_з., \quad (7.18)$$

де: $Ц_з$ – вартість розхідних матеріалів друкуючих пристроїв: відновлення та заправка картриджу для Canon i-SENSYS LBP6030W – 574 грн.; картридж для Epson Stylus Photo P50 – 558 грн.; відновлення картриджу для MF217W – 570 грн.

$$З_{М3} = 574 + 558 + 570 = 1702 \text{ грн.}$$

$$З_М = (105 + 107 + 1702)/19 = 101 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційних систем за нормативом ($Н_п$ = 15%) від основної зарплати виконавців:

$$О_п = З_о \cdot Н_п \cdot 0{,}01, \quad (7.19)$$

де: $Н_п$ – норматив витрат на освоєння нових мов програмування, %.

$$О_п = 6600 \cdot 15 \cdot 0{,}01 = 990 \text{ грн.}$$

КПЗ 2023

Величини ціна підприємства, податок на додану вартість, відпускна ціна програмної продукції визначаються за формулами, приведеними в таблиці 7.9

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн
1	2	3
1. Основна зарплата виконавців	$З_о$	6600
2. Додаткова зарплата виконавців	$З_д$	660
3. Відрахування на соціальні потреби	$С_{оц}$	2686
4. Загальногосподарські витрати	$Г_{осп}$	990
5. Витрати на матеріали	$З_М$	101
6. Освоєння нових операційних систем, мов програмування	$О_п$	990
7. Амортизація основних фондів	$А_м$	2073
8. Повна собівартість програмного забезпечення	$С_п$	14100
9. Плановий прибуток	$П_р$	7050
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	21150
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	4230
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	25380

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 19 прим.):

$$А_м = \frac{А_р \cdot N_{міс}}{N_е \cdot 12}, \quad (7.20)$$

де: $А_р$ – загальна річна сума амортизаційних відрахувань, грн.

КОПЗ – 2023

$$A_м = 157540 \cdot 3/(19 \cdot 12) = 2073 \text{ грн.}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції:

$$C_п = З_о + З_д + C_{оц} + Г_{осп} + З_м + О_п + А_м. \quad (7.21)$$

$$C_п = 6600+660+2686+990+101+990+2073 = 14100 \text{ грн.}$$

Визначимо плановий прибуток за рівнем рентабельності ($P_п$) програмної продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 50%.

$$П_р = 0{,}01 \cdot P_п \cdot C_п, \quad (7.22)$$

де: $P_п$ – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 50 \cdot 14100 = 7050 \text{ грн.}$$

7.6 Визначення об'єму капітальних вкладень у споживача програмної продукції

Об'єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах. Результати розрахунків зводимо у таблицю 7.10.

Таблиця 7.10 – Розрахунок об'єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	25380
Всього капітальних витрат	–	25380

КПЗ_2023

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаємо при умові роботи підсистеми на протязі року. Результати зводимо до таблиці 7.11.

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на технічне обслуговування	$З_р$	29524	13420
2. Витрати на електроенергію	$З_{ел}$	5320	4560
3. Витрати на амортизацію	$З_{ам}$	0	6345
Всього витрат за рік	$І$	34844	24325

Витрати на профілактичні роботи:

$$З_р = Т_р \cdot З_г \cdot (1 + 0{,}01 \cdot Н_q) \cdot (1 + 0{,}01 \cdot Н_c), \qquad (7.23)$$

де: $Т_р$ – кількість годин обслуговування кожного комп'ютера за рік, *год.*;

$З_г$ – заробітна плата обслуговуючого персоналу, *грн / год*.

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 220 годин на рік до 100 годин на рік, тому витрати на технічне обслуговування зменшилися з:

$$З_{р\,баз} = 220 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 29524 \text{ грн},$$

до:

$$З_{р\,нов} = 100 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 13420 \text{ грн}.$$

Витрати на електроенергію визначаються з урахуванням спожитої потужності ($П_{ел}$) в кіловатах, часу експлуатації технічних засобів ($Т_р$) в годинах та ціни однієї кіловат-години ($Ц_{ел}$):

КПІЗ – 2023

$$З_{ел} = П_{ел} \cdot Т_р \cdot Ц_{ел}. \quad (7.24)$$

$$З_{ел\ баз} = 0{,}35 \cdot 4000 \cdot 3{,}8 = 5320 \text{ грн.}$$

$$З_{ел\ нов} = 0{,}3 \cdot 4000 \cdot 3{,}8 = 4560 \text{ грн.}$$

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.12.

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	25	–	25380	–	6345
Всього відрахувань	-	–	25380	–	6345

7.8 Визначення економічної ефективності програмної продукції

Економічна ефективність програмного забезпечення визначається для виготовлювача і споживача за такими показниками.

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою:

$$E_в = (Ц_п - С_п) \cdot N_е - \sum_{i=1}^{m} E_{p_m} \cdot К_{p_m}, \quad (7.25)$$

де: $К_р$ – балансова вартість основних фондів розробника, грн.; $Е_р$ – розрахунковий коефіцієнт капіталовкладень.

$$Ев = (21150 - 14100) \cdot 19 - (0{,}05 \cdot 1408000 + 0{,}5 \cdot 114885 +$$
$$+0{,}25 \cdot 33190 + 0{,}2 \cdot 97500 + 0{,}1 \cdot 19000) \cdot 3/12 = 94565 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$Т_в = \frac{К_р^*}{(Ц_п - С_п) \cdot N_е}, \quad (7.26)$$

КПЗ – 2023

де: K_p^* – балансова вартість основних фондів розробника.

$$T_в = \frac{1672575}{(21150 - 14100) \cdot 19 \cdot 12 / 3} = 3{,}1 \text{ років}.$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	19
2. Повна собівартість розробленої програми	Грн.	14100
3. Ціна розробленої програми	Грн.	21150
4. Плановий прибуток від реалізації розробленої програми	Грн.	7050
5. Рентабельність програмної продукції	%	50
6. Об’єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1672575
7. Загальний прибуток від реалізації програмної продукції	Грн.	133950
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	94565
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років.	3,1
10.Об’єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	25380
11.Величина економічного ефекту у користувача програмної продукції	Грн.	4174
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	2,4

КОПЗ – 2023

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б - I_н) - E_н(K_н - K_б), \quad (7.27)$$

де: $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно; $K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються.

$$E_{сп} = (34844\text{-}24325)\text{-}0{,}25 \cdot 25380 = 4174 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат:

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н}, \quad (7.28)$$

$$T_{сп} = \frac{25380}{34844 - 24325} = 2{,}4 \text{ року}.$$

7.9 Висновки

Розроблена програма економічно вигідна. За рахунок впровадження програмного забезпечення досягається скорочення часу обробки інформації, підвищується культура праці, підвищення якості приймаючих управлінських рішень.

КбПЗ – 2023

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Аналізуючи умови працівників ІТ-сфери, на перший погляд, може здатися, що працівники сфери інформаційних технологій не схильні до ризиків на виробництві, та якщо більш глибоко розглянути умови і специфіку праці фахівців сфері ІТ-індустрії, можна виявити ряд факторів які будуть мати негативний сплив на стан охорони праці, та на самого ІТ-фахівця зокрема. Сюди можна віднести як невідповідність освітлення, так і високий рівень шуму, що негативно позначатимуться як на емоційному так і на фізичному стані фахівця, призводитимуть до зниження ефективності праці та виробничих травм. Також, важливим моментом охорони праці ІТ-фахівця є врахування його психологічних можливостей (швидкість реакції, особливості пам'яті та уваги, емоційний стан тощо). Для того, щоб забезпечити ефективну роботу ІТ-фахівця, потрібно враховувати та максимально компенсувати такі негативні фактори як: надмірне нервово-емоційне навантаження, довготривалі статичні перевантаження, обмежена рухова активність. Всі ці чинники призводить до різноманітних відхилень у стані здоров’я, зокрема до перевтоми, зниження фізичної та розумової працездатності, неврозів, захворювань серцево-судинної системи тощо. Метою даного розділу є огляд конкретних умов праці спеціаліста у сфері ІТ-індустрії. Завданнями для даного розділу є: аналіз умов праці на робочому місці фахівця ІТ-індустрії, розробка конкретних рекомендацій щодо покрашення умов праці фахівців ІТ-індустрії, огляд пожежної безпеки на ІТ-підприємстві та розрахунок системи загального штучного освітлення виробничого приміщення де працюють ІТ – фахівці.

На робочому місці ІТ-фахівця (або програміста) виникають небезпечні та шкідливі для безпечної життєдіяльності фактори:

КБПЗ-2023

– підвищений рівень шуму;

– несприятливі мікрокліматичні умови;

– недостатній рівень освітленості;

– шкідливі речовини;

– підвищений рівень електромагнітних випромінювань радіочастот;

– висока напруга електричної мережі;

– статична електрика та інші.

8.2 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

Розглянемо умови праці у приміщенні, в якому працюють програмісти. Геометричні розміри приміщення наведено у таблиці 8.1.

Таблиця 8.1 – Розміри приміщення

Найменування	Значення, м
Ширина	3
Довжина	4,6
Висота	3

Таблиця 8.2 – Площа та обсяг приміщення, на одного працюючого*

Геометрична характеристика	Одиниця виміру	Нормативне значення*	Фактичне значення
Площа, S	$м^2$	не менше 6.0	6,9
Об'єм, V	$м^3$	не менше 20.0	20,7

* Згідно ДСанПіН 3.3.2.007-98 (Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин).

КОПЗ_2023

У зазначеному приміщенні працюють двоє людей. За даними, які наведено у табл. 8.1, та табл. 8.2, можна зробити висновок, що площа та об'єм приміщення у розрахунку на одно робоче місце програміста не відповідають нормативним вимогам ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [2], але відповідають нормативним вимогам Наказу Міністерства соціальної політики України № 207, від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» та НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин»). Таким чином можна зробити висновок, що санітарно-гігієнічні умови праці на робочому місці програміста відповідають вимогам.

Температура повітря в приміщенні визначається впливом температури зовнішнього повітря і тепловою енергією, яка виділяється всередині приміщення. Джерелами виділення теплоти в даному приміщенні є електроустаткування, освітлювальні прилади, а також люди. У світлий час доби джерелом надлишкового тепла є сонячна радіація. Згідно Постанови № 42 від 01.12.1999 Головного державного санітарного лікаря України, робота, виконувана в даному приміщенні, відноситься до категорії Іа. В цьому випадку людина витрачає енергії до 120 ккал у годину. Вологість повітря в приміщенні визначається впливом багатьох факторів, серед яких: вологість атмосферного повітря, виділення вологи людьми (при диханні та випарами з поверхні шкіри).

Мікроклімат повітряного середовища в приміщенні характеризується запиленістю та загазованістю повітря. Мікроклімат приміщення визначається діючим на організм людини поєднанням, вологості, температури, швидкості руху повітря та інтенсивності теплового випромінювання. Аналіз мікроклімату складається з визначення зазначених вище факторів і порівняння результатів із встановленими нормами.

КОПІЯ 2023

У таблиці 8.3 наведено оптимальні та фактичні значення параметрів мікроклімату як для категорії ваги робіт Іа, так і розглянутого приміщення. У приміщеннях, де встановлено ЕОМ, рекомендується застосування тільки оптимальних значень показників мікроклімату.

Проведений аналіз показує, що показники мікроклімату в приміщенні відповідають установленим нормам. Штучне опалення застосовується у холодний період року.

Таблиця 8.3 – Оптимальні і фактичні значення параметрів мікроклімату

Пора року	Оптимальні для Іа			Фактичні		
	Температура, °С	Вологість,%	Швидкість повітря, м/с	Температура,°С	Вологість%	Швидкість повітря, м/с
Холодна	22-24	40-60	0,1	22-24	40-55	0,12
Тепла	23-25	50-70	0,1	24-25	50-65	0,9

В літню пору застосовується кондиціонер.

Для боротьби з пилом робляться регулярні провітрювання та вологі прибирання приміщенні.

У приміщенні знаходяться наступні джерела шуму: принтер Xerox WorkCentre 3025BI (3025VBI), електродвигуни вентиляторів ЕОМ.

Одним з найважливіших факторів, які впливають на ефективність трудової діяльності людини, та попереджають травматизм і професійні захворювання програмістів є освітлення на робочому місці.

З 2019 року діють Державні будівельні норми україни "Природне і штучне освітлення" – ДБН В.2.5-28:2018 [1], у яких прописані вимоги до використання всіх освітлювальних приладів, у т.ч. світлодіодних.

Працю працівника, який постійно працює за комп'ютером, згідно ДБН В.2.5-28:2018 [1], можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом

КОПЗ_2023

(під розряд зорової роботи В). Приміщення можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при поєднаному, спільному освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 Лк. [1], Крім того все поле зору повинне бути освітлено достатньо рівномірно – ця основна гігієнічна вимога. Так як яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими.

8.3 Розробка заходів з умов поліпшення охорони праці

Згідно аналізу умов праці в розглянутому приміщенні, ми одержали наступні результати:

– розмірі приміщення, у розрахунку на одному працюючого, відповідають нормативам;

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значеннь;

Таким чином можна припустити, що основною причиною можнивого зниження працездатності програміста є психофізіологічний фактор, тому основна пропозиція буде така: дотримання позитивної психологічної атмосфери в колективі та регламентованого режиму праці та відпочинку, організація робочого місця з урахуванням ергономічних вимог.

Рекомендовані заходи: регулярні періодичні наочні огляди персоналом шляхів для евакуації людей із приміщення, відповідно до плану евакуації (який повинен розташовуватись на видному місці у приміщенні), включення до колективного договору мінімально можливого вмісту аптечок з обов’язковою

КПЗ - 2023

наявністю масок-клапанів, або іншого спорядження для штучного дихання. Регулярна періодична перевірка параметрів заземлення та занулення (вимірювання опору ланцюга) [9].

Регулярна наочне знайомство персоналу із шляхами для евакуації людей із приміщення відповідно до плану евакуації, забезпечення розподільних щитів спеціальними розетками з заземлюючими контактами; організація заземлення всіх приладів і пристроїв, які працюють при напрузі вище 36 В.

Так як при ураженні електричним струмом у людини може статися фібриляція шлуночків серця, в організації бажано мати дефібрилятор і підготовлений персонал для роботи з ним.

8.4 Розрахункова частина

Питання охорони праці та правила безпеки при роботі з офісною, комп'ютерною та мережевою технікою розглянуті у працях вітчизняних вчених [11].

Запорукою безпечної роботи є виконання вимог електричної безпеки, оскільки все офісне обладнання заживлюється від електричної мережі. Одним з необхідних засобів електричної безпеки є встановлення захисного заземлення. Початкові дані, необхідні для розрахунку захисного заземлення:

– допустимий опір розповсюдженню струму в землі від заземлю-вального пристрою $R_{зн}$ = 10 Ом;

– питомий опір грунту в місці встановлення заземлювача $\rho_з$ = 100 Ом/м;

– тип грунту – суглинок;

– тип заземлювача – труба, діаметром d=0.045 м і довжиною l =2.2 м; – конструкція заземлювача – розташування заземлювачів по контуру. Розрахунок проводимо за стандартною методикою [7].

Визначимо розрахунковий опір землі:

КБПЗ – 2023

$$\rho_{рз} = \phi \cdot \rho_3$$

де ϕ – коефіцієнт сезонності, що враховує коливання питомого опору при зміні вологості ґрунту протягом року; при використанні заземлювача довжиною l = 2.2 м при глибині закладання від вершини h = 0.6 м ϕ = 1.1 для четвертої кліматичної зони.

Схема розташування заземлювачів показана на рисунку 8.1.

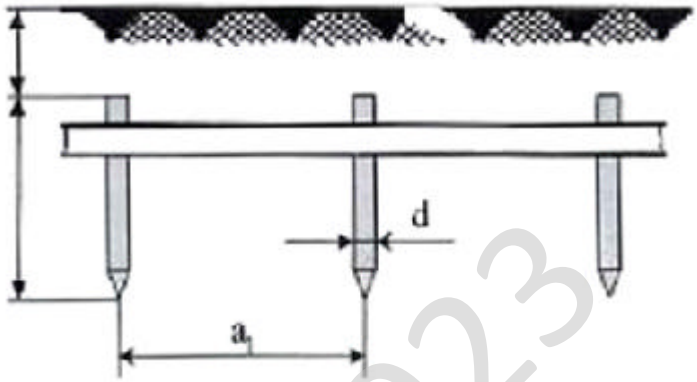


Рисунок 8.1 – Схема розташування заземлювачів

Опір землі:

$$\rho_{рз} = 1{,}1 \cdot 100 = 110 \text{ Ом·м}$$

Опір Rв, розповсюдженню струму в землі від одного вертикального заземлювача:

$$R_B = \frac{\rho_{рз}}{2\pi \cdot l}\left(\ln\frac{2 \cdot l}{d} + 0.5\ln\frac{4t + l}{4t - l}\right)$$

де l – довжина заземлювача (l = 2.2 м);

d = 0.045 м – діаметр заземлювача при U<1 кВ та при S<100 кВА;

t – відстань від поверхні до середини заземлювача:

$$t = h + l/2 = 0.6 + 2.2/2 = 1.7 \text{ м.}$$

$$R_в = \frac{110}{2 \cdot 3.14 \cdot 2.2}\left(\ln\left(\frac{2 \cdot 2.2}{0.045}\right) + 0.5 \cdot \ln\left(\frac{4 \cdot 1.7 + 2.2}{4 \cdot 1.7 - 2.2}\right)\right) = 38.9 \text{ Ом}$$

Визначаємо потрібну кількість заземлювачів:

КбПЗ – 2023

$$n' = \frac{R_в}{R_{зн}} = \frac{38.9}{10} = 3.9 \approx 4 \, шт.$$

Коефіцієнт використання вертикальних заземлювачів враховує ефект екранування. При вибраному значенні k = a/l, де a – відстань між вертикальними заземлювачами, м; k = 1 при a = 1.8 м. Таким чином коефіцієнт використання вертикального заземлювача за довідковими даними дорівнює $\eta_В = 0{,}6$.

Кількість вертикальних заземлювачів з урахуванням коефіцієнту використання $\eta_В$ приблизно складає

$$n = \frac{R_в}{R_{зн} \cdot \eta_в} = \frac{38.9}{10 \cdot 0.6} = 6.48 \approx 7 \, шт.$$

Довжина горизонтального заземлювача, необхідна для розміщення вертикальних заземлювачів по контуру

$$L = a \cdot n = 1.8 \cdot 7 = 12.6 \text{ м}$$

Опір горизонтального заземлювача $R_Г$, Ом, прокладеного на глибині h = 0.6 м від поверхні землі буде

$$R_г = \frac{R_{рз}}{2 \cdot 3.14 \cdot L} \cdot \ln \frac{2 \cdot L^2}{b \cdot h} = \frac{110}{2 \cdot 3.14 \cdot 12.6} \cdot \ln \frac{2 \cdot 12.6^2}{0.06 \cdot 0.6} = 13.57 \text{ Ом}$$

де b = 0.06 м – ширина сталевої смуги, з якої виготовлений заземлювач.

Обчислюємо загальний опір:

$$R_з = \frac{R_в \cdot R_г}{n \cdot R_г \cdot \eta_в + R_в \cdot \eta_г} = \frac{38.9 \cdot 13.57}{6 \cdot 13.57 \cdot 0.6 + 38.9 \cdot 0.34} = 7.93 \text{ Ом}$$

де $\eta_Г$ – коефіцієнт використання горизонтального заземлювача ($\eta_Г$ = 0.34).

Маємо 7.93 Ом < 10 Ом, отже нормативне обмеження $R_з < R_{з.норм}$ виконується.

8.5 Список використаних джерел інформації

1. Державні будівельні норми України: ДБН В.2.5-28:2018. – Режим доступу до ресурсу: https://goo.su/9AkQ

КПІ 2023

2. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин: ДСанПІН 3.3.2-007-98. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/v0007282-98

3. Закон України «Про охорону праці» від 14.10.1992 р. № 2694-XII. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2694-12

4. Зеркалов Д. В. Охорона праці в Галузі: Загальні вимоги: навч. посіб. Київ: Основа. 2011. 551 с.

5. Наказ Міністерства соціальної політики України 14.02.2018 № 207 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z0508

6. Охорона праці. Ч. 1. Захисне заземлення: метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ сумісного типу / Кіровоград. ін-т с.-г. машинобуд.; [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака]. – Кіровоград: КІСМ, 1997. – 20 с. Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/4358

7. Постанова № 42 від 01.12.1999 Головного державного санітарного лікаря України «Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99**.** – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/va042282-99 (дата звернення 19.09.23).

8. Оришака, О. В. Основи охорони праці: навч. посіб. / О. В. Оришака, Г. П. Горбачова, К. М. Марченко; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. – Кропивницький : ЦНТУ, 2022. – 175 с. – Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/12161 (дата звернення 19.09.22).

9. Методичні рекомендації до виконання розділу "Заходи з охорони праці та техніки безпеки" випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти для здобувачів вищої освіти спеціальностей 123 "Комп'ютерна інженерія" та 122 "Комп'ютерні науки" / М-во освіти і науки

КПІ – 2023

України, Центральноукраїн. нац. техн. ун-т, каф. кібербезпеки та програм. забезпечення; [укл. О.В. Оришака, К.М. Марченко]. – Кропивницький: ЦНТУ, 2022. – 19 с. [Електронний ресурс]. – Режим доступу: http://dspace.kntu.kr.ua/jspui/handle/123456789/12240

10. Катренко Л.А., Катренко А.В. Охорона праці в галузі комп'ютингу: Підручник. За науковою редакцією В.В. Пасічника. – Львів: «Магнолія 2006», 2012. – 544 с.

КбПЗ_2023

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи управління файлами в ОС Windows на основі технології BitLocker Drive Encryption.

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

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів управління файлами в ОС Windows на основі технології BitLocker Drive Encryption.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем управління файлами в ОС Windows на основі технології BitLocker Drive Encryption.

– Досліджена система управління файлами в ОС Windows на основі технології BitLocker Drive Encryption.

– На основі отриманих результатів досліджень створена програмна реалізація системи управління файлами в ОС Windows на основі технології BitLocker Drive Encryption.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання управління файлами в ОС Windows на основі технології BitLocker Drive Encryption.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

КПЗ_2023

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

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

Програма реалізована на мові високого рівня Delphi 10.4. Дана мова програмування дозволяє найбільш ефективно обробляти дані. 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 й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм FEAL.

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

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 4174 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 2,4 роки.

КПЗ_2023

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Пісаренко Д.О. Дослідження та програмна реалізація системи управління файлами в ОС Windows на основі технології BitLocker Drive Encryption // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2023.
2. Loren Kohnfelder. Designing Secure Software. No Starch Press. 2022. 332 p.
3. Samir Kumar Rakshit. Ethical Hacker's Penetration Testing Guide. BPB Online. 2022. 509 p.
4. Corey J. Ball. Hacking APIs. No Starch Press. 2022. 353 p.
5. Kevin Beaver. Hacking for Dummies. John Wiley & Sons. 2022. 419 p.
6. Mark S. Merkow. Practical Security for Agile and DevOps. CRC Press. 2022. 236 p
7. Derek Fisher. Application Security Program Handbook. Manning Publications. 2021. 155 p.
8. Cameron Wyatt PH.D. Kali Linux Tutorial. Independently published. 2021. 60 p.
9. Alex Matrosov, Eugene Rodionov, Sergey Bratus. Rootkits and Bootkits. No Starch Press. 2019. 450 p.
10. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» *CEUR Workshop Proceedings,* Volume 3187, 2022,
11. Smirnov, O., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebeshko, B. «Selection of a Rational Composition of İnformation Protection Means Using a Genetic Algorithm». *In: Rajakumar, G., Du, KL., Vuppalapati, C., Beligiannis, G.N. (eds) Intelligent Communication Technologies and Virtual Mobile Networks.*

КПЗ_2023

Lecture Notes on Data Engineering and Communications Technologies, vol 131. 2023. Springer, Singapore. pp. 21-34.

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

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

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

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

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

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

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

КОПЗ – 2023

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

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

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

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

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

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

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

26. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In:

КОПЗ-2023

Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

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

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

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

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

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

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

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

КОПЗ – 2023

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

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

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

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

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

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

40. Smirnov, O., Kuznetsov, A., Kiian, A., Babenko, B., Zhosan, H., Prokopovych-Tkachenko, D., «Soft Decoding Method for Turbo-Productive Codes»,

КБПЗ-2023

2019 3rd International Conference on Advanced Information and Communications Technologies, AICT 2019, Lviv, Ukraine, 2-6 July, 2019, P. 129-134.

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

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

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

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

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

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

47. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої

КОПЗ-2023

функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку,* 2022, № 3(69). С. 93-98.

48. 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.

49. 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.

50. 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.

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

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

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

КБПЗ_2023

Додаток А
(обов'язковий)

Технічне завдання

Зміст

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

КПЗ–2023

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи управління файлами в ОС Windows на основі технології BitLocker Drive Encryption.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 37-13 від 04.08.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи управління файлами в ОС Windows на основі технології BitLocker Drive Encryption.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за другим (магістерським) 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 з ОС та з користувачем;

КПЗ_2023

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи управління файлами в ОС Windows на основі технології BitLocker Drive Encryption;

– 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.

КбПЗ_2023

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 Delphi 10.4.

КбПЗ – 2023

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 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2023 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинен бути розглянутий аналіз санітарно-гігієнічних умов праці на робочому місці програміста.

КбПЗ_2023

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 107 аркушів.

10 Етапи розробки

10.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.

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

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 10.12.2023 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 14.12.2023 р.

КПЗ – 2023

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

_____ Лисенко І.А.

Дослідження та програмна реалізація системи управління файлами в ОС Windows на основі технології BitLocker Drive Encryption

КбПЗ_2023

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 59

Літера: РП

Кропивницький – 2023 року

Файл FmxUtils.pas – робота з файлами

```
unit FmxUtils;

interface

uses SysUtils, Windows, Classes, Consts;

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

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

implementation

uses Forms, ShellAPI, RtlConsts;

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

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

КОПЗ – 2023

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


{ Процедура переміщення файла }

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

{ GetFileSize функція, повертає розмір файлу }

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

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

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

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

end.
```

КОПЗ_2023

Файл filmanex.dpr основної програми

```
program FilManEx;

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

{$R *.RES}

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

КбПЗ_2023

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
```

КбПЗ 2023

```
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;
```

Кобз 2023

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

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

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

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

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

implementation

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

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

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

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

КОПЗ_2023

```
  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
```

КбПЗ_2023

```
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;
```

КБПЗ_2023

```
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;
```

КОПЗ_2023

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

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

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

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

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

КбПЗ_2023

```
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.
```

КбПЗ_2023

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);
```

КОПЗ_2023

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

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

var
  FMForm: TFMForm;

implementation

uses FmxUtils, FAttrDlg, FChngDlg;

{$R *.dfm}

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

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

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

КОПІЯ - 2023

```
        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
```

КОПІЮВАННЯ 2023

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

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

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

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

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

КСТВ - 2023

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

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

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

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

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

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

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

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

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

КбПЗ_2023

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

end.
```

КбПЗ_2023

Файл AES_key.pas – введення користувачем секретного ключа AES

```
unit AES_key;

interface

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

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

var
  AES: TDES;

implementation

{$R *.dfm}

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

end.
```

КбПЗ_2023

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

```
unit RSA_keys;

interface

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

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

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



{$R *.dfm}

procedure TRSA_keys.Button1Click(Sender: TObject);
begin

// Генерація p та q
```

КСЗ_2023

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


 // Обчислення N

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


// Обчислення E – ключ шифрування

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

// Обчислення D – ключ дешифрування

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


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


end;

//дешифрування ключа AES

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

//запис ключів RSA у файли

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

КСТЗ_2023

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

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

end;

//шифрування ключа AES


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

       Edit3.Text:=test;

end;


//Запис ключа AES у файл

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

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

end.
```

КбТЗ_2023

Файл FGIntRSA.pas – алгоритм шифрування RSA

```
Unit FGIntRSA;

Interface

Uses Windows, SysUtils, Controls, FGInt;

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


Implementation


{$H+}



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

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

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

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


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

КОПЗ_2023

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

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

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

KCT3_2023

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


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


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

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

End.
```

КОПЗ_2023

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

```
Unit FGInt;

Interface

Uses Windows, SysUtils, Controls, Math;

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

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

КОПІЯ 2023

```
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,
```

FGIS 2023

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,

KCIS-2023

```
      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);
```

КбПЗ_2023

```
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);
```

КОПЗ_2023

```
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
```

КбТЗ_2023

```
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);
```

КбПЗ_2023

```
      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
```

КбПЗ_2023

```
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 := '';
```

КоПЗ_2023

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


// перетворюємо  FGInt в рядок 10 біт

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



// Видаляємо FGInt для звільнення пам'яті

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



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


// Додаємо 2 FGInts, FGInt1 + FGInt2 = Sum

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

КбТВ-2023

```
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
```

КОПЗ - 2023

```
   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
```

КбТЗ_2023

```
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
```

КПЗ_2023

```
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
```

КОПЗ_2023

```
   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
```

КбПЗ_2023

```
      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;
```

КОПЗ_2023

```
   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 := '';
```

КбПЗ_2023

```
   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
```

КОПЗ_2023

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


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

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

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

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


// FGInt = FGInt * 2147483648

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


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

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

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

КБІЗ_2023

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

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

KSTS_2023

```
   FGIntDestroy(zero);

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


// Розраховуємо MFGInt

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

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

КОПІЯ - 2023

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

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

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


// MFGInt = FGInt1 mod FGInt2

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

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

КОПІЯ 2023

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

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

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


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

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


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

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


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

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


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

КбІЗ_2023

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

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


// Процедура підводення у ступень за Монтгомері

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


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

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

КОПІЯ_2023

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

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


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


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

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

КбГТ 2023

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

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

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


// розраховуємо найбільший загальний дільник 2 FGInts

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

КОПЗ_2023

```
      End;
End;


// розраховуємо найменше загальне кратне 2 FGInts

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


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

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


// Генератор випадкових чисел

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


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

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

КБПЗ_2023

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

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

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

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

      End
   End;

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


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

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

КОПІЯ 2023

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

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

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

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

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

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


// Знахродимо мультпликативне зворотнє FGInt у кінцевому кільці позитивного
порядку

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

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

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

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

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

КОПІЯ_2023

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

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


// Простй комбінований тест на простоту FGIntp

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


//Розрахунок символу Лагранжа

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

КПЗ_2023

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


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

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

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

КОПЗ_2023

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

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


End.
```

КбПЗ_2023

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

```
Unit FGIntPrimeGeneration;

Interface

Uses Windows, SysUtils, Controls, FGInt;

Procedure PrimeSearch(Var GInt : TFGInt);

Implementation


{$H+}


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

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

End.
```

КбПЗ_2023

Файл about.pas – довідка

```
unit about;

interface

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

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

var
  Form1: TForm1;

implementation

{$R *.dfm}

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

end.
```

КбПЗ_2023