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

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2022 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи для забезпечення конфіденційності інформації з’ємних носіїв”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-21М-1,4
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Коваленко В.Ю.
«____» _____ 2022 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Коваленко О.В.
«____» _____ 2022 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _12 _____*"Інформаційні технології"*_____

Спеціальність 123 *"Комп'ютерна інженерія"*

Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерна інженерія"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 6 » вересня 2022 року

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

Коваленку Владиславу Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи для забезпечення конфіденційності інформації з'ємних носіїв*

2. Керівник роботи *Коваленко Олександр Володимирович, докт. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 19-13 від 17.08.2022 року

3. Строк подання студентом роботи до захисту *10.12.2022 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи для забезпечення конфіденційності інформації з'ємних носіїв*

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*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В.	05.10.2022	14.11.2022
Охорона праці	Оришака О.В.	06.10.2022	16.11.2022

7. Дата видачі завдання « 6 » вересня 2022 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	10.10.2022 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2022 р.	
3.	Розробка моделі компонента	20.10.2022 р.	
4.	Розробка структур даних	25.10.2022 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2022 р.	
6.	Програмування алгоритмів	10.11.2022 р.	
7.	Розрахунок економічної ефективності	13.11.2022 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2022 р.	
9.	Оформлення ПЗ	17.11.2022 р.	
10.	Попередній захист роботи	10.12.2022 р.	

Дата видачі завдання
« 6 » вересня 2022 р.

Підпис керівника
_____ Коваленко О.В.
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2022 р.

Підпис здобувача
_____ Коваленко В.Ю.
(прізвище та ініціали)

АНОТАЦІЯ

Коваленко В.Ю. Дослідження та програмна реалізація системи для забезпечення конфіденційності інформації з'ємних носіїв. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2022.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи для забезпечення конфіденційності інформації з'ємних носіїв.

Метою розробки є дослідження та програмна реалізація системи для забезпечення конфіденційності інформації з'ємних носіїв.

Об'єктом дослідження є процес для забезпечення конфіденційності інформації з'ємних носіїв.

Предметом дослідження є методи для забезпечення конфіденційності інформації з'ємних носіїв.

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

Результат роботи – програмна реалізація системи для забезпечення конфіденційності інформації з'ємних носіїв.

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 RAD Studio Delphi 10.

Ключові слова: комп'ютерна інженерія, захисту доступу, з'ємних носії

ABSTRACT

Kovalenko V.Yu. Research and software implementation of a system to ensure the confidentiality of information on removable media. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2022.

In this final qualification work for the second (master's) level of higher education, software is developed, which is intended for a system to ensure the confidentiality of information on removable media.

The purpose of the development is research and software implementation of a system to ensure the confidentiality of information on removable media.

The object of the study is the process for ensuring the confidentiality of information on removable media.

The subject of research is methods for ensuring the confidentiality of information on removable media.

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

The result of the work is the software implementation of the system to ensure the confidentiality of information on removable media.

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 RAD Studio Delphi 10 environment.

Keywords: computer engineering, access protection, removable media

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ГПВЧ – генератор псевдовипадкових чисел

ДСТ – державний стандарт

ЕОМ – електронна обчислювальна машина

ІБ – інформаційна безпека

ІС – інформаційна система

КСЗІ – комплекс системи захисту інформації

НЗД – найбільший загальний дільник

НСД – несанкціонований доступ

ПЗ – програмне забезпечення

ПЗП – Постійно Запам'ятовувальний Пристрій

EEPROM – Electronically EPROM

EPROM – Erasable Programmable ROM або як Electrically Programmable ROM

Flash – Flash Erase EEPROM

NVRWM – nonvolatile read-write memory

PROM – Programmable ROM

RAM – Random Access Memory

ROM – Read Only Memory

ВСТУП

Актуальність теми. Широке застосування комп'ютерних технологій і постійне збільшення обсягу інформаційних потоків викликає постійний ріст інтересу до криптографії. Останнім часом збільшується роль програмних засобів захисту інформації, просто модернізуємих, не потребуючих великих фінансових витрат у порівнянні з апаратними криптосистемами [1-5]. Сучасні методи шифрування гарантують практично абсолютний захист даних, але завжди залишається проблема надійності їхньої реалізації. Іншою важливою проблемою застосування криптографії є протиріччя між бажанням громадян захистити свою інформацію й прагненням державних спецслужб мати можливість доступу до деякої інформації для припинення незаконної діяльності [6]. Надзвичайно важко знайти незаперечно оптимальне рішення цієї проблеми. Як оцінити співвідношення втрат законослухняних громадян і організацій від незаконного використання їхньої інформації й збитків держави від неможливості одержання доступу до захищеної інформації окремих груп, що приховують свою незаконну діяльність? Чи можна гарантированно не допустити незаконне використання криптоалгоритмів особами, які порушують і інші закони? Крім того, завжди існують способи схованого зберігання й передачі інформації. Хоча стримування відкритих досліджень в області криптографії й криптоаналізу є найпростішим шляхом, але це принесе значний негативний ефект. Застосування ненадійних засобів не захистить користувачів, але викличе поширення комп'ютерних злочинів, навпроти, виявлення своєчасне виявлення помилок у системах захисту інформації дозволить запобігти збитку [5-8]. У цей час особливо актуальною стала оцінка вже використовуваних криптоалгоритмів. Завдання визначення ефективності засобів захисту найчастіше більше трудомістка, чим їхня розробка, вимагає наявності спеціальних знань і, як правило, більше високої кваліфікації, ніж завдання розробки. Ці обставини приводять до того, що на ринку з'являється

безліч засобів криптографічного захисту інформації, про які ніхто не може сказати нічого певного. При цьому розроблювачі тримають криптоалгоритм (як показує практика, часто нестійкий) у секреті. Однак задача точного визначення даного криптоалгоритму не можуть бути гарантированно складною хоча б тому, що він відомий розроблювачам. Крім того, якщо порушник знайшов спосіб подолання захисту, то не в його інтересах про це заявляти. Тому суспільству повинне бути вигідно відкрите обговорення безпеки систем захисту інформації масового застосування, а приховання розроблювачами криптоалгоритму повинне бути неприпустимим.

На сьогоднішній день існують добре відомі й апробовані криптоалгоритми (як із симетричними, так і несиметричними ключами), криптостійкость яких або доведена математично, або заснована на необхідності рішення математично складного завдання (факторизації, дискретного логарифмування й т.п.) [7-9]. З іншого боку, у комп'ютерному світі весь час з'являється інформація про помилки або "діри" у тій або іншій програмі (у т.ч. що застосовує криптоалгоритми), або про те, що вона була зламана. Це створює недовіру, як до конкретних програм, так і до можливості взагалі захистити що-небудь криптографічними методами не тільки від спецслужб, але й від простих хакерів. Тому знання атак і дір у криптосистемах, а також розуміння причин, по яких вони мали місце, є однією з необхідних умов розробки захищених систем і їхнього використання.

У зв'язку з тим, що на даному етапі часто для переносу інформації використовуються пристрої на flash-накопичувачах, дуже актуальним є завдання їхнього захисту. Існує два підходи до захисту інформації на flash-накопичувачах: з використанням доступу до flash-накопичувача за допомогою біопараметричних характеристик і за допомогою шифрування інформації, що зберігається на flash-накопичувачі.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи для забезпечення конфіденційності інформації з’ємних носіїв.

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

– Огляд існуючих систем для забезпечення конфіденційності інформації з’ємних носіїв.

– Дослідження системи для забезпечення конфіденційності інформації з’ємних носіїв.

– Програмна реалізація системи для забезпечення конфіденційності інформації з’ємних носіїв.

Об’єктом дослідження є процес для забезпечення конфіденційності інформації з’ємних носіїв.

Предметом дослідження є методи для забезпечення конфіденційності інформації з’ємних носіїв.

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 для забезпечення конфіденційності інформації з’ємних носіїв.

– Розроблено вітчизняний продукт для забезпечення конфіденційності інформації з’ємних носіїв, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі для забезпечення конфіденційності інформації з’ємних носіїв.

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.

Робота апробована на LVI Науково-технічній конференції здобувачів вищої освіти «Наука – виробництву», 2022, основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №13.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи для забезпечення конфіденційності інформації з'ємних носіїв, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

На сучасному рівні розвитку flash-накопичувачів, як було відмічено вище існує два шляхи захисту інформації, що знаходиться на них:

– використання біопараметричних характеристик для доступу інформації, яка зберігається на flash-накопичувачі;

– використання криптографічних методів, для захисту інформації, яка зберігається на flash-накопичувачі.

Розглянемо перший шлях. Це flash-диск із убудованим сканером відбитків пальців, та унікальне програмне забезпечення, що дозволяє захистити доступ до файлів як на флешці, так і на самому комп'ютері, а так само замінити введення паролів простим дотиком пальця. Приклад флешки приведено на рисунку 1.1.

Рисунок 1.1 – Flash-накопичувач з використанням біопараметричних характеристик

Але наш дипломний проект призначений для реалізації захисту інформації на флешці другим шляхом. Виходячи із цього проведемо вибір виду шифру для реалізації вищепоставленої задачі.

Розглянемо захист інформації на флешці, як один з варіантів канального шифруваня, тобто такого шифрування при, якому інформації йде суцільним потоком. Основна вимога, пропонована до вибору алгоритмів криптографічного захисту інформації при канальному шифруванні полягає в можливості закриття нефіксованих обсягів переданої інформації з мінімальними часовими затримками.

При використанні наскрізного шифрування вимоги до нерозмноження криптоалгоритмом помилок, внесених каналом зв'язку не настільки важливо. При наскрізному шифруванні на нижчестоящих рівнях виконується завадостійке кодування. Але криптоалгоритми наскрізного шифрування повинні крім закриття переданої інформації давати можливість надійної автентифікації.

Вимоги, пропоновані до вибору алгоритмів криптографічного захисту інформації при наскрізному шифруванні:

- Інтерактивні дані закривати з мінімальною часовою затримкою.
- До закриття неінтерактивних даних особливих вимог не пред'являється.

При комбінованому шифруванні робота із ключами ведеться так: КСЗІ відповідають за ключі, використовувані при канальному шифруванні, а про ключі, застосовуваних при наскрізному шифруванні, піклуються самі користувачі. Застосовуються потокові (можна потоковий (накладення гами) режим роботи блокового шифру), згорточні шифри

Основним недоліком потокових шифрів є необхідність передачі інформації для початкової ініціалізації (синхропосилка) перед заголовком повідомлення, що повинна бути прийнята до розшифровування будь-якого повідомлення. Це пов'язане з тим, що синхропосилка може також бути й секретним ключем (частиною секретного ключа). Передача синхропосилки у вигляді, у якому вона буде застосовуватися в алгоритмі шифрування по каналу передачі даних може створити погрозу криптографічної стійкості системи, і тому

завжди необхідно застосовувати додатковий ключ, за допомогою якого сихропосилка буде закриватися, або використовувати в алгоритмі шифрування ключезалежну модифікацію синхропосилки, як це реалізовано в ДСТУ 28147:2009.

1.2 Область застосування

У ДСТУ 28147:2009 синхропосилка (початкове заповнення) передається у відкритому виді, але в алгоритмі шифрування використовується результат перетворення початкового заповнення по циклу 32-3: $\Omega_0 = Ц_{32\text{-}3}(S)$, де Ω_0 – вектор початкового заповнення рекурентного генератора послідовності чисел (РГПЧ), $Ц_{32\text{-}3}$ – базовий цикл зашифрування, S – синхропосилка.

На підставі наведених міркувань одержимо наступні результати:

Канальне шифрування – потокові шифри (потоковий режим роботи блокового шифру може не підійти).

Наскрізне шифрування:

Для закриття ітерактивних даних можуть підійти згорточні шифри, або блокові шифри в режимі гаммування. Для закриття неітерактивних даних можуть підійти як згорточні шифри (для спрощення конструкції КСЗІ) так і блокові шифри. Криптосистеми з відкритим ключем можуть використовуватися для механізмів автентифікації КСЗІ й ключового обміну між КСЗІ.

Вибір параметрів криптографічних алгоритмів і ключів

Потокові шифри ґрунтуються на псевдовипадкові (випадкових) ключових послідовностях – згенерованих певним чином послідовностях символів із заданими властивостями непередбачуваності (випадковості) появи чергового символу. Генератори ключових послідовностей звичайно базуються на комбінаціях регістрів зрушення й нелінійних булевих функціях.

Генератори ключових послідовностей характеризуються:

- періодом повторення послідовності;

- статистичними властивостями породжуваних послідовностей;
- криптографічною стійкістю генератора ключової послідовності.

Якщо використовується потокове шифрування, тоді мінімальний період повинен задовольняти умові:

$$T_{\Omega} >> T_{max} \cdot V \cdot n, \quad (1.1)$$

де

T_{Ω} – мінімальний період повторення послідовності,

T_{max} – максимальний час безперервної роботи КСЗІ,

V – швидкість передачі в каналі зв'язку,

t – розрядність послідовності на виході генератора ключової послідовності.

Безпека будь-якого алгоритму зосереджена в ключі. Якщо використовується криптографічно слабкий процес для генерації ключів, то криптосистема в цілому слабка. Зловмисникові не потрібно криптоаналізувати алгоритм шифрування, він може криптоаналізувати алгоритм генерації ключів.

Теоретично, будь-який шифрувальний алгоритм із використанням ключа може бути розкритий методом перебору всіх значень ключа. Якщо ключ підбирається методом грубої сили (brute force), необхідна потужність комп'ютера росте експоненційно зі збільшенням довжини ключа. Ключ довжиною в 32 біта вимагає 2^{32} (близько 10^9) кроків.

Тоді мінімальну довжину ключа в бітах N_{min} можна обчислити по формулі:

$$N_{\min} > \lceil \log_2 (T \cdot V) \rceil \quad (1.2)$$

де

T – час життя переданих даних (визначається відповідно до законодавства по ступені таємності переданих даних). У цьому випадку час життя переданих даних повинне вибиратися не менш 48 годин.

V – швидкість підбора ключів, залежить від класу алгоритму шифрування й використовуваних зловмисником ресурсів і може коливатися від 103 до 106 і вище, при використанні суперЕОМ або розподілених обчислень.

$\lceil\ \rceil$- оператор округлення до найближчого більшого цілого.

Варто врахувати, що деякі алгоритми можуть мати еквівалентні ключі (в іноземній літературі їх називають ключі-доповнення) K^* такі, що виконується рівність:

$$f^{-1}(C, K)=f^{-1}(C, K^*)=f^*(C, K^*)=P, \quad (1.3)$$

де f^* – функція, при використанні еквівалентного ключа K^* , що дає такий же результат, як і функція розшифрування f^{-1} алгоритму.

При цьому може виявитися, що зловмисникові потрібно перебрати тільки деяку частину безлічі ключів.

Приклад для режиму накладення гами: $f(P, K)=P \oplus Г=C$, тоді легальний користувач обчислить $f^{-1}(C, K)=C \oplus Г=P$.

Зловмисник може знайти такий ключ $K^* = \overline{Г}_i$ і обчислити $f(C, K^*) = C \oplus \overline{Г} = P$. У цьому випадку, еквівалентним ключем є інверсія основного ключа, а f^* – інверсія функції додавання по модулю 2.

У підсумку зловмисникові необхідно замість 2^m операцій перебору виконати 2^{m-1} операцій, де m – розрядність ключової комбінації.

З обліком вищесказаного перепишемо формулу (1.2) у вигляді:

$$N_{\min} > \lceil \log_2(T \cdot V \cdot \Psi) \rceil, \quad (1.4)$$

Ψ – коефіцієнт, що враховує наявність еквівалентних ключів. Для алгоритмів шифрування на основі гаммування Ψ = 2÷3.

Для обчислення Ψ для інших алгоритмів криптографічного захисту інформації необхідно проводити дослідження для виявлення еквівалентних ключів.

Варто врахувати, що обчислювальна потужність обчислювальних засобів подвоюється кожні 18÷24 місяців (Δt = 1,5÷2 роки) – емпіричний закон Мура. Якщо необхідно, щоб ключі були стійкі до розкриття грубою силою протягом 5 років, то необхідно відповідним чином планувати використання ключів.

З обліком вищесказаного перепишемо формулу (1.4) у вигляді:

$$N_{\min} > \left\lceil \log_2 \left(\Psi \cdot \sum_{i=0}^{ceil\left(\frac{\tau}{\Delta t}\right)-1} \left[2^i \cdot \left(V \cdot 31536 \cdot 10^3 \right) \right] \right) \right\rceil \quad (1.5)$$

де

$[T]=[рік]$;

Δt – строк, за який обчислювальна потужність обчислювальних засобів подвоюється (параметр подвоєння), рік.

Варто врахувати, що параметр Δt потрібно час від часу переглядати, особливо в періоди зміни поколінь мікросхем і впровадження нових досягнень у технології. Поки обґрунтованим виглядає подання динаміки росту обчислювальної потужності з наступними значеннями параметра подвоєння: $\Delta t = 2$ роки в період з 1971 р. по 1993 р., $\Delta t = 4$ роки в період з 1993 р. по 1999 р. і $\Delta t = 0.6$ року від 1999 р. Але ці числа дають тільки частину відповіді. Справа в тому, що машина свідомо розкриває ключ за рік, має 8% шанс розкрити ключ за місяць. Якщо при цьому ключ міняють 1 раз на місяць, тобто 8% імовірність розкрити ключ ще під час його використання. Більше того, нехай є машина, що відшукує ключ за місяць, а ключ міняється щогодини. Незважаючи на те, що ймовірність знайти даний ключ за годину всього 0,14%, імовірність знайти правильний ключ до його зміна за місяць використання такої схеми = 63%, причому ця цифра не залежить від частоти зміна ключа. Таким чином, часта зміна ключів дозволяє хіба що мінімізувати наслідки злому системи, але в багатьох системах це однаково неприпустимий ризик.

Деякі алгоритми шифрування мають криптографічно слабкі й напівслабкі ключі – специфічні ключі, менш безпечні чим інші ключі. Варто використовувати спеціальні алгоритми, які використовуються для перевірки при генерації ключів для усунення можливості виробітку криптографічно слабких (напівслабких) ключів.

Таким чином провівши дослідження використання різних алгоритмів захисту інформації, можна дійти висновку, що використання потокових шифрів

дозволяє у режимі реального часу забезпечувати захист інформації на flash-накопичувачі. Тому у наступних розділах необхідно розглянути такі питання, як:

– архітектуру flash-накопичувачів;

– вибрати та описати потоковий шифр;

– розглянути генератори псевдовипадкових чисел (ГПВЧ).

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи для забезпечення конфіденційності інформації з'ємних носіїв, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

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

Огляд flash-пам'яті

Flash-пам'ять – особливий вид енергонезалежної перезаписуваної напівпровідникової пам'яті.

- Енергонезалежна – не потребує додаткової енергії для зберігання даних (енергія потрібно тільки для запису).

- Перезаписувана – перезапис, що допускає зміну збережених у ній даних.

- Напівпровідникова (твірдотільна) – не утримуюча механічно, що рухаються частин (як звичайні жорсткі диски або CD/DVD), побудована на основі інтегральних мікросхем (IC-Chip).

На відміну від багатьох інших типів напівпровідникової пам'яті, осередок flash-пам'яті не містить конденсаторів – типовий осередок flash-пам'яті складається всього-на-всього з одного транзистора особливої архітектури. Осередок flash-пам'яті прекрасно масштабується, що досягається не тільки завдяки успіхам у мініатюризації розмірів транзисторів, але й завдяки конструктивним знахідкам, що дозволяють в одному осередку flash-пам'яті зберігати трохи біт інформації. Flash-пам'ять історично походить від ROM (Read Only Memory) пам'яті, і функціонує подібно RAM (Random Access Memory). Дані flash зберігає в комірках пам'яті, схожих на осередки в DRAM. На відміну від DRAM, при відключенні живлення дані з flash-пам'яті не пропадають. Заміни пам'яті SRAM і DRAM flash-пам'яттю не відбувається через дві особливості flash-пам'яті: flash працює істотно повільніше й має обмеження по кількості циклів

перезапису (від 10.000 до 1.000.000 для різних типів). Надійність/довговічність: інформація, записана на flash-пам'ять, може зберігатися дуже тривалий час (від 20 до 100 років), і здатна витримувати значні механічні навантаження (в 5-10 разів перевищуючі гранично припустимі для звичайних жорстких дисків). Основна перевага flash-пам'яті перед жорсткими дисками й носіями CD/DVD -ROM полягає в тому, що flash-пам'ять споживає значно (приблизно в 10-20 і більше раз) менше енергії під час роботи. У пристроях CD/DVD-ROM, жорстких дисках, касетах і інших механічних носіях інформації, більша частина енергії йде на надавання руху механіки цих пристроїв. Крім того, flash-пам'ять компактніше більшості інших механічних носіїв. Flash-пам'ять історично відбулася від напівпровідникового ROM, однак ROM-пам'яттю не є, а всього лише має схожу на ROM організацію. Безліч джерел (як вітчизняних, так і закордонних) найчастіше помилково відносять flash-пам'ять до ROM. Flash ніяк не може бути ROM хоча б тому, що ROM (Read Only Memory) переводиться як "пам'ять тільки для читання". Ні про яку можливість перезапису в ROM мови бути не може! Невелика, по початку, неточність не обертала на себе уваги, однак з розвитком технологій, коли flash-пам'ять стала витримувати до 1 мільйона циклів перезапису, і стала використовуватися як накопичувач загального призначення, цей недолік у класифікації почав впадати в око. Серед напівпровідникової пам'яті тільки два типи відносяться до "чистого" ROM – це Mask-ROM і PROM. На відміну від них EPROM, EEPROM і Flash відносяться до класу енергонезалежної перезаписуваної пам'яті (англійський еквівалент – nonvolatile read-write memory або NVRWM).

ROM:

- ROM (Read Only Memory) – пам'ять тільки для читання. Український еквівалент – ПЗП (Постійно Запам'ятовувальний Пристрій). Якщо бути зовсім точним, даний вид пам'яті називається Mask-ROM (Масочние ПЗП). Пам'ять улаштована у вигляді адресуємого масиву осередків (матриці), кожний осередок якого може кодувати одиницю інформації. Mask-ROM відрізняється складністю

модифікації вмісту (тільки шляхом виготовлення нових мікросхем), а також тривалістю виробничого циклу (4-8 тижнів). Тому, а також у зв'язку з тим, що сучасне програмне забезпечення найчастіше має багато недоробок і часто вимагає відновлення, даний тип пам'яті не одержав широкого поширення.

Переваги:

1. Низька вартість готової запрограмованої мікросхеми (при великих обсягах виробництва).

2. Висока швидкість доступу до комірки пам'яті.

3. Висока надійність готової мікросхеми й стійкість до електромагнітних полів.

Недоліки:

1. Неможливість записувати й модифікувати дані після виготовлення.

2. Складний виробничий цикл.

- PROM – (Programmable ROM), або однократно Програмувальні ПЗП. Як осередки пам'яті в даному типі пам'яті використовувалися плавкі перемички. PROM практично повністю вийшов із уживання наприкінці 80-х років.

Переваги:

1. Висока надійність готової мікросхеми й стійкість до електромагнітних полів.

2. Можливість програмувати готову мікросхему, що зручно для штучного й дрібносерійного виробництва.

3. Висока швидкість доступу до комірки пам'яті.

Недоліки:

1. Неможливість перезапису

2. Великий відсоток браку

3. Необхідність спеціального тривалого термічного тренування, без якої надійність зберігання даних була невисокою

NVRWM:

- EPROM – Різні джерела по-різному розшифровують абревіатуру EPROM – як Erasable Programmable ROM або як Electrically Programmable ROM (стираємо програмувальні ПЗП або електрично програмувальні ПЗП). В EPROM перед записом необхідно зробити стирання. Стирання осередків EPROM виконується відразу для всієї мікросхеми за допомогою опромінення чипа ультрафіолетовими або рентгенівськими променями протягом декількох мінут.

Переваги: Можливість перезаписувати вміст мікросхеми

Недоліки:

1. Невелика кількість циклів перезапису.

2. Неможливість модифікації частини збережених даних.

3. Висока ймовірність "недотерти" або перетримати мікросхему під уф-світлом, що може зменшити термін служби мікросхеми й навіть привести до її повної непридатності.

- EEPROM (EEPROM або Electronically EPROM) – стираєма електрично ППЗП. Головною відмінною рисою EEPROM (у т.ч. Flash) від раніше розглянутих нами типів енергонезалежної пам'яті є можливість перепрограмування при підключенні до стандартної системної шини мікропроцесорного пристрою. В EEPROM з'явилася можливість робити стирання окремого осередку за допомогою електричного струму. Для EEPROM стирання кожного осередку виконується автоматично при записі в неї нової інформації, тобто можна змінити дані в будь-якому осередку, не торкати інші. Процедура стирання звичайно істотно довше процедури запису.

Переваги EEPROM у порівнянні з EPROM:

1. Збільшений ресурс роботи.

2. Простіше в обігу.

Недолік: Висока вартість

Flash (повна історична назва Flash Erase EEPROM) -використовується трохи відмінний від EEPROM тип осередку-транзистора. Технологічно flash-

пам'ять родинна як EPROM, так і EEPROM. Основна відмінність flash-пам'яті від EEPROM полягає в тому, що стирання вмісту осередків виконується або для всієї мікросхеми, або для певного блоку (кластера, кадру або сторінки). Звичайний розмір такого блоку становить 256 або 512 байт, однак у деяких видах flash-пам'яті обсяг блоку може досягати 256 Кб. Варто помітити, що існують мікросхеми, що дозволяють працювати із блоками різних розмірів (для оптимізації швидкодії). Стирати можна як блок, так і вміст всієї мікросхеми відразу. Таким чином, у загальному випадку, для того, щоб змінити один байт, спочатку в буфер зчитується весь блок, де втримується підлягаючій зміні байт, стирається вміст блоку, змінюється значення байта в буфері, після чого виробляється запис зміненого в буфері блоку. Така схема істотно знижує швидкість запису невеликих обсягів даних у довільні області пам'яті, однак значно збільшує швидкодію при послідовному записі даних більшими порціями.

Переваги flash-пам'яті в порівнянні з EEPROM:

1. Більш висока швидкість запису при послідовному доступі за рахунок того, що стирання інформації в flash виробляється блоками.

2. Собівартість виробництва flash-пам'яті нижче за рахунок більше простої організації.

Недолік: Повільний запис у довільні ділянки пам'яті.

Опис потокових шифрів

У потокових шифрах, тобто при шифруванні потоку даних, кожний біт вихідної інформації шифрується незалежно від інших за допомогою гаммування. Гаммування – накладення на відкриті дані гами шифру (випадкової або псевдовипадкової послідовності одиниць і нулів) за певним правилом. Звичайно використовується "АБО що виключає ", називане також додаванням по модулю 2 і реалізоване в ассемблерних програмах командою XOR. Для розшифровування та ж гама накладається на зашифровані дані.

При однократному використанні випадкової гами однакового розміру із зашифровуваними даними взлом коду неможливий (так звані криптосистеми з

одноразовим або нескінченним ключем). У цьому випадку "нескінченний" означає, що гама не повторюється.

У деяких потокових шифрах ключ коротше повідомлення. Так, у системі Вернама для телеграфу використовується паперове кільце, що містить гаму. Звичайно, стійкість такого шифру не ідеальна.

Зрозуміло, що обмін ключами розміром із шифруємою інформацію не завжди доречний. Тому частіше використовують гаму, одержувану за допомогою генератора псевдовипадкових чисел (ПВЧ). У цьому випадку ключ – число, що породжує (початкове значення, вектор ініціалізації, initializing value, IV) для запуску генератора ПВЧ. Кожний генератор ПВЧ має період, після якого генеруєма послідовність повторюється. Очевидно, що період псевдовипадкової гами повинен перевищувати довжину шифруємої інформації.

Генератор ПВЧ вважається коректним, якщо спостереження фрагментів його виходу не дозволяє відновити пропущені частини або всю послідовність при відомому алгоритмі, але невідомому початковому значенні.

При використанні генератора ПВЧ можливі кілька варіантів:

1. Побітове шифрування потоку даних. Цифровий ключ використовується як початкове значення генератора ПВЧ, а вихідний потік біт підсумується по модулю 2 з вихідною інформацією. У таких системах відсутнє властивість поширення помилок.

2. Побітове шифрування потоку даних зі зворотним зв'язком (ЗЗ) по шифртексту. Така система аналогічна попередній, за винятком того, що шифртекст вертається як параметр у генератор ПВЧ. Характерна властивість – поширення помилок. Область поширення помилки залежить від структури генератора ПВЧ.

3. Побітове шифрування потоку даних з ЗЗ по вихідному тексті. Базою генератора ПВЧ є вихідна інформація. Характерна властивість необмеженого поширення помилки.

4. Побітове шифрування потоку даних з ЗЗ по шифртексту й по вихідному тексту.

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. Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

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

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

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

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

RAD Studio 10.4 Короткий огляд:

− Істотні розширення для Windows. Створення застосунків, що чудово виглядають, із чіткими елементами інтерфейсу на 4k моніторах High DPI за допомогою нової гнучкої підтримки стилів елементів керування на екрані. Інтеграція із сучасними, безпечними web-технологіями від Microsoft – новим WebView2 на базі Chromium. Використання сучасних розширених title bars, таких

же, як в Office, Explorer, Google Chrome, у своїх проектах. Істотні поліпшення надійності налагодження в новому відладнику для C++ Windows 64-bit.

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

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

Істотне поліпшення Delphi Code Insight

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

Delphi Custom Managed Records

Ключове розширення мови Delphi: тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання.

Управляйте тем, як ці структури створюються, копіюються й звільняються з допомогу вашого коду, який буде виконуватися у відповідний момент.

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

Єдине керування пам'яттю

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

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

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

Розширена підтримка бібліотек C++

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

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

Win 64-відладник і збирач для C++

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

стабільному й багатому на можливості процесу налагодження, більш докладним перегляду й інспекції в debug-time.

Підвищення якості й швидкодії інструментів

– Велика кількість поліпшень STL від Dinkumware.

– Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.

– Поліпшена підтримка Cmake.

– Велика кількість виправлень для підвищення стабільності і якості.

– Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.

– Загальні вдосконалення в бібліотеці доступу до БД FireDAC, включаючи оновлені драйвера для FireBird, PostgreSQL і SQLite. Вибір статичного або динамічного підключення SQLite до застосунку.

Змінені стилі VCL для High DPI

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

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

Нові High DPI стилі й стилізація окремих VCL компонент

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

Розроблювачі VCL застосунків тепер можуть використовувати трохи VCL стилів на різних формах в одному застосунку або в різних компонентів на одній формі. Це також включає стилізацію компонентів загальною темою для платформи. Крім застосункової гнучкості використання стилів, це дозволяє

використовувати нестилізуємі компоненти із зовнішніх бібліотек в VCL застосунках, що використовуть стиль.

Поліпшена кроссплатформеність

– Додана підтримка Metal Driver GPU для macOS і iOS.

– Крім підтримки останнього iOS SDK, в RAD Studio 10.4 розроблювачі можуть задовольнити нові вимоги Apple до набору стартових екранів.

– Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

– Користувачам редакцій Enterprise або Architect доступна повна інтеграція Fmxlinux з IDE для створення клієнтських застосунків Linux з GUI.

– Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.

– Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

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

Універсальний інсталятор для установки Online і Offline

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

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

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи для забезпечення конфіденційності інформації з'ємних носіїв.

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 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.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Опис генератора псевдовипадкових чисел

Послідовності випадкових чисел (ВЧ) є невід'ємним інструментом рішення багатьох математичних завдань, і в тому числі завдань захисту інформації. Для потокових шифрів він використовується у ролі гами для накладання на дані, які потрібно захищати. Найчастіше необхідні послідовності випадкових чисел, які рівномірно й рівноймовірно розподілені на деякому відрізку. Більшість природних процесів є випадковими, і, відповідно до теорії математичної статистики, вони підкоряються різним законам розподілу випадкових величин. Для породження випадкових чисел, рівномірно розподілених на відрізку [0..*M*], досить для деякого випадкового процесу, що підкоряється закону рівномірного розподілу, увести міру випадкової величини. А потім послідовно проводити експерименти, вимірювати значення випадкової величини й після нормування одержувати необхідну рівномірну випадкову послідовність чисел.

Подібні методи дадуть дуже гарні статистичні результати, але зажадають колосального часу для одержання скільки-небудь довгої послідовності. На щастя, математики розробили рекурентні формули одержання псевдовипадкових чисел (ПВЧ). Назва «псевдовипадкові» обумовлена хоча б тим фактом, що якщо відомо деяке і-*е* число, то по формулі однозначно обчислимо $(i + 1)$-ий елемент послідовності. Більше того, з рекурентної природи знайденої формули виходить, що при тому самому x_0 ми одержимо при повторній генерації ту ж саму послідовність. До того ж, всі арифметичні алгоритми генерації ПВЧ періодичні, тобто існує деяке $p \in N$, для якого виконується $x_i = x_{i+pn}$ при будь-якому натуральному n. Тому будь-який ГПВЧ повинен бути ініціалізований випадковою

величиною, якимось зовнішнім джерелом випадкових значень. Такими «апаратними» у ПК генераторами можуть виступати, наприклад, електричні шуми в напівпровідникових пристроях, а також поточна кількість виконаних процесором тактів або поточне значення часу.

Недоліки ГПВЧ

- порівняно короткий період генеруємої послідовності
- залежність між сусідніми послідовними значеннями
- нерівномірність розподілу значень, у тому числі через різний ступінь

Ініціалізація ГПВЧ

Читання поточного значення мілісекунд

Як уже було сказано вище, рекурентну формулу обчислення і-*го* члена ПВЧ при реалізації на ПК можна ініціалізувати значенням мілісекунд поточного часу в момент запуску програми. Для цього можна викликати функцію «2Ch» переривання 21h операційної системи DOS і одержати «майже» випадкове число на відрізку [0..99].

Читання значення лічильника тактів процесора

Починаючи з лінійки процесорів Pentium в архітектурі x86 з'явилася інструкція, що дозволяє прочитати лічильник тактів процесора з моменту останнього скидання. Для інструкції *rdtsc* (Read Time Stamp Counter) заданий машинний код 0F 31. 64-бітне значення лічильника вертається в парі регістрів <EDX:EAX>. Якщо для ініціалізації генератора досить 32-бітного значення, то необхідно використовувати найбільше «чутливі» молодші 32 біта лічильника тактів. При використанні старого компілятора мови асемблера для звертання до 32-бітного регістра EAX знадобиться вручну проставити префікс із кодом 66*h*.

Лінійний конгруентний метод

Рекурентна формула виглядає в такий спосіб:

$$x_n = (ax_{n-1} + c) \bmod m. \qquad (3.1)$$

Період породжуваної послідовності не перевищує m. Природно, що від вибору параметрів a, c, m і значення першого члена x_0 істотно залежать основні

властивості породжуваної послідовності. Довжина періоду буде максимальною (рівна m) тільки в тому випадку, коли:

- НЗД(c, m) = 1 (тобто c і m взаємно прості)
- $a - 1$ кратно всім простим дільникам m
- якщо m кратно 4, то й $(a - 1)$ *mod* 4 = 0

Алгоритм Блюма-блюма-Шуба

Алгоритм ГПВЧ, стійкий до зворотних перетворень. Основна рекурентна формула алгоритму:

$$x_n = (x_{n-1})^2 \ mod \ pq, \qquad z_n = parity\ (x_n), \tag{3.2}$$

де p і q – два великих простих числа. Для підвищення якості одержуваної послідовності на черговому кроці вибираються не всі біти x_n, а тільки молодші, або навіть тільки біт парності. З отриманих «випадкових біт» формуються двійкові ПВЧ довільної розрядності. Однієї з особливостей обчислювальної формули є наскрізна можливість обчислити x_n без генерації всіх попередніх членів послідовності.

$$x_n = (x_0)^{2n \ mod \ (p-1)(q-1)} \ mod \ pq, \tag{3.3}$$

Даний алгоритм більше вимогливий до обчислювальних ресурсів, але, з іншого боку, має гарні статистичні характеристики.

Алгоритм xor-shift

Професором університету Флориди Джорджем Марсаглією був розроблений дуже швидкий генератор, що був названий « xor-shift».

$$x = 3456789,\ y = 62436069,\ z = 1288629,\ w = 675123 \tag{3.4}$$

$$t = (x \text{ xor } (x \text{ shl } 11)), \quad x = y, \quad y = z, \quad z = w, \tag{3.5}$$

$$\text{xor128} = w = (w \text{ xor } (w \text{ shr } 19)) \text{ xor } (t \text{ xor } (t \text{ shr } 8))) \tag{3.6}$$

Існує комбінація даного алгоритму з лінійним конгруентним алгоритмом і алгоритмом Фібоначчі із запізнюваннями.

$$x = 123456789, \quad y = 362436000, \quad z = 521288629, \quad c = 7654321, \tag{3.7}$$

$$x = \text{int64}(69069) \cdot x + 12345, \quad y = y \text{ xor } (y \text{ shl } 13), \tag{3.8}$$

$$y = y \text{ xor } (y \text{ shr } 17), \quad y = y \text{ xor } (y \text{ shl } 5), \tag{3.9}$$

$$t = \text{int64}(698769069)\cdot z + c, \quad c = t \text{ shr } 32, \tag{3.10}$$

$$z = t, \quad \text{Random} = x + y + z, \tag{3.11}$$

Опис алгоритму шифрування

У якості криптоалгоритму спрямованого на захист інформації, яка утримується в flash-пристрої візьмемо алгоритм RC4. Розглянутий нами криптоалгоритм RC4 відноситься до класу потокових шифрів, які останнім часом стали популярними завдяки високій швидкості роботи. Потокові шифри перетворять відкритий текст у шифротекст по одному біті за операцію. Генератор потоку ключів (іноді називаний генератором із ключем, що біжить) видає потік біт: k_1, k_2, k_3, ..., k_i. Цей потік ключів і потік біт відкритого тексту, p_1, p_2, p_3, ..., p_i, піддаються операції “або, що виключає", і в результаті виходить потік біт шифротексту.

$$c_i = p_i \oplus k_i \tag{3.12}$$

При дешифруванні операція XOR виконується над бітами шифротексту й тим же самим потоком ключів для відновлення біт відкритого тексту.

$$p_i = c_i \oplus k_i \tag{3.13}$$

Безпека системи повністю залежить від властивостей генератора потоку ключів. Генератор потоку ключів створює бітовий потік, що схожий на випадковий, але в дійсності детермінований і може бути безпомилково відтворений при дешифруванні. Чим ближче вихід генератора потоку ключів до випадкового, тим більше часу буде потрібно для взлому шифру.

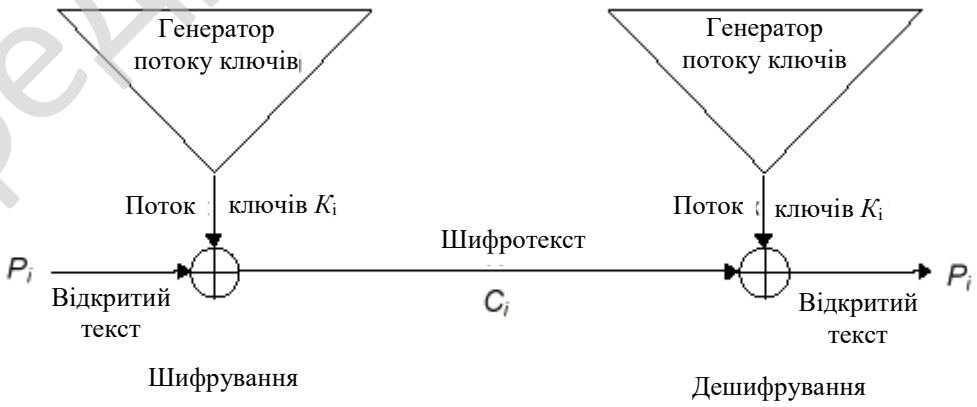

Рисунок 3.1 – Потоковий шифр

Для всіх потокових шифрів використовуються ключі. Вихід генератора потоку ключів є функцією ключа. Тепер, якщо одержати пару відкритий текст/шифротекст, то можна читати тільки ті повідомлення, які зашифровані тим же ключем. Потокові шифри особливо корисні для шифрування нескінченних потоків комунікаційного трафіку, наприклад, при записі данних на flash-пам'ять.

Генератор потоку ключів складається із трьох основних частин:

– Внутрішній стан описує поточний стан генератора потоку ключів.

– Два генератори потоку ключів, з однаковим ключем і однаковим внутрішнім станом, видають однакові потоки ключів.

– Функція виходу по внутрішньому стану генерує біт потоку ключів.

– Функція наступного стану по внутрішньому стану генерує новий внутрішній стан.

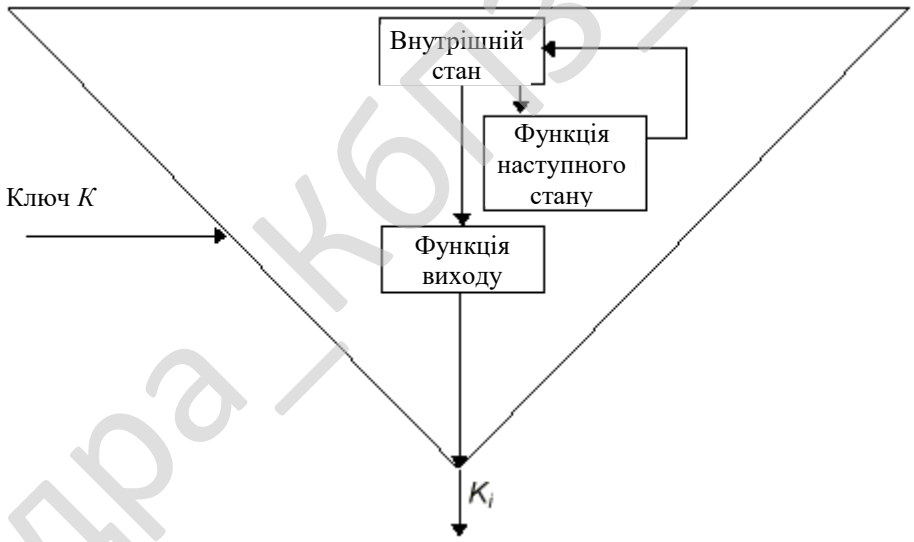

Рисунок 3.2 – Пристрій генератора потоку ключів

Криптоалгоритм RC4 відноситься до так званих шифрів, що самосинхронізуються. У потокових шифрах, що самосинхронізуються, кожний біт потоку ключів є функцією фіксованого числа попередніх біт шифротексту. Військові називають цей шифр автоключом шифротексту.

Потоковий шифр, що самосинхронізується, показаний на рисунку 3.3. Внутрішній стан є функцією попередніх n біт шифротексту. Криптографічно

складною є вихідна функція, що використовує внутрішній стан для генерації біта потоку ключів.

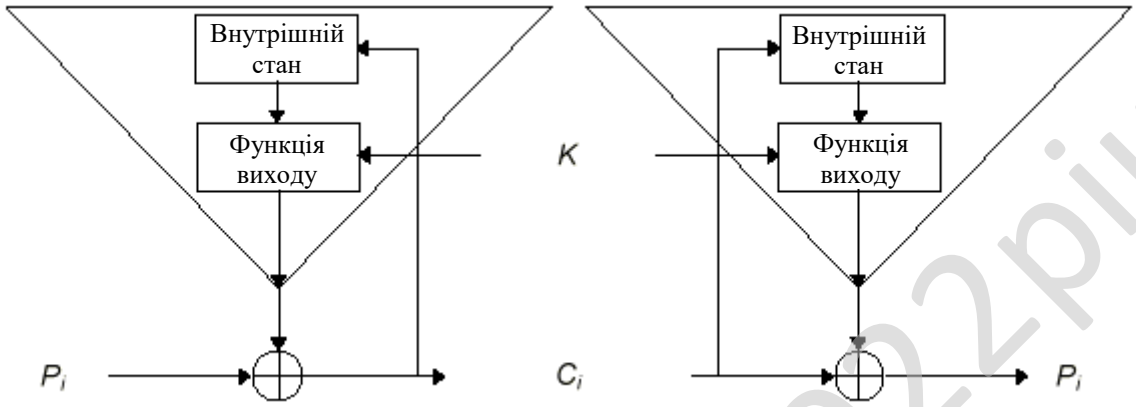

Рисунок 3.3 – Генератор потоку ключів, що самосинхронізується

Так як внутрішній стан повністю залежить від попередніх *n* шифротексту, дешифруючий генератор потоку ключів автоматично синхронізується з генератором, що шифрує, потоку ключів, прийнявши *n* біт шифротексту. В інтелектуальних реалізаціях цього режиму кожне повідомлення починається випадковим заголовком довжиною *n* біт.

Цей заголовок шифрується, передається й потім розшифровується. Розшифровка буде неправильною, але після цих *n* біт обидва генератори потоку ключів будуть синхронізовані.

Слабкою стороною потокового шифру, що самосинхронізується, є поширення помилки. Для кожного біта шифротексту, зіпсованого при передачі, дешифруючий генератор потоку ключів видає n неправильних біт потоку ключів. Отже, кожному неправильному біту шифротексту відповідають n помилок у відкритому тексті, поки зіпсований біт не перестане впливати на внутрішній стан.

Алгоритм RC4 і його криптоаналіз

Істотне підвищення продуктивності мікропроцесорів в 80-і роки викликало в криптографії посилення інтересу до програмних методів реалізації

криптоалгоритмів як можливій альтернативі апаратним схемам на регістрах зрушення.

Одним з найперших подібних криптоалгоритмів, що получили широке поширення, став RC4. Алгоритм RC4 – це потоковий шифр зі змінною довжиною ключа.

Він володіє наступними властивостями:

- адаптивністю для апаратних засобів і програмного забезпечення, що означає використання в ньому тільки примітивних обчислювальних операцій, звичайно присутніх на типових мікропроцесорах;

- алгоритм швидкий, тобто в базисних обчислювальних операціях оператори працюють на повних словах даних;

- адаптивністю на процесори різних довжин слова;

- компактністю в термінах розміру коду, і особливо зручний для процесорів з побайтно-орієнтованою обробкою;

- низькою вимогою до пам'яті, що дозволяє реалізовувати алгоритм на пристроях з обмеженою пам'яттю;

- використанням циклічних зрушень, залежних від даних, з "змінним" числом;

- простотою й легкістю виконання.

У цей час алгоритм RC4 реалізований у десятках комерційних криптографічних продуктів, включаючи Lotus Notes, Apple Computer's AOCE, Oracle Secure SQL, а також є частиною специфікації стандарту стільникового зв'язка CDPD.

Криптогенератор функціонує незалежно від відкритого тексту. Генератор має підстановочну таблицю (S-бокс 8 х 8): S_0, S_1, . . ., S_{255}. Входами генератора є замінені по підстановці числа від 0 до 255, і ця підстановка є функцією від ключа змінюваної довжини. Генератор має два лічильники i і j, ініціалізуємих нульовим значенням.

Для генерації випадкового байта гами виконуються наступні операції:

$$i = (i+1) \bmod 256 \quad (3.14)$$

$$j = (j+S_i) \bmod 256 \quad (3.15)$$

$$\text{swap}\ (S_i, S_j) \quad (3.16)$$

$$t = (S_i+S_j) \bmod 256 \quad (3.17)$$

$$K = S_t \quad (3.18)$$

Байт K складається операцією XOR з відкритим текстом для виробітку шифротексту, або із шифротекстом для одержання байта відкритого тексту. Шифрування відбувається досить швидко – приблизно в 10 разів швидше DES-Алгоритму. Ініціалізація S-боксу настільки ж проста. На першому кроці він заповнюється лінійно: $S_0 = 0$, $S_1 = 1$, . . ., $S_{255} = 255$.

Потім ще один 256-байтний масив повністю заповнюється ключем, для чого ключ повторюється відповідне число раз залежно від довжини: K_0, K_1,...,K_{255}. Індекс j обнуляется. Потім:

```
for (i=0; i<= 255; i++)
{
     j = (j+Si+Ki) mod 256;
     swap (Si , Sj);
}
```

Схема показує, що RC4 може приймати приблизно 2^{1700} (256! * 256^2) можливих станів. S-бох повільно змінюється в процесі роботи: параметр i забезпечує зміну кожного елемента, а j відповідає за те, щоб ці елементи змінювалися випадковим образом.

Фактично, RC4 являє собою сімейство алгоритмів, що задаються параметром n, що є позитивним цілим з рекомендованим типовим значенням $n = 8$.

Внутрішній стан генератора RC4 у момент часу t складається з таблиці $S_t = (S_t(L))_{t=0}^{n2^n-1}$, що містить 2^n n-бітних слів і із двох n-бітних слів-покажчиків i_t і j_t. Таким чином, розмір внутрішньої пам'яті становить $M = n2^n + 2n$ біт. Нехай вихідне n-бітне слово генератора в момент t позначається як Z_t.

Нехай початкові значення $i_0 = j_0 = 0$. Тоді функція наступного стану й функція виходу RC4 для кожного $t > 1$ задається наступними співвідношеннями:

$$i_t = i_{t-1} + 1 \quad (3.19)$$

$$j_t = j_{t-1} + S_{t-1}(i_t) \quad (3.20)$$

$$S_t(i_t) = S_{t-1}(j_t) \quad (3.21)$$

$$S_t(j_t) = S_{t-1}(i_t) \quad (3.22)$$

$$Z_t = S_t(S_t(i_t) + S_t(j_t)), \quad (3.23)$$

де всі додавання виконуються по модул. 2^n. Мається на увазі, що всі слова, крім тих, які піддаються перестановці, залишаються тими ж самими. Вихідна послідовність n-бітних слів позначається як $Z_t = (Z_t)_{t=1}^{\infty}$.Початкова таблиця S_0 задається в термінах ключової послідовності:

$$K = (K_L)_{t=0}^{2n-1} \quad (3.24)$$

з використанням тієї ж самої функції наступного стану, починаючи від таблиці одиничної підстановки $(L)^{2n}{}_{L=0}{}^{2n-1}$. Більш строго, нехай $j_0 = 0$ і для кожного $1 \le t \le 2^n$ обчислюється $j_t = (j_t - 1 + S_{t-1}(t\text{-}1) + K_{t-1}) \bmod 2n$, а потім переставляються місцями $S_{t-1}(t\text{-}1)$ і $S_{t-1}(j_t)$.

На останньому кроці породжується таблиця, що представляє S_0. Ключова послідовність K складається із секретного ключа, що можливо повторюється, і випадкового ключа, переданого у відкритому виді з метою ресинхронизації.

До останнього часу у відкритій літературі практично не було публікацій по криптоаналізу алгоритму RC4. Компанія RSA Data Security оголосила, що шифр має імунітет до методів лінійного й диференціального криптоаналізу, високо не лінієн і не схоже, щоб у нього були короткі цикли.

Відзначається, що для послідовностей, генеруємих RC4, не підходять методи статистичного аналізу. Але, з іншого боку, для блоків, розмір яких перевищує M (розмір внутрішньої пам'яті генератора), завжди існує лінійна статистична слабість або так звана "лінійна модель". Таку модель можна ефективно визначати за допомогою методу апроксимації лінійною послідовною схемою. Лінійна статистична слабість – це лінійне співвідношення між бітами гами, що виконується з імовірністю, що відрізняється від 1/2.

За допомогою методу АЛПС були виведені лінійні моделі для RC4. Метод АЛПС полягає в знаходженні й рішенні послідовної лінійної схеми, що апроксимує генератор гами й приводить до лінійних моделей з відносно великим кореляційним коефіцієнтом c, де ймовірність відповідного лінійного співвідношення між бітами гами становить $(1 + c)/2$. При аналізі використовувалася техніка двійкових похідних. Нехай $Z = (Z_t)_{t=1}^{\infty}$ позначає послідовність самих молодших біт слів виходу RC4, і нехай $Z' = (Z'_t = Z_t + Z_{t+1})_{t=1}^{\infty}$ і $Z'' = (Z''_t = Z_t + Z_{t+2})_{t=1}^{\infty}$ позначають її перші й другу двійкові похідні, відповідно. Показано, що Z' не корелює ні з 1, ні з 0, але Z'' корелює з 1 з кореляційним коефіцієнтом, близьким до $15^* 2\text{-}3^n$ при великих $2n$, де n – довжина ключа. Оскільки довжина вихідної послідовності, необхідна для виявлення статистичної слабості з кореляційним коефіцієнтом c, становить O(c-2), те ця довжина дорівнює приблизно $64^n /225$. Наприклад, якщо $n = 8$, як рекомендується в більшості додатків, те необхідна довжина близька до 2^{40}.

Результати комп'ютерних експериментів погодяться з теоретичними пророкуваннями. Оскільки результуючий коефіцієнт кореляції істотно перевищує величину $2^{M/2}$, те встановлену лінійну модель варто розглядати як статистичну слабість генератора, принаймні в теоретичному аспекті.

Був проведений криптоаналіз узагальненої схеми вузла, що комбінує, з довільним розміром пам'яті. Досліджено кореляційні властивості таких вузлів, обґрунтовані нові конструктивні критерії, пропоновані до схем подібного типу. Розроблено ефективний метод апроксимації лінійною послідовною схемою для побудови лінійних функцій від входу й виходу з порівняно більшим коефіцієнтом взаємної кореляції. Це практична процедура, що дозволяє з високою ймовірністю знаходити пари взаємно коррельованих лінійних функцій (від якнайбільше $M + 1$ послідовних вихідних біт і якнайбільше $M + 1$ послідовних векторів входу) з порівняно більшими коефіцієнтами кореляції. Метод АЛПС складається в завданні й рішенні лінійної послідовної схеми (ЛПС), що апроксимує вузол, що комбінує, з пам'яттю. Ця ЛПС має додаткові незбалансовані входи й заснована на

лінійних апроксимаціях функції виходу й всіх компонентів функції наступного стану. Лінійна апроксимація булевої функції – це будь-яка лінійна функція, з якою задана булева функція скорельована. Описаний метод застосуємо до довільних вузлів, що комбінують, з пам'яттю без обмежень на функції виходу й наступний стан.

Спочатку відшукуються лінійні апроксимації функції виходу f і кожної з функцій-компонентів функції наступного стану F. Це еквівалентно вираженню кожної із цих $M + 1$ функцій у вигляді суми лінійної функції й незбалансованої функції. Якщо підлягаючої декомпозиції функція вже несбалансована, то можна вибрати константно-нульову лінійну функцію. Якщо підлягаюча декомпозиції функція статистично незалежна від деякої підмножини змінних, то кожна лінійна апроксимація з необхідністю повинна задіяти принаймні одну зі змінних цієї підмножини. Основна вимога – щоб відповідні кореляційні коефіцієнти відрізнялися від нуля. Також бажано, щоб вибиралися лінійні апроксимації з кореляційними коефіцієнтами, абсолютні значення яких близькі до максимального. Кореляційні коефіцієнти можна визначати за допомогою техніки перетворення Уолша.

На наступному кроці, одержавши лінійні апроксимації, у матричній формі записують базові рівняння вузла, що комбінує, з пам'яттю

$$S_{t+1} = A \cdot S_t + B \cdot X_t + \Delta(X_t, S_t),\ t \geq 0, \tag{3.25}$$

$$y_t = C \cdot S_t + D \cdot X_t + \varepsilon(X_t, S_t),\ t \geq 0, \tag{3.26}$$

де вектори розглядаються як матриці-стовпці; A, B, C, D – двійкові матриці; а ε і кожний компонент в $D = (d_1, \ldots, d)$ – незбалансовані булеві функції, іменовані функціями шуму. Основна ідея полягає в тому, щоб розглядати $\{\varepsilon(X_t, S_t)\}_{t=0}^{\infty}$ і $\{\delta(X_t, S_t)\}_{t=0}^{\infty}$, $1 \leq i \leq M$, як вхідні послідовності, так що останні рівняння виявляються задаючими неавтономну лінійну машину з кінцевим числом станів або ЛПС, іменовану АЛПС вузла, що комбінує, з пам'яттю. Тоді можна вирішувати цю ЛПС із використанням техніки виробляючих функцій (D-перетворень). Зокрема, нехай S, X, Δ, ε, у позначають виробляючі функції від

змінної z для послідовностей $\{S_t\}$, $\{X_t\}$, $\Delta(X_t, S_t)\}$, $\varepsilon(X_t, S_t)\}$, $y_t\}$, відповідно. Тоді рівняння зводяться до виду:

$$y = \left(D - \frac{C \cdot adj(zA - I)B}{\det(zA - I)} \right) X - \frac{C \cdot adj(zA - I)}{\det(zA - I)} (z\Delta + S_0) + \varepsilon \qquad (3.27)$$

де I – одинична матриця, $\det(z - I) = \phi(z)$, $\phi(0) = 1$, – багаточлен, зворотний до характеристичного багаточлена матриці переходів A ступеня, що не перевищує ранг A ($\leq M$); а елементи (приєднаної) матриці $\mathrm{adj}(z - I)$ – це поліноми від z ступеня не більше M-1. Обчислювальна складність для відшукання такого рішення становить $O(M^3(N+1))$. В іншому виді рішення можна переписати як

$$y = \frac{1}{\varphi(z)} \sum_{i=1}^{N} g_i(z) x_i + \frac{1}{\varphi(z)} \sum_{j=1}^{M} h_j(z)(z\delta_j + s_{j0}) + \varepsilon \qquad (3.28)$$

де x_i і δ_j позначають виробляючі функції для $\{x_{it}\}$ і $\{\delta_j(X_t, S_t)\}$, а ступеня поліномів $g_i(z)$ і $h_j(z)$ якнайбільше рівні M і M-1, $1 \leq i \leq N$, $1 \leq j \leq M$, відповідно. Взявши

$$\varphi(z) = \sum_{k=0}^{M} \varphi_k z^k, \quad g_i(z) = \sum_{k=0}^{M} g_{ik} z^k, \quad h_j(z) = \sum_{k=0}^{M-1} h_{jk} z^k \qquad (3.29)$$

рішення можна перетворити до виду:

$$\sum_{k=0}^{M} \varphi_k y_{l-k} = \sum_{i=1}^{N} \sum_{k=0}^{M} g_{ik} x_{i,l-k} + e\left(X_i^{M+1}, S_{l-M}\right), \qquad t \geq M, \qquad (3.30)$$

$$e\left(X_i^{M+1}, S_{l-M}\right) = \sum_{j=1}^{N} \sum_{k=0}^{M-1} h_{jk} \delta_j \left(X_{t-1-k}, S_{t-1-k}\right) + \sum_{k=0}^{M-1} \varphi_k \varepsilon\left(X_{t-k}, S_{t-k}\right), \qquad t \geq M, \qquad (3.31)$$

де мається на увазі, що вектор стану S_{t-k} – це функція від ($X_{t-k-1}{}^{M-k}$, S_{t-M}) для кожного $0 \leq k \leq M$-1. Лінійні функції входу й виходу в (3.30) скоррельовані тоді й тільки тоді, коли функція шуму e незбалансована. Коефіцієнт кореляції не залежить від часу, якщо функція наступного стану збалансована. Якщо ця умова не задовольняється, то кореляційний коефіцієнт може залежати від часу, оскільки від S_t більше не потрібна збалансованість для кожного $t \geq 0$. Функція шуму e в (3.31) визначена як сума індивідуальних шумових функцій, які незбалансовані за умови, що збалансовано функцію наступного стану. Оскільки від індивідуальних шумових функцій не потрібно бути незалежними, у принципі не можна

виключати можливість, що коефіцієнт кореляції *e* з константною нульовою функцією дорівнює нулю або дуже близький до цього значення.

У розглянутому випадку індивідуальні шумові функції можна трактувати як булеві функції від $n = MN + N + M$ змінних в ($X^{M+1}{}_t$, $S_{t\text{ -M}}$). Отже, за винятком деяких особливих випадків, у загальному випадку можна з високою ймовірністю очікувати, що загальний кореляційний коефіцієнт дуже близький до добутку індивідуальних і, таким чином, відрізняється від нуля. Відповідно, метод АПЛС не тільки з високою ймовірністю дає взаємно коррельовані лінійні функції від входу й виходу, але також дозволяє оцінити значення відповідного кореляційного коефіцієнта, використовуючи незалежність або інші імовірнісні припущення. Оскільки в ідеальному випадку хотілося б одержати такі АЛПС, у яких кореляційні коефіцієнти за абсолютним значенням близькі до максимуму, те індивідуальні кореляційні коефіцієнти повинні бути великими по величині, а кількість шумових членів в (3.31) повинне бути маленьким. Звичайно, ці вимоги можуть суперечити один одному. Тому гарним підходом буде повторення процедури АЛПС кілька разів, починаючи з найкращих лінійних апроксимацій для функції виходу й компонент функції наступного стану. Ця процедура може також виконуватися для всіх можливих лінійних апроксимацій, що представляється єдиним систематичним способом перевірити всі кореляції, виявлені в процесі застосування методу АЛПС. У загальному випадку є якнайбільше $(M+1)2^{M+N}$ таких лінійних апроксимацій. Однак, у принципі завжди можна перевірити всі можливі лінійні апроксимації навіть при великому *M*, оскільки в практичних реалізаціях функції виходу й наступного стану залежать від порівняно невеликої кількості змінних або ж складені з таких булевих функцій.

Із практичної точки зору дана лінійна модель може бути використана для виділення по шифротексту генератора RC4 серед інших криптосистем, а також для відновлення параметра *n*. В 2000 році була опублікована стаття присвячена статистичному аналізу потокового генератора RC4, у якій були використані

результати роботи для знаходження значення компонент *S*-боксу. Приблизний час роботи цього методу становить 2^{6n}, де n – порція біт у вихідному потоці, довжина вихідної послідовності, необхідна для виявлення статистичної слабості, близька до 2^{30}. Отриманий результат указує на істотну слабість генератора й можливість відновити параметри *i* і *n*. *S*-бокс може приймати 2^{nk}, де n_k – число біт ключа.

3.2 Розробка структурної схеми

На рисунку 3.4 зображена структурна схема роботи системи. Схема розділена на три основних компоненти:

- Flash накопичувач;
- розроблена програма;
- користувач.

Коли користувач вставляє в персональний комп'ютер Flash накопичувач, відбувається розпізнання операційною системою типу пристрою й виводу меню вироблених дій.

Розроблена програма перехоплює системне повідомлення операційній системі про виклик меню вироблених дій над Flash накопичувачем і активізує власний інтерфейс програми.

Розроблена програма складається з декількох частин:

- інтерфейсу програми;
- модуля потокового шифрування RC4;
- модуля потокового дешифрування RC4;
- налаштовування програми й захисту програми.

Розроблена програма управляє процесом обміну інформацією між Flash накопичувачем і персональним комп'ютером, використовуючи потоковий алгоритм шифрування інформації RC4. Завдяки такому підходу, можливо використовувати всі існуючі на даний момент Flash накопичувачі не

зупиняючись на окремих реалізаціях з підвищеними вимогами захищеності Flash накопичувача (Рисунок 3.4).

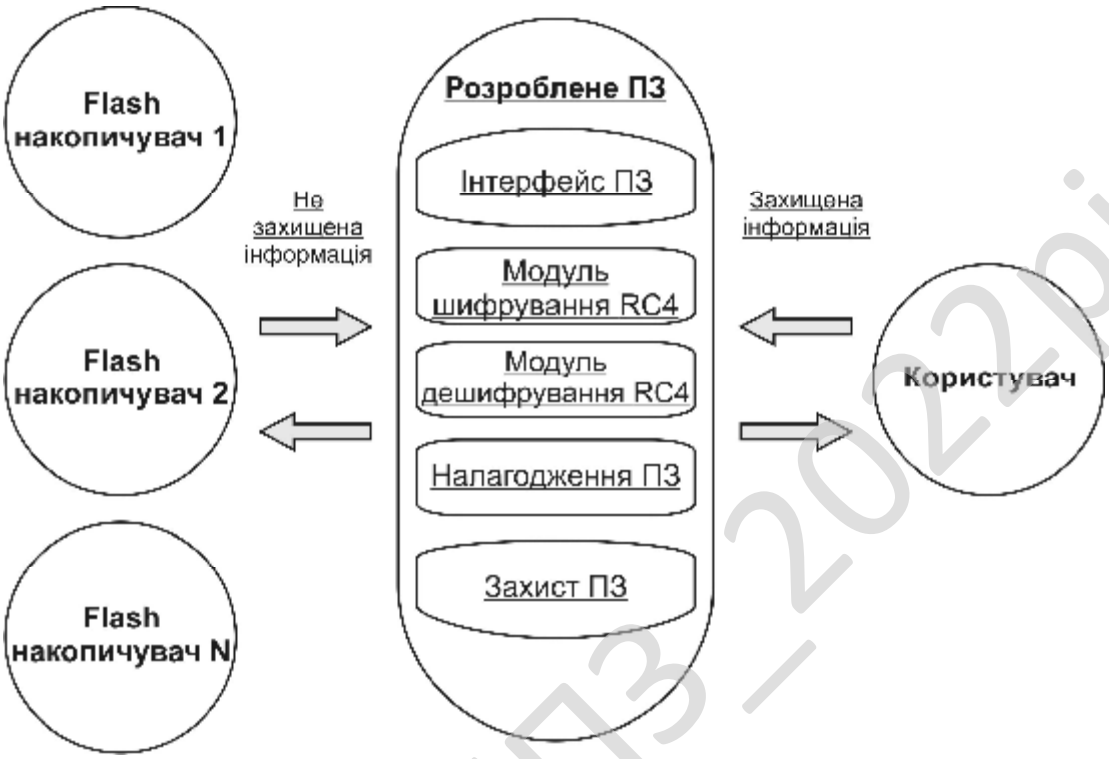

Рисунок 3.4 – Структурна схема роботи системи

3.3 Розробка функціональної схеми

Розглянемо поетапно функціональну схему (Рисунок 3.5). Функціональна схема – це схема, яка описує взаємодію й обробку даних. На функціональній схемі роботи системи представлений процес обробки інформації з Flash накопичувачів з використанням алгоритму потокового шифрування RC4.

Інформація надходить із Flash накопичувача в ядро алгоритму. Ядро алгоритму складається з функції генерації ключового потоку. Ця функція генерує послідовність біт, що потім поєднується з відкритим текстом за допомогою підсумовування по модулю два.

Процес дешифрації складається з регенерації цього ключового потоку й підсумовування його із шифрограммою по модулю два, відновлюючи вихідний текст. Також важливий елемент алгоритму функція ініціалізації, використовує ключ змінної довжини для створення початкового стану генератора ключового потоку.

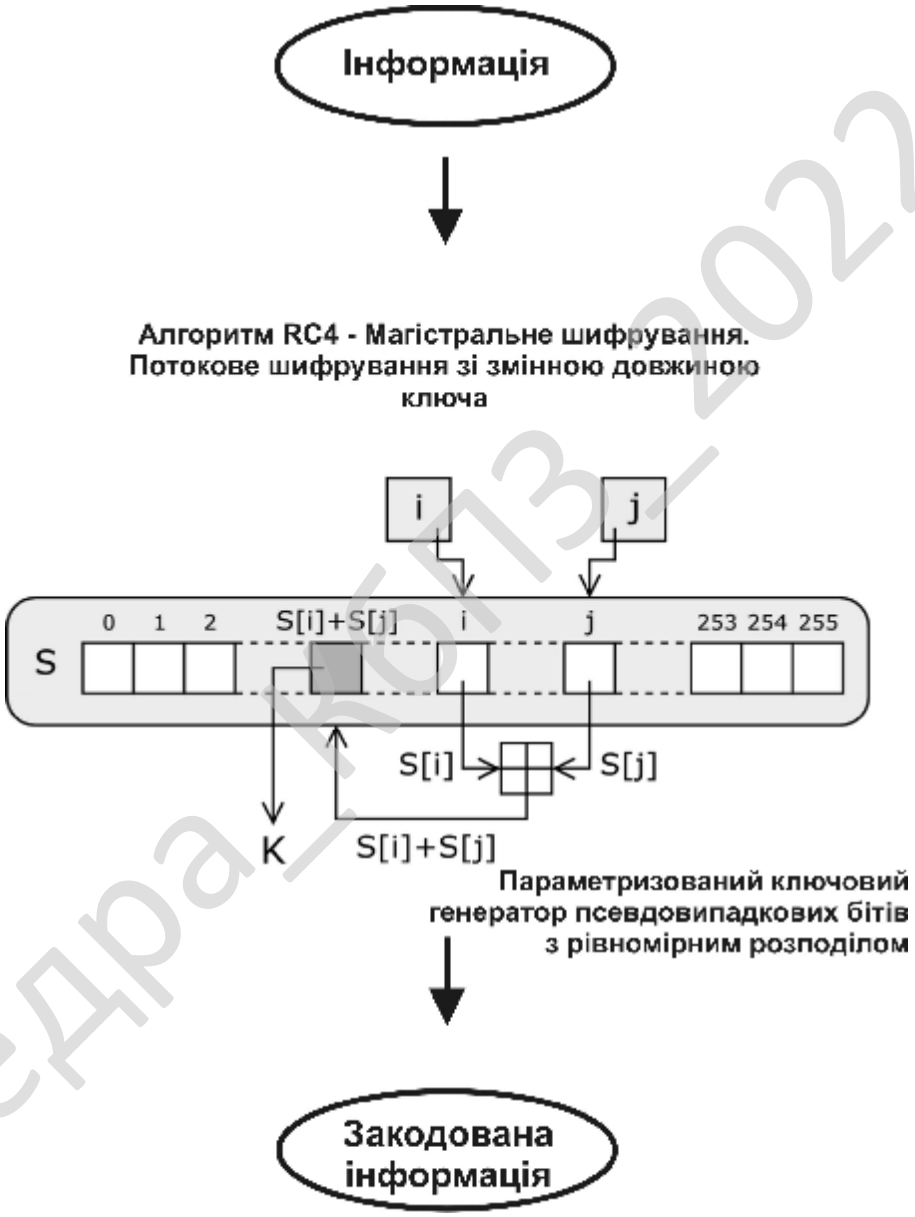

Рисунок 3.5 – Функціональна схема роботи системи

Параметр n є розміром слова для алгоритму. За замовчуванням програма встановлює значення, $n = 8$. для підвищення рівня безпеки необхідно збільшити це значення.

Внутрішній стан RC4 складається з масиву S розміром 2^n слів і двох лічильників, кожний розміром в одне слово. Масив містить перестановку 2^n можливих значень слова. На плакаті два лічильники позначені через i і j.

Ключем як і іншими налаштовуваннями алгоритму управляє користувач через меню налаштовування програми.

3.4 Розробка діаграми процесів

Важливим критерієм при розробці програмного забезпечення є грамотна розробка структури роботи системи, потоків і процесів.

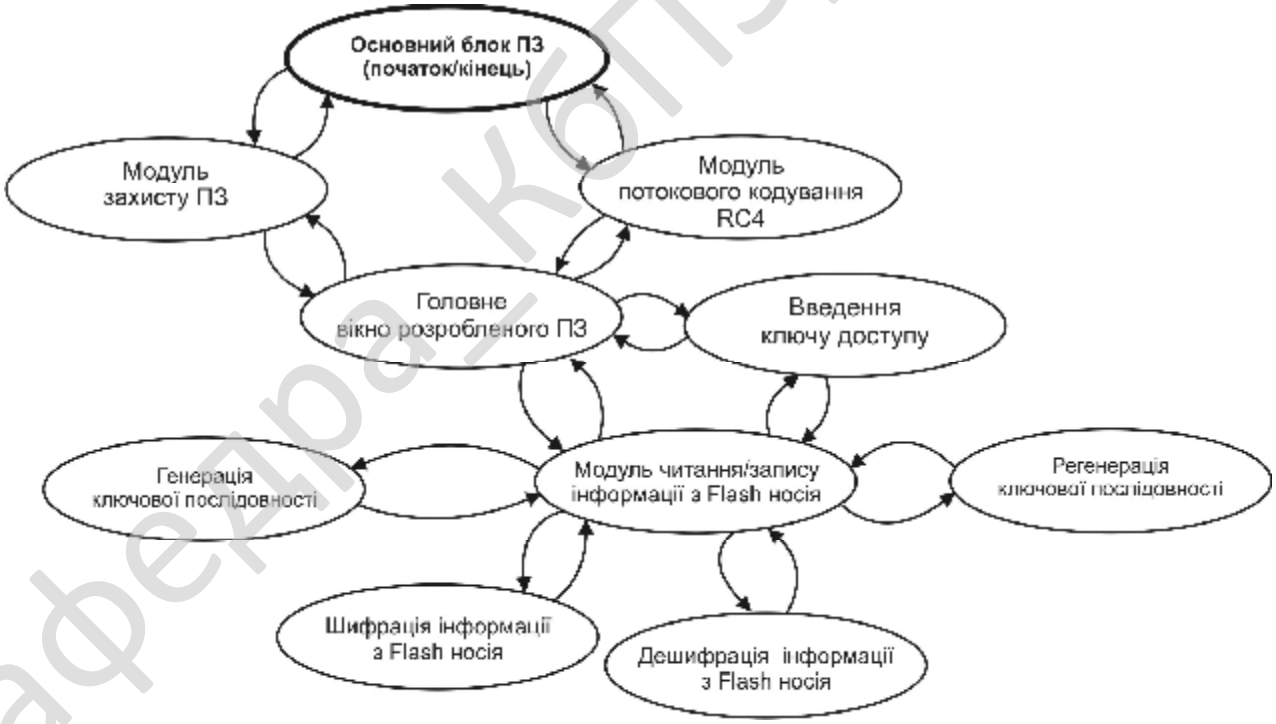

Рисунок 3.6 – Діаграма процесів

При розробці діаграми процесів показаної на рисунку 3.6, ураховувалися особливості, що накладаються на програму застосуванням алгоритму потокового шифрування RC4.

Розглядаючи розроблену діаграму процесів можна точно зрозуміти, як працює ПЗ і алгоритм потокового шифрування в цілому.

Починається й закінчується програма в першому блоці діаграми, який є основною крапкою звіту, при переміщенні по стрілках відбувається взаємодія блоків діаграми і їхнє входження друг у друга.

При розгляді діаграми процесів можна чітко простежити, що головне вікно програми перед проходить через блоки зашиті програмного забезпечення й модуль потокового кодування RC4. Також шифрування або дешифрування інформації з Flash накопичувача відбувається через блок читання/запису інформації на Flash накопичувач із застосуванням блоків введення ключа доступу й генерації/регенерації ключової послідовності.

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно розробка блок-схем. Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності й детальної блок-схеми залежить результат всієї програми. При виборі початкової крапки відліку при побудові схем було враховано, що виходячи з вибору мови програмування й інших технічних засобів, програма буде об’єкно-орієнтовна, проект, що розробляється, вимагає оптимізації програми високого рівня, також ті, що при розробці програми слід надати особливу увагу захисту програмного забезпечення та алгоритму потокового кодування даних RC4.

На рисунку 4.1 зображений основний алгоритм програми. При детальному розгляді програма розбита на декілька важливих блоків.

Початкові блоки ініціалізації та перевірки робото спроможності:

- ініціалізація початкових значень ПЗ;
- підключення модуля захисту ПЗ;
- підключення модулю потокового шифрування RC4;
- перевірка цілісності ПЗ.

Блоки безпосередньої роботи програми:

– цикл чекання дії з сторони користувача;
– запитий FLASH носія;
– підпрограма обробки запиту “Flash_CODE”.

Кінцеві блоки завершення роботи програми:

- Звільнення виділених ресурсів програми.

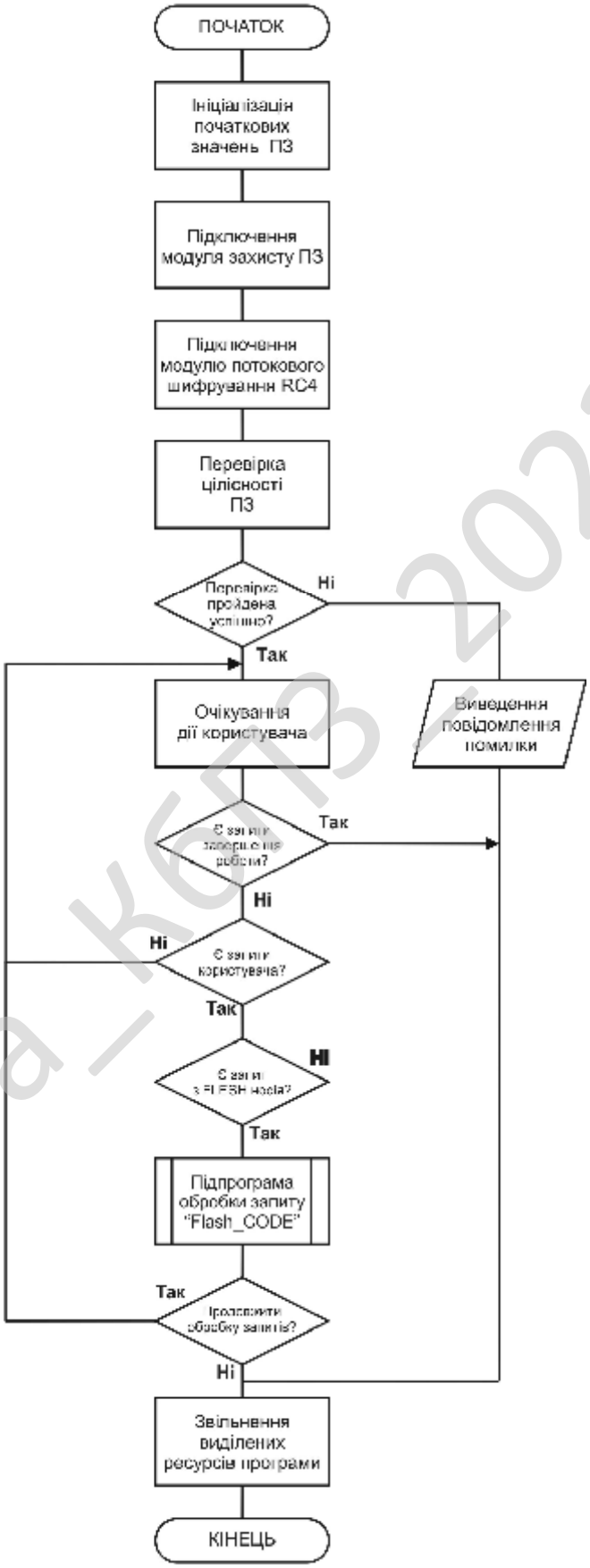

Рисунок 4.1 – Блок схема основної програми

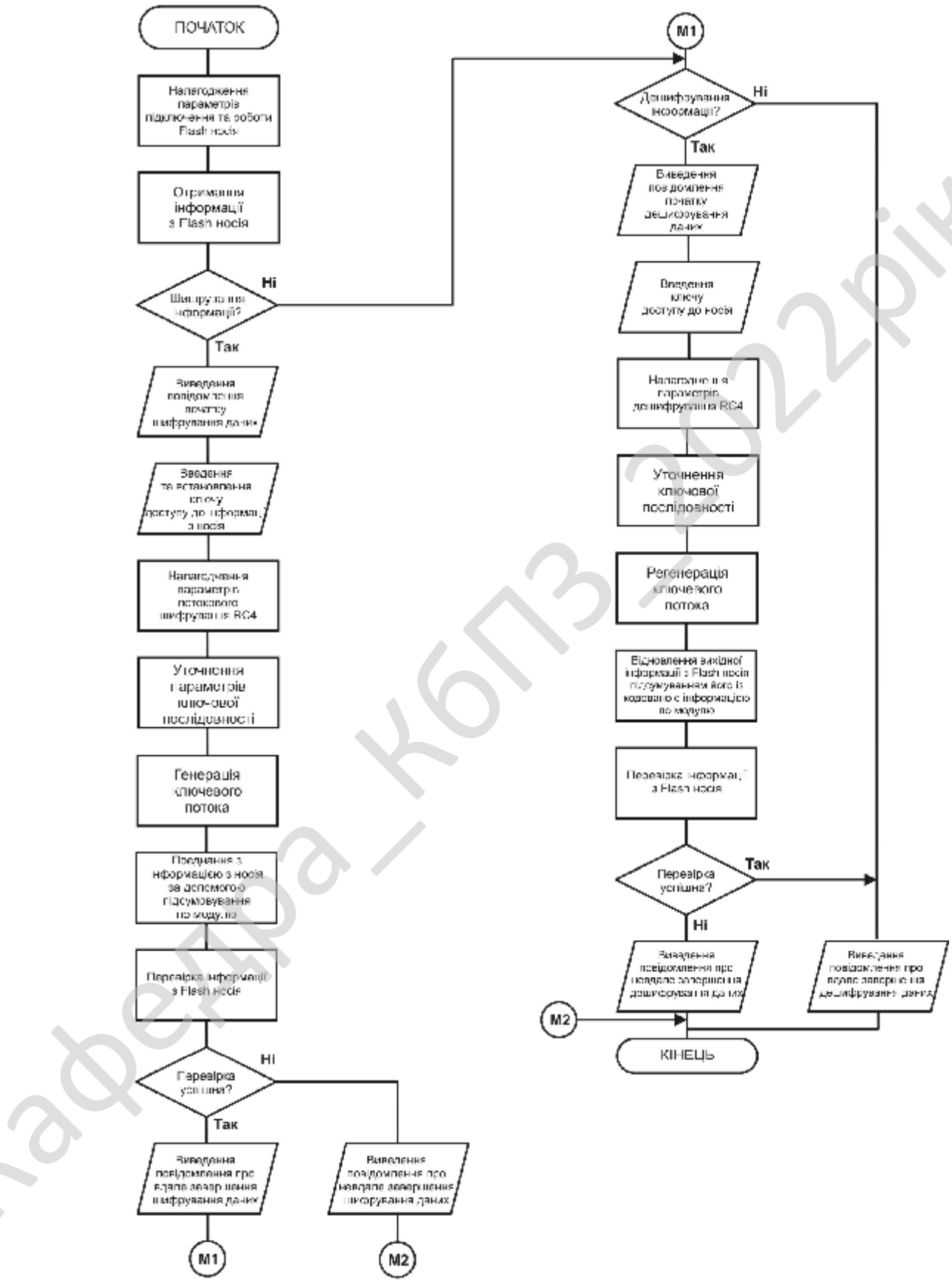

Рисунок 4.2 – Блок схема підпрограм

Ядро алгоритму потокового шифрування RC4 розглянуто в підпрограмі “Flash_CODE”, вона складається з наступних блоків.

Блоки початкової перевірки:

- налагодження параметрів підключення та роботи Flash носія;
- отримання інформації Flash носія.

Блоки шифрування інформації:

- виведення повідомлення качану шифрування даних;
- введення та встановлення ключу доступу до інформації з носія;
- налагодження параметрів потокового шифрування RC4;

уточнення параметрів ключової послідовності;

- генерація ключового потоку;
- поєднання з інформацією з носія за допомогою підсумовування по модулю;
- перевірка інформації з Flash носія;
- виведення повідомлення про вдале чи не вдале завершення дешифрування даних.

Блоки дешифрування інформації:

- виведення повідомлення качану дешифрування даних;
- введення ключу доступу до носія;
- налагодження параметрів дешифрування RC4;
- уточнення ключової послідовності;
- регенерація ключового потоку;
- відновлення вихідної інформації з Flash носія підсумуванням його йз кодованою інформацією по модулю;
- перевірка інформації з Flash носія;
- виведення повідомлення про вдале чи не вдале завершення дешифрування даних.

Реалізація алгоритму потокового шифрування інформації RC4

Розглянемо розроблену реалізацію алгоритму RC4 у дипломній програмі. RC4 будується як і будь-який потоковий шифр на основі параметризованного ключем генератора псевдовипадкових битов з рівномірним розподілом.

Основні переваги шифру – висока швидкість роботи й змінний розмір ключа. Типова реалізація виконує 19 машинних команд на кожний байт тексту. RC4 може шифрувати зі швидкістю близько 10 Мбайт/з на процесорі з тактовою частотою 330 Мгц, що є оптимальним випадком для використання алгоритму в дипломній програмі.

Генератор має подстановочную таблицю (S-Бокс 8x8): S_0, S_1,..., S_{255}. Входами генератора є замінені по підстановці числа від 0 до 255, і ця підстановка є функцією від ключа змінюваної довжини. Генератор має два лічильники *i* і *j*, ініціалізуємих нульовим значенням

Для генерації випадкового байта гами в програмі виконуються наступні операції:

```
I: = (i+1) mod 256;
J:= (j+Si) mod 256;
swap (Si and Sj);
t:= (Si+Sj) mod 256;
K:=St;
```

Байт *K* складається операцією XOR з відкритим текстом для виробітку шифртекста або для одержання байта відкритого тексту.

Шифрування відбувається досить швидко – приблизно в 10 разів швидше DES-Алгоритму.

Ініціалізація S-Боксу проста. На першому кроці він заповнюється лінійно: $S_0 = 0$, $S_1 = 1$,..., $S_{255} = 255$. Потім ще один 256-байтний масив повністю заповнюється ключем, для чого ключ повторюється відповідне число раз залежно від довжини: K_0, K_1,..., K_{255}. Індекс j виставляється рівним нулю.

Потім у програмі реалізуємо наступні дії:

```
for i :=0 to 255 do
begin
     j:= (j+Si+Ki) mod 256;
```

```
            swap (Si and Sj);
        end;
```

RC4 може приймати приблизно 21700 (256x2562) можливих станів, а це дуже багато. S-Бокс повільно змінюється в процесі роботи: параметр і забезпечує зміну кожного елемента, а j відповідає за те, щоб ці елементи змінювалися випадковим образом.

RC4 являє собою сімейство алгоритмів, що задаються параметром n, що є позитивним цілим з рекомендованим типовим значенням n = 8. Внутрішній стан генератора RC4 у момент часу t складається з таблиці утримуючої 2n n-бітних слів, і із двох n-бітних слів-покажчиків it і jt. Таким чином, розмір внутрішньої пам'яті становить M = n2n + 2n біт дозволяючи оптимізувати процес шифрування даних з Flash носіїв.

Виходячи з вищевикладених особливостей і переваг алгоритму, був розроблений вихідний код потокового шифрування RC4 у дипломній програмі.

```
{
 ************************************
       Реалізація потокового шифрування, алгоритм RC4
************************************
 }
 unit PoltovecRC4;
 interface
 uses
  Windows, Sysutils;
 type
  TRC4Data= record
    Key: array[0..255] of byte;        { поточний ключ }
    OrgKey: array[0..255] of byte;    { початковий ключ }
  end;
       var
    s: array [0..255] of Byte;
 i,j: Byte;
       implementation

       //Ініціалізація S-Box'a
 procedure InitRC4Cipher(key: ShortString);
 var
     k: array [0..255] of Byte;
```

```
     t: Byte;
     l: Cardinal;
 i0,j0: Byte;
 begin
  for i0:= 0 to 255 do s[i0]:= i0;
  j0:= 1; l:= Length(key);
  for i0:= 0 to 255 do
    begin
      k[i0]:= Ord(key[j0]);
      if j0 = l then j0:= 0;
      Inc(j0);
    end;
  for i0:= 0 to 255 do
    begin
      j0:= (j0 + k[i0] + s[i0]) mod 256;
      t:= s[i0];
      s[i0]:= s[j0];
      s[j0]:= t;
    end;
  i:= 0;
  j:= 0;
 end;
 //Ініціалізація RC4
procedure RC4Init;
 var
  xKey: array[0..255] of byte;
  i, j: integer;
  t: byte;
 begin
  if (Len<= 0) or (Len> 256) then
    raise Exception.Create('RC4: неправильна довжина ключа');
  for i:= 0 to 255 do
  begin
    Data.Key[i]:= i;
    xKey[i]:= PByte(integer(Key)+(i mod Len))^;
  end;
  j:= 0;
  for i:= 0 to 255 do
  begin
    j:= (j+Data.Key[i]+xKey[i]) and $FF;
    t:= Data.Key[i];
    Data.Key[i]:= Data.Key[j];
```

```
    Data.Key[j]:= t;
   end;
   Move(Data.Key,Data.OrgKey,256);
 end;
 procedure RC4Burn;
 begin
   FillChar(Data,Sizeof(Data),$FF);
 end;
// Тестування
function RC4SelfTest;
 const
   InBlock: array[0..4] of byte= ($dc,$ee,$4c,$f9,$2c);
   OutBlock: array[0..4] of byte= ($f1,$38,$29,$c9,$de);
   Key: array[0..4] of byte= ($61,$8a,$63,$d2,$fb);
 var
   Block: array[0..4] of byte;
   Data: TRC4Data;
 begin
   RC4Init(Data,@Key,5);
   RC4Crypt(Data,@InBlock,@Block,5);
   Result:= CompareMem(@Block,@OutBlock,5);
   RC4Reset(Data);
   RC4Crypt(Data,@Block,@Block,5);
   Result:= Result and CompareMem(@Block,@InBlock,5);
   RC4Burn(Data);
 end;
//Зашифрувати конкретний символ
 function GetRC4ByteCiphered(bt: Byte): Byte;
 var
   t: Byte;
 begin
   i:= (i + 1)   mod 256;
   j:= (j + s[i]) mod 256;
   t:= s[i];
   s[i]:= s[j];
   s[j]:= t;
   t:= (s[i] + s[j]) mod 256;
   Result:= bt XOR s[t];
 end;
 //Застосувати RC4 шифр до потоку даних
  function ApplyRC4ToData(Data: TStream; var Buffer: TStream; key: ShortString):
Boolean; stdcall;
```

```
 var
   i: Cardinal;
   d: Byte;
 pos: Cardinal;
 begin
  if (key = '')OR(Buffer = Data)OR(Buffer = nil)OR(Data = nil)OR(Data.Size = 0) OR
(Buffer.Size <> 0) then
    begin
      Result:= false;
      Exit;
    end;
  pos:= Data.Position;
  Data.Position:= 0;
  Buffer.CopyFrom(Data,Data.Size);
  Buffer.Position:= 0;
  Data.Position:= 0;
  try
    InitRC4Cipher(key);
    for i:= 0 to Buffer.Size-1 do
      begin
        Data.ReadBuffer(d,1);
        d:= GetRC4ByteCiphered(d);
        Buffer.WriteBuffer(d,1);
      end;
  except
    Result:= false;
    Exit;
  end;
  Data.Position:= pos;
         Buffer.Position:= 0;
  Result:= true;
 end;
```

У даному вихідному коді реалізоване саме ядро алгоритму потокового шифрування RC4, самі виклики й обробка інформації відбувається в іншому модулі (дивитеся доповнення Б) з викликом розглянутої функції.

Для реалізації алгоритму необхідно викликати функцію:

`function ApplyRC4ToData(Data: TStream; var Buffer: TStream; key: ShortString): Boolean.`

Параметри виклику функції:

– Data передається створений раніше, не порожній потік з вихідними даними;

– Buffer передається створений раніше, порожній потік;

– Key передається рядок, що є ключем, довжина ключа від 1 до 256.

Функція повертає true, якщо шифрування вдалося, і відповідно false, якщо воно провалилося.

Якщо шифрування пройшло успішно, то Buffer зберігає зашифровані дані, вихідні дані в Data не змінюються.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм REDOC III, який оперує з 80-бітовим блоком. Довжина ключа може змінюватися й досягати 2560 байт (204800 біт). Алгоритм складається тільки з операцій XOR над байтами ключа й відкритого тексту, перестановки й підстановки не використовуються.

1. Створюють таблицю ключів з 256 10-байтових ключів, використовуючи секретний ключ.

2. Створюють два 10-байтових блоки масок M1 і M2. M1 являє собою результат операції XOR перших 128 10-байтових ключів, а M2 – результат операції XOR других 128 10-байтовых ключів.

3. Для шифрування 10-байтового блоку:

a. Виконують операцію XOR з першим байтом блоку даних і першим байтом M1. Вибирають ключ у таблиці ключів, розрахованої в раунді 1. Використовують обчислене значення XOR як індекс таблиці. Виконують операцію XOR з кожним, крім першого, байтом блоку даних і відповідним байтом обраного ключа.

b. Виконують операцію XOR із другим байтом блоку даних і другим байтом M1. Вибирають ключ у таблиці ключів, розрахованої в раунді 1.

Використовують обчислене значення XOR як індекс таблиці. Виконаєте операцію XOR з кожним, крім другого, байтом блоку даних і відповідним байтом обраного ключа.

c. Продовжують ці дії з усім блоком даних (з 3-10 байтами), поки не буде використаний кожний байт для вибору ключа з таблиці після виконання операції XOR з ним і відповідним значенням M1. Потім виконують операцію XOR з кожним, крім використаного для вибору ключа, байтом, і ключем.

d. Повторюють етапи a-c для M2.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Після написання програмного продукту його потрібно впровадити в експлуатацію. При цьому виправляються помилки, які були помічені, система настроюється на відповідний режим роботи. Для налаштування системи на оптимальні умови роботи необхідно виконати наступні кроки:

– перевірити правильність роботи й дозвіл використання Flash накопичувачів в операційній системі;

– установити розроблене програмне забезпечення на комп'ютер

– підключити Flash накопичувач.

На рисунку 5.1 представлене головне вікно розробленого програмного забезпечення, зашиті інформації на Flash накопичувачах із застосуванням потокових шифрів. Як видно з рисунка розроблена програма складається із трьох складових частин – верхнє меню дій, інтерфейс введення даних, нижній рядок відображення поточної дії.

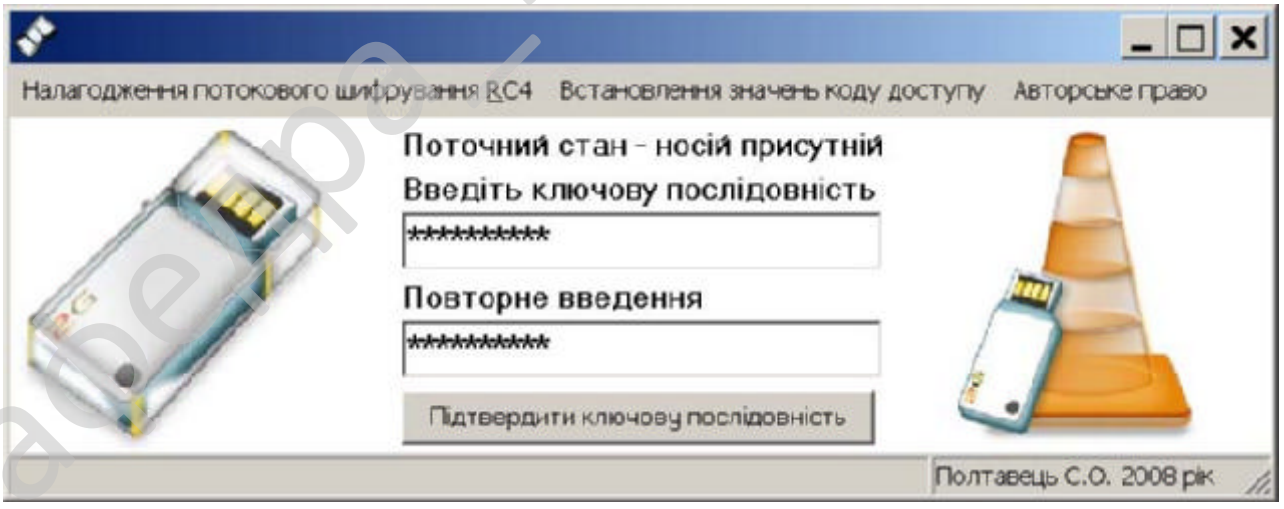

Рисунок 5.1 – Головне вікно програми

Розроблена програма використовує алгоритм потокового шифрування RC4 (розділ 3), завдяки чому дії, проведені для зашиті інформації на Flash носії, проводяться зі швидкістю 10 Мбайт/с на процесорі з тактовою частотою 330 Мгц. Розроблена програма під час роботи перебуває в одному з 4 режимів роботи .

Розглянемо їх більш детальніше.

1. Режим очікування (за замовчуванням) – програма очікує дій користувача або підтвердження операційної системи про підключення Flash носія в персональний комп'ютер. При підключенні Flash носія відбувається активізація програми й вивід головного вікна програми (Рисунок 5.1) на передній тло.

Під час запуску програма звертається в трей операційної системи й переходить у режим очікування.

2. Режим шифрування даних (Рисунок 5.2) – Після підтвердження операційною системою про підключення Flash носія до персонального комп'ютера виробляється перевірка накопичувача на наявність шифрованих даних формату розробленої програми, якщо таких даних немає користувач вводить новий ключ доступу й програма проводить потокове шифрування інформації на носії з видаленням незашифрованої копії.

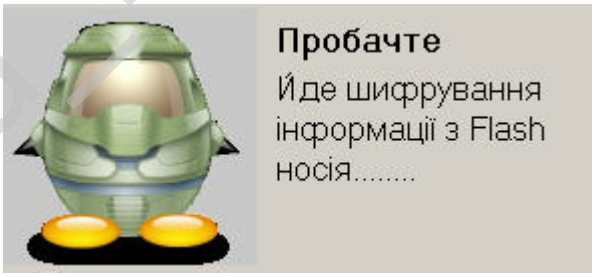

Рисунок 5.2 – Вікно проведення шифрування інформації

3. Режим дешифрування даних (Рисунок 5.3) – Після підтвердження операційною системою про підключення Flash носія до персонального комп'ютера виробляється перевірка накопичувача на наявність шифрованих даних формату розробленої програми якщо такі дані присутні користувач

вводить ключ доступу до інформації й програма проводить потокову дешифрування інформації на носії із заміщенням зашифрованої копії.

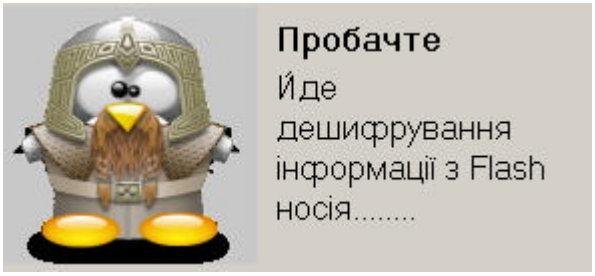

Рисунок 5.3 – Вікно проведення дешифрування інформації

4. Режим діагностики даних (Рисунок 5.4) – Після проведення дій шифрування/дешифрування програма автоматично переходить у режим перевірки інформації й створення контрольної суми та підтверджувальної мітки про відповідність даних.

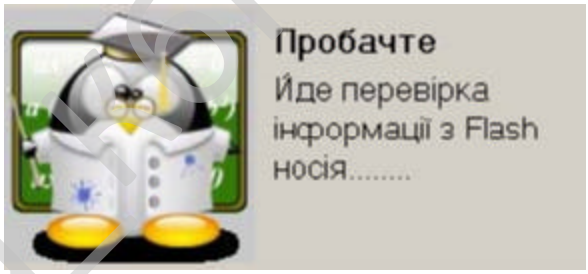

Рисунок 5.4 – Вікно проведення перевірки інформації

Крім вищерозглянутих можливостей програма містить форму авторського права підтверджувальні авторські права програми з темою розробки, даних студента й році розробки.

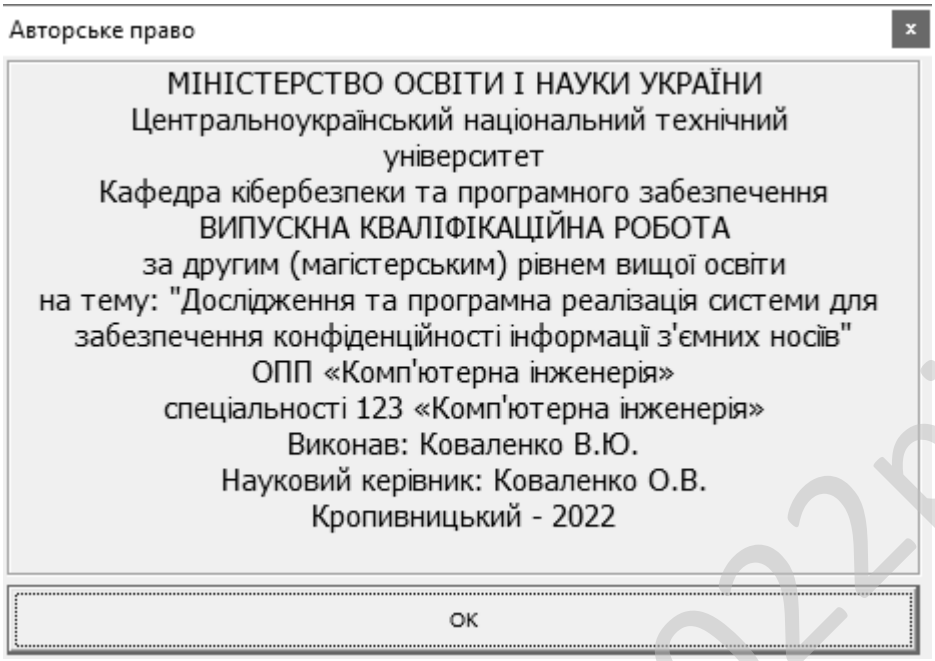

Рисунок 5.5 – Вікно авторського права

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи для забезпечення конфіденційності інформації з’ємних носіїв.

Метою розробки є дослідження та програмна реалізація системи для забезпечення конфіденційності інформації з’ємних носіїв.

Об’єктом дослідження є процес для забезпечення конфіденційності інформації з’ємних носіїв.

Предметом дослідження є методи для забезпечення конфіденційності інформації з’ємних носіїв.

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 для забезпечення конфіденційності інформації з’ємних носіїв.

– Розроблено вітчизняний продукт для забезпечення конфіденційності інформації з’ємних носіїв, який має більш широкі можливості, на відміну від існуючих аналогів.

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 24 днів (один місяць).

В магістерській роботі було проведене дослідження та виконана програмна реалізація системи для забезпечення конфіденційності інформації з'ємних носіїв

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

а) невеликий розмір;

б) невеликі системні потреби;

в) незалежність від встановлених на комп'ютері баз даних;

г) 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	180
3. Запланований термін розробки, днів	Fpq	24 (1 місяць)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	В
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

1	2	3
7. Кількість макетів вхідної інформації	–	8
8. Кількість форм вихідної інформації.	–	6
9. Мова програмування (1-6)	–	1
10. Попередній досвід (1-6)	–	3
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	–	1
13. Рівень спрацьованості колективу (1-6)	–	2
14. Ступінь вимірності процесів (1-6)	–	3
15. Необхідна надійність програмного забезпечення (1-6)	–	3
16. Розмір бази даних (порівняно з розміром програми) (1-6)	–	4
17. Складність кінцевого програмного продукту (1-6)	–	5
18. Необхідний рівень забезпечення повторного використання (1-6)	–	2
19. Документованість відповідно до планованого життєвого циклу (1-6)	–	3
20. Вимоги до швидкодії ПП (1-6)	–	3
21. Обмеження на розміри основного сховища даних (1-6)	–	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	–	4
23. Професійний рівень аналітиків (1-6)	–	3
24. Професійний рівень програмістів (1-6)	–	4
25. Постійність складу команди розробників (1-6)	–	2
26. Досвід розробки додатків (1-6)	–	1
27. Досвід роботи з обчислювальною платформою (1-6)	–	2

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)	–	3
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	18000
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	40
38. Ставка податку на додану вартість, %	Ндв	20

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.

$$B = 1{,}01+0{,}001(2{,}43+3{,}64+4{,}22+3{,}95+2{,}73) = 1{,}027.$$

$$T_{ном} = 2{,}45 \cdot 2{,}2^{1{,}027} = 5{,}5 \text{ люд-міс.}$$

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.

$$T_{уточн} = 5{,}5 \cdot (1 \cdot 1{,}09 \cdot 1{,}30 \cdot 0{,}91 \cdot 1 \cdot 1 \cdot 1 \cdot 1{,}15 \cdot 1 \cdot 0{,}87 \cdot 1{,}10 \cdot$$

$$\cdot 1{,}22 \cdot 1{,}12 \cdot 1{,}10 \cdot 1 \cdot 1 \cdot 1{,}10) = 12{,}9 \text{ люд-міс.}$$

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);

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

$$T_{РП} = 0{,}3 \cdot 3{,}23 \cdot 12{,}9^{0{,}33+0{,}2(1{,}027-1{,}01)} \cdot 40 = 91 \text{ люд/день.}$$

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

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	15	Д7
Робочий проект	91	Ф 7.1-7.4
Впровадження	15	Д13
Всього	140	–

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.

$$Ч = \frac{140 \cdot 1}{24\text{-}3} = 6,7 \text{ ставки.}$$

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.

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	90	10	900	15
Монітор	60	10	600	10
Клавіатура	30	10	300	5
Маніпулятор «мишка»	30	10	300	5
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	1	120	2
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор-маршрутизатор	30	2	60	1
Кабельні господарства ЛОМ на 1 м.п.	2,5	250	625	10,42
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	52,08

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф_{др}^{с} = \frac{З_ч \cdot n_{тіс}}{1{,}2}, \tag{7.6}$$

$$Ф_{др}^{с} = \frac{52 \cdot 1}{1{,}2} = 43 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф_{др}^{с}}{F_{др} \cdot T_{зм}}, \tag{7.7}$$

$$Ч_{ел} = 43/(24 \cdot 8) = 0{,}2 \text{ ставки.}$$

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	

Продовження таблиці 7.4

Посада	Вид роботи	Час	К-ть штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	2	0,5
	Підтримка постійних клієнтів	1	
	Оформлення договорів, ведення тендерів	0,5	
	Контроль взаєморозрахунків з постачальниками	0,5	
Всього		4	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	2	0,5
	Створення графічних і стилістичних елементів сайту	1	
	Оформлення банерів і промо-сторінок	0,5	
	Розміщення графіки і контенту на Інтернет сторінках	0,5	
Всього		4	
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	

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	0,5	20000	10000
Продакт-менеджер	0,5	13106	6553
Інженер-програміст	6,7	16700	111890
Інженер-електронщик	0,2	12000	2400
Інженер-системотехнік	0,25	12000	3000
Адміністратор мережі	0,5	13106	6553
Системний програміст	0,25	12000	3000
Дизайнер WEB	0,5	13000	6500
Інженер-верстальник	0,25	11700	2925
Бухгалтер-економіст	0,5	12500	6250
Всього за період розробки	$R_{сп}$ = 10,15	-	$Ф_{роб}$ = 159071

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.

$$З_{сд} = \frac{159071}{10{,}15 \cdot 24} = 653 \; грн.$$

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:

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

де: $R^1_{сп}$ – кількість робочих місць виконавців, шт. Приймаємо 13 робочих місць;

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

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

Згідно даних інтернет ресурсу DOM.RIA (https://dom.ria.com) ціна одного квадратного метра площі, вік якої не перевищує 30 років, по місту складає 500…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 38 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:

$$Б_{уд} = 13 \cdot 8 \cdot 20000 = 2080000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 208000 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.

$$І_{нв} = 13 \cdot 3500 = 45500 \text{ грн.}$$

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 за прайсом Інтернет магазину Компбест за 01.11.22 – джерело https://compbest.com.ua.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		11457
Системний блок		7509
Процесор	Intel Core i5 2500k 3.30GHz, socket1155, box	–
Системна плата	Gigabyte GA-H61M-S2PV	–
Відеокарта	AMD Radeon HD 8570, 1 GB GDDR3, 128 bit	–
Жорсткий диск	SSD: 240 Gb Kingston	–
Оперативна пам’ять	Kingston DDR3 8GB (KVR1333D3N9 Intel/AMD 2x4 GB	–
DVD-привод	DVD -RW/+RW , LG SATA SuperMulti Bul 22x, SecurDisc, black	–
Корпус	ATX Middle Tower GIGABYTE GZ-X Silver 500W (GZ-X4 Silver)	–
Кулер	–	–
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-E int. 3.5", 1*USB2.0+AUDIO+1394, multi: A Type Cards, black	–
інше	Клавіатура, мишка	Подару-нок
Монітор	22" TFT, ASUS VW223D (5ms, 300/3000: 1 170/160, D-SUB, Wide)	2600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Сканер	Epson Perfection V37 Photo	2970
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965
Пристрій безперебійног живлення	UPS APC BACK-UPS ES 525VA 230V RUSSIA (BE525-RS)	1348

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 7.7 – 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	13	11457	14894,1	163835,1
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	1	2970	297	3267
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	185653,6

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	2080000	-	-
2. Передавальні пристрої	208000	-	-
Всього по групі	2288000	5	114400
Група 4			
3. Обчислювальна техніка	185654	-	-
Всього по групі	185654	50	92827
Група 5, 6			
4. Вимірювальні пристрої	3999	25	999,75
5. Транспортні засоби	97500	20	19500
6. Господарський інвентар	45500	25	11375
Всього по групі	146999	-	31874,75
7. Нематеріальні активи	18000	10	1800
Разом	$К_р$ = 2666153		$А_р$ = 250476,75

Примітка: вартість автомобіля взята по даним з автосалону автотрейдінг, вкладки автобазар, джерело http: //www.auto-trading.com.ua/sale/lot20772.html, 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.

$$З_о = 653 \cdot 140/180 = 508 \text{ грн.}$$

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, %.

$$З_д = 508 \cdot 10 \cdot 0{,}01 = 51 \text{ грн.}$$

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

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

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

$$С_{оц} = 0{,}01 \cdot 22(508+51) = 123 \text{ грн.}$$

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, %.

$$Г_{осп} = 508 \cdot 15 \cdot 0{,}01 = 76 \text{ грн.}$$

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

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

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

Згідно прийнятих норм на підприємстві $n_{вит}$ приймаємо 0,2 пачек паперу на період розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=200 грн., визначаємо вартість паперу за період розробки:

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

$$З_{М1} = 200 \cdot 0{,}2 = 40 \text{ грн.}$$

Згідно прийнятих норм по комплектації до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків. Їх кількість дорівнює кількості коробочних версій запропонованого продукту (приймаємо 70):

$$З_{М2} = \sum Ц_д\ , \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR box – 23 грн./шт., DVD-R box – 45 грн./шт.

$$З_{М2} = 69 \cdot 23 + 45 = 1632 \text{ грн.}$$

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.

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

$$З_М = (40 + 1632 + 1702)/180 = 18{,}7 \text{ грн.}$$

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, %.

$$О_п = 508 \cdot 15 \cdot 0{,}01 = 76 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 180 прим.):

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

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

$$A_м = 250477 \cdot 1/(180 \cdot 12) = 116 \text{ грн.}$$

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

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

$$C_п = 508+51+123+76+18,7+76+116 = 968,7 \text{ грн.}$$

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 40%.

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

де: $P_п$ – рівень рентабельності, %.

$$П_р = 0,01 \cdot 40 \cdot 968,7 = 378 \text{ грн.}$$

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	$З_о$	508
2. Додаткова зарплата виконавців	$З_д$	51
3. Відрахування на соціальні потреби	$C_{оц}$	123
4. Загальногосподарські витрати	$Г_{осп}$	76
5. Витрати на матеріали	$З_М$	18,7
6. Освоєння нових операційних систем, мов програмування	$О_п$	76

Продовження таблиці 7.9

1	2	3
7. Амортизація основних фондів	$А_м$	116
8. Повна собівартість програмного забезпечення	$С_п$	968,7
9. Плановий прибуток	$П_р$	378
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	1346,7
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	269,3
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	1616

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	–	1616
Всього капітальних витрат	–	1616

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 обслуговування системи	$З_р$	46970	26840
2. Витрати на електроенергію	$З_{ел}$	376	314
3. Витрати на амортизацію	$З_{ам}$	0	404
Всього витрат за рік	$І$	47346	27558

Витрати на обслуговування системи:

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

де: $Т_р$ – кількість годин обслуговування за рік, *год.*;

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

Після купівлі нового програмного забезпечення кількість годин на обслуговування системи зменшилася з 350 годин на рік до 200 годин на рік, тому витрати на технічне обслуговування зменшилися з:

$$З_{р\,баз} = 350 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 46970 \text{ грн},$$

до:

$$З_{р\,нов} = 200 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 26840 \text{ грн}.$$

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

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

$$З_{ел\ баз} = 0{,}15 \cdot 1140 \cdot 2{,}2 = 376 \text{ грн.}$$

$$З_{ел\ нов} = 0{,}15 \cdot 950 \cdot 2{,}2 = 314 \text{ грн.}$$

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	–	1616	–	404
Всього відрахувань	-	–	1616	–	404

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:

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

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

$$Ев = (1346{,}7 - 968{,}7) \cdot 180 - (0{,}05 \cdot 2288000 + 0{,}5 \cdot 185654 +$$
$$+0{,}25 \cdot 49499 + 0{,}2 \cdot 97500 + 0{,}1 \cdot 18000) \cdot 1/12 = 47968 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

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

де: $K_р$ – балансова вартість основних фондів розробника.

$$T_в = \frac{2666153}{(1346,7 - 968,7) \cdot 180 \cdot 12 / 1} = 3,3 \ року .$$

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	180
2. Повна собівартість розробленої програми	Грн.	968,7
3. Ціна розробленої програми	Грн.	1346,7
4. Плановий прибуток від реалізації розробленої програми	Грн.	378
5. Рентабельність програмної продукції	%	40
6. Об’єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	2666153
7. Загальний прибуток від реалізації програмної продукції	Грн.	68040
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	47968
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Рік	3,3
10.Об’єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	1616
11.Величина економічного ефекту у користувача програмної продукції	Грн.	19384
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Рік	0,1

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

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

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.

$$E_{сп} = (47346\text{-}27558)\text{-}0{,}25 \cdot 1616 = 19384 \text{ грн.}$$

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

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

$$T_{сп} = \frac{1616}{47346 - 27558} = 0{,}1 \text{ року}.$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

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.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Незважаючи на стрімке розповсюдження комп'ютерних мереж, з’ємні носії інформації не втрачають актуальності у ряді сфер професійної діяльності.

Стрімкий розвиток галузі інформаційних технологій (ІТ) не тільки у сфері управління виробництвом, в банківській систему, бізнесі, системі освіти, але також на транспорті, сфері обслуговування призвів до того, що десятки мільйонів людей у всьому світі виявились втягнутими у взаємодію людини з комп'ютером. Природно виникає запитання: настільки безпечною є ця взаємодія для людини? Адже відома аксіома про те, що будь-яка взаємодія людини та засобів праці двостороння.

Впровадження комп'ютерних технологій принципово змінило характер праці різних категорій фахівців. Працівники, використовують комп'ютерну техніку, на своєму досвіді оцінили її величезні можливості. Одночасно виникла певна безтурботність при її експлуатації.

Недотримання вимог безпеки призводить до того, що й через кілька днів роботи за комп'ютером співробітник починає відчувати певний дискомфорт: в нього виникає головний біль і різь у власних очах, з'являються почуття виснаження й дратівливість. В окремих людей порушується сон, погіршується зір, занедужують руки, шия, поперек тощо.

До недоліків умов праці користувачів комп’ютерної техніки можна віднести:

– недостатню площу і обсяг виробничого приміщення;

– недотримання вимог, мікроклімату на робочих місцях;

– низький рівень освітленості у приміщеннях і на робочих поверхнях апаратури;

– підвищений рівень низькочастотних магнітних полів від моніторів;

– порушення вимог організації робочих місць;

– недотримання вимог до режимам праці та відпочинку;

– надмірне виробничу навантаження працівників;

– відсутність навичок зниження впливу психоемоційного напруги.

Відповідно до ст.14 Закону «Про охорони праці» [1] на роботодавця покладено обов'язок забезпечити: безпеку працівників при експлуатації устаткування; застосування коштів індивідуальної захисту працівників; відповідні вимоги охорони праці, умови праці в кожному робоче місце; дотримання режиму праці та відпочинку працівників; навчання безпечним методам і прийомам виконання; інструктаж з охорони праці; організацію контролю над станом умов праці в робочих місць; проведення атестації робочих місць в умовах праці.

Максимально зменшити кількість шкідливих впливів на людину при високій продуктивності праці, створити комфортні умови для роботи людей – ось одна з головних задач охорони праці.

8.2 Аналіз умов праці

Приміщення розташовано на третьому поверсі п'ятиповерхового будинку. У приміщенні розташовано 3 робочих місць з комп'ютерами (далі ПК). Відповідно до норм «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98 [2] площа, що відводиться для робочого місця з комп'ютером повинна бути не менше 6 м^2, об'єм не менше 20 м^3. Розміри даного приміщень складають: довжина – 6 м, ширина – 4,5 м, висота – 3,5 м, тобто загальна фактична площа складає 27 м^2. Необхідна площа на 3 робочих місця із установленими ПК складає 18 м^2, що не перевищує фактичну. Обсяг кабінету на одного працюючого складає 31,5м^3, отже відповідає нормі ДСанПіН 3.3.2-007-98 – не менше 20 м^3 [2].

При роботі з ПК людина може піддатися впливу шкідливих та небезпечних факторів. Під шкідливими виробничими факторами розуміють фактори, тривалий вплив яких викликає розвиток професійних захворювань. Небезпечні виробничі фактори – вплив яких на працюючого викликає травму, тобто пошкодження організму. Шкідливі і небезпечні чинники, з якими стикається бібліограф при роботі з ПК, приведені в таблиці 8.1.

Таблиця 8.1 – Перелік шкідливих та небезпечних виробничих факторів

Найменування факторів	Можливі джерела їх виникнення	Характер дії
Небезпека ураження електричним струмом	Мережа живлення	Небезпечний
Пожежонебезпечність приміщень	Наявність матеріалів, що згорають і джерел запалення (електроапаратура)	Небезпечний та шкідливий
Іонізація повітря	Статична електрика випромінювання	Шкідливий
Підвищений рівень шуму	Шум створюється перетворювачем напруги ЕОМ, її технічною периферією, а також людьми, що працюють в приміщенні	Шкідливий
Несприятлива освітленість	Недостатнє штучне і природне освітлення	Шкідливий
Незадовільні параметри мікроклімату	Незадовільний стан системи опалення і вентиляції	Шкідливий
Психофізіологічні напруження	Монотонність праці, перенапруженість зорових аналізаторів, розумова напруженість, незручність і статичність пози	Шкідливий

По категорії вибухо- і пожежонебезпеки, згідно дане приміщення відноситься до категорії В – пожежонебезпечне, тому що присутні тверді матеріали, що горять, такі як дерев'яні столи, папір і інше. Виходячи з категорії пожежонебезпеки і поверховості будинку, ступінь вогнестійкості будівлі II. Згідно з ДБН В 1.1-7-2016 «Пожежна безпека об’єктів будівництва» [13] ЕОМ повинні розташовуватись в будівлі не менше ніж II ступню вогнестійкості.

По ступені небезпеки поразки людей електричним струмом відділ, згідно, класифікується як приміщення з підвищеною небезпекою, тому що не виключена можливість одночасного дотику людини до маючих з'єднання з землею конструкціям будинку, з одного боку, і до металевих корпусів електроустаткування, що можуть виявити під напругою – з іншого.

Для забезпечення вищевказаних оптимальних метеорологічних умов у помешканні передбачена система опалення (загальне парове) в холодному періоді, та вентиляція і кондиціювання в теплий період року, згідно ДБН2.5–67–2013 «Опалення, вентиляція та кондиціювання» [4]. При виконанні замірів параметрів мікроклімату, значення їх відповідали оптимальним та допустимим параметрам відповідно до ДСанПіН3.3.2.007 – 98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно – обчислювальних машин» [2].

Припустимий рівень іонізації повітря помешкання відповідно до СН 21.52-80 повинен складати 1500 – 3000 один./м^3.

Нормування освітлення здійснюється відповідно до ДБН В.2.5 – 28 – 2006 «Природне та штучне освітлення». [5]

Відділ забезпечений комбінованим освітленням. В темний час доби передбачається загальне і/або місцеве рівномірне штучне, а в світлий – бокове одностороннє природне освітлення два віконних прорізи.

8.3 Техніка безпеки та протипожежна профілактика

Відповідно ДБН В 1.1-7-2016 «Пожежна безпека об’єктів будівництва» [13] будинок можна віднести до II групи по ступені вогнестійкості й до категорії Д по ступені пожежонебезпеки.

Від розподільного щита по праву й ліву сторони встановлені кондиціонери, зовнішня електропроводка, поміщена в ізольований кабель. Висота проводки становить 2,2м від рівня підлоги, її кріплення здійснюється за допомогою металевих власників. Біля кожного стола організований розподільний щит, розташований на текстолітовій пластинці, закріпленої на стіні на рівні 1м від підлоги. Усього до складу входять п'ять розеток і дві клеми заземлення. Всі обчислювальні машини з'єднані із клемами заземлення. Чотири з п'яти розеток забезпечують подачу напруги 220 V, а одна, забезпечує подачу напруги в 36 V. Про це є відповідні написи на кожному розподільному щиті.

Вимоги до пожежної безпеки на підприємстві неухильно повинен дотримуватися кожен співробітник, а організаційна складова при цьому покладається на посадових осіб за відповідним рішенням керівництва і прописується в посадових інструкціях і положеннях по структурним підрозділам.

Зокрема, вказуються конкретні території, ділянки, зони, об’єкти, цілі будівлі і їх частини, поверхи, на яких відповідального співробітника повинне проводити такі організаційні роботи.

Відповідальні особи зобов’язуються розробити, впровадити та підтримувати в певному інструкцією і положенням на ввірених їм об’єктах протипожежний режим і інструкції відповідно до вимог, викладених в нормативних актах.

Передбачено також створення підрозділу добровільної пожежної охорони та пожежно-рятувальної команди в його складі.

Встановлений режим включає порядки з описом місць спеціального призначення та правила їх користування та утримання, наприклад:

– евакуаційних шляхів;

– так званих «курилок»;

– місць складування продукції та сировини;

– стоянки транспорту.

Також встановлюється порядок роботи та технічного обслуговування:

– вентиляційного устаткування;

– засобів пожежогасіння і захисту від загорянь;

– нагрівальних приладів;

– електрообладнання.

Розробляються і впроваджуються правила роботи з відкритим вогнем і горючими матеріалами. Створюються графіки проходження інструктажів з пожежної безпеки співробітників, а також порядок і терміни перевірок знань пожежно-технічного мінімуму, в тому числі, тих працівників, які відповідальні за цю ділянку роботи на підприємстві. При цьому можуть передбачатися внутрішні лекції, семінари, тренінги та практичні заняття на підприємстві, а також зовнішні – на базі спеціалізованих навчальних центрів з професійними викладачами.

Важливою складовою протипожежного режиму на будь-якому об’єкті є розробка і впровадження порядку дій при виникненні пожежі. Неодмінно має бути план евакуації, описано, як повинні відключатися електроустановки, що і в якій послідовності необхідно робити співробітникам.

Відповідно, для кожного об’єкта, кожного приміщення (крім коридорів, санвузлів, басейнів і подібних приміщень), окремих видів робіт складаються інструкції, за якими повинен працювати персонал, залучений на певних ділянках і в виконанні окремих видів робіт. За інструкціями проводиться навчання (інструктаж) персоналу з подальшим контролем знань.

Відповідно НПАОП 40.1-1.21-98 “Правил безпечної експлуатації електроустановок споживачів” [6], приміщення можна віднести до приміщень без

підвищеної небезпеки, оскільки це приміщення, сухе, з нормальною температурою й ізолюючими підлогами, що не має заземлених металоконструкцій.

Персональні ЕОМ можна віднести до першого класу електротехнічних виробів по способі захисту людини від поразки електричним струмом, оскільки їхні корпуси зроблені з ізолюючої пластмаси й кожен пристрій має заземлення. Відповідно правилам пристрою електроустановок ЕОМ можна віднести до електроустановок з робочою напругою до 1000 В.

Однієї з достовірних причин пожежі в приміщенні з обчислювальною технікою може бути коротке замикання, що спричиняє спалах електропроводки. Для його попередження вся обчислювальна техніка, а також інші електричні пристрої повинні бути обладнані плавкими запобіжниками, а на вході електромережі повинен бути передбачений автомат захисту. Не слід користуватися електричними подовжувачами й трійниками, що не мають сертифікатів відповідності вимогам безпеки.

Необхідно передбачити наявність у межах досяжності первинних засобів гасіння пожежі (вогнегасників) для локалізації вогню власними засобами до приїзду команди пожежної охорони. Повинен бути розроблений план екстреної евакуації персоналу при виникненні загоряння. Кількість евакуаційних виходів повинне бути не менш двох. Допускається використання одного евакуаційного виходу, якщо відстань найбільш віддаленого робочого місця до цього виходу не перевищує 25 м.

8.4 Розрахункова частина

Для захисного штучного заземлення застосовуються ветикальні електроди: метелевий куток *63·63·6 мм*., (згідно з ДСТУ 2251-93 «Кутики сталеві гарячекатані рівнополичні. Сортамент») довжиною *L=1,6 м.,* та горизонтальний електрод – металева полоса з перетином *60·5 мм*. Напруга –

220/380 В. Розрахункова схема розташування заземлюючих електродів – по контуру (прямокутником).

Розрахунок проведем за допустимим опором розтіканню струму заземлювача.

Початкові дані для розрахунку захисного заземлення: тип верхнього шару грунта – чорнозем, нижнього шару грунта – глина (питомий опір $\rho_2 = 40$ *Ом·м)*. Умовна товщина верхнього шару грунта: *Н=0,55 м*. Відстань між вертикальними заземлювачами (електродами) *А=3 м*. Глибина закладення горизонтального контура заземлення *t=0,6 м*. Опір заземлювача, який нормується: $R_{ЗН} = 4$ *Ом*. Необхідно визначити необхідну кількість вертикальних заземлювачів та длину полоси (горизонтального заземлювача).

Розрахунок

Відстань від центра вертикального заземлювача до поверхні землі:

$$T=t+L/2=0,6+1,6/2=1,4 \text{ м.}$$

Розрахунковий питомий опір грунта (з врахуванням того, що фактично вся конструкція заземлювача розташовується у нижньому шарі грунта):

$$\rho_{=} \psi\rho = 1,36 \cdot 40 = 54,5 \text{ Ом·м.}$$

де $\psi = 1,36$ – табличне значення коефіцієнта сезонності для відповідної климатичной зони у багатошаровому грунті [12];

$\rho_2 = 40$ *Ом·м.* – табличне значення питомого опору нижнього шару грунта (глина) [12].

Еквівалентний діаметр ветикальнього електрода (кутка) [12]:

$$Dв=0,95 \cdot K = 0,95 \cdot 63 = 59,85 \text{ мм.} = 0,0598 \text{ м.}$$

де *К= 63 мм.* – розмір метелевого кутка (задан).

Відношення *А/L=3/1,6=1,87*

Опір розтіканню електричного струму одного електрода вертикального заземлювача з урахуванням заглиблення заземлювача [12]:

$$R_O=0,366(\rho/L)[\lg(2L/Dв)+(1/2)\lg((4T+L)/(4T-L))] =$$

$$=0,366(54,5/1,6)[(\lg(2 \cdot 1,6/0,0598)+(1/2)\lg((4 \cdot 1,4+1,6)/(4 \cdot 1,4-1,6))] = =23 \text{ Ом.}$$

Визначаємо коефіцієнт екранування ветикальньних електродів *Кев=0,62* при орієтовній кількості ветикальньних електродів, яке дорівнює 5 [12].

Визначаємо необхідну кількість вертикальних електродів заземлювача (без вхарування горизонтального заземлювача), при $R_{ЗН}$ = 4 *Ом*:

$$N=R_O /(Кев\ R_{ЗН})= 23 / (0,62\cdot4)= 9,3 \approx 9\ шт.$$

Визначаємо длину з'єднуючої полоси:

$$L_П =1,05\cdot A\cdot N= 1,05\cdot3\cdot9=28,35\approx 28\ м.$$

Опір розтіканню електричного струму з'єднуючої полоси з урахуванням кліматичного коефіцієнта питомого опору грунта $K_П$ [12]:

$$R_П =0,366(\rho\cdot K_П/L_П)lg(2\cdot L_П^{\,2} /(B\cdot t))=$$

$$=0,366(40\cdot5/28)\cdot lg((2\cdot28^{\,2})/(0,06\cdot0,6))=12,2\ Ом.$$

де $K_П=5$ – табличне значення кліматичного коефіцієнта питомого опору грунта для відповідної климатичної зони для з'єднуючої полоси [12]:

$B = 60\ мм. = 0,06\ м.$- ширина з'єднуючої полоси (задана).

Загальньний опір розтіканню електричного струму заземлювача [12]:

$$R =(R_O\cdot R_П)/(R_O\cdot\eta_П+ N\cdot R_П\cdot Кев)=$$

$$=(23\cdot 12,2)/(23\cdot 0,6+ 9\cdot 12,2 \cdot 0,62)=3,48\ Ом.$$

де $\eta_П = 0,6$ – табличне значення коефіцієнта екранування з'єднуючої полоси [12].

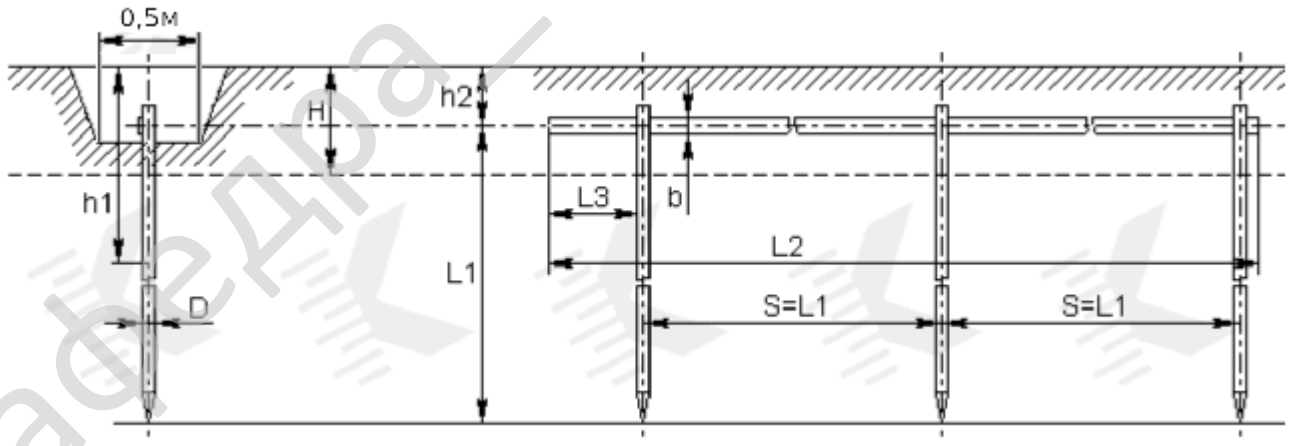

Рисунок 8.1 – Схема штучного заземлення.

Умова $R \leq R_{ЗН}$ виконується ($3,48 \leq 4$).

Так як при *9* вертикальних електродах *R* суттево менше $R_{ЗН}$, зменшимо кількість вертикальних електродів *N* до *8* і виконаємо перерахунок. У результаті остаточно отримали: *R =3,91 Ом. при* кількості вертикальних електродів *N=8*.

8.5 Висновки до розділу

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров'я працівників, а також підвищує ефективність виробництва вцілому.

З цих міркувань було здійснено аналіз приміщення, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи.

Тільки повна усвідомленість працівника про можливі небезпеки, що можуть підстерігати його на робочому місці та дотримання вимог нормативних актів з питань охорони праці та відповідних рекомендацій фахівців, дозволять значною мірою знизити негативний вплив шкідливих та небезпечних факторів при роботі з комп'ютером на організм людини.

Виконано розрахунок захисного штучного заземлення, як одного з ключових факторів безпеки програміста.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи для забезпечення конфіденційності інформації з’ємних носіїв.

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 для забезпечення конфіденційності інформації з’ємних носіїв.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем для забезпечення конфіденційності інформації з’ємних носіїв.

– Досліджена система для забезпечення конфіденційності інформації з’ємних носіїв.

– На основі отриманих результатів досліджень створена програмна реалізація системи для забезпечення конфіденційності інформації з’ємних носіїв.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання для забезпечення конфіденційності інформації з’ємних носіїв.

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.

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

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

Програма реалізована на мові високого рівня RAD Studio Delphi 10. 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 REDOC III.

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 19384 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,1 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Коваленко В.Ю. Дослідження та програмна реалізація системи для забезпечення конфіденційності інформації з'ємних носіїв // Збірник праць молодих науковців ЦНТУ. – Вип. 13. – Кропивницький: ЦНТУ, 2022.

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

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

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

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

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

7. Smirnov O., Kovalenko O., Kovalenko A., Kavun S. «Quantitative Risk Assessment Method Development in the Context of the SDLC-model». 2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology

(PIC S&T), 2021, pp. 203-208, doi: 10.1109/PICST54195.2021.9772143 (Scopus).

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

35. 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.

36. 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.

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

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

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

39. 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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

55. 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

56. 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

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

58. 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

59. 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

60. 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

61. Сакулин В.П., Шептовицкий В.М. Безопасность труда при монтаже и эксплуатации электроустановок / В.П.Сакулин, В.М.Шептовицкий. – Л. : “Колос”, 1973. – 238 с.

62. Центр післядипломної освіти та підвищення кваліфікації. – Режим доступу до ресурсу: https://cpo.stu.cn.ua

63. 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).

Додаток А
(обов'язковий)

Технічне завдання

Зміст

					ВКРМ-123.22.0013.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Коваленко В.Ю.			Дослідження та програмна реалізація системи для забезпечення конфіденційності інформації з'ємних носіїв	Літ. М	Аркуш 1 / Аркушів 6
Перевірив		Коваленко О.В.					
Н. Контр.		Гермак В.С.				ЦНТУ КІ-21М-1,4	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи для забезпечення конфіденційності інформації з’ємних носіїв.

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. № 19-13 від 17.08.2022 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи для забезпечення конфіденційності інформації з’ємних носіїв.

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;

– 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 для забезпечення конфіденційності інформації з'ємних носіїв;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище RAD Studio Delphi 10.

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 2022 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинна бути розглянута техніка безпеки та протипожежна профілактика.

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 – 103 аркуші.

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 10.12.2022 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 24.12.2022 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

другим (магістерським) рівнем вищої освіти

_____ Коваленко О.В.

Дослідження та програмна реалізація

системи для забезпечення конфіденційності інформації з’ємних носіїв

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 47

Літера: РП

Кропивницький – 2022 року

Головний файл програми

```
program RC4;

uses
  Forms,
  Frm1 in 'Frm1.pas' {Form1},
  Frm2 in 'Frm2.pas' {AboutBox},
  _LOG in 'RC4.pas' {Form3},
  Frm3 in 'RC4_DATA.pas' {Form4};
Var
  MemHnd: HWND;

{$R *.res}

begin
  Application.Initialize;
M:=CreateFileMapping(HWND($FFFFFFFF), Nil, PAGE_READWRITE, 0, MemFileSize,
MemFileName);
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(TAboutBox, AboutBox);
  Application.CreateForm(TForm3, Form3);
  Application.CreateForm(TForm4, Form4);
  Application.Run;
end.
```

Модуль алгоритму потокового шифрування RC4.pas

```
{Poltovec S.O. 2008 yr}
unit PoltovecRC4;
interface
uses
  Windows, Sysutils;
type
  TRC4Data= record
    Key: array[0..255] of byte;
    OrgKey: array[0..255] of byte;
  end;
var
   s: array [0..255] of Byte;
 i,j: Byte;
implementation

//Ініціалізація S-Box'а
procedure InitRC4Cipher(key: ShortString);
var
     k: array [0..255] of Byte;
     t: Byte;
     l: Cardinal;
 i0,j0: Byte;
begin
  for i0:= 0 to 255 do s[i0]:= i0;
  j0:= 1; l:= Length(key);
  for i0:= 0 to 255 do
    begin
      k[i0]:= Ord(key[j0]);
      if j0 = l then j0:= 0;
      Inc(j0);
    end;
  for i0:= 0 to 255 do
    begin
      j0:= (j0 + k[i0] + s[i0]) mod 256;
      t:= s[i0];
      s[i0]:= s[j0];
      s[j0]:= t;
    end;
  i:= 0;
  j:= 0;
end;
procedure RC4Init;
var
  xKey: array[0..255] of byte;
```

```
  i, j: integer;
  t: byte;
begin
  if (Len<= 0) or (Len> 256) then
    raise Exception.Create('RC4: неправильна довжина ключа');
  for i:= 0 to 255 do
  begin
    Data.Key[i]:= i;
    xKey[i]:= PByte(integer(Key)+(i mod Len))^;
  end;
  j:= 0;
  for i:= 0 to 255 do
  begin
    j:= (j+Data.Key[i]+xKey[i]) and $FF;
    t:= Data.Key[i];
    Data.Key[i]:= Data.Key[j];
    Data.Key[j]:= t;
  end;
  Move(Data.Key,Data.OrgKey,256);
end;
procedure RC4Burn;
begin
  FillChar(Data,Sizeof(Data),$FF);
end;
// Тестування
function RC4SelfTest;
const
  InBlock: array[0..4] of byte= ($dc,$ee,$4c,$f9,$2c);
  OutBlock: array[0..4] of byte= ($f1,$38,$29,$c9,$de);
  Key: array[0..4] of byte= ($61,$8a,$63,$d2,$fb);
var
  Block: array[0..4] of byte;
  Data: TRC4Data;
begin
  RC4Init(Data,@Key,5);
  RC4Crypt(Data,@InBlock,@Block,5);
  Result:= CompareMem(@Block,@OutBlock,5);
  RC4Reset(Data);
  RC4Crypt(Data,@Block,@Block,5);
  Result:= Result and CompareMem(@Block,@InBlock,5);
  RC4Burn(Data);
end;
//Зашифрувати конкретний символ
function GetRC4ByteCiphered(bt: Byte): Byte;
var
  t: Byte;
```

```
begin
  i:= (i + 1)    mod 256;
  j:= (j + s[i]) mod 256;
  t:= s[i];
  s[i]:= s[j];
  s[j]:= t;
  t:= (s[i] + s[j]) mod 256;
  Result:= bt XOR s[t];
end;
//Застосувати RC4 шифр до потоку даних
function ApplyRC4ToData(Data: TStream; var Buffer: TStream; key: ShortString):
Boolean; stdcall;
var
   i: Cardinal;
   d: Byte;
 pos: Cardinal;
begin
  if (key = '')OR(Buffer = Data)OR(Buffer = nil)OR(Data = nil)OR(Data.Size =
0)OR(Buffer.Size <> 0) then
    begin
      Result:= false;
      Exit;
    end;
  pos:= Data.Position;
  Data.Position:= 0;
  Buffer.CopyFrom(Data,Data.Size);
  Buffer.Position:= 0;
  Data.Position:= 0;
  try
    InitRC4Cipher(key);
    for i:= 0 to Buffer.Size-1 do
      begin
        Data.ReadBuffer(d,1);
        d:= GetRC4ByteCiphered(d);
        Buffer.WriteBuffer(d,1);
      end;
  except
    Result:= false;
    Exit;
  end;
  Data.Position:= pos;
  Buffer.Position:= 0;
  Result:= true;
end;
```

Модуль обробки інформації та визова функцій RC4_DATA.pas

```
{Poltovec S.O. 2008 yr}
Unit RC4DATA;

Interface
Type
TDA = ^longint;

TCompare = (Lt, St, Eq, Er);

TSign = (negative, positive);

TRC4DATA = Record
Sign: TSign;
Number: TDA;
End;
Procedure RC4DATADestroy(Var GInt: TRC4DATA);
Procedure RC4DATACopy(Var GInt, Copied: TRC4DATA);
Procedure zeronetochar8(Var g: char; Var x: String);
Procedure zeronetochar6(Var g: integer; Var x: String);
Function IntToStr1(Var b: integer): String;
Function StrToInt4(Var S: String): longint;
Function IntToStr4(i: longint): String;
Function min(i1, i2: longint): longint;
Procedure Setlength(Var number: TDA; oldsize, newsize: longint);
Procedure initialize8(Var trans: Array Of 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 RC4DATAToBase2String(RC4DATA: TRC4DATA; Var S: String);
Procedure Base2StringToRC4DATA(S: String; Var RC4DATA: TRC4DATA);
Procedure RC4DATAToBase256String(Const RC4DATA: TRC4DATA; Var str256: String);
Procedure Base256StringToRC4DATA(str256: String; Var RC4DATA: TRC4DATA);
Procedure PGPMPIToRC4DATA(PGPMPI: String; Var RC4DATA: TRC4DATA);
Procedure RC4DATAToBase10String(Var RC4DATA: TRC4DATA; Var Base10: String);
Function RC4DATACompareAbs(Var RC4DATA1, RC4DATA2: TRC4DATA): TCompare;
Procedure RC4DATAChangeSign(Var RC4DATA: TRC4DATA);
Procedure RC4DATAAdd(Var RC4DATA1, RC4DATA2, Sum: TRC4DATA);
Procedure RC4DATASub(Var RC4DATA1, RC4DATA2, dif: TRC4DATA);
Procedure RC4DATAMulByInt(Var RC4DATA, res: TRC4DATA; by: longint);
Procedure RC4DATAMulByIntbis(Var RC4DATA: TRC4DATA; by: longint);
```

```
Procedure RC4DATADivByInt(Var RC4DATA, res: TRC4DATA; by: longint; Var modres:
Procedure RC4DATAAbs(Var RC4DATA: TRC4DATA);
Procedure RC4DATAShiftLeft(Var RC4DATA: TRC4DATA);
Procedure RC4DATAShiftRight(Var RC4DATA: TRC4DATA);
Procedure RC4DATAShiftLeftBy15(Var RC4DATA: TRC4DATA);
Procedure RC4DATAShiftRightBy15(Var RC4DATA: TRC4DATA);
Procedure RC4DATAAddBis(Var RC4DATA1, RC4DATA2: TRC4DATA);
Procedure RC4DATASubBis(Var RC4DATA1, RC4DATA2: TRC4DATA);
Procedure RC4DATAMul(Var RC4DATA1, RC4DATA2, Prod: TRC4DATA);
Procedure RC4DATASquare(Var RC4DATA, Square: TRC4DATA);
Procedure RC4DATAExp(Var RC4DATA, exp, res: TRC4DATA);
Procedure RC4DATAFac(Var RC4DATA, res: TRC4DATA);
Procedure RC4DATADivMod(Var RC4DATA1, RC4DATA2, QRC4DATA, MRC4DATA: TRC4DATA);
Procedure RC4DATADiv(Var RC4DATA1, RC4DATA2, QRC4DATA: TRC4DATA);
Procedure RC4DATAMod(Var RC4DATA1, RC4DATA2, MRC4DATA: TRC4DATA);
Procedure RC4DATAModBis(Var RC4DATA, RC4DATAOut: TRC4DATA; b, head: longint);
Procedure RC4DATAMontgomeryModExp(Var RC4DATA, exp, modb, res: TRC4DATA);
Procedure RC4DATAModExp(Var RC4DATA, exp, modb, res: TRC4DATA);
Procedure RC4DATATrialDiv9999(Var RC4DATA: TRC4DATA; Var ok: boolean);
Procedure RC4DATARandom1(Var Seed, RandomRC4DATA: TRC4DATA);
Procedure RC4DATALegendreSymbol(Var a, p: TRC4DATA; Var L: integer);
Procedure RC4DATASquareRootModP(Square, Prime: TRC4DATA; Var SquareRoot:
TRC4DATA);

Implementation

Const
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', '+', '/');
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);

Var
trans: Array[0..255] Of String;

Procedure RC4DATADestroy(Var GInt: TRC4DATA);
```

```
Begin
If GInt.Number <> Nil Then Freemem(GInt.Number);
GInt.Number:= Nil;
End;

Procedure RC4DATACopy(Var GInt, Copied: TRC4DATA);
Var
i: longint;
Begin
Copied.Sign:= GInt.Sign;
Getmem(Copied.Number, 4 * (GInt.Number[0] + 1));
For i:= 0 To GInt.Number[0] Do Copied.Number[i]:= GInt.Number[i];
End;

Procedure zeronetochar8(Var g: char; Var 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; Var 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;

Function IntToStr1(Var b: integer): String;
Begin
If b = 0 Then IntToStr1:= '0' Else IntToStr1:= '1';
```

```
End;

Function StrToInt4(Var S: String): longint;
Var
i, r: integer;
Begin
r:= 0;
For i:= 1 To length(S) Do
Begin
r:= r * 10;
r:= r + (ord(S[i]) - 48);
End;
StrToInt4:= r;
End;

Function IntToStr4(i: longint): String;
Var
j: integer;
r: String;
Begin
r:= '';
For j:= 1 To 4 Do
Begin
r:= chr((i Mod 10) + 48) + r;
i:= i Div 10;
End;
IntToStr4:= r;
End;

Function min(i1, i2: longint): longint;
Begin
If i1 < i2 Then min:= i1 Else min:= i2;
End;

Procedure Setlength(Var number: TDA; oldsize, newsize: longint);
Var
temp: TDA;
i, t: longint;
Begin
t:= min(oldsize, newsize);
getmem(temp, newsize * 4);
For i:= 0 To (t - 1) Do temp[i]:= number[i];
freemem(number);
number:= temp;
End;
```

```
Procedure initialize8(Var trans: Array Of String);
Var
c1, c2, c3, c4, c5, c6, c7, c8: integer;
x: String;
g: char;
Begin
For c1:= 0 To 1 Do
For c2:= 0 To 1 Do
For c3:= 0 To 1 Do
  For c4:= 0 To 1 Do
     For c5:= 0 To 1 Do
        For c6:= 0 To 1 Do
           For c7:= 0 To 1 Do
              For c8:= 0 To 1 Do
              Begin
                 x:= '';
x:=inttostr1(c1)+inttostr1(c2)+inttostr1(c3)+inttostr1(c4)+inttostr1(c5)+inttost
r1(c6)+inttostr1(c7)+inttostr1(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:= inttostr1(c1) + inttostr1(c2) + inttostr1(c3) + inttostr1(c4) +
inttostr1(c5) + inttostr1(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:=inttostr1(c1)+inttostr1(c2)+inttostr1(c3)+
                     inttostr1(c4)+inttostr1(c5)+inttostr1(c6);
             zeronetochar6(g, x);
             trans[ord(PGPchr64[g])]:= x;
          End;
End;

Procedure ConvertBase256to64(Var str256, str64: String);
Var
temp, x: 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
x:= copy(temp, 1, 6);
zeronetochar6(g, x);
str64:= str64 + chr64[g];
delete(temp, 1, 6);
End;
End;

Procedure ConvertBase64to256(Var str64, str256: String);
Var
temp, x: 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
x:= copy(temp, 1, 8);
zeronetochar8(g, x);
str256:= str256 + g;
delete(temp, 1, 8);
End;
End;

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

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

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

```
delete(str2, 1, 8);
End;
End;


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


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;


Procedure ConvertHexStringToBase256String(HexStr: String; Var Str256: String);
Var
i: longint;
b, h1, h2: byte;
Begin
```

```
Str256:= '';
For i:= 1 To (length(Hexstr) Div 2) Do
Begin
h2:= ord(HexStr[2 * i]);
h1:= ord(HexStr[2 * i - 1]);
If h1 < 58 Then b:= ((h1 - 48) Shl 4) Else b:= ((h1 - 55) Shl 4);
If h2 < 58 Then b:= (b Or (h2 - 48)) Else b:= (b Or (h2 - 55));
Str256:= Str256 + chr(b);
End;
End;


Procedure PGPConvertBase256to64(Var str256, str64: String);
Var
temp, x, a: String;
i, len6: longint;
g: integer;
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;

Procedure PGPConvertBase64to256(str64: String; Var str256: String);
Var
```

```
temp, x: String;
i, j, len8: longint;
g: char;
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;

Procedure PGPConvertBase64to2(str64: String; Var str2: String);
Var
i, j: longint;
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;

Procedure RC4DATAToBase2String(RC4DATA: TRC4DATA; Var S: String);
Var
i: longint;
j: integer;
Begin
S:= '';
For i:= 1 To RC4DATA.Number[0] Do
Begin
For j:= 0 To 14 Do
 If (1 And (RC4DATA.Number[i] Shr j)) = 1 Then S:= '1' + S Else S:= '0' + S;
End;
While (length(S) > 1) And (S[1] = '0') Do delete(S, 1, 1);
If S = '' Then S:= '0';
```

```
End;

Procedure Base2StringToRC4DATA(S: String; Var RC4DATA: TRC4DATA);
Var
i, j, size: longint;
Begin
size:= length(S) Div 15;
If (length(S) Mod 15) <> 0 Then size:= size + 1;
Getmem(RC4DATA.Number, (size + 1) * 4);
RC4DATA.Number[0]:= size;
j:= 1;
RC4DATA.Number[j]:= 0;
i:= 0;
While length(S) > 0 Do
Begin
If S[length(S)] = '1' Then
 RC4DATA.Number[j]:= RC4DATA.Number[j] Or (1 Shl i);
i:= i + 1;
If i = 15 Then
Begin
 i:= 0;
 j:= j + 1;
 If j <= size Then RC4DATA.Number[j]:= 0;
End;
delete(S, length(S), 1);
End;
RC4DATA.Sign:= positive;
End;

Procedure RC4DATAToBase256String(Const RC4DATA: TRC4DATA; Var str256: String);
Var
temp1, x: String;
i, len8: longint;
g: char;
Begin
RC4DATAToBase2String(RC4DATA, temp1);
While (length(temp1) Mod 8) <> 0 Do temp1:= '0' + temp1;
len8:= length(temp1) Div 8;
str256:= '';
For i:= 1 To len8 Do
Begin
x:= copy(temp1, 1, 8);
zeronetochar8(g, x);
str256:= str256 + g;
delete(temp1, 1, 8);
End;
```

```
End;

Procedure Base256StringToRC4DATA(str256: String; Var RC4DATA: TRC4DATA);
Var
temp1: String;
i: longint;
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);
Base2StringToRC4DATA(temp1, RC4DATA);
End;

Procedure PGPMPIToRC4DATA(PGPMPI: String; Var RC4DATA: TRC4DATA);
Var
temp: String;
Begin
temp:= PGPMPI;
delete(temp, 1, 2);
Base256StringToRC4DATA(temp, RC4DATA);
End;


Procedure RC4DATAToPGPMPI(RC4DATA: TRC4DATA; Var PGPMPI: String);
Var
len, i: word;
c: char;
b: byte;
Begin
RC4DATAToBase256String(RC4DATA, 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;

Procedure Base10StringToRC4DATA(Base10: String; Var RC4DATA: TRC4DATA);
Var
i, size: longint;
j: integer;
S, x: String;
sign: TSign;
```

```
Procedure GIntDivByIntBis1(Var GInt: TRC4DATA; by: longint; Var modres:
integer);
Var
i, size: longint;
rest: longint;
Begin
size:= GInt.Number[0];
modres:= 0;
For i:= size Downto 1 Do
Begin
 modres:= modres * 10000;
 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, GInt.Number[0] + 1, 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 4;
If (length(Base10) Mod 4) <> 0 Then size:= size + 1;
Getmem(RC4DATA.Number, 4 * (size + 1));
RC4DATA.Number[0]:= size;
For i:= 1 To (size - 1) Do
Begin
x:= copy(Base10, length(Base10) - 3, 4);
RC4DATA.Number[i]:= StrToInt4(x);
delete(Base10, length(Base10) - 3, 4);
End;
RC4DATA.Number[size]:= StrToInt4(Base10);
```

```
S:= '';
While (RC4DATA.Number[0] <> 1) Or (RC4DATA.Number[1] <> 0) Do
Begin
GIntDivByIntBis1(RC4DATA, 2, j);
S:= inttostr1(j) + S;
End;
S:= '0' + S;
RC4DATADestroy(RC4DATA);
Base2StringToRC4DATA(S, RC4DATA);
RC4DATA.Sign:= sign;
End;


Procedure RC4DATADivByIntBis(Var RC4DATA: TRC4DATA; by: longint; Var modres:
longint);
Var
i, size, rest: longint;
Begin
size:= RC4DATA.Number[0];
modres:= 0;
For i:= size Downto 1 Do
Begin
modres:= modres Shl 15;
rest:= modres Or RC4DATA.Number[i];
RC4DATA.Number[i]:= rest Div by;
modres:= rest Mod by;
End;
While (RC4DATA.Number[size] = 0) And (size > 1) Do size:= size - 1;
If size <> RC4DATA.Number[0] Then
Begin
SetLength(RC4DATA.Number, RC4DATA.Number[0] + 1, size + 1);
RC4DATA.Number[0]:= size;
End;
End;


Procedure RC4DATAToBase10String(Var RC4DATA: TRC4DATA; Var Base10: String);
Var
S: String;
j: longint;
temp: TRC4DATA;
Begin
RC4DATACopy(RC4DATA, temp);
Base10:= '';
While (temp.Number[0] > 1) Or (temp.Number[1] > 0) Do
```

```
Begin
RC4DATADivByIntBis(temp, 10000, j);
S:= IntToStr4(j);
While Length(S) < 4 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 RC4DATA.Sign = negative Then Base10:= '-' + Base10;
End;


Function RC4DATACompareAbs(Var RC4DATA1, RC4DATA2: TRC4DATA): TCompare;
Var
size1, size2, i: longint;
Begin
RC4DATACompareAbs:= Er;
size1:= RC4DATA1.Number[0];
size2:= RC4DATA2.Number[0];
If size1 > size2 Then RC4DATACompareAbs:= Lt Else
If size1 < size2 Then RC4DATACompareAbs:= St Else
Begin
 i:= size2;
 While (RC4DATA1.Number[i] = RC4DATA2.Number[i]) And (i > 1) Do i:= i - 1;
 If RC4DATA1.Number[i] = RC4DATA2.Number[i] Then RC4DATACompareAbs:= Eq Else
    If RC4DATA1.Number[i] < RC4DATA2.Number[i] Then RC4DATACompareAbs:= St Else
       If RC4DATA1.Number[i] > RC4DATA2.Number[i] Then RC4DATACompareAbs:= Lt;
End;
End;

Procedure RC4DATAChangeSign(Var RC4DATA: TRC4DATA);
Begin
If RC4DATA.Sign = negative Then RC4DATA.Sign:= positive Else RC4DATA.Sign:=
negative;
End;


Procedure RC4DATAAdd(Var RC4DATA1, RC4DATA2, Sum: TRC4DATA);
Var
i, size1, size2, size: longint;
rest: integer;
Trest: longint;
Begin
size1:= RC4DATA1.Number[0];
size2:= RC4DATA2.Number[0];
If size1 < size2 Then RC4DATAAdd(RC4DATA2, RC4DATA1, Sum) Else
```

```
Begin
If RC4DATA1.Sign = RC4DATA2.Sign Then
Begin
 Sum.Sign:= RC4DATA1.Sign;
 Getmem(Sum.Number, (size1 + 2) * 4);
 rest:= 0;
 For i:= 1 To size2 Do
 Begin
    Trest:= RC4DATA1.Number[i] + RC4DATA2.Number[i] + rest;
    Sum.Number[i]:= Trest And 32767;
    rest:= Trest Shr 15;
 End;
 For i:= (size2 + 1) To size1 Do
 Begin
    Trest:= RC4DATA1.Number[i] + rest;
    Sum.Number[i]:= Trest And 32767;
    rest:= Trest Shr 15;
 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, Sum.number[0] + 1, size +
1);
 Sum.Number[0]:= size;
End
Else
Begin
 If RC4DATACompareAbs(RC4DATA2, RC4DATA1) = Lt Then RC4DATAAdd(RC4DATA2,
RC4DATA1, Sum)
 Else
 Begin
    Getmem(Sum.Number, 4 * (size1 + 1));
    rest:= 0;
    For i:= 1 To size2 Do
    Begin
       Trest:= 32768 + RC4DATA1.Number[i] - RC4DATA2.Number[i] + rest;
       Sum.Number[i]:= Trest And 32767;
       If (Trest > 32767) Then rest:= 0 Else rest:= -1;
    End;
    For i:= (size2 + 1) To size1 Do
    Begin
       Trest:= 32768 + RC4DATA1.Number[i] + rest;
       Sum.Number[i]:= Trest And 32767;
       If (Trest > 32767) 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 SetLength(Sum.Number, size1 + 1, size + 1);
    Sum.Number[0]:= size;
    Sum.Sign:= RC4DATA1.Sign;
 End;
End;
End;
End;

Procedure RC4DATASub(Var RC4DATA1, RC4DATA2, dif: TRC4DATA);
Begin
RC4DATAChangeSign(RC4DATA2);
RC4DATAAdd(RC4DATA1, RC4DATA2, dif);
RC4DATAChangeSign(RC4DATA2);
End;

Procedure RC4DATAMulByInt(Var RC4DATA, res: TRC4DATA; by: longint);
Var
i, size: longint;
Trest, rest: longint;
Begin
size:= RC4DATA.Number[0];
Getmem(res.Number, 4 * (size + 2));
rest:= 0;
For i:= 1 To size Do
Begin
Trest:= RC4DATA.Number[i] * by + rest;
res.Number[i]:= Trest And 32767;
rest:= Trest Shr 15;
End;
If rest <> 0 Then
Begin
size:= size + 1;
Res.Number[size]:= rest;
End
Else SetLength(Res.Number, size + 2, size + 1);
Res.Number[0]:= size;
Res.Sign:= RC4DATA.Sign;
End;

Procedure RC4DATAMulByIntbis(Var RC4DATA: TRC4DATA; by: longint);
Var
i, size: longint;
Trest, rest: longint;
Begin
```

```
size:= RC4DATA.Number[0];
Setlength(RC4DATA.Number, size + 1, size + 2);
rest:= 0;
For i:= 1 To size Do
Begin
Trest:= RC4DATA.Number[i] * by + rest;
RC4DATA.Number[i]:= Trest And 32767;
rest:= Trest Shr 15;
End;
If rest <> 0 Then
Begin
size:= size + 1;
RC4DATA.Number[size]:= rest;
End
Else SetLength(RC4DATA.Number, size + 2, size + 1);
RC4DATA.Number[0]:= size;
End;

Procedure RC4DATADivByInt(Var RC4DATA, res: TRC4DATA; by: longint; Var modres:
longint);
Var
i, size: longint;
rest: longint;
Begin
size:= RC4DATA.Number[0];
Getmem(res.Number, 4 * (size + 1));
modres:= 0;
For i:= size Downto 1 Do
Begin
modres:= modres Shl 15;
rest:= modres Or RC4DATA.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;
If size <> RC4DATA.Number[0] Then SetLength(res.Number, RC4DATA.Number[0] + 1,
size + 1);
res.Number[0]:= size;
Res.Sign:= RC4DATA.Sign;
End;

Procedure RC4DATAModByInt(Var RC4DATA: TRC4DATA; by: longint; Var modres:
longint);
Var
i, size, rest: longint;
Begin
```

```
size:= RC4DATA.Number[0];
modres:= 0;
For i:= size Downto 1 Do
Begin
modres:= modres Shl 15;
rest:= modres + RC4DATA.Number[i];
modres:= rest Mod by;
End;
End;


Procedure RC4DATAAbs(Var RC4DATA: TRC4DATA);
Begin
RC4DATA.Sign:= positive;
End;

Procedure RC4DATAShiftLeft(Var RC4DATA: TRC4DATA);
Var
l, m, i, size: longint;
Begin
size:= RC4DATA.Number[0];
l:= 0;
For i:= 1 To Size Do
Begin
m:= RC4DATA.Number[i] Shr 14;
RC4DATA.Number[i]:= ((RC4DATA.Number[i] Shl 1) Or l) And 32767;
l:= m;
End;
If l <> 0 Then
Begin
setlength(RC4DATA.Number, size + 1, size + 2);
RC4DATA.Number[size + 1]:= l;
RC4DATA.Number[0]:= size + 1;
End;
End;


Procedure RC4DATAShiftRight(Var RC4DATA: TRC4DATA);
Var
l, m, i, size: longint;
Begin
size:= RC4DATA.Number[0];
l:= 0;
For i:= size Downto 1 Do
Begin
m:= RC4DATA.Number[i] And 1;
```

```
RC4DATA.Number[i]:= (RC4DATA.Number[i] Shr 1) Or l;
l:= m Shl 14;
End;
If (RC4DATA.Number[size] = 0) And (size > 1) Then
Begin
setlength(RC4DATA.Number, size + 1, size);
RC4DATA.Number[0]:= size - 1;
End;
End;

Procedure RC4DATAShiftLeftBy15(Var RC4DATA: TRC4DATA);
Var
f1, f2, i, size: longint;
Begin
size:= RC4DATA.Number[0];
SetLength(RC4DATA.Number, size + 1, size + 2);
f1:= 0;
For i:= 1 To size Do
Begin
f2:= RC4DATA.Number[i];
RC4DATA.Number[i]:= f1;
f1:= f2;
End;
RC4DATA.Number[size + 1]:= f1;
RC4DATA.Number[0]:= size + 1;
End;


Procedure RC4DATAShiftRightBy15(Var RC4DATA: TRC4DATA);
Var
size, i: longint;
Begin
size:= RC4DATA.Number[0];
If size > 1 Then
Begin
For i:= 1 To size - 1 Do
Begin
 RC4DATA.Number[i]:= RC4DATA.Number[i + 1];
End;
SetLength(RC4DATA.Number, size + 1, Size);
RC4DATA.Number[0]:= size - 1;
End
Else RC4DATA.Number[1]:= 0;
End;

Procedure RC4DATAAddBis(Var RC4DATA1, RC4DATA2: TRC4DATA);
```

```
Var
i, size1, size2, Trest: longint;
rest: integer;
Begin
size1:= RC4DATA1.Number[0];
size2:= RC4DATA2.Number[0];
rest:= 0;
For i:= 1 To size2 Do
Begin
Trest:= RC4DATA1.Number[i] + RC4DATA2.Number[i] + rest;
rest:= Trest Shr 15;
RC4DATA1.Number[i]:= Trest And 32767;
End;
For i:= size2 + 1 To size1 Do
Begin
Trest:= RC4DATA1.Number[i] + rest;
rest:= Trest Shr 15;
RC4DATA1.Number[i]:= Trest And 32767;
End;
If rest <> 0 Then
Begin
SetLength(RC4DATA1.Number, size1 + 1, size1 + 2);
RC4DATA1.Number[0]:= size1 + 1;
RC4DATA1.Number[size1 + 1]:= rest;
End;
End;


Procedure RC4DATASubBis(Var RC4DATA1, RC4DATA2: TRC4DATA);
Var
i, size1, size2: longint;
rest: integer;
Trest: longint;
Begin
size1:= RC4DATA1.Number[0];
size2:= RC4DATA2.Number[0];
rest:= 0;
For i:= 1 To size2 Do
Begin
Trest:= 32768 + RC4DATA1.Number[i] - RC4DATA2.Number[i] + rest;
If (Trest > 32767) Then rest:= 0 Else rest:= -1;
RC4DATA1.Number[i]:= Trest And 32767;
End;
For i:= size2 + 1 To size1 Do
Begin
Trest:= 32768 + RC4DATA1.Number[i] + rest;
```

```
If (Trest > 32767) Then rest:= 0 Else rest:= -1;
RC4DATA1.Number[i]:= Trest And 32767;
End;
i:= size1;
While (RC4DATA1.Number[i] = 0) And (i > 1) Do i:= i - 1;
If i <> size1 Then
Begin
SetLength(RC4DATA1.Number, size1 + 1, i + 1);
RC4DATA1.Number[0]:= i;
End;
End;

Procedure RC4DATAMul(Var RC4DATA1, RC4DATA2, Prod: TRC4DATA);
Var
i, j, size, size1, size2: longint;
rest, Trest: longint;
Begin
size1:= RC4DATA1.Number[0];
size2:= RC4DATA2.Number[0];
size:= size1 + size2;
Getmem(Prod.Number, 4 * (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] + RC4DATA1.Number[j] * RC4DATA2.Number[i] +
rest;
 Prod.Number[j + i - 1]:= Trest And 32767;
 rest:= Trest Shr 15;
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, prod.Number[0]+1, size+1);
Prod.Number[0]:= size;
End;
If RC4DATA1.Sign = RC4DATA2.Sign Then Prod.Sign:= Positive Else prod.Sign:=
negative;
End;
```

```
Procedure RC4DATASquare(Var RC4DATA, Square: TRC4DATA);
Var
size, size1, i, j: longint;
rest, Trest: longint;
Begin
size1:= RC4DATA.Number[0];
size:= 2 * size1;
Getmem(Square.Number, 4 * (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] + RC4DATA.Number[i] * RC4DATA.Number[i];
Square.Number[2 * i - 1]:= Trest And 32767;
rest:= Trest Shr 15;
For j:=i+1 To size1 Do
Begin
 Trest:=Square.Number[i+j-1]+2*RC4DATA.Number[i]*RC4DATA.Number[j]+rest;
 Square.Number[i + j - 1]:= Trest And 32767;
 rest:= Trest Shr 15;
End;
Square.Number[i + size1]:= rest;
End;
Square.Sign:= positive;
While (Square.Number[size] = 0) And (size > 1) Do size:= size - 1;
If size <> (2 * size1) Then
Begin
SetLength(Square.Number, 2 * size1 + 1, size + 1);
Square.Number[0]:= size;
End;
End;

Procedure RC4DATAExp(Var RC4DATA, exp, res: TRC4DATA);
Var
temp2, temp3: TRC4DATA;
S: String;
i: longint;
Begin
RC4DATAToBase2String(exp, S);
If S[length(S)] = '0' Then Base10StringToRC4DATA('1', res) Else
RC4DATACopy(RC4DATA, res);
RC4DATACopy(RC4DATA, temp2);
If length(S) > 1 Then
For i:= (length(S) - 1) Downto 1 Do
Begin
```

```
 RC4DATASquare(temp2, temp3);
 RC4DATADestroy(temp2);
 RC4DATACopy(temp3, temp2);
 RC4DATADestroy(temp3);
 If S[i] = '1' Then
 Begin
    RC4DATAMul(res, temp2, temp3);
    RC4DATADestroy(res);
    RC4DATACopy(temp3, res);
    RC4DATADestroy(temp3);
 End;
End;
RC4DATADestroy(temp2);
End;

Procedure RC4DATAFac(Var RC4DATA, res: TRC4DATA);
Var
one, temp, temp1: TRC4DATA;
Begin
RC4DATACopy(RC4DATA, temp);
Base10StringToRC4DATA('1', res);
Base10StringToRC4DATA('1', one);

While Not (RC4DATACompareAbs(temp, one) = Eq) Do
Begin
RC4DATAMul(temp, res, temp1);
RC4DATACopy(temp1, res);
RC4DATADestroy(temp1);
RC4DATASubBis(temp, one);
End;

RC4DATADestroy(one);
RC4DATADestroy(temp);
End;

Procedure RC4DATADivMod(Var RC4DATA1, RC4DATA2, QRC4DATA, MRC4DATA: TRC4DATA);
Var
one, zero, temp1, temp2: TRC4DATA;
s1, s2: TSign;
i, j, s, t: longint;
Begin
s1:= RC4DATA1.Sign;
s2:= RC4DATA2.Sign;
RC4DATAAbs(RC4DATA1);
RC4DATAAbs(RC4DATA2);
RC4DATACopy(RC4DATA1, MRC4DATA);
```

```
RC4DATACopy(RC4DATA2, temp1);

If RC4DATACompareAbs(RC4DATA1, RC4DATA2) <> St Then
Begin
s:= RC4DATA1.Number[0] - RC4DATA2.Number[0];
Getmem(QRC4DATA.Number, 4 * (s + 2));
QRC4DATA.Number[0]:= s + 1;
For t:= 1 To s Do
Begin
 RC4DATAShiftLeftBy15(temp1);
 QRC4DATA.Number[t]:= 0;
End;
j:= s + 1;
QRC4DATA.Number[j]:= 0;
While RC4DATACompareAbs(MRC4DATA, RC4DATA2) <> St Do
Begin
 While RC4DATACompareAbs(MRC4DATA, temp1) <> St Do
 Begin
    If MRC4DATA.Number[0] > temp1.Number[0] Then
       i:= (32768 * MRC4DATA.Number[MRC4DATA.Number[0]] +
MRC4DATA.Number[MRC4DATA.Number[0] - 1]) Div (temp1.Number[temp1.Number[0]] + 1)
    Else i:= MRC4DATA.Number[MRC4DATA.Number[0]] Div
(temp1.Number[temp1.Number[0]] + 1);
    If (i <> 0) Then
    Begin
       RC4DATACopy(temp1, temp2);
       RC4DATAMulByIntBis(temp2, i);
       RC4DATASubBis(MRC4DATA, temp2);
       QRC4DATA.Number[j]:= QRC4DATA.Number[j] + i;
       If RC4DATACompareAbs(MRC4DATA, temp2) <> St Then
       Begin
          QRC4DATA.Number[j]:= QRC4DATA.Number[j] + i;
          RC4DATASubBis(MRC4DATA, temp2);
       End;
       RC4DATADestroy(temp2);
    End Else
    Begin
       QRC4DATA.Number[j]:= QRC4DATA.Number[j] + 1;
       RC4DATASubBis(MRC4DATA, temp1);
    End;
 End;
 If MRC4DATA.Number[0] <= temp1.Number[0] Then
    If RC4DATACompareAbs(temp1, RC4DATA2) <> Eq Then
    Begin
       RC4DATAShiftRightBy15(temp1);
       j:= j - 1;
```

```
    End;
End;
End
Else Base10StringToRC4DATA('0', QRC4DATA);
s:= QRC4DATA.Number[0];
While (s > 1) And (QRC4DATA.Number[s] = 0) Do s:= s - 1;
If s < QRC4DATA.Number[0] Then
Begin
setlength(QRC4DATA.Number, QRC4DATA.Number[0] + 1, s + 1);
QRC4DATA.Number[0]:= s;
End;
QRC4DATA.Sign:= positive;

RC4DATADestroy(temp1);
Base10StringToRC4DATA('0', zero);
Base10StringToRC4DATA('1', one);
If s1 = negative Then
Begin
If RC4DATACompareAbs(MRC4DATA, zero) <> Eq Then
Begin
 RC4DATAadd(QRC4DATA, one, temp1);
 RC4DATADestroy(QRC4DATA);
 RC4DATACopy(temp1, QRC4DATA);
 RC4DATADestroy(temp1);
 RC4DATAsub(RC4DATA2, MRC4DATA, temp1);
 RC4DATADestroy(MRC4DATA);
 RC4DATACopy(temp1, MRC4DATA);
 RC4DATADestroy(temp1);
End;
If s2 = positive Then QRC4DATA.Sign:= negative;
End
Else QRC4DATA.Sign:= s2;
RC4DATADestroy(one);
RC4DATADestroy(zero);

RC4DATA1.Sign:= s1;
RC4DATA2.Sign:= s2;
End;

Procedure RC4DATADiv(Var RC4DATA1, RC4DATA2, QRC4DATA: TRC4DATA);
Var
one, zero, temp1, temp2, MRC4DATA: TRC4DATA;
s1, s2: TSign;
i, j, s, t: longint;
Begin
s1:= RC4DATA1.Sign;
```

```
s2:= RC4DATA2.Sign;
RC4DATAAbs(RC4DATA1);
RC4DATAAbs(RC4DATA2);
RC4DATACopy(RC4DATA1, MRC4DATA);
RC4DATACopy(RC4DATA2, temp1);

If RC4DATACompareAbs(RC4DATA1, RC4DATA2) <> St Then
Begin
s:= RC4DATA1.Number[0] - RC4DATA2.Number[0];
Getmem(QRC4DATA.Number, 4 * (s + 2));
QRC4DATA.Number[0]:= s + 1;
For t:= 1 To s Do
Begin
 RC4DATAShiftLeftBy15(temp1);
 QRC4DATA.Number[t]:= 0;
End;
j:= s + 1;
QRC4DATA.Number[j]:= 0;
While RC4DATACompareAbs(MRC4DATA, RC4DATA2) <> St Do
Begin
 While RC4DATACompareAbs(MRC4DATA, temp1) <> St Do
 Begin
    If MRC4DATA.Number[0] > temp1.Number[0] Then
   i:= (32768 * MRC4DATA.Number[MRC4DATA.Number[0]] +
MRC4DATA.Number[MRC4DATA.Number[0] - 1])
      Div (temp1.Number[temp1.Number[0]] + 1)
    Else i:= MRC4DATA.Number[MRC4DATA.Number[0]] Div
(temp1.Number[temp1.Number[0]] + 1);
    If (i <> 0) Then
    Begin
       RC4DATACopy(temp1, temp2);
       RC4DATAMulByIntBis(temp2, i);
       RC4DATASubBis(MRC4DATA, temp2);
       QRC4DATA.Number[j]:= QRC4DATA.Number[j] + i;
       If RC4DATACompareAbs(MRC4DATA, temp2) <> St Then
       Begin
          QRC4DATA.Number[j]:= QRC4DATA.Number[j] + i;
          RC4DATASubBis(MRC4DATA, temp2);
       End;
       RC4DATADestroy(temp2);
    End Else
    Begin
       QRC4DATA.Number[j]:= QRC4DATA.Number[j] + 1;
       RC4DATASubBis(MRC4DATA, temp1);
    End;
 End;
```

```
 If MRC4DATA.Number[0] <= temp1.Number[0] Then
    If RC4DATACompareAbs(temp1, RC4DATA2) <> Eq Then
    Begin
       RC4DATAShiftRightBy15(temp1);
       j:= j - 1;
    End;
End;
End
Else Base10StringToRC4DATA('0', QRC4DATA);
s:= QRC4DATA.Number[0];
While (s > 1) And (QRC4DATA.Number[s] = 0) Do s:= s - 1;
If s < QRC4DATA.Number[0] Then
Begin
setlength(QRC4DATA.Number, QRC4DATA.Number[0] + 1, s + 1);
QRC4DATA.Number[0]:= s;
End;
QRC4DATA.Sign:= positive;
RC4DATADestroy(temp1);
Base10StringToRC4DATA('0', zero);
Base10StringToRC4DATA('1', one);
If s1 = negative Then
Begin
If RC4DATACompareAbs(MRC4DATA, zero) <> Eq Then
Begin
 RC4DATAadd(QRC4DATA, one, temp1);
 RC4DATADestroy(QRC4DATA);
 RC4DATACopy(temp1, QRC4DATA);
 RC4DATADestroy(temp1);
 RC4DATAsub(RC4DATA2, MRC4DATA, temp1);
 RC4DATADestroy(MRC4DATA);
 RC4DATACopy(temp1, MRC4DATA);
 RC4DATADestroy(temp1);
End;
If s2 = positive Then QRC4DATA.Sign:= negative;
End
Else QRC4DATA.Sign:= s2;
RC4DATADestroy(one);
RC4DATADestroy(zero);
RC4DATADestroy(MRC4DATA);
RC4DATA1.Sign:= s1;
RC4DATA2.Sign:= s2;
End;

Procedure RC4DATAMod(Var RC4DATA1, RC4DATA2, MRC4DATA: TRC4DATA);
Var
one, zero, temp1, temp2: TRC4DATA;
```

```
s1, s2: TSign;
i, s, t: longint;
Begin
s1:= RC4DATA1.Sign;
s2:= RC4DATA2.Sign;
RC4DATAAbs(RC4DATA1);
RC4DATAAbs(RC4DATA2);
RC4DATACopy(RC4DATA1, MRC4DATA);
RC4DATACopy(RC4DATA2, temp1);

If RC4DATACompareAbs(RC4DATA1, RC4DATA2) <> St Then
Begin
s:= RC4DATA1.Number[0] - RC4DATA2.Number[0];
For t:= 1 To s Do RC4DATAShiftLeftBy15(temp1);
While RC4DATACompareAbs(MRC4DATA, RC4DATA2) <> St Do
Begin
 While RC4DATACompareAbs(MRC4DATA, temp1) <> St Do
 Begin
    If MRC4DATA.Number[0] > temp1.Number[0] Then
    i:= (32768 * MRC4DATA.Number[MRC4DATA.Number[0]] +
MRC4DATA.Number[MRC4DATA.Number[0] - 1])
         Div (temp1.Number[temp1.Number[0]] + 1)
    Else i:= MRC4DATA.Number[MRC4DATA.Number[0]] Div
(temp1.Number[temp1.Number[0]] + 1);
    If (i <> 0) Then
    Begin
       RC4DATACopy(temp1, temp2);
       RC4DATAMulByIntBis(temp2, i);
       RC4DATASubBis(MRC4DATA, temp2);
       If RC4DATACompareAbs(MRC4DATA, temp2) <> St Then RC4DATASubBis(MRC4DATA,
temp2);
       RC4DATADestroy(temp2);
    End Else RC4DATASubBis(MRC4DATA, temp1);
  If RC4DATACompareAbs(MRC4DATA, temp1) <> St Then
RC4DATASubBis(MRC4DATA,temp1);
 End;
 If MRC4DATA.Number[0] <= temp1.Number[0] Then
    If RC4DATACompareAbs(temp1, RC4DATA2) <> Eq Then
RC4DATAShiftRightBy15(temp1);
End;
End;

RC4DATADestroy(temp1);
Base10StringToRC4DATA('0', zero);
Base10StringToRC4DATA('1', one);
If s1 = negative Then
```

```
Begin
If RC4DATACompareAbs(MRC4DATA, zero) <> Eq Then
Begin
 RC4DATASub(RC4DATA2, MRC4DATA, temp1);
 RC4DATADestroy(MRC4DATA);
 RC4DATACopy(temp1, MRC4DATA);
 RC4DATADestroy(temp1);
End;
End;
RC4DATADestroy(one);
RC4DATADestroy(zero);

RC4DATA1.Sign:= s1;
RC4DATA2.Sign:= s2;
End;

Procedure RC4DATASquareMod(Var RC4DATA, Modb, RC4DATASM: TRC4DATA);
Var
temp: TRC4DATA;
Begin
RC4DATASquare(RC4DATA, temp);
RC4DATAMod(temp, Modb, RC4DATASM);
RC4DATADestroy(temp);
End;

Procedure RC4DATAAddMod(Var RC4DATA1, RC4DATA2, base, RC4DATAres: TRC4DATA);
Var
temp: TRC4DATA;
Begin
RC4DATAadd(RC4DATA1, RC4DATA2, temp);
RC4DATAMod(temp, base, RC4DATAres);
RC4DATADestroy(temp);
End;


Procedure RC4DATAMulMod(Var RC4DATA1, RC4DATA2, base, RC4DATAres: TRC4DATA);
Var
temp: TRC4DATA;
Begin
RC4DATAMul(RC4DATA1, RC4DATA2, temp);
RC4DATAMod(temp, base, RC4DATAres);
RC4DATADestroy(temp);
End;

Procedure RC4DATAGCD(Var RC4DATA1, RC4DATA2, GCD: TRC4DATA);
Var
```

```
k: TCompare;
temp1, temp2, temp3: TRC4DATA;
Begin
k:= RC4DATACompareAbs(RC4DATA1, RC4DATA2);
If (k = Eq) Then RC4DATACopy(RC4DATA1, GCD) Else
If (k = St) Then RC4DATAGCD(RC4DATA2, RC4DATA1, GCD) Else
Begin
 RC4DATACopy(RC4DATA1, temp1);
 RC4DATACopy(RC4DATA2, temp2);
 While (temp2.Number[0] > 1) Or (temp2.Number[1] > 0) Do
 Begin
    RC4DATAMod(temp1, temp2, temp3);
    RC4DATADestroy(temp1);
    RC4DATACopy(temp2, temp1);
    RC4DATADestroy(temp2);
    RC4DATACopy(temp3, temp2);
    RC4DATADestroy(temp3);
 End;
 RC4DATACopy(temp1, GCD);
 RC4DATADestroy(temp2);
 RC4DATADestroy(temp1);
End; End;

Procedure RC4DATALCM(Var RC4DATA1, RC4DATA2, LCM: TRC4DATA);
Var
temp1, temp2: TRC4DATA;
Begin
RC4DATAGCD(RC4DATA1, RC4DATA2, temp1);
RC4DATAmul(RC4DATA1, RC4DATA2, temp2);
RC4DATAdiv(temp2, temp1, LCM);
RC4DATADestroy(temp1);
RC4DATADestroy(temp2);
End;

Procedure RC4DATABezoutBachet(Var RC4DATA1, RC4DATA2, a, b: TRC4DATA);
Var
zero, r1, r2, r3, ta, gcd, temp, temp1, temp2: TRC4DATA;
stop: boolean;
Begin
If RC4DATACompareAbs(RC4DATA1, RC4DATA2) <> St Then
Begin
RC4DATAcopy(RC4DATA1, r1); RC4DATAcopy(RC4DATA2, r2);
Base10StringToRC4DATA('0', zero); Base10StringToRC4DATA('1', a);
Base10StringToRC4DATA('0', ta);
Repeat
 RC4DATAdivmod(r1, r2, temp, r3); RC4DATADestroy(r1);
```

```
 RC4DATACopy(r2, r1); RC4DATADestroy(r2);
 RC4DATACopy(r3, r2);
 RC4DATAmul(ta, temp, temp1);  RC4DATAsub(a, temp1, temp2);
 RC4DATACopy(ta, a);  RC4DATACopy(temp2, ta);
 RC4DATADestroy(temp1); RC4DATADestroy(temp);
 If RC4DATACompareAbs(r3, zero) = Eq Then stop:= true Else stop:= false;
 RC4DATADestroy(r3);
Until stop;
RC4DATAGCD(RC4DATA1, RC4DATA2, gcd); RC4DATAmul(a, RC4DATA1, temp1);
RC4DATAsub(gcd, temp1, temp2); RC4DATADestroy(temp1);
RC4DATAdiv(temp2, RC4DATA2, b); RC4DATADestroy(temp2);
RC4DATADestroy(ta); RC4DATADestroy(r1);
RC4DATADestroy(r2); RC4DATADestroy(gcd);
End
Else RC4DATABezoutBachet(RC4DATA2, RC4DATA1, b, a);
End;

Procedure RC4DATAModInv(Var RC4DATA1, base: TRC4DATA; Var Inverse: TRC4DATA);
Var
zero, one, r1, r2, r3, tb, gcd, temp, temp1, temp2: TRC4DATA;
stop: boolean;
Begin
Base10StringToRC4DATA('1', one);
RC4DATAGCD(RC4DATA1, base, gcd);
If RC4DATACompareAbs(one, gcd) = Eq Then
Begin
RC4DATAcopy(base, r1);
RC4DATAcopy(RC4DATA1, r2);
Base10StringToRC4DATA('0', zero);
Base10StringToRC4DATA('0', inverse);
Base10StringToRC4DATA('1', tb);
Repeat
 RC4DATAdivmod(r1, r2, temp, r3);
 RC4DATADestroy(r1);
 RC4DATACopy(r2, r1);
 RC4DATADestroy(r2);
 RC4DATACopy(r3, r2); RC4DATAmul(tb, temp, temp1);
 RC4DATAsub(inverse, temp1, temp2);
 RC4DATADestroy(inverse);
 RC4DATADestroy(temp1); RC4DATACopy(tb, inverse);
 RC4DATADestroy(tb); RC4DATACopy(temp2, tb);
 RC4DATADestroy(temp2); RC4DATADestroy(temp);
 If RC4DATACompareAbs(r3, zero) = Eq Then stop:= true Else stop:= false;
 RC4DATADestroy(r3);
Until stop;
```

```
If inverse.Sign = negative Then
Begin
 RC4DATAadd(base, inverse, temp);
 RC4DATACopy(temp, inverse);
End;

RC4DATADestroy(tb);
RC4DATADestroy(r1);
RC4DATADestroy(r2);
End;
RC4DATADestroy(gcd);
RC4DATADestroy(one);
End;
Procedure RC4DATAModBis(Var RC4DATA, RC4DATAOut: TRC4DATA; b, head: longint);
Var
i: longint;
Begin
If b <= RC4DATA.Number[0] Then
Begin
Getmem(RC4DATAOut.Number, 4 * (b + 1));
For i:= 0 To b Do RC4DATAOut.Number[i]:= RC4DATA.Number[i];
RC4DATAOut.Number[b]:= RC4DATAOut.Number[b] And head;
i:= b;
While (RC4DATAOut.Number[i] = 0) And (i > 1) Do i:= i - 1;
If i < b Then SetLength(RC4DATAOut.Number, b + 1, i + 1);
RC4DATAOut.Number[0]:= i;
RC4DATAOut.Sign:= positive;
End Else RC4DATACopy(RC4DATA, RC4DATAOut);
End;
Procedure RC4DATAMulModBis(Var RC4DATA1, RC4DATA2, Prod: TRC4DATA; b, head:
longint);
Var
i, j, size, size1, size2, t, rest, Trest: longint;
Begin
size1:= RC4DATA1.Number[0];
size2:= RC4DATA2.Number[0];
size:= min(b, size1 + size2);
Getmem(Prod.Number, 4 * (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] + RC4DATA1.Number[j] * RC4DATA2.Number[i] +
rest;
 Prod.Number[j + i - 1]:= Trest And 32767;
 rest:= Trest Shr 15;
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, prod.Number[0] + 1, size + 1);
Prod.Number[0]:= size;
End;
If RC4DATA1.Sign = RC4DATA2.Sign Then Prod.Sign:= Positive Else prod.Sign:=
negative;
End;
Procedure RC4DATAMontgomeryMod(Var GInt, base, baseInv, MGInt: TRC4DATA; b,
head: longint);
Var
m, temp, temp1: TRC4DATA;
r, i: longint;
Begin
RC4DATAModBis(GInt, temp, b, head);
RC4DATAMulModBis(temp, baseInv, m, b, head);
RC4DATAMul(m, base, temp1);
RC4DATADestroy(temp);
RC4DATAAdd(temp1, GInt, temp);
RC4DATADestroy(temp1);
Getmem(MGInt.Number, 4 * (temp.Number[0] - b + 2));
For i:= 0 To temp.Number[0] - b + 1 Do MGInt.Number[i]:= temp.Number[b - 1 + i];
MGInt.Sign:= positive;
MGInt.Number[0]:= temp.Number[0] - b + 1;
RC4DATADestroy(temp);
If (head Shr 14) = 0 Then RC4DATADivByIntBis(MGInt, head + 1, r)
Else RC4DATAShiftRightBy15(MGInt);
If RC4DATACompareAbs(MGInt, base) <> St Then RC4DATASubBis(MGInt, base);
RC4DATADestroy(temp);
RC4DATADestroy(m);
End;

Procedure RC4DATAMontgomeryModExp(Var RC4DATA, exp, modb, res: TRC4DATA);
Var
temp2, temp3, baseInv, r: TRC4DATA;
```

```
i, j, t, b: longint;
S: String;
head: longint;
Begin
RC4DATAToBase2String(exp, S);
t:= modb.Number[0];
b:= t;

If (modb.Number[t] Shr 14) = 1 Then t:= t + 1;
Getmem(r.Number, 4 * (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:= 32767;
For j:= 13 Downto 0 Do
Begin
 head:= head Shr 1;
 If (modb.Number[t] Shr j) = 1 Then
 Begin
    r.Number[t]:= 1 Shl (j + 1);
    break;
 End;
End;
End
Else
Begin
r.Number[t]:= 1;
head:= 32767;
End;

RC4DATAModInv(modb, r, temp2);
If temp2.Sign = negative Then RC4DATACopy(temp2, BaseInv)
Else
Begin
RC4DATACopy(r, BaseInv);
RC4DATASubBis(BaseInv, temp2);
End;
RC4DATAAbs(BaseInv);
RC4DATADestroy(temp2);
RC4DATAMod(r, modb, res);
RC4DATAMulMod(RC4DATA, res, modb, temp2);
RC4DATADestroy(r);

For i:= length(S) Downto 1 Do
```

```
Begin
If S[i] = '1' Then
Begin
 RC4DATAmul(res, temp2, temp3);
 RC4DATADestroy(res);
 RC4DATAMontgomeryMod(temp3, modb, baseinv, res, b, head);
 RC4DATADestroy(temp3);
End;
RC4DATASquare(temp2, temp3);
RC4DATADestroy(temp2);
RC4DATAMontgomeryMod(temp3, modb, baseinv, temp2, b, head);
RC4DATADestroy(temp3);
End;
RC4DATADestroy(temp2);
RC4DATAMontgomeryMod(res, modb, baseinv, temp3, b, head);
RC4DATACopy(temp3, res);
RC4DATADestroy(temp3);
RC4DATADestroy(baseinv);
End;

Procedure RC4DATAModExp(Var RC4DATA, exp, modb, res: TRC4DATA);
Var
temp2, temp3: TRC4DATA;
i: longint;
S: String;
Begin
If (Modb.Number[1] Mod 2) = 1 Then
Begin
RC4DATAMontgomeryModExp(RC4DATA, exp, modb, res);
exit;
End;
RC4DATAToBase2String(exp, S);
Base10StringToRC4DATA('1', res);
RC4DATAcopy(RC4DATA, temp2);

For i:= length(S) Downto 1 Do
Begin
If S[i] = '1' Then
Begin
 RC4DATAmulMod(res, temp2, modb, temp3);
 RC4DATADestroy(res);
 RC4DATACopy(temp3, res);
End;
RC4DATASquareMod(temp2, Modb, temp3);
RC4DATACopy(temp3, temp2);
End;
```

```
RC4DATADestroy(temp2);
End;

Procedure RC4DATATrialDiv9999(Var RC4DATA: TRC4DATA; Var ok: boolean);
Var
j: longint;
i: integer;
Begin
If ((RC4DATA.Number[1] Mod 2) = 0) Then ok:= false
Else
Begin
i:= 0;
ok:= true;
While ok And (i < 1228) Do
Begin
 i:= i + 1;
 RC4DATAmodbyint(RC4DATA, primes[i], j);
 If j = 0 Then ok:= false;
End;
End;
End;

Procedure RC4DATARandom1(Var Seed, RandomRC4DATA: TRC4DATA);
Var
temp, base: TRC4DATA;
Begin
Base10StringToRC4DATA('281474976710656', base);
Base10StringToRC4DATA('44485709377909', temp);
RC4DATAMulMod(seed, temp, base, RandomRC4DATA);
RC4DATADestroy(temp);
RC4DATADestroy(base);
End;

Procedure RC4DATARabinMiller(Var RC4DATAp: TRC4DATA; nrtest: integer; Var ok:
boolean);
Var
j, b, i: longint;
m, z, temp1, temp2, temp3, zero, one, two, pmin1: TRC4DATA;
ok1, ok2: boolean;
Begin
randomize;
j:= 0;
Base10StringToRC4DATA('0', zero);
Base10StringToRC4DATA('1', one);
Base10StringToRC4DATA('2', two);
RC4DATAsub(RC4DATAp, one, temp1);
```

```
RC4DATAsub(RC4DATAp, one, pmin1);

b:= 0;
While (temp1.Number[1] Mod 2) = 0 Do
Begin
b:= b + 1;
RC4DATAShiftRight(temp1);
End;
m:= temp1;

i:= 0;
ok:= true;
Randomize;
While (i < nrtest) And ok Do
Begin
i:= i + 1;
Base10StringToRC4DATA(inttostr4(Primes[Random(1227) + 1]), temp2);
RC4DATAMontGomeryModExp(temp2, m, RC4DATAp, z);
RC4DATADestroy(temp2);
ok1:= (RC4DATACompareAbs(z, one) = Eq);
ok2:= (RC4DATACompareAbs(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
          RC4DATASquaremod(z, RC4DATAp, temp3);
          RC4DATADestroy(z);
          RC4DATACopy(temp3, z);
          ok1:= (RC4DATACompareAbs(z, one) = Eq);
          ok2:= (RC4DATACompareAbs(z, pmin1) = Eq);
          If ok2 Then j:= b;
       End
       Else If (Not ok2) And (j >= b) Then ok:= false;
    End;
 End;

End
End;
```

```
RC4DATADestroy(zero);
RC4DATADestroy(one);
RC4DATADestroy(two);
RC4DATADestroy(m);
RC4DATADestroy(z);
RC4DATADestroy(pmin1);
End;

Procedure RC4DATAPrimetest(Var RC4DATAp: TRC4DATA; nrRMtests: integer; Var ok:
boolean);
Begin
RC4DATATrialdiv9999(RC4DATAp, ok);
If ok Then RC4DATARabinMiller(RC4DATAp, nrRMtests, ok);
End;

Procedure RC4DATALegendreSymbol(Var a, p: TRC4DATA; Var L: integer);
Var
temp1, temp2, temp3, temp4, temp5, zero, one: TRC4DATA;
i: longint;
ok1, ok2: boolean;
Begin
Base10StringToRC4DATA('0', zero);
Base10StringToRC4DATA('1', one);
RC4DATAMod(a, p, temp1);
If RC4DATACompareAbs(zero, temp1) = Eq Then
Begin
RC4DATADestroy(temp1);
L:= 0;
End
Else
Begin
RC4DATADestroy(temp1);
RC4DATACopy(p, temp1);
RC4DATACopy(a, temp2);
L:= 1;
While RC4DATACompareAbs(temp2, one) <> Eq Do
Begin
 If (temp2.Number[1] Mod 2) = 0 Then
 Begin
    RC4DATASquare(temp1, temp3);
    RC4DATASub(temp3, one, temp4);
    RC4DATADestroy(temp3);
    RC4DATADivByInt(temp4, temp3, 8, i);
    If (temp3.Number[1] Mod 2) = 0 Then ok1:= false Else ok1:= true;
    RC4DATADestroy(temp3);
    RC4DATADestroy(temp4);
```

```
    If ok1 = true Then L:= L * (-1);
    RC4DATADivByIntBis(temp2, 2, i);
 End
 Else
 Begin
    RC4DATASub(temp1, one, temp3);
    RC4DATASub(temp2, one, temp4);
    RC4DATAMul(temp3, temp4, temp5);
    RC4DATADestroy(temp3);
    RC4DATADestroy(temp4);
    RC4DATADivByInt(temp5, temp3, 4, i);
    If (temp3.Number[1] Mod 2) = 0 Then ok2:= false Else ok2:= true;
    RC4DATADestroy(temp5);
    RC4DATADestroy(temp3);
    If ok2 = true Then L:= L * (-1);
    RC4DATAMod(temp1, temp2, temp3);
    RC4DATADestroy(temp1);
    RC4DATACopy(temp2, temp1);
    RC4DATADestroy(temp2);
    RC4DATACopy(temp3, temp2);
 End;
End;
RC4DATADestroy(temp1);
RC4DATADestroy(temp2);
End;
RC4DATADestroy(zero);
RC4DATADestroy(one);
End;


Procedure RC4DATASquareRootModP(Square, Prime: TRC4DATA; Var SquareRoot:
TRC4DATA);
Var
one, n, b, s, r, temp, temp1, temp2, temp3: TRC4DATA;
a, i, j: longint;
L: integer;
Begin
Base2StringToRC4DATA('1', one);
Base2StringToRC4DATA('2', n);
a:= 0;
RC4DATALegendreSymbol(n, Prime, L);
While L <> -1 Do
Begin
RC4DATAAddBis(n, one);
RC4DATALegendreSymbol(n, Prime, L);
End;
```

```
RC4DATACopy(Prime, s);
s.Number[1]:= s.Number[1] - 1;
While (s.Number[1] Mod 2) = 0 Do
Begin
RC4DATAShiftRight(s);
a:= a + 1;
End;
RC4DATAMontgomeryModExp(n, s, Prime, b);
RC4DATAAdd(s, one, temp);
RC4DATAShiftRight(temp);
RC4DATAMontgomeryModExp(Square, temp, Prime, r);
RC4DATADestroy(temp);
RC4DATAModInv(Square, Prime, temp1);

For i:= 0 To (a - 2) Do
Begin
RC4DATASquareMod(r, Prime, temp2);
RC4DATAMulMod(temp1, temp2, Prime, temp);
RC4DATADestroy(temp2);
For j:= 1 To (a - i - 2) Do
Begin
 RC4DATASquareMod(temp, Prime, temp2);
 RC4DATADestroy(temp);
 RC4DATACopy(temp2, temp);
 RC4DATADestroy(temp2);
End;
If RC4DATACompareAbs(temp, one) <> Eq Then
Begin
 RC4DATAMulMod(r, b, Prime, temp3);
 RC4DATADestroy(r);
 RC4DATACopy(temp3, r);
 RC4DATADestroy(temp3);
End;
RC4DATADestroy(temp);
RC4DATADestroy(temp2);
If i = (a - 2) Then break;
RC4DATASquareMod(b, Prime, temp3);
RC4DATADestroy(b);
RC4DATACopy(temp3, b);
RC4DATADestroy(temp3);
End;

RC4DATACopy(r, SquareRoot);
RC4DATADestroy(r);
RC4DATADestroy(s);
RC4DATADestroy(b);
```

```
RC4DATADestroy(temp1);
RC4DATADestroy(one);
RC4DATADestroy(n);
End;

End.
```