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 повідомлень електронної пошти”

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

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

Науменку Ігорю Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи повідомлень електронної пошти*

2. Керівник роботи *Смірнов Сергій Анатолійович, канд. 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 повідомлень електронної пошти*

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
_____ Смірнов С.А.
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2022 р.

Підпис здобувача
_____ Науменко І.О.
(прізвище та ініціали)

АНОТАЦІЯ

Науменко І.О. Дослідження та програмна реалізація системи повідомлень електронної пошти. 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 повідомлень електронної пошти.

Метою розробки є дослідження та програмна реалізація системи повідомлень електронної пошти.

Об'єктом дослідження є процес повідомлень електронної пошти.

Предметом дослідження є методи повідомлень електронної пошти.

Методи дослідження базуються на методах теорії комп'ютерних мереж, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи повідомлень електронної пошти.

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 10.4.

Ключові слова: комп'ютерна інженерія, електронної пошти

ABSTRACT

Naumenko I.O. Research and software implementation of the e-mail message system. 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 the e-mail message system.

The purpose of the development is the research and software implementation of the e-mail message system.

The object of research is the process of e-mail messages.

The subject of research is methods of e-mail messages.

Research methods are based on computer network theory methods, mathematical statistics methods, and software development methods.

The result of the work is a software implementation of the e-mail message system.

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

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

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

The program was developed in the RAD Studio Delphi 10.4 environment.

Keywords: computer engineering, e-mail

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ПЗ	–	програмне забезпечення
УЦ	–	удостовірюючий центр
ЦС	–	центр сертифікації
AH	–	Authentication Header – протокол автентифікуючого заголовку
DES	–	симетричний алгоритм шифрування
ESP	–	протокол інкапсулюючий захист вмісту
IKE	–	протокол обміну ключами в Інтернеті
ISAKMP	–	протокол керування ключами й контекстами безпеки Інтернету
HMAC	–	геш-код повідомлення
MAC	–	код автентифікації
MD5	–	геш-функція
MLA	–	агент списку розсилання
MOSS	–	MIME Object Security Services
OAKLEY	–	протокол обчислення ключів
PEM	–	Privacy Enhanced Mail
PGP	–	Pretty Good Privacy
PKI	–	інфраструктура відкритих ключів
RSA	–	несиметричний алгоритм шифрування
S/MIME	–	Secure Multipurpose Internet Mail Extension
SHA-1	–	геш-функція
SPI	–	Security Parameters Index – індекс параметрів безпеки
SSL	–	Secure Socket Layer
TCP	–	транспортний протокол
TLS	–	Transport Layer Security – протокол безпеки транспортного рівня
VPN	–	Virtual Private Networks – віртуальна приватна мережа

ВСТУП

Актуальність теми. Адміністратори багатьох організацій намагаються захистити повідомлення електронної пошти співробітників. Secure MIME (S/MIME) – рішення безпеки, реалізоване в більшості сучасних поштових програм, що допоможе зберегти конфіденційність, цілісність поштових повідомлень і перевірити дійсність даних. S/MIME забезпечує наскрізний захист – не тільки в процесі пересилання повідомлень, але й при зберіганні в базі даних поштового сервера. S/MIME забезпечує перевірку дійсності даних, конфіденційність і цілісність повідомлень у форматі MIME. S/MIME – відмінний приклад гібридного рішення шифрування, у якому об'єднані достоїнства симетричного й асиметричного шифрів і функцій гешування. Якщо TLS забезпечує безпеку даних у момент пересилання по незахищеній мережі, такій як Інтернет, то S/MIME забезпечує безпеку даних між кінцевими користувачами: S/MIME повідомлення шифрується відправником (з використанням багатобічного шифрування) і передається на сервер відправника в зашифрованій формі. Та ж зашифрована форма використовується, коли повідомлення передається через мережу, коли воно зберігатися на проміжних серверах, і коли воно міститься в папках одержувачів. Тільки одержувачі, використовуючи свої закриті ключі, можуть розшифрувати повідомлення й в той момент, коли вони фактично читають повідомлення: саме повідомлення залишається зашифрованим у папках одержувачів. Для того, що б кінцеві користувачі могли використовувати S/MIME безпеку, всі вони повинні мати своїми власними PKI-ключі. Кожний користувач повинен мати закритий ключ, що безпечно зберігається в місці, доступному тільки для цього користувача, і відповідний йому відкритий ключ, убудований у сертифікат. Цей Сертифікат повинен бути виданий центром сертифікації (удостовірюючим центром), якому довіряють інші користувачі. S/MIME забезпечує потужні розширення функцій поштової безпеки, які можуть бути дуже корисними користувачам.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи повідомлень електронної пошти.

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

– Огляд існуючих систем повідомлень електронної пошти.

– Дослідження системи повідомлень електронної пошти.

– Програмна реалізація системи повідомлень електронної пошти.

Об'єктом дослідження є процес повідомлень електронної пошти.

Предметом дослідження є методи повідомлень електронної пошти.

Методи дослідження базуються на методах теорії комп'ютерних мереж, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод повідомлень електронної пошти.

– Розроблено вітчизняний продукт повідомлень електронної пошти, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі повідомлень електронної пошти.

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 повідомлень електронної пошти, 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

S/MIME (Secure / Multipurpose Internet Mail Extensions) – стандарт для шифрування й підпису в електронній пошті за допомогою відкритого ключа.

S/MIME призначена для забезпечення криптографічної безпеки електронної пошти. Забезпечуються автентифікація, цілісність повідомлення й гарантія збереження авторства, безпека даних (за допомогою шифрування). Більша частина сучасних поштових програм підтримує S/MIME.

Для використання S/MIME необхідно одержати й установити індивідуальний ключ/сертифікат від центра сертифікації (ЦС). Найкраще використовувати різні ключі/сертифікати для цифрового підпису й шифрування, так як це дозволить розкрити за певних умов ключ шифрування (наприклад, за рішенням суду), не дискредитуючи при цьому цифрові підписи. Для шифрування повідомлення потрібно знати сертифікат прийомної сторони, що зазвичай забезпечується автоматично при одержанні листа із сертифікатом. Хоча технічно можливо послати повідомлення, зашифроване сертифікатом одержувача й не підписувати повідомлення власним, наприклад, через відсутність його, на практиці, програми з підтримкою S/MIME зажадають установки сертифіката відправника перед тим як дозволити шифрування повідомлень.

Звичайний основний особистий сертифікат засвідчує ідентичність власника тільки шляхом зв'язування воєдино поштової адреси й сертифіката. Він не засвідчує ні імена, ні рід діяльності. Більше повне посвідчення можна одержати, звернувшись до спеціалізованих ЦС, які надають додаткові (нотаріально еквівалентні) послуги або безпечну інфраструктуру відкритого ключа.

Залежно від політик ЦС, ваш сертифікат і весь його вміст можуть бути відкрито опубліковані для ознайомлення й перевірки. У такому випадку, ваше ім'я й поштова адреса стають доступними для всіх, у тому числі й для пошуку. Інші ЦС можуть публікувати тільки серійні номери й ознаку відозваності. Це необхідний мінімум для забезпечення цілісності інфраструктури відкритого ключа.

Перешкоди при практичному використанні S/MIME

– Не всі додатки електронної пошти можуть обробляти S/MIME, що приводить до листів із прикладеним файлом «smime.p7m», що може привести до непорозуміння.

– Іноді вважається, що S/MIME не сильно підходить для використання вебпошти. Так як вимоги безпеки вимагають, щоб сервер ніколи не зміг одержати доступ до закритого ключа, що зменшує таку перевагу вебпошти, як доступність із будь-якої точки.

– Багато хто розрізняють закриті ключі для розшифровки й для цифрового підпису. Тих, хто готовий надати деякому агентові перший набагато більше, ніж тих, хто готовий надати другий. Якщо необхідно безпечне підтвердження авторства (як і забезпечення відсутності помилкового підтвердження), то другий ключ повинен бути під строгим контролем власника, і тільки його, протягом усього циклу його життя, від створення, до знищення.

– S/MIME спеціально призначено для забезпечення безпеки на шляху від відправника до одержувача. Шкідливе ПЗ, що потрапило в лист, без перешкод дійде до одержувача, так як немає засобів виявити його в цьому проміжку. Отже, необхідно забезпечувати безпеку на кінцевому пристрої.

1.2 Область застосування

У силу важливості мережного сервісу S/MIME вживано багато спроб стандартизувати рішення захищеної електронної пошти. Одна з перших схем

захисту – Privacy Enhanced Mail (PEM) – була запропонована в 1985 році. Наступна схема захисту з'явилася в 1995 році й одержала назву MIME Object Security Services (MOSS). Незважаючи на те, що в них утримувалася множина гарних ідей, негнучкість і відсутність сумісності відсунули PEM і MOSS на другі ролі в сфері створення захищеного обміну повідомленнями й були витиснуті системами Secure/MIME (S/MIME) і Pretty Good Privacy (PGP).

Протоколи S/MIME і PGP підтримуються більшістю програмних продуктів, призначених для колективної роботи й захисту електронної пошти. Найпоширенішим і широко визнаним стандартом обміну повідомленнями став розроблений в 1996 році стандарт S/MIME. Спочатку компанією RSA Data Security була запропонована друга версія специфікації S/MIME v2 [1], [2], пізніше вона була дороблена робочою групою організації IETF і в результаті з'явилася нова, третя версія – S/MIME v3 [3], [4].

S/MIME забезпечує захист від трьох типів порушень безпеки: "підслуховування", або перлюстрації, перекручування повідомлень і фальсифікації [5]. Захист від перлюстрації досягається шляхом шифрування повідомлень. Для захисту від перекручування поштового повідомлення або фальсифікації в S/MIME застосовують цифрові підписи. Цифровий підпис гарантує, що повідомлення не було змінено в процесі передачі. Крім того, її наявність не дозволяє відправникові повідомлення відмовитися від свого авторства. Однак цифровий підпис не гарантує конфіденційність повідомлення. В S/MIME цю функцію виконує цифровий конверт. Шифрування здійснюється за допомогою симетричного криптографічного алгоритму типу DES, Triple-DES або RS2. Симетричний ключ шифрується за допомогою відкритого ключа одержувача, а зашифроване повідомлення й ключ передаються разом.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи повідомлень електронної пошти, 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

У даному розділі розглянемо види поштових клієнтів, і проаналізуємо як у цих клієнтах реалізується захищеність повідомлень. Результати аналізу представимо у вигляді таблиць. Дані таблиці містять порівняння загальної й технічної інформації для ряду поштових клієнтів. У таблицях використовуються наступні скорочення:

– «+» – так, використовує або використовується;

– «–» – ні, не використовує або не використовується;

– «?» – невідомо;

– «З» – підтримка завершена;

– «+-» – частково використовується.

Таблиця 2.1 – Підтримка операційних систем

	Windows	Mac OS X	GNU/Linux	BSD	Інші Unix
Citadel	–	+	+	+	+
Courier	+	–	–	–	–
Eudora	+	+	–	–	–
Foxmail	+	–	–	–	–
Gnus	+	+	+	+	+
GroupWise	+	+	+	?	?
Insight	DOS	–	–	–	–
KMail	+	+	+	+	+
Lotus Notes	+	+	+	–	–
Opera Mail (M2)	+	+	+	+	+
Mail	–	+	–	–	–
Microsoft Office Outlook	+	+	–	–	–

Продовження таблиці 2.1

	Windows	Mac OS X	GNU/Linux	BSD	Інші Unix
Mozilla Mail & Newsgroups	З	З	З	З	З
Mozilla Thunderbird	+	+	+	+	+
Mulberry	+	+	+	+	+
Mutt	+	+	+	+	+
Netscape Messenger	З	З	З	З	З
Novell Evolution	+	+	+	+	+
Open-Xchange	?	?	?	?	?
Outlook Express	+	–	–	–	–
Pegasus Mail	+	–	–	–	–
Pine	+	+	+	+	+
i.Scribe/InScribe	+	–	+	–	–
SeaMonkey Mail & Newsgroups	+	+	+	+	+
Sylpheed	+	+	+	+	+
Sylpheed Claws	+	+	+	+	+
The Bat!	+	–	–	–	–
Windows Live Mail Desktop	+	–	–	–	–
Windows Mail	+	–	–	–	–
Zimbra	+	+	+	+	+

Далі розглянемо які протоколи підтримують ті, або інші поштові клієнти.

Таблиця 2.2 – Підтримка протоколів

Client	POP3			IMAP4	SMTP	NNTP	LDAP		IMSP	ACAP	Feeds				iCalendar
	скачування всіх повідомлень	вибірка шляхом фільтрації	вибірка користувачем				v2	v3			RSS 0.91	RSS 1.0	RSS 2.0	ATOM	
Citadel	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
Courier	+	+	+	–	+	–	+	+	–	–	–	–	–	–	–
Eudora	+	–	–	+	+	–	+	+	–	+	–	–	–	–	–
Foxmail	+	+	+	–	+	–	+-	?	?	?	+	+	+	–	–
Gnus	+	–	–	+	+	+	?	?	?	?	+	+	+	–	–
GroupWise	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
Insight	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
KMail	+	–	–	+	+	–	+	?	?	?	–	–	–	–	–
Lotus Notes	+	–	–	+	+	+	+	+	–	–	+	+	+	+	+
Opera Mail (M2)	+	–	–	+	+	+	–	–	?	?	+	+	+	+	–
Mail	+	–	–	+	+	–	?	?	?	?	–	–	–	–	–
MS Outlook	+	–	–	+	+	–	+		?	?	+	+	+	+	+
Mozilla Mail & Newsgroups	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
Mozilla Thunderbird	+	–	–	+	+	+	+-	?	–	–	+	+	+	+	+
Mulberry	+	–	–	+	+	+	+		+	+	–	–	–	–	+
Mutt	+	–	–	+	–	–	+-		?	?	–	–	–	–	–
Netscape Messenger	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Продовження таблиці 2.2

Client	POP3			IMAP4	SMTP	NNTP	LDAP		IMSP	ACAP	Feeds				iCalendar
	скачування всіх повідомлень	вибірка шляхом фільтрації	вибірка користувачем				v2	v3			RSS 0.91	RSS 1.0	RSS 2.0	ATOM	
Novell Evolution	+	–	–	+	+	+	?	?	?	?	–	–	–	–	+
Open-Xchange	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
Outlook Express	+	–	–	+	+	+	+		?	?	–	–	–	–	–
Pegasus Mail	+	+	+	+	+		+								
Pine	+	–	–	+	+	+	+	–	–	–	–	–	–	–	–
i.Scribe/InScribe	+	–	–	+	+	–	+	+	–	–	–	–	–	–	–
SeaMonkey Mail & Newsgroups	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
Sylpheed Claws	+	–	–	+	+	+	+		?	?	–	+	+	+	+
TheBat!	+	+	+	+	+	+	+	–	–	–	+	+	+	+	–
Windows Live Mail Desktop	+	–	–	+	+	+	+				+	+	+	+	–
Windows Mail	+	–	–	+	+	?	?	?	?	?	?	?	?		
Zimbra	+	–	–	+	?	?	?	?	?	?	?	?	?	?	?

Далі роглянемо, які протокли та механізми автентифікації використовують поштові клієнти.

Таблиця 2.3 – Підтримка автентификации

Клієнт	Звичайна		MSN (NTLM) browserlogin	Compuserve (RPA)	MD5 APOP	CRAM-HMAC			DIGEST-MD5	Апаратна PKCS #11	Біометрична	SMTP Auth	SSL/TLS Client Certificate
	LOGIN	PLAIN				MD5	SHA1	RIPEMD					
Citadel	?	?	?	?	?	?	?	?	?	?	?	?	?
Courier	+	+	–	–	+	+	–	–	–	–	–	+	–
Eudora	+	+	+	+	+	+	–	–	–	–	–	+	
Foxmail	?	?	?	?	?	?	?	?	?	?	?	?	?
Gnus	?	?	?	?	?	?	?	?	?	?	?	?	?
GroupWise	?	?	?	?	?	?	?	?	?	?	?	?	?
Insight	?	?	?	?	?	?	?	?	?	?	?	?	?
KMail	+	+	+	–	+	+	–	–	+	–	–	+	?
Lotus Notes	+		+	–	–	–				+	–	+	
Opera Mail (M2)	+	+	–	–	+	+	–	–	–	–	–	+	
Mail	?	+	+	?	?	+	?	?	?	?	?	+	
MS Office Outlook	+	–	+	–	–	–	–	–	–	–	–		
Mozilla Mail & Newsgroups	?	?	?	?	?	?	?	?	?	?	?	?	?
Mozilla Thunderbird	+	+	+	–	+	+	–	–	–	–	–	+	
Mulberry	+	+	+	–	–	+	–	–	+	–	–	+	+
Mutt													
Netscape Messenger	?	?	?	?	?	?	?	?	?	?	?	?	?
Novell Evolution	+	+	+	–	+	+	–	–	+	–	–	+	
Open-Xchange	?	?	?	?	?	?	?	?	?	?	?	?	?

Продовження таблиці 2.3

Клієнт	Звичайна		MSN (NTLM) browserlogin	Compuserve (RPA)	MD5 APOP	CRAM-HMAC			DIGEST-MD5	Апаратна PKCS #11	Біометрична	SMTP Auth	SSL/TLS Client Certificate
	LOGIN	PLAIN				MD5	SHA1	RIPEMD					
Outlook Express	+	–	+	–	–	–	–	–	–	–	–	–	+
Pegasus Mail	+		–	–	–	–	–	–	–	–	+	–	
Pine	+		–	–	+				–	–	+		
i.Scribe / InScribe	+	+	+	–	+	+	–	–	+	–	+	–	
SeaMonkey Mail & Newsgroups	?	?	?	?	?	?	?	?	?	?	?	?	?
Sylpheed Claws	+				+	–						+	
TheBat!	+	+	+	+	+	–	–	–	+	+	+		
Windows Live Mail Desktop	+	–	+	–	–	–	–	–	–	–	–	–	
Windows Mail	?	?	?	?	?	?	?	?	?	?	?	?	
Zimbra	?	?	?	?	?	?	?	?	?	?	?	?	

Таблиця 2.4 – Підтримка SSL і TLS

Клієнт	Secure POP3		Secure IMAP4		Secure SMTP		Secure NNTP		Secure LDAP		OCSP	CRL
	SSL	TLS	SSL	TLS	SSL	TLS	SSL	TLS	SSL	TLS		
Courier	–	–	?	?	–	–	?	?	–	–	–	–
Eudora	+	+	+	+	+	+	–	–	+	+		
Mail	+	?	+	+	+	?	–	–	+	?		

Продовження таблиці 2.4

Клієнт	Secure POP3		Secure IMAP4		Secure SMTP		Secure NNTP		Secure LDAP		OCSP	CRL
	SSL	TLS	SSL	TLS	SSL	TLS	SSL	TLS	SSL	TLS		
Gnus	?	?	+	+	?	?	?	?	?	?		
KMail	+	+	+	+	+	+	–	–				
Lotus Notes	+		+		+	+	+		+			+
Opera Mail (M2)	+	+-	+	+-	+	+-	+	–	–	–		
Mozilla Thunderbird	+	+	+	+	+	+	+		+		+	+
Mulberry												
Mutt												
Novell Evolution	+	+	+	+	+	+	–	–	?	?		
Microsoft Office Outlook	+	–	+	–	+	–						
Outlook Express	+	–	+	–	+	–						
Pegasus Mail	+		+		+				+			
Pine	+	+	+	+	+	+	+	+	–	–		
i.Scribe / InScribe	+	+	+	+	+	+	?	?	–	–	?	?
Sylpheed Claws	+	+	+	+	+	+	+	+	+	+		
TheBat!	+	+	+	+	+	+	–	–	–	–	–	–
Windows Live Mail Desktop	+	–	+	–	+	–	?	?	?	?		
Windows Mail	?	?	?	?	?	?	?	?	?	?		
Zimbra	?	?	?	?	?	?	?	?	?	?		

Таблиця 2.5 – Інформація про те, які властивості підтримує кожний клієнт

Клієнт	HTML e-mail	Підтримка UTF-UTF-8	Блокування картинок	Фільтрація Junk		Перегляд тредов	Підтримка PGP	Підтримка S/MIME			
				локально	на сервері			протокол	OCSP	CRL	підтримка сертифікатів токенов, смарт-карлов
Courier	+	–	–	+	+	–					
Citadel	+	+	–	+		–					
Eudora	+	–	+	+		+		+			
Gnus	+	+	+	+		+	+	+			
GroupWise	+	+	+	+	?	+	?	?	?	?	?
Lotus Notes	+	+	+	+	+	+	+	+		+	+
KMail	+	+	+	+		+	+	+			
Mail	+	+	+	+		+	+	+			
Mozilla Thunderbird	+	+	+	+	–	+	+	+			
Mulberry	+	+	+	–	+	+	+	+			
Mutt	+-	+	+	+		+	+	+			
Novell Evolution	+	+	+	+		+	+	+			
Opera Mail (M2)	+-	+	+	+		+	–	–			
Microsoft Office Outlook	+	+	+	+		+		+			
Outlook Express	+	+	+	–		+	+	+			
Pegasus Mail	+	+	+	+	+	+	+		–	–	–
Pine	+-	+	+	–		+		+			
i.Scribe / InScribe	+-	+	+	+	–	+	+	–	–	–	–
SeaMonkey Mail & Newsgroups	+	+	+	+		+	+	+			

Продовження таблиці 2.5

Клієнт	HTML e-mail	Підтримка UTF-UTF-8	Блокування картинок	Фільтрація Junk		Перегляд тредов	Підтримка PGP	Підтримка S/MIME			
				локально	на сервері			протокол	OCSP	CRL	підтримка сертифікатів токенов, смарт-кардов
Sylpheed	–	+	+	+		+					
Sylpheed Claws	+-	+	+	+		+	+	+			
The Bat!	+	+	+	+	+	+	+	+	+-	+-	+
Windows Live Mail Desktop	+	+	+	+		+	+	+			
Windows Mail	?	?	?	?	?	?	?	?	?	?	?
Zimbra	+	+	+	+		+	?	?			

Таблиця 2.6 – Зведення характеристик S/MIME поштових клієнтів Microsoft

Функція	Поштовий клієнт		
	Outlook 2003	Outlook Express 6.0	OWA
Поштовий протокол	MAPI/RPC, POP3, IMAP4, SMTP, HTTP	POP3, IMAP4, SMTP	HTTP
Постачальник сертифікатів	Exchange KMS, внутрішній CA, комерційний CA	Внутрішній CA, комерційний CA	Внутрішній CA, комерційний CA
Необхідний сервер обробки повідомлень Exchange	–	–	–

Продовження таблиці 2.6

Функція	Поштовий клієнт		
	Outlook 2003	Outlook Express 6.0	OWA
Шифрування пошти (захист конфіденційності)	–	–	–
Підписування пошти (перевірка дійсності й цілісності)	–	–	–
Безпечні повідомлення	–	–	–
Явні й неявні підписи	–	–	+–
Оформлення й відновлення сертифікатів користувачів	+–	+–	+–
Архівування й відновлення ключа шифрування	+–	+–	+–
Перевірка відкликаних сертифікатів включена за замовчуванням	–	–	–
Використання точок поширення CRLDP для перевірки відкликаних сертифікатів	–	–	–

Таким чином розлянувши існуючи методи захисту даних, які передаються по електронній пошті у сучасних поштовиках, перейдемо до обґрунтування мови програмування, на якій буде реалізована спроектована у даному магістерському проекті система передачі захищених повідомлень S/MIME, на основі технології PKI.

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 повідомлень електронної пошти.

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 S/MIME на основі PKI

Специфікація S/MIME визначає два типи MIME-конверта: один для цифрових підписів, іншої – для шифрованих повідомлень. Обидва типи базуються на синтаксисі криптографічних повідомлень стандарту PKCS#7 [6]. Якщо повідомлення повинне бути зашифроване, а шифртексту повинні бути привласнені деякі атрибути, використовуються вкладені конверти. Зовнішній і внутрішній конверт призначаються для захисту цифрових підписів, а проміжний конверт – для захисту шифртексту.

Крім забезпечення цілісності повідомлення під час передачі, S/MIME ідентифікує власника конкретного відкритого ключа за допомогою сертифіката X.509. Цифровий сертифікат засвідчує, що відкритий ключ дійсно належить тому, хто є суб'єктом сертифіката.

S/MIME v3 підтримує наступні важливі властивості, які відсутні в S/MIME v2:

- завірені цифровим підписом квитанції;
- мітки безпеки;
- списки розсилання;
- гнучке керування ключами.

Завірені цифровим підписом **квитанції** дозволяють відправникові повідомлення впевнитися в тому, що воно було отримано адресатами без змін. Одержувач повідомлення не може згенерувати валідну квитанцію доти, поки не перевірить підпис відправника на отриманому повідомленні.

Мітки безпеки дозволяють відправникові задавати керуючі вимоги до змісту повідомлення. Найчастіше мітка безпеки свідчить про включення в зміст повідомлення приватної або конфіденційної інформації.

Зазвичай відправник повідомлення шифрує його один раз за допомогою симетричного ключа. Потім відправник повинен зашифрувати симетричний ключ окремо, використовуючи відкритий ключ того адресата, якому він направляє своє повідомлення. Якщо кількість адресатів велика, виникає необхідність делегувати функції шифрування довіреному серверу. Такий сервер називають агентом списку розсилання (Mail List Agent – MLA). Якщо є агент MLA, то відправник може послати зашифроване повідомлення йому, а той, у свою чергу, після виконання відповідних операцій виконує розсилання повідомлення всім адресатам зі списку відправника. При цьому MLA не одержує доступ до ключа шифрування повідомлення й не розшифровує повідомлення, що пересилається.

Якщо адресати повідомлення територіально розподілені (наприклад, перебувають на різних континентах), то відправник посилає зашифроване повідомлення головному агентові MLA, головний агент пересилає його регіональним агентам MLA, і, нарешті, кожний регіональний агент доставляє зашифроване повідомлення локальним одержувачам [7]. У цьому випадку повідомлення бере участь у кожній міжконтинентальній комунікації тільки один раз. Правда, якщо в списки розсилання кожного регіонального MLA включені адреси всіх інших регіональних агентів, може відбуватися багаторазове пересилання повідомлення по колу. Для виявлення циклів аналізується атрибут "історія поширення списку розсилання", що втримується в зовнішньому конверті повідомлення. Коли агент MLA одержує повідомлення, то перевіряє історію поширення, щоб визначити, чи не оброблялося вже дане повідомлення. Якщо повідомлення оброблялося, воно просто віддаляється. Зовнішній конверт із цифровим підписом забезпечує цілісність історії поширення списку розсилання.

S/MIME v2 забезпечує тільки транспортування ключів шифрування повідомлень, а S/MIME v3 додатково підтримує механізми узгодження ключів і зовнішнього поширення симетричних ключів шифрування ключів.

Підтримка захищеної електронної пошти на основі PKI

Всі сервіси, пропонованими S/MIME (обох версій), покладаються на сертифікати й надійність зв'язування електронної адреси суб'єкта з його відкритим ключем. Адреса електронної пошти (часто називаєма адресою RFC 822 [8]) повинна бути представлена у доповненні сертифіката `Subject Alternative Name` (альтернативне ім'я суб'єкта). Якщо використовується друга версія S/MIME, то адреса електронної пошти вказується як відмітне ім'я суб'єкта (`emailAddress`).

Відправник зашифрованого повідомлення повинен бути впевнений, що відкритий ключ належить саме тому одержувачеві, якому він адресує своє повідомлення, у противному випадку доступ до змісту повідомлення може одержати сторонній. Аналогічно, поширюючи ключі шифрування симетричного ключа тільки членам списку розсилання відправника, агент MLA покладається на правильність зв'язування ідентичності одержувача і його відкритого ключа. Відправник і агент MLA порівнюють ідентифікаційну ознаку одержувача, зазначену в сертифікаті, з адресою електронної пошти одержувача, якому посилає своє повідомлення відправник.

Одержувач повідомлення, завіреного цифровим підписом, повинен бути впевнений, що відкритий ключ підпису, що необхідний для верифікації підпису на повідомленні, належить відправникові. Для цього він порівнює адресу електронної пошти, зазначену в полі `SENDER` (або `FROM`) отриманого повідомлення, з адресою, представленою в сертифікаті.

Аналогічні дії виконуються при перевірці квитанцій, завірених цифровим підписом, – для цього що перевіряє порівнює адресу електронної пошти із запиту на квитанцію з адресою електронної пошти в сертифікаті особи, що подписали квитанцію.

Засоби безпеки транспортного рівня

Протокол безпеки транспортного рівня Transport Layer Protocol (TLS) [9] забезпечує захист комунікацій між додатками, розробленими в архітектурі "клієнт-сервер", в основному між web-браузером і web-сервером. World Wide Web є найбільш популярний Інтернет-сервісом після електронної пошти. TLS найбільше часто застосовується для захисту web-контента, але може використовуватися з будь-яким протоколом прикладного рівня. Специфікація TLS базується на популярному протоколі Secure Socket Layer (SSL) [10], розробленому корпорацією Netscape. Ці протоколи створювалися для забезпечення автентифікації, цілісності й конфіденційності даних, якими обмінюються взаємодіючі один з одним додатки. Обидва протоколи мають дворівневу організацію: протокол установлення з'єднання (Handshake Protocol) і протокол передачі записів (Record Protocol).

Протокол установлення з'єднання дозволяє серверу й клієнтові виконати взаємну автентифікацію, погодити застосовуваний алгоритм шифрування й криптографічні параметри перед тим, як протокол прикладного рівня почне передачу даних. **Протокол передачі записів** забезпечує захист протоколів більше високого рівня, включаючи протокол установлення з'єднання. Протокол передачі записів залежить від надійності транспортного протоколу, такого як TCP.

Протоколи SSL і TLS незалежні від протоколів прикладного рівня, тому будь-який протокол прикладного рівня може прозоро оперувати поверх SSL і TLS. Протоколи SSL і TLS забезпечують три сервіси безпеки [11]:

– автентифікацію (підтвердження ідентичності з'єднання: протокол установлення з'єднання використовує сертифікати й верифікацію цифрових підписів для підтвердження ідентифікаційних ознак і повноважень вилученого додатка);

– цілісність (захист даних протоколу від несанкціонованої модифікації: протокол передачі записів використовує значення біта контролю цілісності для підтвердження того, що передані дані не змінювалися);

– конфіденційність (забезпечення таємності з'єднання: після узгодження симетричного ключа шифрування на основі протоколу встановлення з'єднання виконується шифрування даних, якими обмінюються сторони під час сеансу зв'язку).

Протоколи SSL і TLS здатні підтримувати взаємну автентифікацію сторін, але зазвичай на базі сертифіката виконується автентифікація сервера клієнтом, а потім клієнт автентифікується іншим способом, наприклад, уводячи по запиту сервера своє ім'я й пароль або номер своєї кредитної карти й дату закінчення її терміну дії.

Протокол установлення з'єднань

Протокол установлення з'єднань Handshake Protocol складається як би із трьох підпротоколів, які дозволяють виконати автентифікацію сторін, погодити алгоритми й параметри безпеки для протоколу передачі записів [11].

Handshake Protocol відповідає за організацію сеансу взаємодії між клієнтом і сервером для протоколу передачі записів, зокрема за узгодження характеристик сеансу:

– ідентифікатора сеансу (`Session identifier`), тобто довільної послідовності біт, обраної сервером для ідентифікації сеансу;

– сертифіката з'єднання (`Peer certificate`), що представляє собою сертифікат X.509; цей елемент може бути відсутнім, якщо автентифікація не потрібна;

– методу стиску (`Compression method`), тобто алгоритму стиску даних перед їхнім шифруванням;

– специфікатора шифрування (`Cipher spec`), що задає ідентифікатори алгоритму шифрування даних і алгоритму гешування, а також деякі криптографічні атрибути (наприклад, розмір геш-коду);

– головного секрету (`Master secret`), що представляє собою секретне значення, розділене між клієнтом і сервером;

– ознаки встановлення нового з'єднання (`Is resumable`) на основі поточного сеансу.

Ці характеристики потім використовуються для установки параметрів безпеки в протоколі передачі записів. Можливість установити кілька захищених з'єднань під час одного сеансу особливо важлива, коли клієнтові й серверу необхідно встановити кілька короткочасних з'єднань.

У результаті роботи протоколу Handshake Protocol формуються криптографічні параметри. Коли клієнт і сервер починають взаємодію, то погоджують версію протоколу, криптографічні алгоритми, автентифікують один одного (за бажанням) і використовують криптографію з відкритими ключами для поділу загального секрету. Робота протоколу Handshake Protocol виконується за шість кроків [12].

1-й крок. Обмін привітальними повідомленнями для узгодження алгоритмів і обміну випадковими числами.

2-й крок. Обмін криптографічними параметрами для узгодження початкового секрету.

3-й крок. Опціональний обмін сертифікатами й криптографічною інформацією для взаємної автентифікації клієнта й сервера.

4-й крок. Генерація головного секрету на основі початкового секрету й обмін випадковими числами.

5-й крок. Формування параметрів безпеки для протоколу передачі записів.

6-й крок. Оповіщення про розрахунок тих же самих параметрів безпеки й коректному завершенні з'єднання.

Коли клієнт бажає встановити нове з'єднання на основі даного сеансу або повторити цей сеанс, то відправляє своє привітальне повідомлення з ідентифікатором сеансу, що повинен бути відновлений. Якщо сервер усе ще зберігає в кеш-пам'яті параметри сеансу й бажає відновити з'єднання, то у

відповідь відправляє своє привітальне повідомлення з тим же самим ідентифікатором сеансу. У цей момент клієнт і сервер обмінюються повідомленнями про зміну параметрів шифрування й завершенні формування з'єднання.

При обміні сертифікатами й ключами передаються дані, необхідні для генерації початкового секрету. Якщо використовується алгоритм RSA, то клієнт генерує випадкове початкове число й шифрує його за допомогою відкритого RSA-ключа сервера. Якщо застосовується алгоритм Діффі-Хеллмана, клієнт і сервер обмінюються відкритими ключами, а потім як початковий секрет використовується результат обчислення ключа узгодження ключів по алгоритму Діффі-Хеллмана.

Головний секрет і симетричні ключі генеруються за допомогою псевдовипадкової функції PRF (PseudoRandom Function), отриманої на основі алгоритмів SHA-1 і MD5. Щоб гарантувати безпеку, застосовуються дві різні односпрямовані геш-функции. При першому застосуванні функції PRF генерується головний секрет з початкового секрету й випадкових чисел, отриманих на основі привітальних повідомлень клієнта й сервера. У результаті повторного застосування функції PRF до тих же самим вихідним даним одержують два симетричних ключі, два значення початкового вектора й два значення MAC (коду автентифікації повідомлення) секрету. При поновленні сеансу використовується вже відоме для даного сеансу значення головного секрету, але генеруються нові випадкові числа на основі нових привітальних повідомлень клієнта й сервера, тому в результаті формується новий ключовий матеріал.

Протокол передачі записів використовує один симетричний ключ, одне значення початкового вектора й один MAC секрет для захисту трафіку "клієнт-сервер", а також інші значення перерахованих параметрів для захисту трафіку "сервер-клієнт", – у цілому, для коректної роботи протоколу передачі записів потрібно шість секретних чисел.

Протокол передачі записів

Протокол передачі записів Record Protocol складається з декількох підрівнів. Обробка даних протоколу прикладного рівня полягає в їхній розбивці на керовані блоки, стиску, постачанні імітовставки, шифруванні й наступній передачі результату. Отримані дані обробляються у зворотному порядку: розшифровуються, перевіряються на цілісність, піддаються декомпресії, складанню, а потім доставляються додатку [6].

На підрівні фрагментації інформація розбивається на записи, довжина кожного запису не перевищує `16384` байтів. Допускається агрегування декількох однотипних повідомлень в один запис, а також розбивка одного повідомлення протоколу прикладного рівня на кілька записів. На підрівні стиску виконується стиск або декомпресія всіх фрагментів. Очевидно, що при компресії не повинне бути втрати даних. Потім обчислюється імітовставка, тобто перевіряється цілісність стислого запису, а отримане значення й стислий запис шифруються. Перевірка цілісності здійснюється за допомогою коду автентифікації повідомлення (MAC) або коду автентифікації, отриманого на основі геш-коду повідомлення (HMAC).

Після прийняття даних текст розшифровується, для перевірки його цілісності повторно обчислюється MAC, причому обчислення виконуються з використанням порядкового номера запису з метою виявлення загублених, зайвих або повторно стислих записів.

Підтримка безпеки транспортного рівня на основі PKI

Сертифікати є центральним компонентом всіх сервісів автентифікації й керування ключами, пропонованих як TLS, так і SSL. Ці сервіси покладаються на зв'язування ідентичності суб'єкта з його відкритим ключем. Для ідентифікації web-серверів рекомендується використовувати DNS-імена (типу www.alpha.com) і вказувати їх у доповненні сертифікатів `Subject Alternative Name` (параметр `dNSName`). Якщо DNS-ім'я не представлене в сертифікаті, то для ідентифікації використовується відмітне ім'я суб'єкта.

Зазвичай при взаємодії клієнта й сервера сервер пред'являє сертифікат, а клієнт – ні, у результаті чого відбувається однобічна автентифікація сервера клієнтом, що зберігає свою анонімність. Сервер може зажадати від клієнта автентифікації, запитуючи сертифікат по протоколі Handshake Protocol. У цьому випадку клієнт повинен мати сертифікат і надати його серверу. Зазвичай клієнт пред'являє сертифікат ключа підпису. У процесі верифікації завіреного цифровим підписом повідомлення, відправленого по протоколу Handshake Protocol, з'ясовується, чи здатний клієнт згенерувати підпис за допомогою свого секретного ключа, що відповідає відкритому ключу підпису, що втримується в сертифікаті.

Сертифікати, використовувані для підтримки безпеки транспортного рівня, також повинні містити доповнення `Key Usage`, що відбиває відповідне призначення відкритого ключа, що втримується в сертифікаті:

– цифровий підпис, якщо необхідно виконувати верифікацію підписів;

– шифрування ключів, щоб забезпечити RSA-шифрування;

– узгодження ключів, якщо необхідно підтримку узгодження ключів методом Діффі-Хеллмана.

Клієнт повинен бути впевнений, що відкритий ключ належить серверу. Якщо використовується некоректний відкритий ключ, то стороння особа може одержати початковий секрет і можливість згенерувати головний секрет і всі інші секретні параметри, що обчислюються з його допомогою. Сертифікат підтверджує приналежність відкритого ключа серверу.

При автентифікації клієнта сервер покладається на приналежність відкритого ключа клієнтові й ухвалює рішення щодо можливості доступу. Якщо використовується некоректний відкритий ключ, то доступ до сервера може одержати сторонню особу, а не та сторона, який доступ необхідний. Сертифікат забезпечує необхідне зв'язування відкритого ключа з ідентичністю клієнта.

Засоби безпеки IP-рівня

Сукупність механізмів IPsec забезпечує основу для захисту мережного трафіку на IP-рівні, безпеку IP-пакетів, захищеного взаємодії мобільних систем з корпоративною мережею, реалізації віртуальних приватних мереж (Virtual Private Networks – VPN) і т.п. Сімейство специфікацій IPsec представлено серією з 10 документів, розроблених робочою групою IP Security Protocol організації IETF і утримуючі відомості про архітектуру IPsec [14], формуванні контекстів безпеки, керуванні ключами й базовими протоколами. Ядро IPsec становлять три протоколи: протокол автентифікуючого заголовку (Authentication Header, AH) [14], протокол інкапсулюючий захист вмісту (Encapsulating Security Payload, ESP) [14] і протокол обміну ключами в Інтернеті (Internet Key Exchange, IKE) [14]. Функції по підтримці захищеного каналу передачі даних по мережах IP розподіляються між цими протоколами в такий спосіб:

– протокол AH забезпечує цілісність IP-пакетів, автентифікацію джерела даних, а також захист від відтворення раніше переданих IP-пакетів;

– протокол ESP підтримує конфіденційність, автентифікацію й цілісність IP-пакетів, а також частковий захист від аналізу трафіку;

– протокол IKE дозволяє взаємодіючим сторонам автоматично генерувати й безпечно розподіляти симетричні секретні ключі.

Контексти безпеки

Контексти безпеки (Security Associations) утворюють основу криптографічних сервісів безпеки на базі протоколів IPsec. Для захисту двостороннього зв'язку між вузлами мережі необхідні два контексти безпеки: один – для вхідних потоків, інший – для вихідних. Контексти безпеки містять інформацію про IP-адреси, тип захисного протоколу (AH або ESP), криптографічних алгоритмах, ключах для автентифікації й шифрування й періоді їхньої дії.

Контекст безпеки унікально ідентифікується трьома елементами:

– індексом параметрів безпеки (`Security Parameters Index – SPI`);

– цільовою IP-адресою;

– ідентифікатором захисного протоколу.

Отже, індекс SPI – це ідентифікатор контексту безпеки, що вказується в протоколі AH або ESP. Цільова IP-адреса ідентифікує з'єднання IPsec, він може бути індивідуальною або груповою адресою або діапазоном адрес. У цей час обмін ключами IKE реалізований тільки для індивідуальних адрес, а для групових адрес або діапазонів розподіл ключів виконується вручну.

Таблиця 3.1 – Режими, використовувані для різних типів з'єднань

	Хост	Маршрутизатор або міжмережний екран
Хост	Транспортний режим або тунельний режим	Тунельний режим
Маршрутизатор або міжмережний екран	Тунельний режим	Тунельний режим

Протоколи забезпечення автентичності й конфіденційності можуть застосовуватися у двох режимах: транспортному й тунельному. Зазвичай транспортний режим використовується хостами, при цьому захищається тільки вміст IP-пакетів і опціонально – деякі поля заголовків. У тунельному режимі захищається весь пакет: він інкапсулюється в інший IP-пакет, при цьому відбувається як би "обгортання", або висновок у конверт, переданої порції даних [6]. Тунельний режим зазвичай реалізують на спеціально виділених захисних шлюзах, у ролі яких можуть виступати маршрутизатори або міжмережні екрани (рисунок 3.1).

Протокол прикладного рівня
Транспортний протокол (TCP, UDP)
Захистний протокол (AH, ESP)
Інтернет-протокол (IPv4, IPv6)
Канальний протокол (Ethernet та ін.)

Транспортний режим

Протокол прикладного рівня
Транспортний протокол (TCP, UDP)
Інтернет-протокол (IPv4, IPv6)
Захистний протокол (AH, ESP)
Інтернет-протокол (IPv4, IPv6)
Канальний протокол (Ethernet та ін.)

Тунельний режим

Рисунок 3.1 – Стеки протоколів у різних режимах

У транспортному режимі заголовок протоколу (AH або ESP) розташовується в стеці протоколів після заголовка вихідного IP-пакету й перед заголовками протоколів більш високого рівня [7]. У тунельному режимі заголовок протоколу (AH або ESP) розташовується в стеці протоколів між двома заголовками: після заголовка зовнішнього IP-пакета й перед заголовком внутрішнього вихідного IP-пакета (рисунок 3.1).

Протокол автентифікуючого заголовка AH

Протокол автентифікуючого заголовка AH забезпечує:

– цілісність IP-пакетів, даних протоколів більше високого рівня й певних полів IP-заголовків;

– автентифікацію джерела даних (на основі IP-адреси вузла мережі або ім'я кінцевого користувача);

– захист від помилкового відтворення раніше переданих IP-пакетів.

Контроль цілісності базується на перевірці коду автентифікації гешированного повідомлення Hashed Message Authentication Code (HMAC), що обчислюється за допомогою геш-функції MD5 або SHA-1 із секретним симетричним ключем, що відомий обом взаємодіючим сторонам.

Рисунок 3.2 ілюструє типові поля даних протоколу AH. Протокол містить п'ять полів: `Next Header`, `Length`, `SPI`, `Sequence Number` і `Authentication Data`.

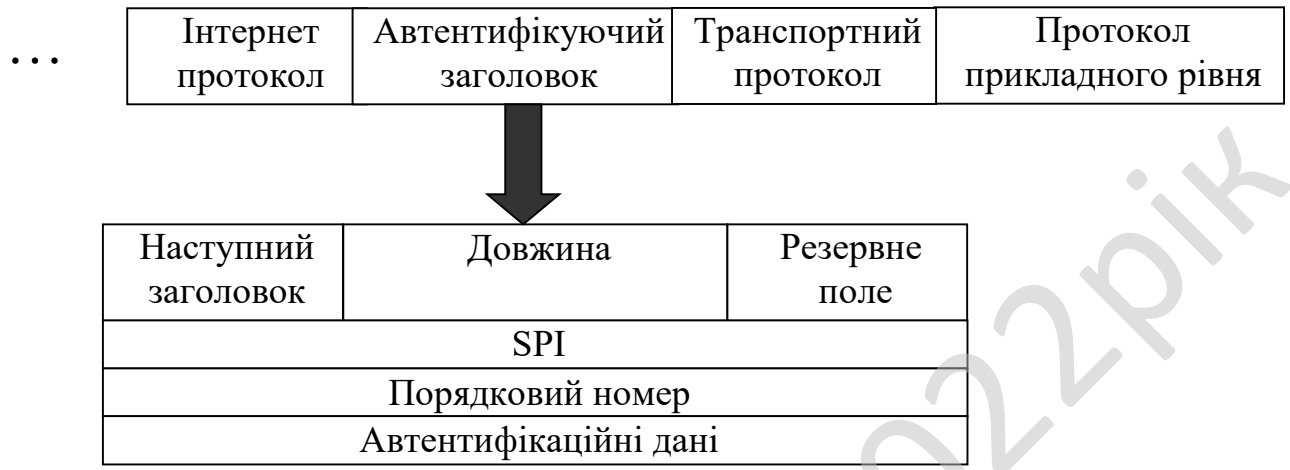

Рисунок 3.2 – Поля даних протоколу AH

Поле `Next Header` (наступний заголовок) указує, який протокол більш високого рівня інкапсулюється за допомогою AH. У тунельному режимі це поле зазвичай містить IP v4 або IP v6, а в транспортному режимі – TCP, UDP або ICMP.

Поле `Length` (довжина) задає розмір заголовка протоколу AH. Розмір залежить від типу використовуваної геш-функції, значення `HMAC` утримується в єдиному полі змінної довжини.

Поле `SPI` (індекс параметрів безпеки) містить 32-розрядне довільне значення, що ідентифікує контекст безпеки.

Поле `Sequence Number` (порядковий номер) використовується для завдання значення лічильника IP-пакетів (32-розрядного монотонно зростаючого) і захисту від відтворення пакетів. Відправник пакета повинен задавати це значення, а одержувач пакета може або обробляти його, або ігнорувати.

Поле `Authentication Data` (автентифікаціонные дані) містить значення `HMAC` для даного IP-пакета. Це поле має змінну довжину, що повинна бути кратна 32 розрядам.

При передачі пакета його порядковий номер, що вказується в поле Sequence Number, збільшується, а потім поля IP-заголовка й протоколу більш високого рівня гешируются для створення HMAC на основі загального секретного симетричного ключа. Після одержання IP-пакета одержувачем виконується та ж сама послідовність операцій. Якщо обчислене їм значення HMAC не відповідає значенню, отриманому по протоколу AH, то пакет не приймається. Крім того, якщо контекст безпеки містить інформацію про застосування засобу захисту від відтворення пакетів, то значення поля Sequence Number зменшується на одиницю, тобто відновлюється колишнє значення лічильника IP-пакетів.

Протокол інкапсулюючий захист вмісту ESP

Протокол інкапсулюючий захист вмісту ESP підтримує конфіденційність, автентифікацію й цілісність IP-пакетів. Конфіденційність забезпечується шляхом шифрування вмісту IP-пакетів, а також частини заголовка й трейлера (хвостової частини) протоколу ESP; надійність шифрування залежить, насамперед, від використовуваного алгоритму шифрування. Автентифікація джерела даних і захист цілісності здійснюється на основі HMAC (як і в протоколі AH). Хоча сервіси конфіденційності й автентифікації (який включає цілісність) є опціональними, у кожному контексті безпеки повинен бути заданий, принаймні, один сервіс безпеки.

Як алгоритми шифрування в протоколі ESP використовуються алгоритми DES і Triple-DES, для обчислення HMAC застосовується геш-функція типу MD5 або SHA-1. Рисунок 3.3 ілюструє типові поля даних протоколу ESP. Заголовок ESP містить два поля: SPI і Sequence Number, їхній синтаксис і семантика збігається з однойменними полями протоколу AH. Трейлер ESP складається із чотирьох полів: Padding, Pad Length, Next Header і Authentication Data.

Поле Padding (заповнювач) використовується для того, щоб розмір шифруемых даних був кратний розміру криптографічного блоку.

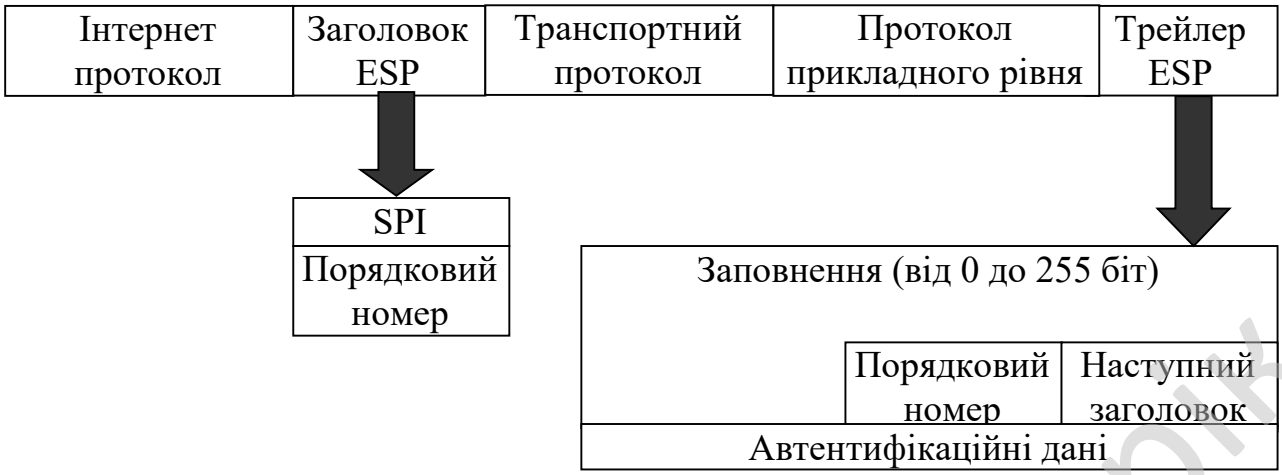

Рисунок 3.3 – Поля даних протоколу ESP

Поле `Pad Length` (довжина заповнювача) характеризує розмір заповнювача й залежить від використовуваного алгоритму шифрування й заданого рівня конфіденційності IP-трафіку.

Поле `Next Header` (наступний заголовок) містить інформацію про те, який протокол більше високого рівня інкапсулюється за допомогою ESP. У тунельному режимі це поле зазвичай містить IP v4 або IP v6, а в транспортному режимі – TCP, UDP або ICMP.

Поле `Authentication Data` (автентифікаційні дані) містить значення `HMAC` для даного IP-пакета. Це поле має змінну довжину, що повинна бути кратна 32 розрядам. Якщо автентифікація джерела даних або захист цілісності не потрібна, то це поле відсутнє або має нульову довжину.

При передачі пакета його порядковий номер, що вказується в поле `Sequence Number`, збільшується, а потім поля заголовка ESP, протоколу більш високого рівня й трейлера ESP гешуються для створення HMAC на основі загального секретного симетричного ключа. Потім поля протоколу більш високого рівня й трейлер ESP (за винятком автентифікаціоних даних) шифруються; якщо необхідно початковий вектор, то він випереджає шифртекст. Після одержання IP-пакета одержувачем виконується розшифрування й розрахунок того ж самого значення HMAC. Якщо обчислене їм значення `HMAC` не

відповідає значенню, отриманому в трейлері ESP, то пакет не приймається. Крім того, якщо контекст безпеки містить інформацію про використання засобу захисту від відтворення пакетів, то значення поля `Sequence Number` зменшується на одиницю, тобто відновлюється колишнє значення лічильника IP-пакетів.

Протокол обміну ключами IKE

Широке використання IPsec вимагає масштабованого, автоматизованого керування контекстами безпеки. Формування контекстів безпеки по запиту й використання засобів захисту від відтворення пакетів у протоколах AH і ESP неможливо без роботи протоколу обміну ключами в Інтернеті IKE [14]. Протокол IKE розроблений на основі протоколу керування ключами й контекстами безпеки Інтернету – Internet Security Associations and Key Management Protocol (ISAKMP) [14] і протоколу обчислення ключів – OAKLEY Key Determination Protocol (OAKLEY) [14]. Протокол ISAKMP забезпечує незалежну від криптографічного механізму автентифікацію й задає структуру обміну ключами. Протокол IKE базується на функціональності протоколу ISAKMP, а для формування симетричного ключа використовує можливості протоколу OAKLEY. Як алгоритм узгодження ключів застосовується алгоритм Діффі-Хеллмана.

Робота протоколу IKE виконується за два етапи. На першому етапі встановлюється автенифікований і шифруємий канал зв'язку. Це вимагає формування двох контекстів безпеки – по одному для зв'язку в одному напрямку. Для автентифікації сторін використовуються сертифікати відкритих ключів підпису (як алгоритм цифрового підпису застосовується DSA). На другому етапі формуються контексти безпеки для AH і ESP.

Підтримка безпеки IP-рівня на основі PKI

Сертифікати служать головними компонентами автентифікації на основі IKE. Коли ідентифікується хост або захисний шлюз, найбільш кращою формою ідентифікаційних даних є DNS-імена або IP-адреси, які вказуються в доповненні сертифіката `Subject Alternative Name` (параметри `dNSName` і `iPAddress` відповідно). При ідентифікації користувача рекомендується використовувати адресу електронної пошти або звичайне ім'я. Адреса електронної пошти втримується в

доповненні сертифіката `Subject Alternative Name`, а звичайне ім'я входить до складу відмітного ім'я суб'єкта. Неможливість зв'язати ідентичність суб'єкта з його відкритим ключем робить автентифікацію неможливою. Неправильна автентифікація на основі протоколу IKE може привести до помилкового зв'язування ключа й реалізації IPsec, у результаті неавторизований користувач (або навіть група комп'ютерів) може одержати доступ, наприклад, до віртуальної приватної мережі.

Сертифікати, призначені для захисту трафіку Інтернету, повинні включати доповнення `Key Usage`, що відбиває відповідне призначення відкритого ключа, що втримується в сертифікаті (наприклад, цифровий підпис, якщо необхідно виконувати верифікацію підписів). Крім того, у доповненні сертифіката `Extended Key Usage` повинні явно вказуватися ті додатки, для підтримки яких призначений даний відкритий ключ, зокрема додаток IPsec.

Отже, сертифікати є базовим компонентом сервісів безпеки, надаваних S/MIME, TLS і IPsec.

S/MIME за допомогою сертифікатів ідентифікує користувачів. Сервіси автентифікації, цілісності, невідказуємості й конфіденційності захищеної електронної пошти залежать від сертифікатів. Сертифікати підтримують шифрування й завірення цифровим підписом поштових повідомлень.

TLS за допомогою сертифікатів ідентифікує користувачів і сервери. Від сертифікатів залежать сервіси автентифікації, цілісності й конфіденційності. Сертифікати підтримують шифрування потоку, автентифікацію потоку й цілісність потоку.

IPsec за допомогою сертифікатів ідентифікує користувачів, хости й шлюзи безпеки. Від сертифікатів залежать сервіси автентифікації. На базі сертифікатів виконується взаємна автентифікація взаємодіючих сторін при обміні відкритими ключами Діффі-Хеллмана, які, у свою чергу, використовуються для безпечного розподілу симетричних секретних ключів. Секретні ключі забезпечують підтримку автентифікації, цілісності й конфіденційності IP-пакетів.

Типові сценарії використання PKI

Повнофункціональна PKI – це ідеальне подання можливої інфраструктури, поки невідомі PKI-продукти, що реалізують всі перераховані в таблиці 3.2 функції. Сучасні PKI зазвичай призначені для рішення певної задачі або ряду задач. Конкретні реалізації PKI являють собою деякі підмножини повнофункціональної архітектури.

Розглянемо чотири розповсюджених на сьогоднішній день сценарія використання PKI [4]. У таблиці 3.3 представлений Інтернет-PKI, що підтримує звичайну електронну пошту (між знайомими) і навігацію в World Wide Web за допомогою SSL-сервера автентифікації. Такий сценарій вимагає наявності УЦ для випуску сертифікатів відкритих ключів і підтримки основних сервісів автентифікації, цілісності й конфіденційності. У цьому сценарії не передбачене використання репозитория (сертифікати пересилаються по протоколі зв'язку), не виконується перевірка статусу одержувача електронної пошти (або навіть сертифіката сервера) і керування життєвим циклом ключів і сертифікатів, відсутнє клієнтське програмне забезпечення (як окремий модуль, викликуваний за допомогою браузера), не потрібно ні крос-сертифікація, ні додаткові сервіси, що базуються на PKI.

Таблиця 3.2 – Повнофункціональна PKI

УЦ	**Репозиторій**	**Анулювання сертифікатів**
Резервне зберігання ключів	**Відновлення ключів**	**Автоматичне відновлення ключів**
Керування історіями ключів	**Крос-сертифікація**	**Клієнтське ПЗ**
Автентифікація	**Цілісність**	**Конфіденційність**
Захищене датування	**Нотаризація**	**Невідказуємість**
Захищений архів даних	**Розробка повноважень/політики**	**Перевірка повноважень/політики**

Таблиця 3.3 – Інтернет-PKI

УЦ	Репозиторій	Анулювання сертифікатів
Резервне зберігання ключів	Відновлення ключів	Автоматичне відновлення ключів
Керування історіями ключів	Крос-сертифікація	Клієнтське ПЗ
Автентифікація	**Цілісність**	**Конфіденційність**
Захищене датування	Нотаризація	Невідказуємість
Захищений архів даних	Розробка повноважень/політики	Перевірка повноважень/політики

Таблиця 3.4 ілюструє функції PKI у сценарії, коли для доступу до корпоративної мережі ззовні використовується браузер і виконується SSL-автентифікація клієнтів. У цьому сценарії повинна підтримуватися перевірка статусу сертифіката, повноважень і політики. Через обмежені можливості браузера неможливо реалізувати керування життєвим циклом ключів і сертифікатів, крос-сертифікацію й інші сервіси, що базуються на PKI.

У таблиці 3.5 представлений набір функцій PKI для сценарію захищеної корпоративної електронної пошти. У цьому сценарії може знадобитися керування життєвим циклом ключів і сертифікатів і вбудоване клієнтське програмне забезпечення, тому що стандартні пакети електронної пошти не завжди підтримують безпеку, засновану на PKI. У цьому випадку не потрібні додаткові сервіси, що базуються на PKI, і крос-сертифікація.

Нарешті, у сценарії підтримки міжкорпоративних транзакцій з використанням цифрових підписів можуть знадобитися багато можливостей повнофункціональної PKI, зокрема сильна автентифікація й авторизація, перевірка статусу сертифікатів, розробка й перевірка повноважень і політики,

сервіс невідказуємості (підтримка множинних пар ключів, зберігання прийнятих електронних документів із цифровим підписом і т.д.). Якщо корпорації мають свої власні PKI, то необхідно крос-сертифікацію. У даному сценарії можна обійтися без архівування даних, датування й нотаризації.

Таблиця 3.4 – Екстранет-безпека (через SSL-автентифікацію клієнтів)

УЦ	Репозиторій	**Анулювання сертифікатів**
Резервне зберігання ключів	Відновлення ключів	Автоматичне відновлення ключів
Керування історіями ключів	Крос-сертифікація	Клієнтське ПЗ
Автентифікація	**Цілісність**	**Конфіденційність**
Захищене датування	Нотаризація	Невідказуємість
Захищений архів даних	**Розробка повноважень/політики**	**Перевірка повноважень/політики**

Таблиця 3.5 – Захищена корпоративна електронна пошта

УЦ	**Репозиторій**	**Анулювання сертифікатів**
Резервне зберігання ключів	**Відновлення ключів**	**Автоматичне відновлення ключів**
Керування історіями ключів	Крос-сертифікація	**Клієнтське ПЗ**
Автентифікація	**Цілісність**	**Конфіденційність**
Захищене датування	Нотаризація	Невідказуємість
Захищений архів даних	Розробка повноважень/політики	Перевірка повноважень/політики

Таблиця 3.6 – Міжкорпоративні транзакції із цифровим підписом

УЦ	**Репозиторій**	**Анулювання сертифікатів**
Резервне зберігання ключів	**Відновлення ключів**	**Автоматичне відновлення ключів**
Керування історіями ключів	**Крос-сертифікація**	**Клієнтське ПЗ**
Автентифікація	**Цілісність**	**Конфіденційність**
Захищене датування	Нотаризація	**Невідказуємість**
Захищений архів даних	**Розробка повноважень/політики**	**Перевірка повноважень/політики**

Таблиці 3.3, 3.4, 3.5 і 3.6 підтверджують, що PKI, що реалізують приватні сценарії, є підмножинами повнофункціональної PKI. Технологія PKI продовжує розвиватися, але вже зараз ясно, що багато постачальників програмного й апаратного забезпечення PKI будуть орієнтуватися на реалізацію повнофункціональних систем, а не на PKI-продукти вузького призначення. Очевидно, що в багатьох випадках простіше й економічно більш ефективно адаптувати повнофункціональний продукт для рішення специфічної проблеми, чим розробляти й підтримувати кілька окремих продуктів, кожний з яких призначений для рішення однієї або двох специфічних проблем. У багатьох середовищах PKI відбудеться неминучий перехід від приватних рішень приватних проблем до повнофункціонального PKI, що пропонує універсальне рішення проблем безпеки для широкого кола додатків.

3.2 Розробка структурної схеми

На рисунку 3.4 зображено структурну схему інфраструктури відкритих ключів, та місце S/MIME у ній. Розглянемо компоненти структурної схеми.

Почнемо розгляд з точену. Замість використання доступу по паролю до корпоративних ресурсів, для входу в домен і при використанні корпоративної пошти можна використовувати апаратну автентифікацію й захист електронної переписки в мережах, побудованих на базі Windows 10/11. У пропонованому рішенні використовуються вбудовані інструменти безпеки Windows і електронні ідентифікатори **токен** як носії ключової інформації.

Що забезпечується при використанні цього продукту? Авторизація користувачів (вхід у домен при підключенні токена й блокування сесії після його від'єднання). При роботі з поштою – електронний цифровий підпис поштових повідомлень і їхнє шифрування. Доступ до корпоративних ресурсів по пред'явленні токена й, звичайно, надійне зберігання й використання сертифікатів. Частина з перерахованих задач може бути використана й на домашньому комп'ютері, наприклад, при роботі з поштою, а також для віддаленого підключення до корпоративних ресурсів. Давайте розберемо, що і як необхідно виконати для реалізації перерахованих задач.

Токен як додатковий пристрій не може бути пізнаний операційною системою, оскільки він не є стандартним устаткуванням. Неможлива й установка токена за допомогою inf-файлу, оскільки для коректної установки потрібний вимір деяких параметрів для автоматичної конфігурації драйвера. Тому доводиться для установки використовувати спеціальне програмне забезпечення. І от що ще варто пам'ятати. Не підключайте токен до комп'ютера до того, як ви встановите драйвер. Якщо все-таки підключення буде виконано раніше, припиніть роботу стандартного майстра установки USB-пристроїв, витягніть ключ із порту й установіть драйвер. Крім драйверів у процесі інсталяції на диск будуть скопійовані й утиліти для роботи з токеном (утиліта адміністрування й браузер сертифікатів). Кількість одночасно використовуваних токенів залежить від операційної системи й кількості USB-портів.

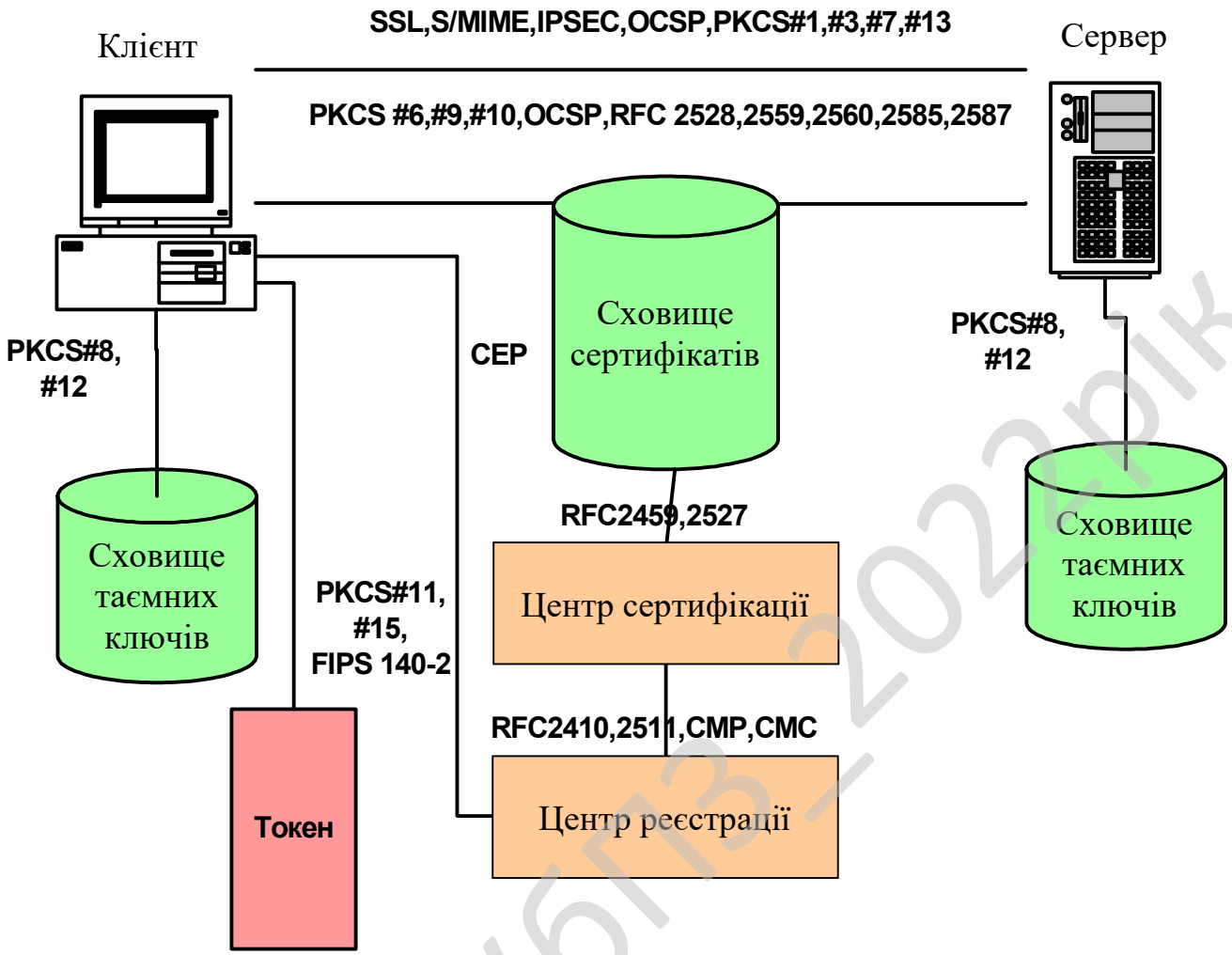

Рисунок 3.4 – Структурна схема PKI

Центр сертифікації (Удостовірюючий центр) (Certification authority, CA) – це організація, що випускає сертифікати ключів електронного цифрового підпису.

Центр сертифікації – це компонента глобальної служби каталогів, відповідальназа керування криптографічними ключами користувачів. Відкриті ключі й інша інформація про користувачів зберігається центрами сертифікації у вигляді цифрових сертифікатів, що мають наступну структуру:

– серійний номер сертифіката;

– об'єктний ідентифікатор алгоритму електронного підпису;

– ім'я центра, що засвідчує;

– строк придатності;

– ім'я власника сертифіката (ім'я користувача, якому належить сертифікат);

– відкриті ключі власника сертифіката (ключів може бути трохи);

– об'єктні ідентифікатори алгоритмів, асоційованих з відкритими ключами власника сертифіката; електронний підпис, згенерований з використанням секретного ключа центра, що засвідчує (підписується результат гешування всієї інформації, що зберігається в сертифікаті).

Центр реєстрації – опціональна компонента інфраструктури, призначена для реєстрації кінцевих користувачів і забезпечення їхньої взаємодії із центром сертифікації.

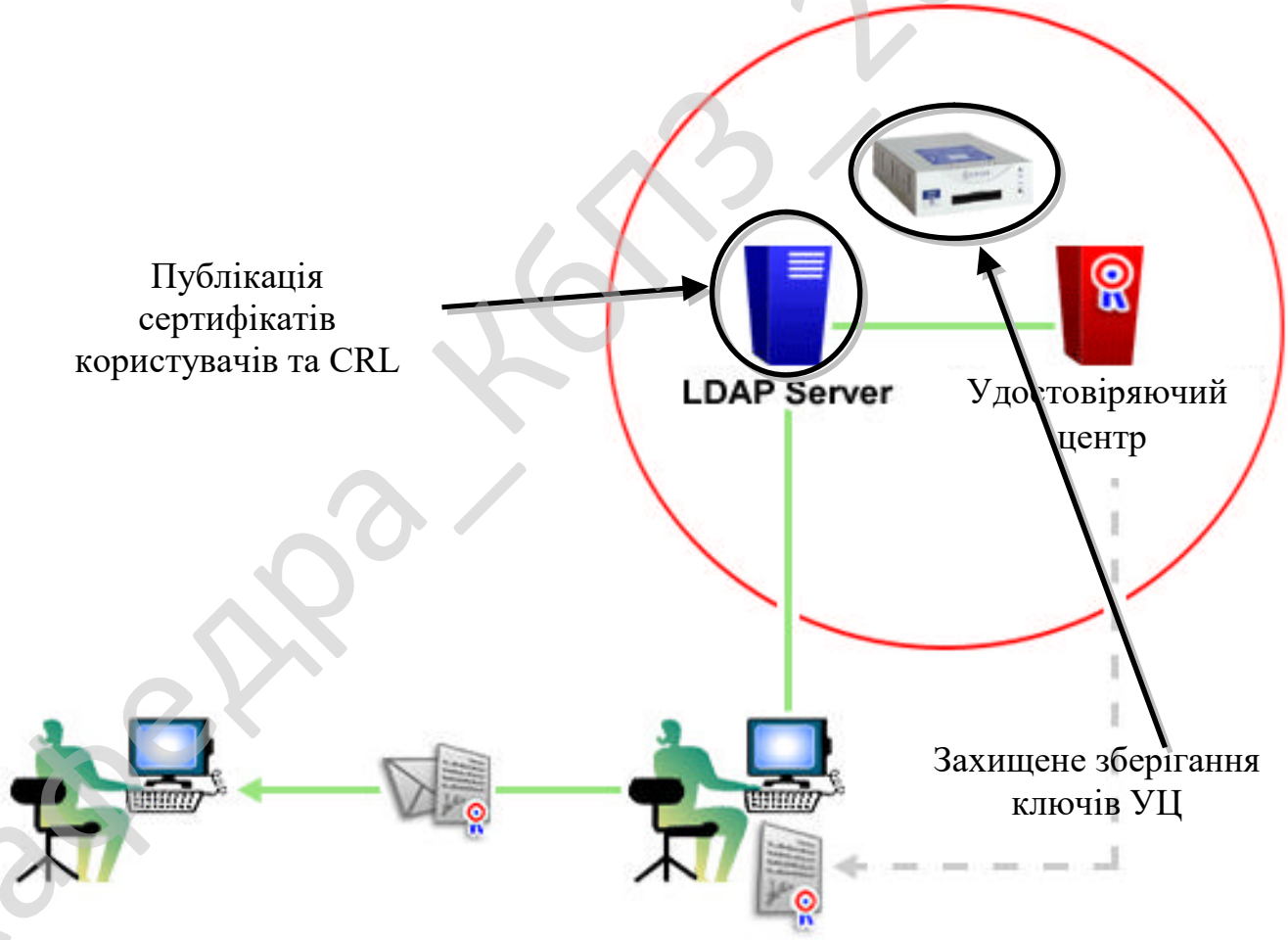

Рисунок 3.5 – Структурна схема захищеного обміну повідомленнями S/MIME

На рисунку 3.5 зображено структурну схему захищеного обміну повідомленями S/MIME.

CRL – список відозваних сертифікатів. Оснащення «Центру сертифікації» можна використовувати для відкликання сертифіката, для адміністрування публікації списків відкликання сертифікатів (CRL) і для завдання точок поширення списків CRL, які опубліковані в кожному сертифікаті, виданому центром сертифікації (ЦС).

Відкликання сертифікатів

Щоб допомогти в досягненні цілісності інфраструктури відкритого ключа (PKI) організації, адміністратор ЦС відзиває сертифікат, якщо суб'єкт сертифіката залишає організацію, порушений закритий ключ сертифіката або існують інші причини, по яких сертифікат більше не може вважатися дійсним. При відкликанні сертифіката ЦС він додається в список відкликання сертифікатів (CRL) цього ЦС. Це може відбуватися зі створенням нового списку CRL або з використанням різницевого списку CRL, що є невеликим списком сертифікатів, відкликаних з моменту останнього заповнення списку CRL.

Розклад публікації списку відкликання сертифікатів (CRL)

Одна з можливостей служб сертифікації складається в автоматичній публікації оновленого списку CRL по закінченні певного періоду часу, заданого адміністратором ЦС. Цей інтервал часу називається періодом публікації CRL. Після початкової установки ЦС період публікації встановлюється рівним одному тижню (відповідно до часу на локальному комп'ютері, починаючи з дати початкової установки ЦС). Періоди публікації списків CRL і різницевого CRL можуть задаватися незалежно.

Адміністратор ЦС повинен розуміти, що існує різниця між періодом публікації списків CRL і терміном дії списків CRL. Термін дії списків CRL – це період часу, протягом якого вони є офіційними списками для сертифікат, що перевіряє. Поки, що перевіряє сертифікат, зберігає дійсний список CRL у локальному кеші, у нього немає необхідності запитувати інший список CRL у ЦС, що них публікує.

LDAP (Lightweight Directory Access Protocol – «полегшений протокол доступу до каталогів») – це мережний протокол для доступу до служби каталогів X.500, розроблений IETF як полегшений варіант розробленого ITU-T протоколу DAP. LDAP – відносно простий протокол, що використовує TCP/IP і дозволяє робити операції автентифікації (bind), пошуку (search) і порівняння (compare), а також операції додавання, зміни або видалення записів. Звичайно LDAP-Сервер приймає вхідні з'єднання на порт 389 по протоколах TCP або UDP. Для LDAP-сеансів, інкапсульованих в SSL, звичайно використовується порт 636.

Усякий запис у каталозі LDAP складається з одного або декількох атрибутів і має унікальне ім'я (DN -Distinguished Name). Унікальне ім'я може виглядати, наприклад, у такий спосіб: «cn=Іван Петров, ou=Співробітники, dc=example, dc=com». Унікальне ім'я складається з одного або декількох відносних унікальних імен (RDN -Relative Distinguished Name), розділених комою. Відносне унікальне ім'я має вигляд Ім'я Атрибута = значення. На одному рівні каталогу не може існувати двох записів з однаковими відносними унікальними іменами. У силу такої структури унікального імені запису в каталозі LDAP можна легко представити у вигляді дерева.

Запис може складатися тільки з тих атрибутів, які визначені в описі класу запису (object class), які, у свою чергу, об'єднані в схеми (schema). У схемі визначено, які атрибути є для даного класу обов'язковими, а які – необов'язковими. Також схема визначає тип і правила порівняння атрибутів. Кожний атрибут запису може зберігати кілька значень.

3.3 Розробка функціональної схеми

S/MIME – відмінний приклад гібридного рішення шифрування, у якому об'єднані достоїнства симетричного й асиметричного шифрів і функцій гешування. На рисунку 3.6 показана функціональна схема використання S/MIME.

Відправник хоче послати безпечне повідомлення (з гарантованими конфіденційністю, цілісністю, дійсністю даних) користувачеві Одержувач. Обмін S/MIME складається з наступних кроків:

1. Відправник створює цифрову сигнатуру для повідомлення з використанням свого приватного ключа.

2. Відправник використовує єдиний ключ симетричного шифрування для шифрування повідомлення.

3. Щоб сформувати безпечний канал, у якому буде захищена конфіденційність ключа шифрування при його передачі через загальнодоступний канал зв'язку, Відправник використовує відкритий ключ Одержувача (із сертифіката Одержувача), щоб зашифрувати ключ шифрування. Результат цього етапу – захищений ящик (lockbox), у якому втримується зашифрована копія ключа шифрування.

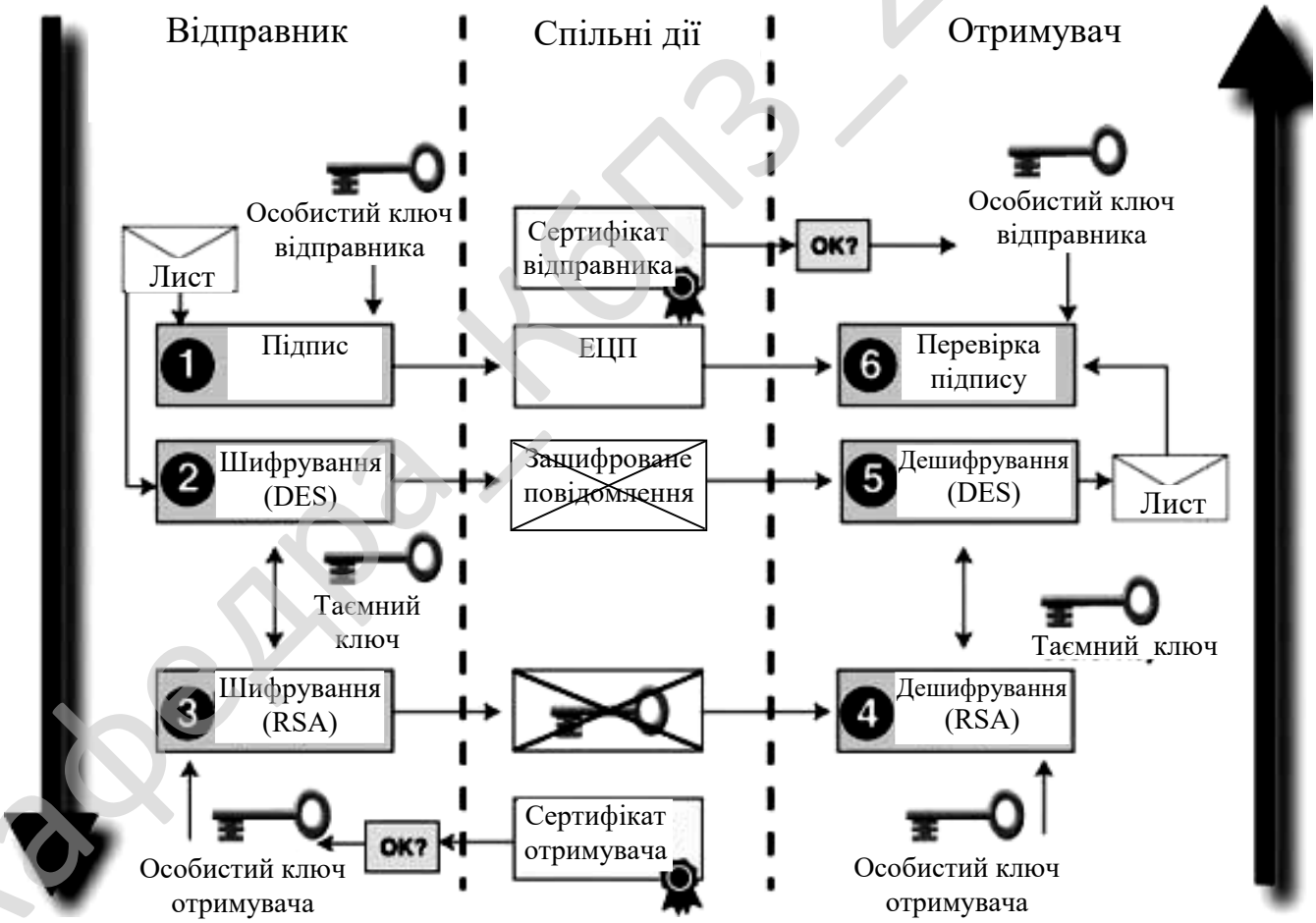

Рисунок 3.6 – Функціональна схема використання S/MIME

4. Одержувач використовує свій приватний ключ, щоб розшифрувати захищений ящик. У процесі дешифрування формується єдиний ключ шифрування.

5. Одержувач розшифровує повідомлення за допомогою єдиного ключа шифрування. Тепер Одержувач може прочитати повідомлення.

6. Одержувач використовує відкритий ключ Відправника для перевірки дійсності й цілісності повідомлення, засвідчуючи цифрову сигнатуру, що є в сертифікаті Відправника. S/MIME забезпечує стійке наскрізне шифрування поштових повідомлень. Це значить, що при використанні S/MIME повідомлення зашифровані не тільки на етапі пересилання, але й при зберіганні в локальній особистій папці й поштовій скриньці.

3.4 Розробка діаграми процесів

На рисунку 3.7 зображена діаграма взаємодії процесів системи. З неї ми бачимо, що у системі відбуваються наступні процеси.

Після початку роботи можливо виконання однієї з операцій:

– Створення/відкриття поштової скриньки.

– Перегляд змісту листа.

У разі вибору дії створення/відкриття поштової скриньки починають виконуватися наступні процеси:

– генерація сертифікату та ключа для підпису листа;

– створення ключа для шифрування;

– написання листа;

– перевіка поштової скриньки.

Після написання листа, він шифрується, на нього накладається цифровий підпис та відбувається відправка листа.

Якщо ж відбувається перевірка поштової скриньки, то переглядаються отримані листи. Для цього перевіряється цифровий підпис та відбувається

дешифрування листа.

Якщо підпис справжній то лист вважається дійсно надісланим від санкціонованого передавача даних. У іншому випадку лист знищується й надсилається повідомлення про помилку відправнику даних.

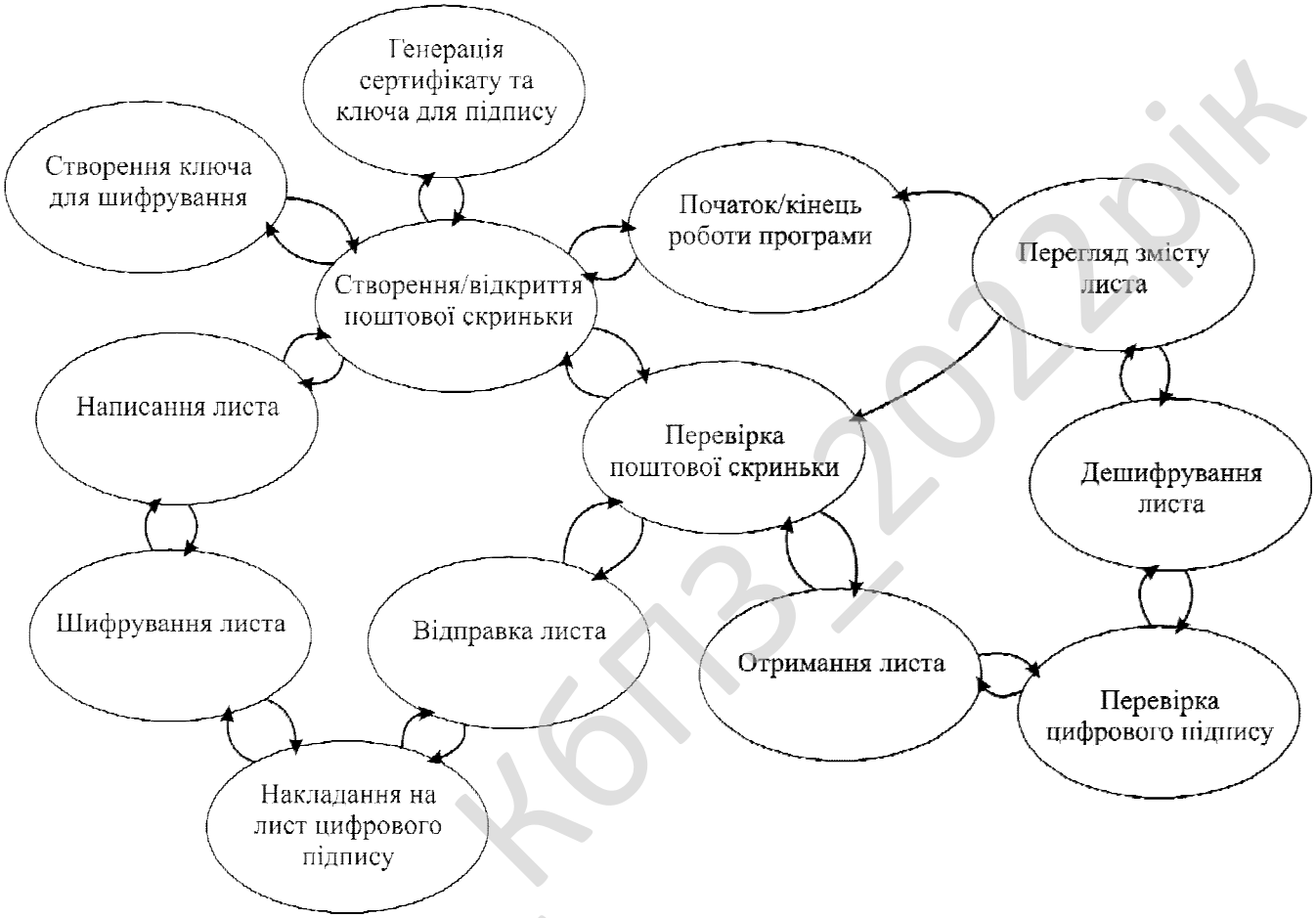

Рисунок 3.7 – Діаграма процесів системи

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рисунку 4.1 зображена блок-схема основної програми.

З неї ми бачимо, що програма працює наступним чином. Спершу відбувається завантаження основного вікна програми. За цим слідує перевірка наявності поштової скриньки. Якщо її немає, вона створюється.

Якщо поштова скринька є то відбувається перевірка наявності ключа шифрування. Якщо його немає, то він генерується.

Якщо ж ключ є то перевіряється наявність ключа та сертифіката для цифрового підпису, яким буде підписуватися лист, уповноваженою особою, що має право надсилати лист.

Якщо ж ключа немає, то відбувається генерація ключа для цифрового підпису та генерація сертифікату.

Після виконання вищеперерахованих дій перевіряється поштова скринька на наявність листів.

Якщо є вхідні листи то читаються їх заголовки.

Відбувається завантаження вхідних листів. Під час завантаження відбувається перевірка цифрових підписів з використанням відповідних алгоритмів накладання та перевірки. Після чого виводяться результати перевірки підписів.

За цим слідує дешифрування алгоритма DES, яким був зашифрований лист. Це дозволяє вивести вихідний текст захищеного листа на екран.

Якщо обрано дію відправки листа, то відбуваються наступні кроки.

Спершу виводиться вікно створення листа, після цього виводиться заголовок листа, на вводиться текст, який буде міститися у листі.

Після цього відбувається процедура шифрування листа за допомогою алгоритма симетричного шифрування DES.

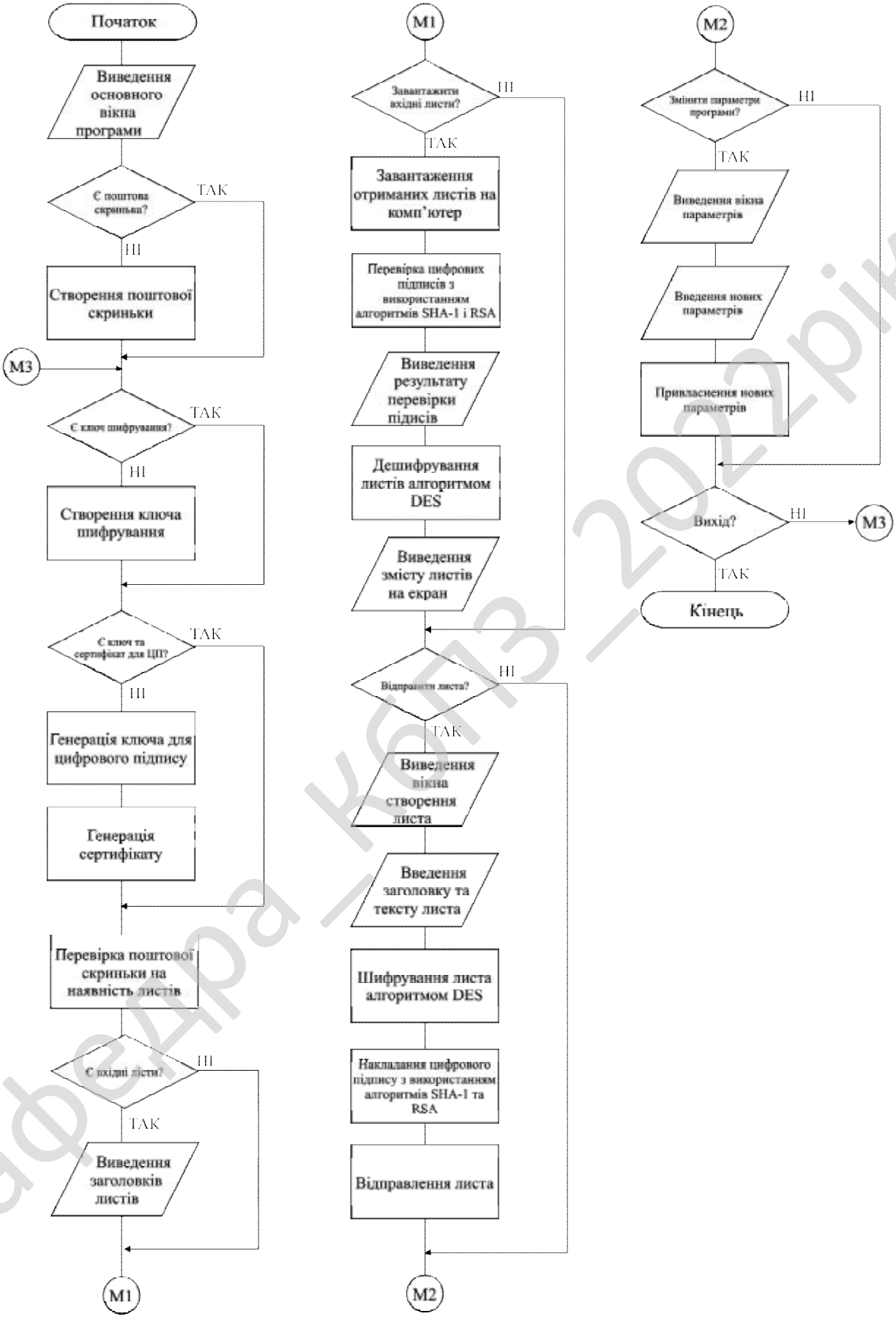

Рисунок 4.1 – Блок-схема основної програми

Зашифрований лист підписується цифроим підписом з використанням алгоритмів гешуваня SHA-1 та цифрового підпису RSA.

Після цього зашифрований лист з цифровим підписом відправляється адресатові.

У програмі існує ще можливість зміни параметрів. Для цього виводиться вікно параметрів у якому можливо змінити параметри методом вибору з існуючих сертифікованих згенерованих нових параметрів.

Накладання ЦВЗ та ЕЦП здійснюється відправником (підписувачем) документу із використанням його таємного ключа.

Перевірка ЦВЗ та ЕЦП виконується одержувачем захищеного електронного документу з використанням відкритого ключа підписувача.

Генерація ключів здійснюється датчиком псевдовипадкових чисел, після чого програма надсилає запит на сертифікацію відкритого ключа.

Заявка на сертифікацію відкритого ключа складається у формі електронного документа, шифрується відкритим ключем центру сертифікатів та надсилається у зашифрованому вигляді по мережі.

В окремих випадках, при угоді сторін, заявка складається і у вигляді документа на паперовому носієві, підписаного власноручним підписом особи та передається з рук у руки.

Заявка на сертифікацію відкритого ключа повинна містити:

– прізвище та ім'я фізичної особи, номер документа, що засвідчує особу;

– інші ідентифікаційні дані особи, а також інформацію, необхідну для передачі їй повідомлень;

– відкритий ключ, що треба сертифікувати;

– інші відомості, встановлені уповноваженим органом.

Володіння сертифікатом дозволяє користувачеві підписувати свої повідомлення цифровим підписом. Після успішної сертифікації закритий та відкритий ключі зберігаються у файлі на флешці чи іншому електронному носії, а відкритий ключ також заноситься у базу даних цифрових сертифікатів.

Випуск сертифіката відкритого ключа здійснюється центром сертифікації. Створення центрів сертифікації здійснюється з метою впорядкування процесу випуску та відкликання («анулювання») сертифікатів відкритого ключа, у підтримці актуальності бази даних сертифікатів і встановлення відповідальності за правильність внесення в сертифікат необхідних відомостей. Ці обов'язки покладені на адміністраторів центрів сертифікації. Конкретні підстави для одержання сертифіката визначаються окремими документами компанії.

На виданий сертифікат встановлюється термін дії. Існує непряма залежність корисного терміну дії від інформації, яку містить сертифікат. Чим більше інформації містить цей документ, тим менше корисний термін його дії. Це пов’язано з тим, що інформація може змінюватися, й власнику доведеться видати новий сертифікат до того, як вибігає термін дії старого.

Новий сертифікат відкритого ключа може бути виданий у випадку закінчення терміну дії сертифіката, втрати або компрометації діючого сертифіката.

Нижче наведено блок-схеми вже конкретно шифрування/дешифрування та формування/читання цифрового підпису.

На рисунку 4.2 наведена блок-схема підпрограми створення цифрового підпису за допомогою алгоритму RSA.

З неї ми бачимо, що відбуваються наступні дії. Спершу відбувається вибір сертифікату зі списку сгенерованих сертифікатів.

Після цього відбувається обчислення геш-функції по відповідному алгоритму SHA-1. Геш-значення шифрується відкритим ключем відправника. Оримане значення записується у кінець листа, про що виводиться відповідне повідомлення.

На рисунку 4.3 наведена блок-схема підпрограми перевірки цифрового підпису створеного за допомогою алгоритму RSA.

Перевірка відбувається наступним чином. Спершу перевіряється сертифікат відправника, якщо сертифікат дійсний, то вважається, що цифровий

підпис справжній, у іншому випадку вважається що цифровий підпис було підроблено, робота програми закінчується.

Якщо ж цифровий підпис правильний, то відбувається дешифрування геш-значення відкритим ключем відправника.

Після цього відбувається обчислення геш-значення отриманого листа за формулами наведеними нижче, та перевіряється справжність геш-функції. Для цього порівнюється геш функція яку сформував отримувач по листу, та геш-функція. Яка була отримана у результаті дешифрування. Якщо вони співпадають, то вважається, що лист справжній. У іншому випадку, вважається. що лист несправжній.

Рисунок 4.2 – Блок-схема підпрограми створення цифрового підпису за допомогою алгоритму RSA

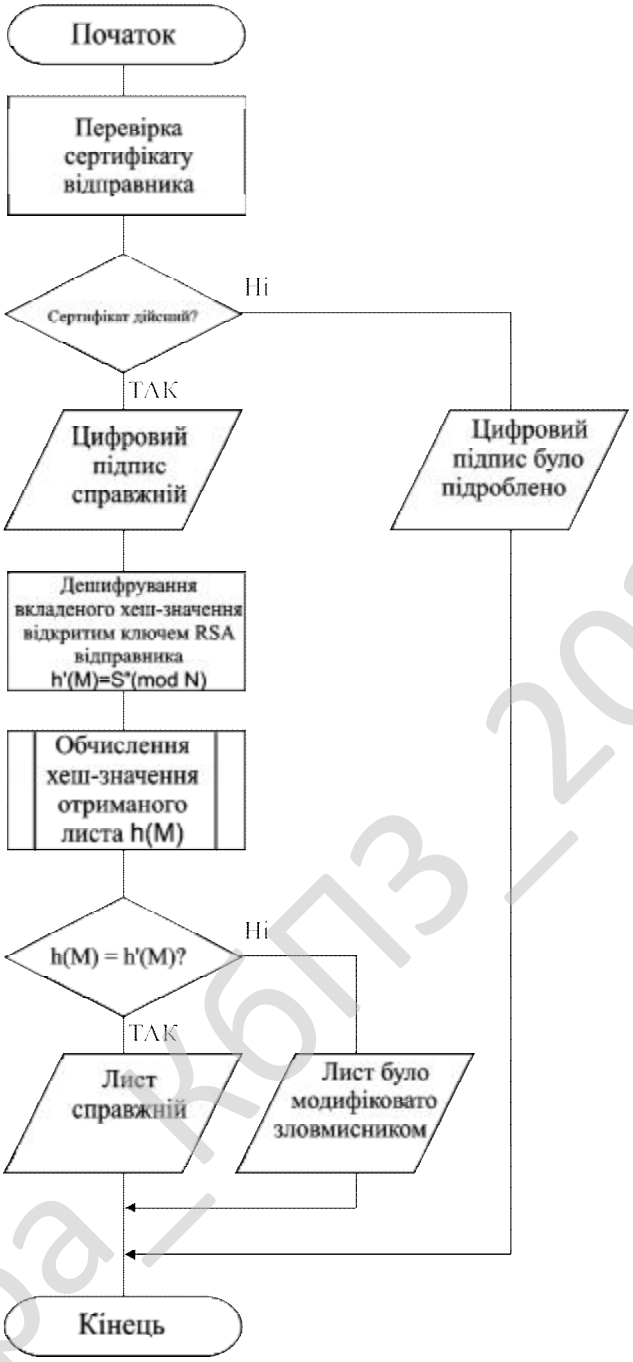

Рисунок 4.3 – Блок-схема підпрограми перевірки цифрового підпису створеного за допомогою алгоритму RSA

Важливе значення в створенні ЕЦП посідає геш-функція. Блок-схема алгоритму обчислення геш-функції наведено на рисунку 4.4.

У даному магістерському проекті для реалізації геш-функції було використано алгоритм SHA-1.

Надамо опис цього алгоритму, більш детальніше.

Криптографічна геш-функція SHA-1 (SHA – Secure Hash Algorithm) призначена для обчислення з повідомлення довжиною до 2^{64} бітів його ущільненого представлення довжиною 160 бітів.

Геш-функція SHA-1 опрацьовує вхідне повідомлення послідовними блоками по 512 бітів, тому якщо довжина повідомлення не є кратною 512 бітам, здійснюється його вирівнювання на границю блока.

Для цього в кінець повідомлення дописується біт 1, а потім дописуються біти 0; це дописування припиняється, коли до кінця блока залишається 64 біта. В останні 64 біта блока записується початкова довжина повідомлення (до вирівнювання); при цьому найменш значущі біти довжини повинні знаходитись у кінці блока.

В процесі обчислення геш-функції застосовуються дві групи по п'ять 32-бітних змінних – A, B, C, D, E і H_0, H_1, H_2, H_3, H_4, а також вісімдесят 32-бітних змінних $W_0,\ldots,W_{79}$. Крім цього, в процесі роботи використовується змінна T довжиною 32 біта.

Нехай вирівняне повідомлення складається з 512-бітних блоків $M_1,\ldots,M_n$. Процес обробки кожного блока M_i включає 80 ітерацій. Перед початком обробки блоків здійснюється ініціалізація змінних H_0, H_1, H_2, H_3, H_4: $H_0=67452301$; $H_1=EFCDAB89$; $H_2=98BADCFE$; $H_3=10325476$; $H_4=C3D2E1F0$.

Обробка блока M_i полягає в наступному:

1. Блок M_i послідовно розбивається на шістнадцять 32-бітних частин, які записуються в змінні $W_0,\ldots,W_{15}$.

2. Для i від 16 до 79 виконуються обчислення:

$$W_i = (W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}) <<<1 \qquad (4.1)$$

3. Здійснюються присвоювання:

$$A = H_0,\ B = H_1,\ C = H_2,\ D = H_3,\ E = H_4. \qquad (4.2)$$

Для i від 0 до 79 виконуються наступні обчислення:

$$T = (A<<<5) \oplus_{32} f(B,C,D) \oplus_{32} E \oplus_{32} W_i \oplus_{32} K, \quad (4.3)$$

$$E = D,\ D = C,\ C = B<<<30,\ B = A,\ A = T, \quad (4.4)$$

$$T = (A<<<5) \oplus_{32} f(B,C,D) \oplus_{32} E \oplus_{32} W_i \oplus_{32} K, \quad (4.5)$$

$$E = D,\ D = C,\ C = B<<<30,\ B = A,\ A = T. \quad (4.6)$$

Здійснюються переприсвоювання:

$$H_0 = H_0 \oplus_{32} A,\ H_1 = H_1 \oplus_{32} B,\ H_2 = H_2 \oplus_{32} C,$$

$$H_3 = H_3 \oplus_{32} D,\ H_4 = H_4 \oplus_{32} E. \quad (4.7)$$

В процесі обробки блоків використовується функція *f*, причому в залежності від номера *i* ітерації ця функція має різний вигляд. Можливі типи функції *f*, наведені нижче.

$$f(B,C,D) = (B\&C) \vee ((\oplus B)\&D) \quad (4.8)$$

$$f(B,C,D) = B \oplus C \oplus D \quad (4.9)$$

$$f(B,C,D) = (B\&C) \vee (B\&D) \vee (C\&D) \quad (4.10)$$

В ітераціях 0-19 використовується функція *f*,типу (4.7), в ітераціях 20-39, 60-79 – функція *f*, типу (4.8), в ітераціях 40-59 – функція *f*, типу (4.9). Крім функції *f*, від номера ітерації *i* також залежить значення константи *K*. Можливі значення цієї константи наведені нижче.

1) K = 5A827999 (ціла частина числа $[\ 2^{30} \times 2^{1/2}\ ]$);

2) K = 6ED9EBA1 (ціла частина числа $[\ 2^{30} \times 3^{1/2}\ ]$);

3) K = 8F1BBCDC (ціла частина числа $[\ 2^{30} \times 5^{1/2}\ ]$);

4) K = CA62C1D6 (ціла частина числа $[\ 2^{30} \times 10^{1/2}\ ]$).

В ітераціях 0-19 використовується константа *K* типу 1, в ітераціях 20-39 – константа *K* типу 2, в ітераціях 40-59 – константа *K* типу 3, в ітераціях 60-79 – константа *K* типу 4.

Після обробки останнього блока повідомлення – блока M_n здійснюється конкатенація змінних H_0, H_1, H_2, H_3, H_4, і таким чином отримується 160-бітний результат обчислення геш-функції від повідомлення.

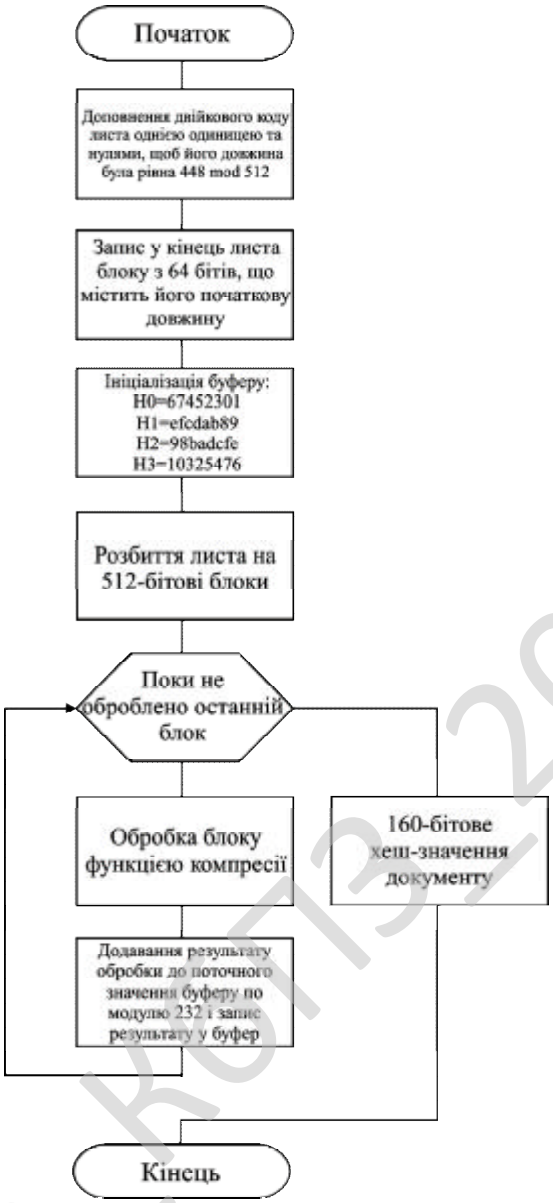

Рисунок 4.4 – Блок-схема підпрограми обчислення геш-значення листа за допомогою алгоритму SHA-1

Нижче опишемо алгоритми шифрування, які застосовуються у програмі. Ми опишемо наступні алгоритми:

– DES – для шифрування повідомлення.

– RSA – для шифрування ключів.

DES

DES являє собою блоковий шифр, він шифрує дані 64-бітовими блоками. З одного кінця алгоритму вводиться 64-бітовий блок відкритого тексту, а з іншого кінця виходить 64-бітовий блок шифротексту. DES є симетричним алгоритмом: для шифрування й дешифрування використовуються однакові алгоритм і ключ (за винятком невеликих розходжень у використанні ключа). довжина ключа дорівнює 56 бітам. (ключ звичайно представляється 64-бітовим числом, але кожний восьмий біт використовується для перевірки парності й ігнорується, біти парності є найменшими значущими бітами байтів ключа). Ключ, що може бути будь-яким 56-бітним числом, можна змінити в будь-який момент часу. Ряд чисел вважаються слабкими ключами, але їх можна легко уникнути. Безпека повністю визначається ключем.

Процес шифрування складається із чотирьох етапів. На першому з них виконується початкова перестановка (IP) 64-бітного вихідного тексту (забілювання), під час якої біти переставляються у відповідності зі стандартною таблицею. Наступний етап складається з 16 раундів однієї й тієї ж функції, що використовує операції зсуву й підстановки. На третьому етапі ліва й права половини виходу останньої (16-ої) ітерації міняються місцями. Нарешті, на четвертому етапі виконується перестановка IP^{-1} результату, отриманого на третьому етапі. Перестановка IP^{-1} інверсна початковій перестановці.

У правій частині рисунку 3.1 показаний спосіб, яким використовується 56-бітний ключ. Спочатку ключ подається на вхід функції перестановки. Потім для кожного з 16 раундів підключ K_i є комбінацією лівого циклічного зсуву й перестановки. Функція перестановки та сама для кожного раунду, але підключи K_i для кожного раунду виходять різні внаслідок повторюваного зсуву бітів ключа.

Рисунок 4.5 – Загальна схема DES

На найпростішому рівні алгоритм не представляє нічого більшого, ніж комбінація двох основних методів шифрування: зсуву й дифузії. Фундаментальним будівельним блоком DES є застосування до тексту одиничної комбінації цих методів (підстановка, а за нею – перестановка), що залежить від ключа. Такий блок називається етапом. DES складається з 16 етапів, однакова комбінація методів застосовується до відкритого тексту 16 разів (рисунок 4.6).

Розглянемо більш докладно алгоритм шифрування. DES працює з 64-бітовим блоком відкритого тексту. Після первісної перестановки блок розбивається на праву й ліву половини довжиною по 32 біта. Потім виконується 16 етапів однакових дій, називаних функцією *f*, у яких дані поєднуються із ключем. Після шістнадцятого етапу права й ліва половини поєднуються й алгоритм завершується заключною перестановкою (зворотною відносно первісної). На кожному етапі біти ключа зрушуються, і потім з 56 біт ключа вибираються 48 біт. Права половина даних збільшується до 48 біт за допомогою перестановки з розширенням, поєднується за допомогою XOR с 48 бітами зміщеного й переставленого ключа, проходить через 8 S-блоків, утворюючи 32 нових біта, і переставляється знову. Ці чотири операції й виконуються функцією

f. Потім результат функції f поєднується з лівою половиною за допомогою іншого XOR. У підсумку цих дій з'являється нова права половина, а стара права половина стає нової лівою. Ці дії повторюються 16 разів, утворюючи 16 етапів DES.

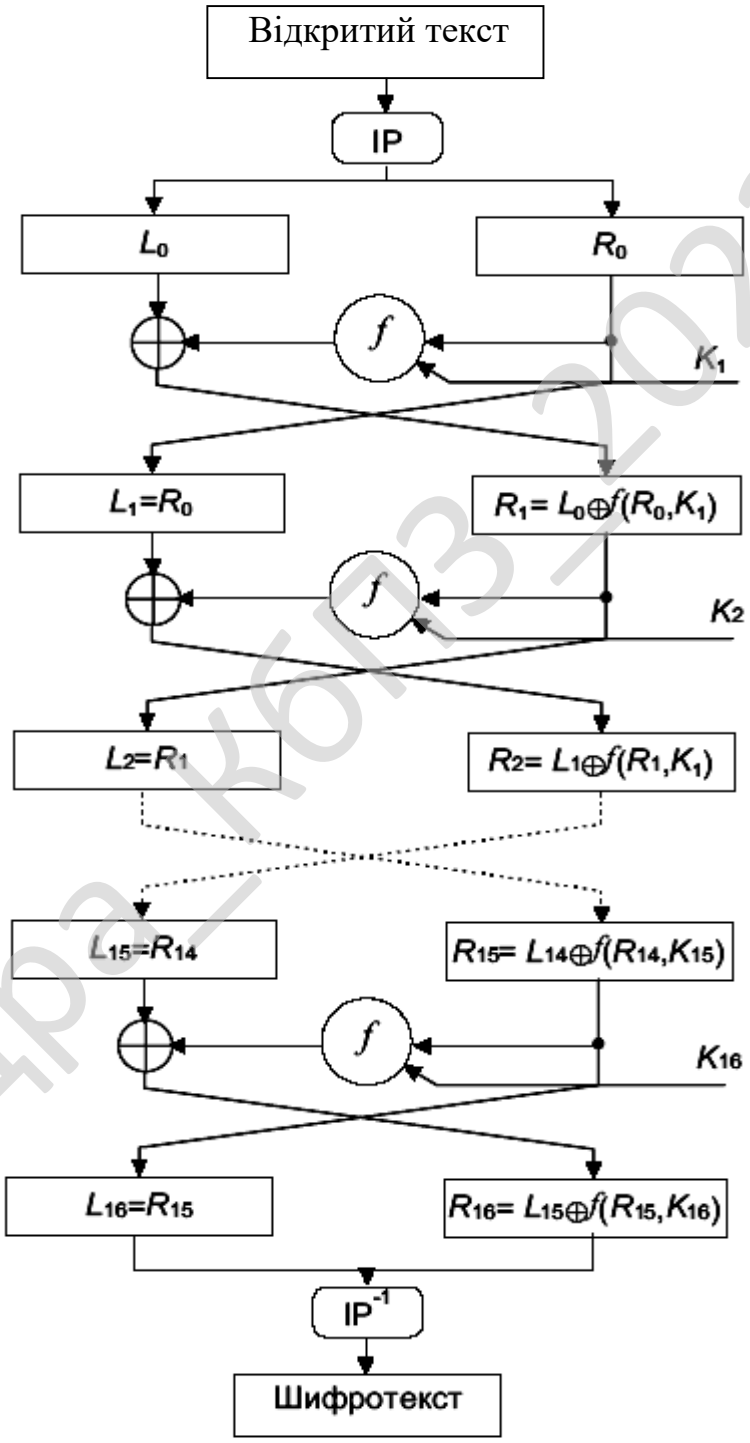

Рисунок 4.6 – Алгоритм DES

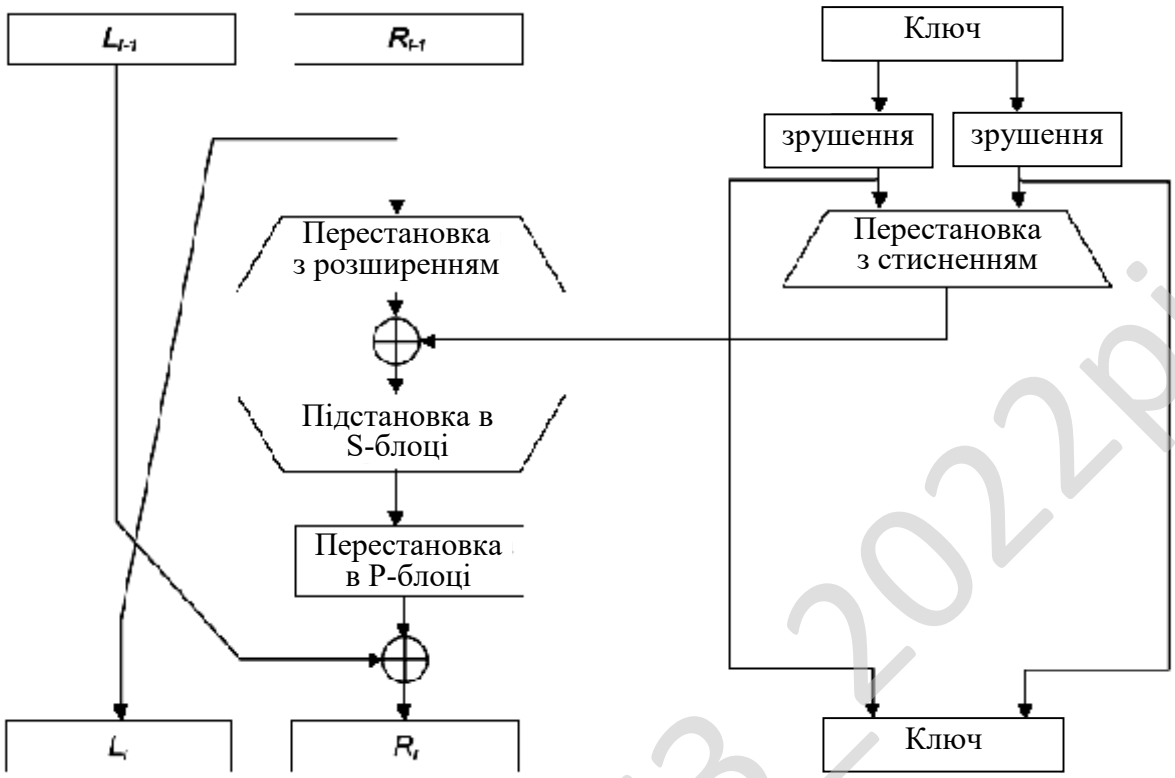

Рисунок 4.7 – Один етап DES

Якщо B_i – це результат i-ої ітерації. L_i, і R_i, – ліва й права половини B_i, K_i – 48-бітовий ключ для етапу i, а f – це функція, що виконують всі підстановки, перестановки й XOR з ключем, то етап можна представити як: $L_i = R_{i-1}$, $R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$.

Початкова перестановка виконується ще до етапу 1, при цьому вхідний блок переставляється.

Таблиця 4.1 – Початкова перестановка шифру DES

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

Початкова перестановка й відповідна заключна перестановка не впливають на безпеку DES. Так як програмна реалізація цієї багатобітної перестановки нелегка, у багатьох програмних реалізаціях DES початкова й заключна перестановки не використовуються. Хоча такий новий алгоритм не менш безпечний, чим DES, він не відповідає стандарту DES і, тому, не може називатися DES.

Перетворення ключа: 64-**б**ітовий ключ DES зменшується до 56-бітового ключа відкиданням кожного восьмого біта. Ці біти використовуються тільки для контролю парності, дозволяючи перевіряти правильність ключа. Після добування 56-бітового ключа для кожного з 16 етапів DES генерується новий 48-бітовий підключ і ці підключи, K_i, визначаються в такий спосіб:

Таблиця 4.2 – Перетворення ключа DES

57	49	41	33	25	17	9	1	58	50	42	34	26	18
10	2	59	51	43	35	27	19	11	3	60	52	44	36
63	55	47	39	31	23	15	7	62	54	46	38	30	22
14	6	61	53	45	37	29	21	13	5	28	20	12	4

По перше, 56-бітовий ключ ділиться на дві 28-бітових половинки. Потім, половинки циклічно зрушуються ліворуч на один або два біти залежно від етапу.

Таблиця 4.3 – Число біт зрушення залежно від етапу DES

Етап	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Число	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Після зрушення вибирається 48 з 56 біт. Так як при цьому не тільки вибирається підмножина біт, але й змінюється їхній порядок, ця операція називається **перестановка зі стиском.** Її результатом є набір з 48 біт. Наприклад, біт зрушеного ключа в позиції 33 переміщається в позицію 35 результату, а 18-й біт зрушеного ключа відкидається.

Таблиця 4.4 – Перестановка зі стиском

14	17	11	24	1	5	3	28	15	6	21	10
23	19	11	4	26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40	51	45	33	48
44	49	39	56	34	53	46	42	50	36	29	32

Через зрушення для кожного підключа використовується відмінна підмножина біт ключа. Кожний біт використовується приблизно в 14 з 16 підключей, хоча не всі біти використовуються в точності однакове число раз

Перестановка з розширенням: ця операція розширює праву половину даних, R_i, від 32 до 48 біт. Так як при цьому не просто повторюються певні біти, але й змінюється їхній порядок, ця операція називається перестановкою з розширенням, У неї дві задачі: привести розмір правої половини у відповідність із ключем для операції XOR і одержати більше довгий результат, який можна буде стиснути в ході операції підстановки. Однак головний криптографічний зміст зовсім в іншому. За рахунок впливу одного біта на дві підстановки швидше зростає залежність біт результату від біт вихідних даних. Це називається **лавинним ефектом.** DES спроектований так, щоб якнайшвидше домогтися залежності кожного біта шифротексту від кожного біта відкритого тексту й кожного біта ключа.

Перестановка з розширенням показана на 9-й. Іноді вона називається Е-б**локом** (від expansion). для кожного 4-бітовий вхідний блоки перший і четвертий біт являють собою два біти вихідного блоку, а другий і третій біти – один біт вихідного блоку. В 7-й показано, які позиції результату відповідають яким позиціям вихідних даних. Наприклад, біт вхідного блоку в позиції 3 переміститься в позицію 4 вихідні блоки, а біт вхідного блоку в позиції 21 – у позиції 30 і 32 вихідного блоку.

Хоча вихідний блок більше вхідного, кожний вхідний блок генерує унікальний вихідний блок

Таблиця 4.5 – Перестановка з розширенням DES

32	1	2	3	4	5	4	5	6	7	8	9
8	9	10	11	12	13	12	13	14	15	16	17
16	17	18	19	20	21	20	21	22	23	24	25
24	25	26	27	28	29	28	29	30	31	32	1

Підстановка за допомогою S-блоків: Після об'єднання стислого блоку з розширеним блоком за допомогою XOR над 48-бітовим результатом виконується операція підстановки. Підстановки виробляються у вісьмох блоках підстановки, або S-блоках (від Su*b*stitution). У кожного 8-блоку 6-бітовий вхід і 4-бітовий вихід, усього використовується вісім різних S-блоків. (для восьми S-блоків DES буде потрібно 256 байтів пам'яті.) 48 біт діляться на вісім 6-бітових підблока. Кожний окремий підблок обробляється окремим S-блоком: перший підблок – S-блоком 1, другий – S-блоком 2 і так далі. Кожний **S-блок** являє собою таблицю з 2 рядків і 16 стовпців. Кожний елемент у блоці є 4– бітовим числом. По 6 вхідних бітах S-блоку визначається, під якими номерами стовпців і рядків шукати вихідне значення.

Перестановка за допомогою P-блоків: 32-бітовий вихід підстановки за допомогою S-блоків, перетасовуються відповідно до P-блоку. Ця перестановка переміщає кожний вхідний біт в іншу позицію, жоден біт не використовується двічі, і жоден біт не ігнорується. Цей процес називається прямою перестановкою або просто перестановкою. Наприклад, біт 21 переміщається в позицію 4, а біт 4 – у позицію 31.

Таблиця 4.6 – перестановка за допомогою P-блоків

16	7	20	21	29	12	28	17	1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9	19	13	30	6	22	11	4	25

Нарешті, результат перестановки за допомогою P-блоку поєднується за допомогою XOR. с лівою половиною первісного 64-бітового блоку. Потім ліва й права половини міняються місцями, і починається наступний етап.

Заключна перестановка є зворотною відносно початкової перестановки. Зверніть увагу, що ліва й права половини не міняються місцями після останнього етапу DES, замість цього об'єднаний блок $R_{16}L_{16}$ використовується як вхід заключної перестановки. У цьому немає нічого особливого, перестановка половинок з наступним циклічним зрушенням привела б до точно такого ж результату. Це зроблено для того, щоб алгоритм можна було використовувати як для шифрування, так і для дешифрування.

Таблиця 4.7 – Заключна перестановка DES

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

Дешифрування DES

Після всіх підстановок, перестановок, операцій XOR. і циклічних зрушень можна подумати, що алгоритм дешифрування, різко відрізняючись від алгоритму шифрування, точно так заплутаний. Навпроти, різні компоненти DES були підібрані так, щоб виконувалася дуже корисна властивість: для шифрування й дешифрування використовується той самий алгоритм. DES дозволяє використовувати для шифрування або дешифрування блоку ту саму функцію. Єдина відмінність полягає в тому, що ключі повинні використовуватися у зворотному порядку. Тобто, якщо на етапах шифрування використовувалися ключі K_1, K_2, K_3, ..., K_{16}, то ключами дешифрування будуть K_{16}, K_{15}, K_{14}, ..., K_1. Алгоритм, що створює ключ для кожного етапу, також циклічний. Ключ зрушується праворуч, а число позицій зрушення дорівнює 0, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1.

RSA

Діффі й Хеллман визначили новий підхід до шифрування, що викликало до життя розробку алгоритмів шифрування, що задовольняють вимогам систем з *відкритим ключем*. Одним з перших результатів був алгоритм, розроблений в 1977 році Роном Ривестом, Ади Шамиром і Льоном Адлеманом і опублікований в 1978 році. З того часу алгоритм Rivest-Shamir-Adlem*a*n (RSA) широко застосовується практично у всіх додатках, що використовують криптографію з *відкритим ключем*.

Алгоритм заснований на використанні того факту, що задача ***факторизації*** (тобто розкладення числа на множники) є важкою, тобто легко перемножити два числа, у той час як не існує поліноміального алгоритму знаходження простих співмножників великого числа.

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

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

$$C = M^e \pmod{n}, \tag{4.10}$$

$$M = C^d \pmod{n} = (M^e)^d \pmod{n} = M^{ed} \pmod{n}. \tag{4.11}$$

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

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

Створення ключів

Вибрати прості p та q.

Обчислити $n = p \cdot q$.

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

Вибрати e взаємно просте з $\Phi(n)$: НСД $(\Phi(n), e) = 1$; $1 < e < \Phi(n)$.

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

Відкритий ключ KU = $\{e, n\}$ – публікується

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

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

Шифрування

Незашифрований текст: $M < n$.

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

Дешифрування

Зашифрований текст: $C < n$.

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

Обчислювальні аспекти

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

Шифрування/дешифрування

Як шифрування, так і дешифрування включають піднесення цілого числа в цілий ступінь по модулю n. При цьому проміжні значення будуть величезними. Для того, щоб частково цього уникнути (тобто, щоб проміжні результати не виходили за границі n), використовується наступна властивість модульної арифметики:

$$(a \cdot b) \bmod n = (a \bmod n) \cdot (b \bmod n). \qquad (4.3)$$

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

Створення ключів

Створення ключів включає наступні задачі:

1. Визначити два простих числа p та q.

2. Вибрати e й обчислити d.

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

Були розроблені різні тести для визначення того, чи є число простим. Це тести імовірнісні, тобто тест показує, що дане число ймовірно є простим. Незважаючи на це, вони можуть виконуватися таким чином, що зроблять імовірність близькою до 1. Якщо n "провалює" тест, то воно не є простим. Якщо n "пропускає" тест, то n може як бути, так і не бути простим. Якщо n пропускає багато таких тестів, то можна з високим ступенем вірогідності сказати, що n є простим. Це досить довга процедура, але вона виконується відносно рідко: тільки при створенні нової пари (KU, KR).

На складність обчислень також впливає те, яка кількість чисел буде відкинута перед тим, як буде знайдене просте число. Результат з теорії чисел, відомий як теорема простого числа, говорить, що простих чисел, розташованих біля n у середньому одне на кожні ln (n) чисел. Таким чином, у середньому потрібно перевірити послідовність із ln (n) цілих, перш ніж буде знайдене просте число. Тому що всі парні числа можуть бути відкинуті без перевірки, то потрібно

виконати приблизно ln (n)/2 перевірок. Наприклад, якщо просте число шукається в діапазоні величин 2^{200}, то необхідно виконати біля ln (2^{200}) / 2 = 70 перевірок.

Вибравши прості числа p та q, далі необхідно вибрати значення e так, щоб НСД(Ф(n), e) = 1 і обчислити значення d, $d = e^{-1} \bmod Ф(n)$. Існує єдиний алгоритм, який називається розширеним алгоритмом Евкліда, що за фіксований час обчислює найбільший спільний дільник двох цілих і якщо цей спільний дільник дорівнює одиниці, визначає інверсне значення одного за модулем іншого. Таким чином, процедура полягає в генерації серії випадкових чисел і перевірці кожного відносно Ф(n) доти, поки не буде знайдене число, взаємопросте з Ф(n). Виникає питання, як багато випадкових чисел доведеться перевірити доти, поки не знайдеться потрібне число, що буде взаємопростим з Ф(n). Результати показують, що ймовірність того, що два випадкових числа є взаємопростими, дорівнює 0.6.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою MARS, який є блочно-симетричним шифром з відкритим ключем. Розмір блоку при шифруванні 128 біта, розмір ключа може варіюватися від 128 до 448 біт включно (кратні 32бітам). Творці прагнули поєднати в своєму алгоритмі швидкість кодування і стійкість шифру. В результаті вийшов один з самих криптостійкий алгоритм з алгоритмів, які брали участь в конкурсі AES.

Алгоритм унікальний тим, що використовував практично всі існуючі технології, застосовувані в криптоалгоритмах, а саме:

- Найпростіші операції (додавання, віднімання, виключаюче або).
- Підстановки з використанням таблиці замін.
- Фіксований циклічний зсув.
- Залежний від даних циклічний зсув.
- Множення за модулем 2^{32}.
- Ключове забілювання.

Використання подвійного перемішування представляє складність для криптоаналізу, що деякі відносять до недоліків алгоритму. У той же час на даний момент не існує будь-яких ефективних атак на алгоритм, хоча деякі ключі можуть генерувати слабкі підключі.

Структура алгоритму

Автори шифру виходили з наступних припущень:

1. Вибір операцій. MARS був спроектований для використання на найсучасніших комп'ютерах того часу. Для досягнення найкращих захисних характеристик в нього були включені самі «сильні операції» підтримувані в них. Це дозволило добитися більшого відносини securityper-instruction для різних реалізації шифру.

2. Структура шифру. Двадцятирічний досвід роботи в області криптографії підштовхнув творців алгоритму до думки, що кожен раунд шифрування грає свою роль в забезпеченні безпеки шифру. Зокрема, ми можемо бачити, що перший і останній раунди зазвичай сильно відрізняються від проміжних («центральних») раундів алгоритму в плані захисту від криптоаналітичних атак. Таким чином, при створенні MARSa використовувалася змішана структура, де перший і останній раунди шифрування істотно відрізняються від проміжних.

3. Аналіз. Швидше за все, алгоритм з гетерогенною структурою буде краще протистояти криптоаналітичних методам майбутнього, ніж алгоритм, всі раунди якого ідентичні. Розробники алгоритму MARS надали йому сильно гетерогенну структуру – раунди алгоритму дуже різняться між собою.

У шифрі MARS використовувалися такі методи шифрування:

1. Робота з 32-х бітними словами. Всі операції застосовуються до 32-бітовим словами. тобто вся початкова інформація розбивається на блоки по 32біта. (Якщо ж блок опинявся меншої довжини, то він доповнювався до 32біт)

2. Мережа Фейстеля. Творці шифру вважали, що це найкращий варіант поєднання швидкості шифрування і криптостійкості. В MARS використана мережа Фейстеля 3-го типу.

3. Симетричність алгоритму. Для стійкості шифру до різних атакам всі його раунди були зроблені повністю симетричними, тобто друга частина раунду є дзеркальне повторення першої його частини.

Структуру алгоритму MARS можна описати таким чином:

1. Попереднє накладення ключа: на 32-бітові субблоки A, B, C, D накладаються 4 фрагмента розширеного ключа k_0 … k_3 операцією складання за модулем 2^{32}.

2. Виконуються 8 раундів прямого перемішування (без участі ключа шифрування).

3. Виконуються 8 раундів прямого криптоперетворення.

4. Виконуються 8 раундів зворотного криптоперетворення. [2]

5. Виконуються 8 раундів зворотного перемішування, також без участі ключа шифрування.

6. Фінальне накладення фрагментів розширеного ключа k_{36} … k_{39} операцією віднімання за модулем 2^{32}.

Пряме перемішування

У першій фазі на кожне слово даних накладається слово ключа, а потім відбувається вісім раундів змішування згідно з мережею Фейстеля третього типу спільно з деякими додатковими змішування. У кожному раунді ми використовуємо одне слово даних (зване, вихідним словом) для модифікації трьох інших слів (звані, цільовими словами). Ми розглядаємо чотири байта вихідного слова як індексів на двох S-блоків, S_0 і S_1 , кожен, що складається з 256 32-розрядних слів, а далі проводимо операції XOR або додавання даних відповідного S-блоку в три інших слова.

Якщо чотири байти вихідного слова b_0 , b_1 , b_2, b_3 (де b_0 є першим байтом, а b_3 є старшим байтом), то ми використовуємо b_0 , b_2, як індекси в блоку S_0 і байти

b_1 , b_3 , як індекси в S-блоці S_1 . Спочатку зробимо XOR S_0 до першого цільовим речі, а потім додамо S_1 до того ж слова. Ми також додаємо S_0 до другого цільовим слову і XOR блоку-S_1 до третього цільовим слову. У висновку, ми обертаємо вихідне слово на 24 біта вправо.

У наступному раунді ми обертаємо наявні у нас чотири слова: таким чином, нинішні перші цільове слово стає наступним вихідним словом, поточним другий цільове слово стає новим першим цільовим словом, третє цільове слово стає наступний другий цільовим словом, і поточне вихідне слово стає третім цільовим словом.

Більш того, після кожного з чотирьох раундів ми додаємо одне з цільових слів назад у вихідне слово. Зокрема, після першого і п'ятого раундів ми додав третю цільове слово назад у вихідне слово, а після другого і шостого раунду ми додаємо першої цільової слово назад у вихідне слово. Причиною цих додаткових операцій змішування, є ліквідація декількох простих диференціальних криптоатаки в фазі перемішування, щоб порушити симетрію у фазі змішування та отримати швидкий потік.

Криптографічне ядро

Криптографічне ядро MARS – мережа Фейстеля 3-го типу, що містить в собі 16 раундів. У кожному раунді ми використовуємо ключову E-функцію, яка є комбінацією множень, обертань, а також звернень до S-блоків. Функція приймає на вхід слово даних, а повертає три слова, з якими згодом буде здійснена операція додавання або XOR до інших трьох слів даними. У доповненні вихідне слово обертається на 13 біт вліво.

Для забезпечення, серйозного опору до криптоатаки, три вихідних значення E-функції (O_1 , O_2, O_3) використовуються в перших восьми раундах і в останніх восьми раундах в різних порядках. У перші вісім раундів ми додаємо O_1 і O_2до першого і другого цільовим речі, відповідно, і XOR O_3 у третьому цільовим слову. За останні вісім раундів, ми додаємо O_1 і O_2 до третього і другого цільовим речі, відповідно, і XOR O_3 до першого цільовим слову.

E-функція

E-функція приймає як вхідні дані одне слово даних і використовує ще два ключових слова, виробляючи на виході три слова. У цій функції ми використовуємо три тимчасові змінні, що позначаються L, M і R (для лівої, середньої та правої).

Спочатку ми встановлюємо в R значення вихідного слова зміщеного на 13 біт вліво, а в M – сума вихідних слів і першого ключового слова. Потім ми використовуємо перші дев'ять бітів M як індекс до однієї з 512S-блоків (яке виходить суміщенням S_0 і S_1 змішуванням фази), і зберігаємо в L значення відповідного S-блоку.

Потім помножимо друге ключове слово на R, зберігши значення в R. Потім обертаємо R на 5 позицій вліво (так, 5 старших бітів стають 5 нижніми бітами R після обертання). Тоді ми робимо XOR R в L, а також переглядаємо п'ять нижніх біт R для визначення величини зсуву (від 0 до 31), і обертаємо M вліво на цю величину. Далі ми обертаємо R ще на 5 позицій вліво і робимо XOR в L. У висновку, ми знову дивимося на 5 молодших бітів R, як на величину обертання і обертаємо L на цю величину вліво. Таким чином результат роботи E-функції – 3 слова (по порядку): L, M, R.

Зворотне перемішування

Зворотне перемішування практично збігається з прямим перемішуванням, за винятком того факту, що дані обробляються в зворотному порядку. Тобто, якби ми поєднали пряме і зворотне перемішування так, щоб їх виходи і входи були б з'єднані в зворотному порядку (D [0] прямого і D [3] зворотного, D [1] прямого і D [2] зворотного), то не побачили б результату перемішування. Як і в прямому змішування, тут ми теж використовуємо одне вихідне слово і три цільових. Розглянемо чотири перших байта вихідного слова: b_0 , b_1 , b_2, b_3 . Будемо використовувати b_0 , b_2як індекс до S-блоку – S_1 , а b_1 b_3 для S_0 . Зробимо XOR S_1 [b_0] в перше цільове слово, віднімемо S_0 [b_3] з другого слова, віднімемо S_1 [b_2] з третього цільового слів і потім проробимо XOR S_0 [b_1] також до третього

цільового слова. Нарешті, ми повертаємо початкове слово на 24 позицій вліво. Для наступного раунду ми обертаємо наявні слова так, щоб нинішнє перше цільове слово стало наступним вихідним словом, поточне друге цільове слово стало першим цільовим словом, поточне третє цільове слово стало другим цільовим словом, і поточне вихідне слово стало третім цільовим словом. Крім того, перед одним з чотирьох «особливих» раундів ми віднімаємо одне з цільових слів з вихідного слова: перед четвертим і восьмим раундами ми віднімаємо першої цільової слово, перед третьому і сьомим раундами ми віднімемо третя цільова слово з вихідного.

Дешифрування

Процес декодування обернений процесу кодування. Код дешифрування схожий (але не ідентичний) на код шифрування.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Головне вікно програми зображене на рисунку 5.1.

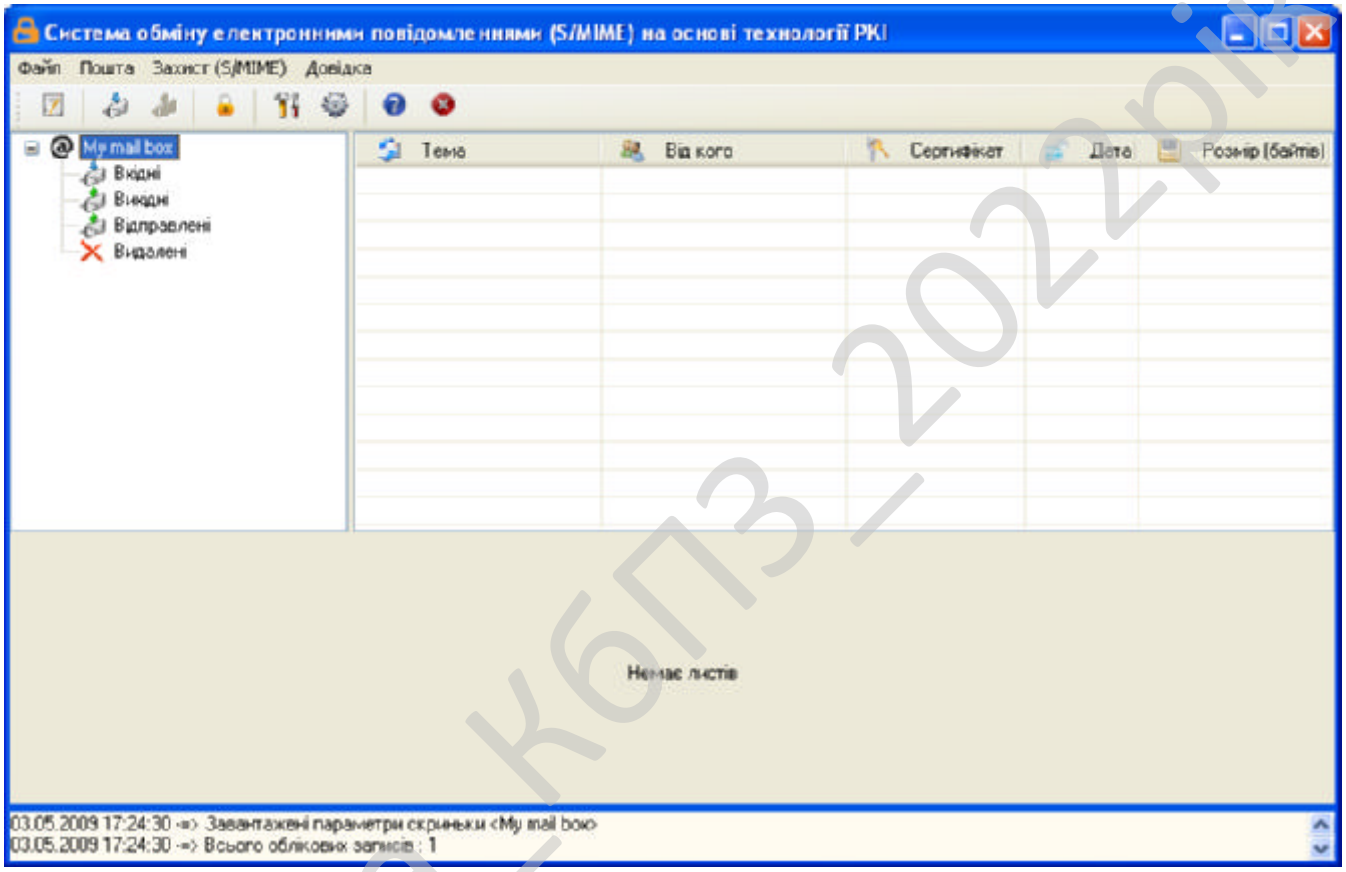

Рисунок 5.1 – Головне вікно програми

Пункти меню користувача зображені на рисунках 5.2-5.5.

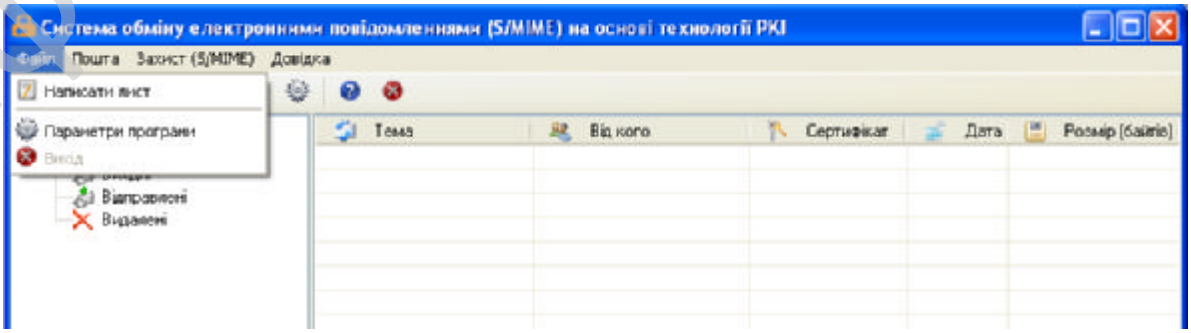

Рисунок 5.2 – Меню «Файл»

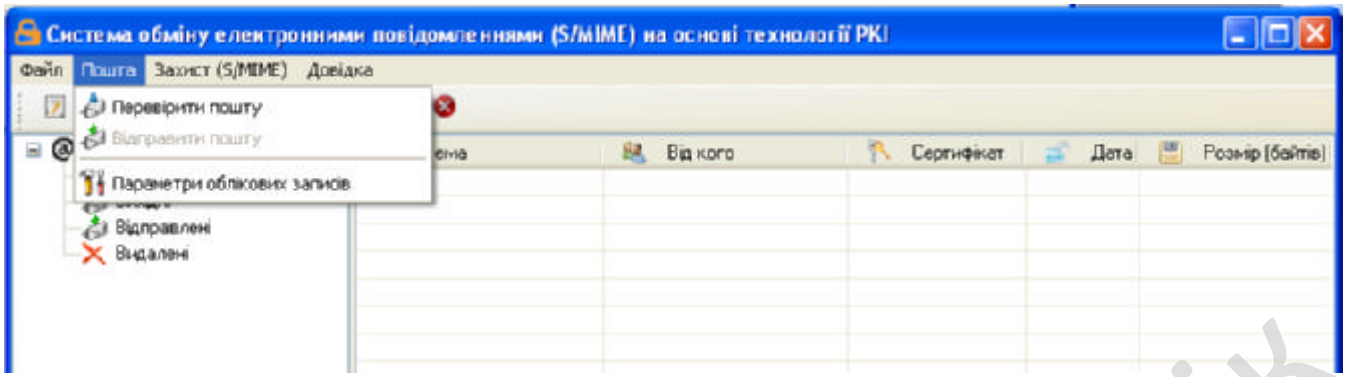

Рисунок 5.3 – Меню «Пошта»

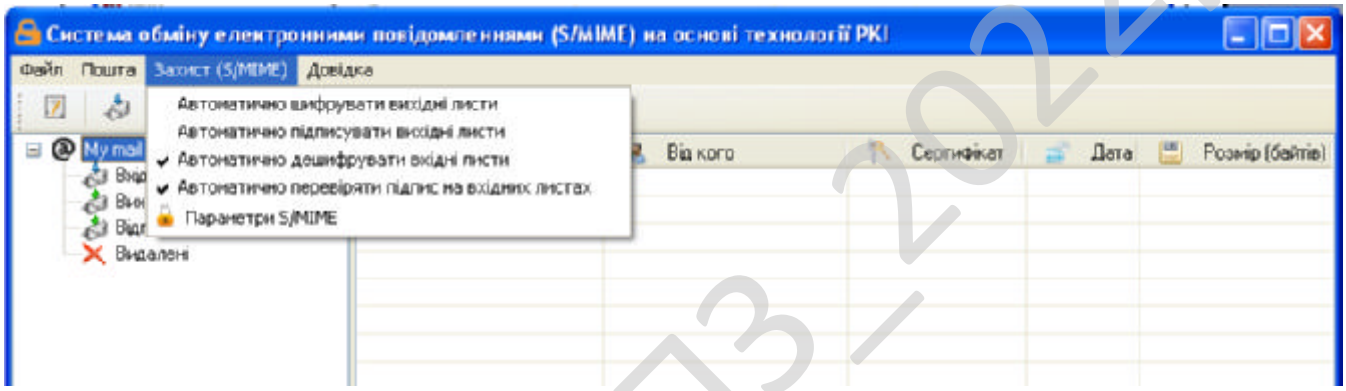

Рисунок 5.4 – Меню «Захист (S/MIME)»

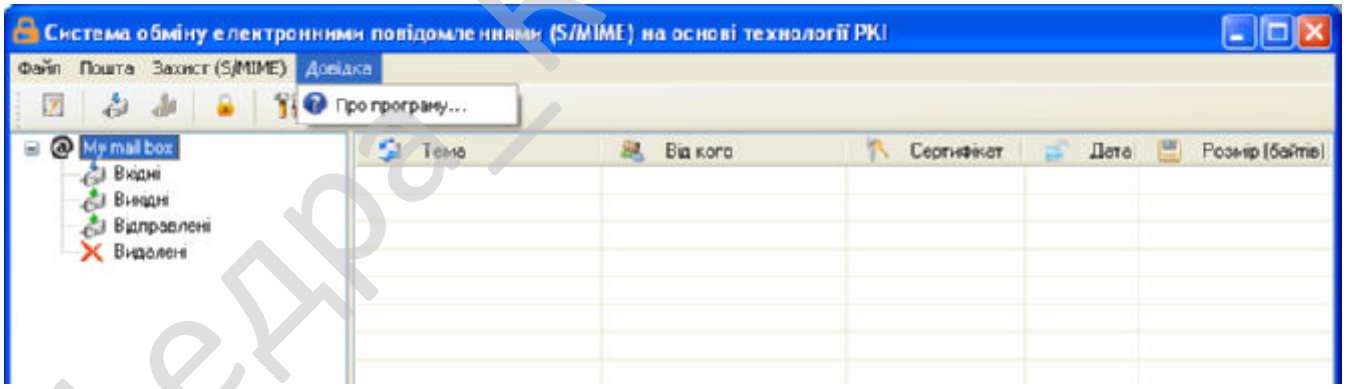

Рисунок 5.5 – Меню «Довідка»

Для генерації сертифікату і ключа підпису та створення ключа шифрування треба вибрати пункт «Параметри S/MIME» в меню «Захист (S/MIME)», після чого з'явиться вікно зображене на рисунку 5.6.

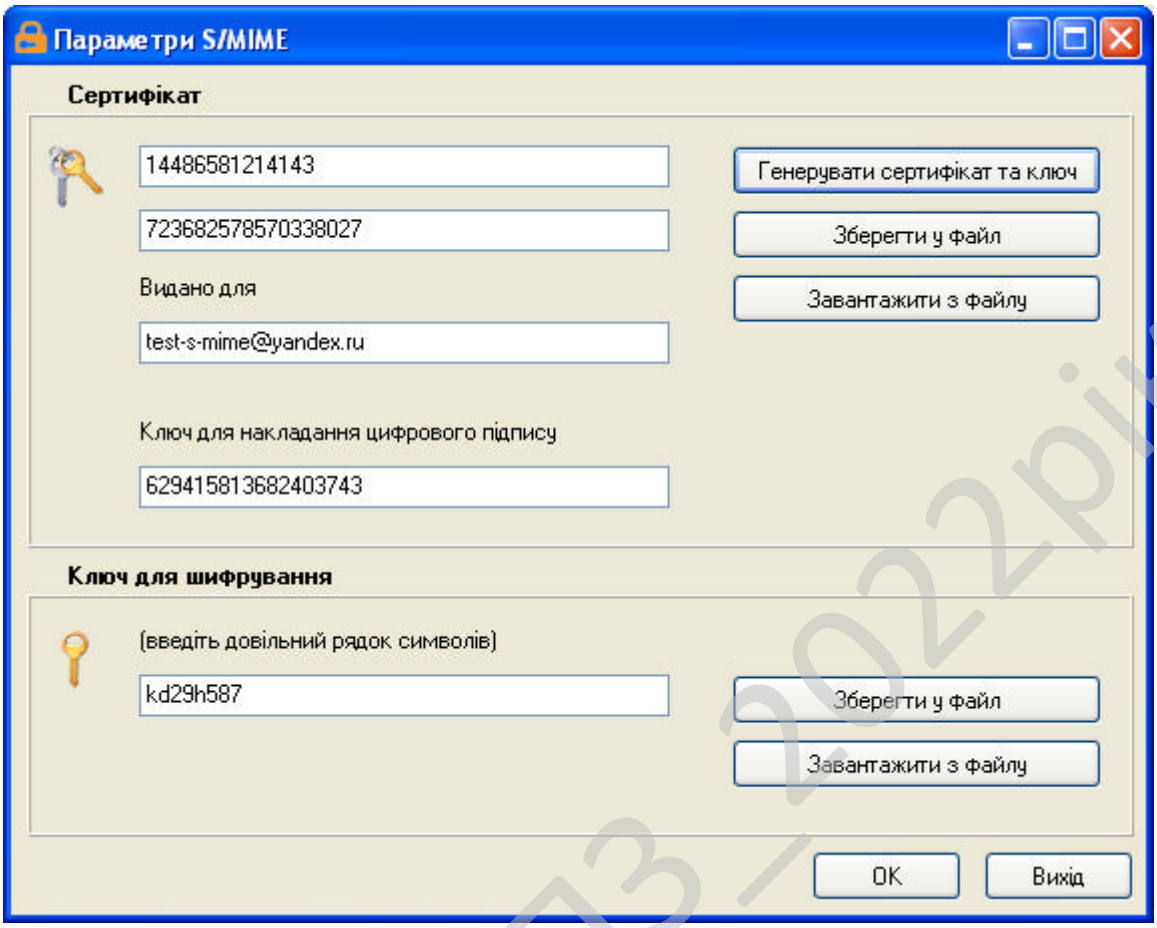

Рисунок 5.6 – Параметри S/MIME

Для створення листа необхідно вибрати пункт «Новий лист» в меню «Файл», після чого з'явиться вікно зображене на рисунку 5.7.

Обов'язково слід вказати електронну адресу одержувача листа у полі «Кому?». Бажано також вказати тему листа у полі «Тема». Інші поля заповнюються за бажанням користувача. Далі необхідно набрати текст повідомлення.

Для здійснення шифрування слід натиснути кнопку «Шифрувати» (функція спрацює тільки в тому разі, якщо в параметрах S/MIME було введено ключ для шифрування). Результат шифрування листа з текстом «Тут повинен бути текст секретного повідомлення» показаний на рисунку 5.8.

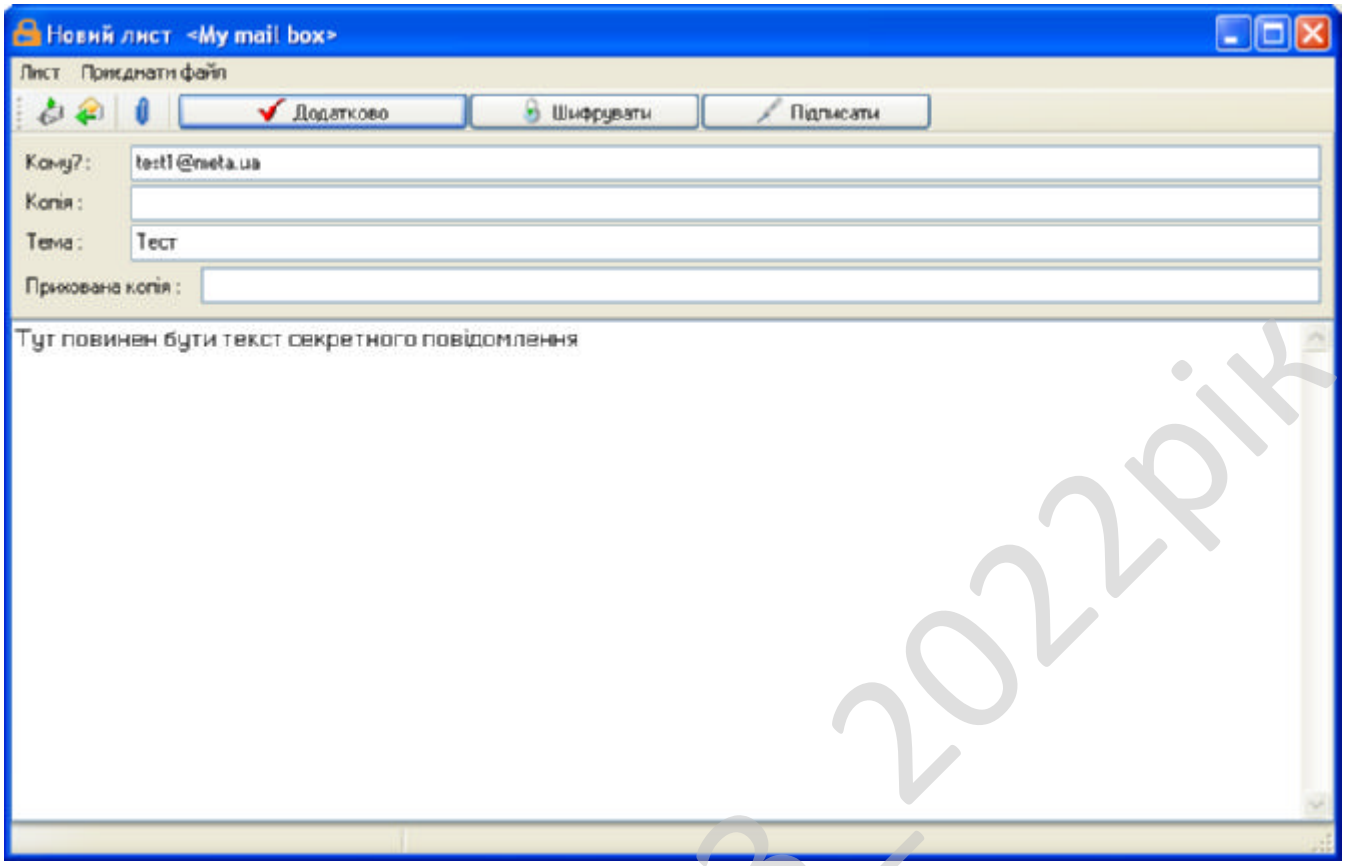

Рисунок 5.7 – Створення електронного листа

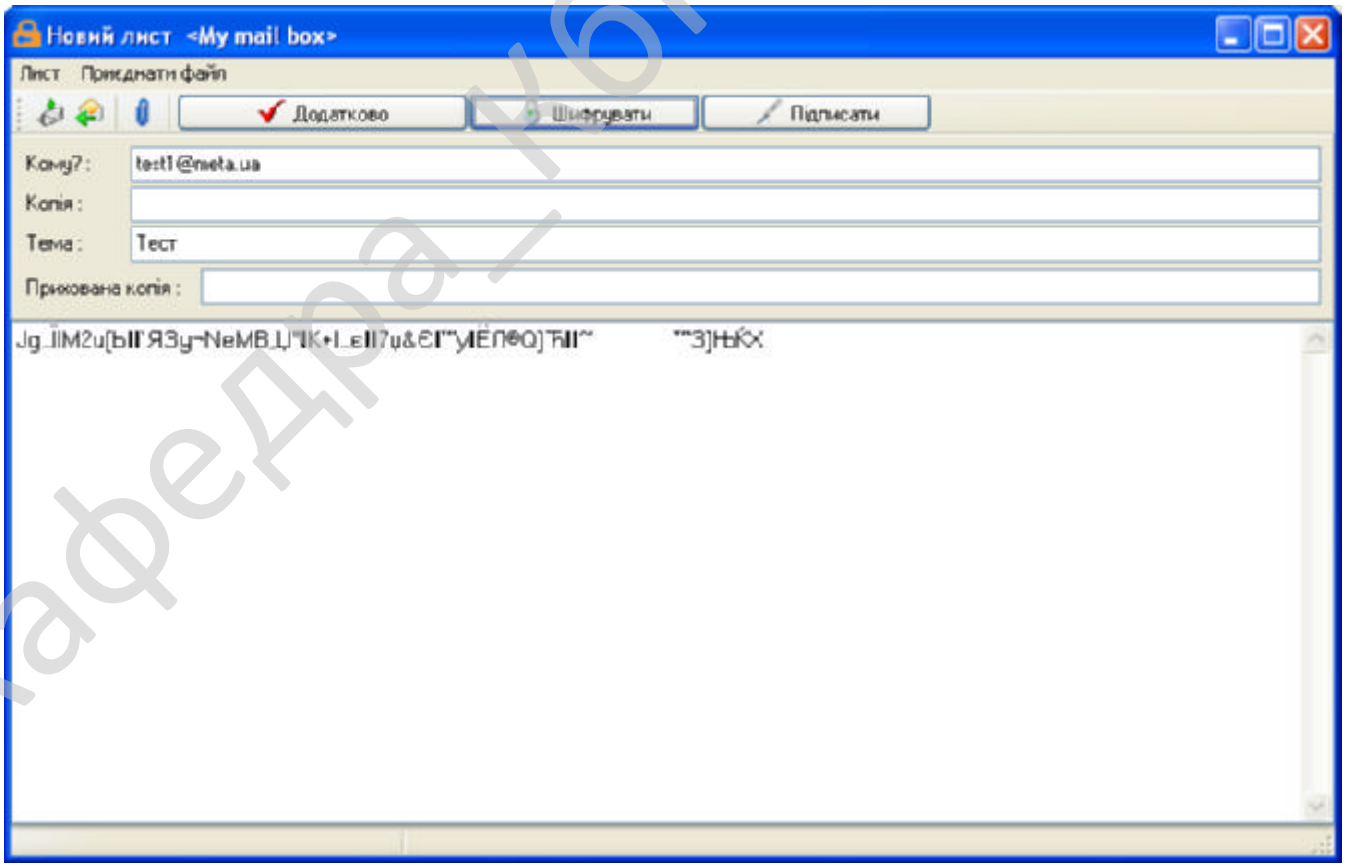

Рисунок 5.8 – Текст листа після шифрування

Для накладання цифрового підпису на повідомлення треба натиснути кнопку «Підписати» (функція спрацює тільки в тому разі, якщо в параметрах S/MIME було згенеровано та завантажено сертифікат та ключ для підпису). Результат підписання листа показаний на рисунку 5.9.

Як видно з малюнку цифровий підпис додається в кінець листа після додавання рядка з дефісами та рядка з текстом «цифровий підпис». Такий форма цифрового підпису було обрано для наглядності роботи системи.

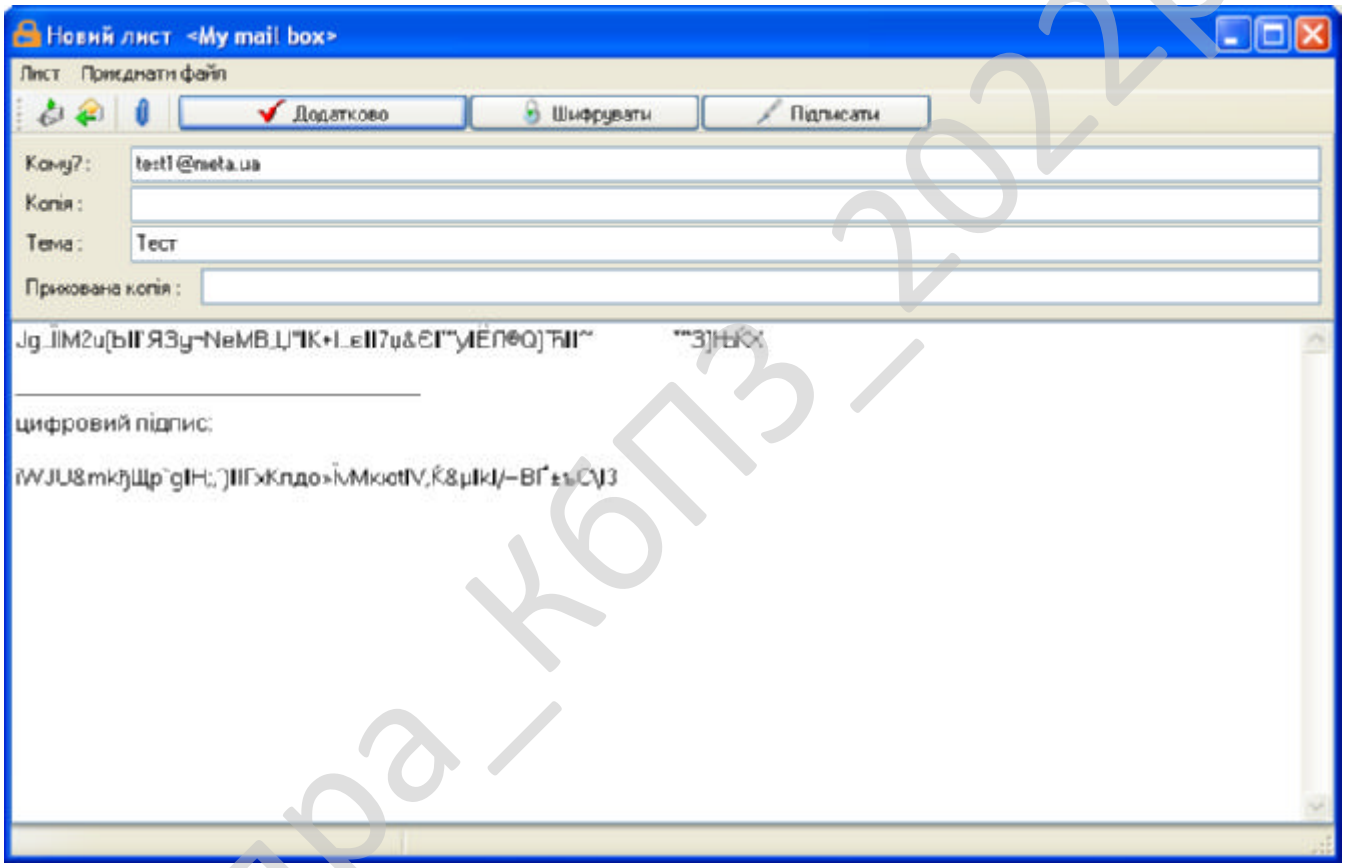

Рисунок 5.9 – Текст листа після накладання цифрового підпису

Для створення нової електронної скриньки, або зміни параметрів вже існуючої треба вибрати пункт «Параметри облікових записів» в меню «Пошта», після чого з'явиться вікно зображене на рисунку 5.10.

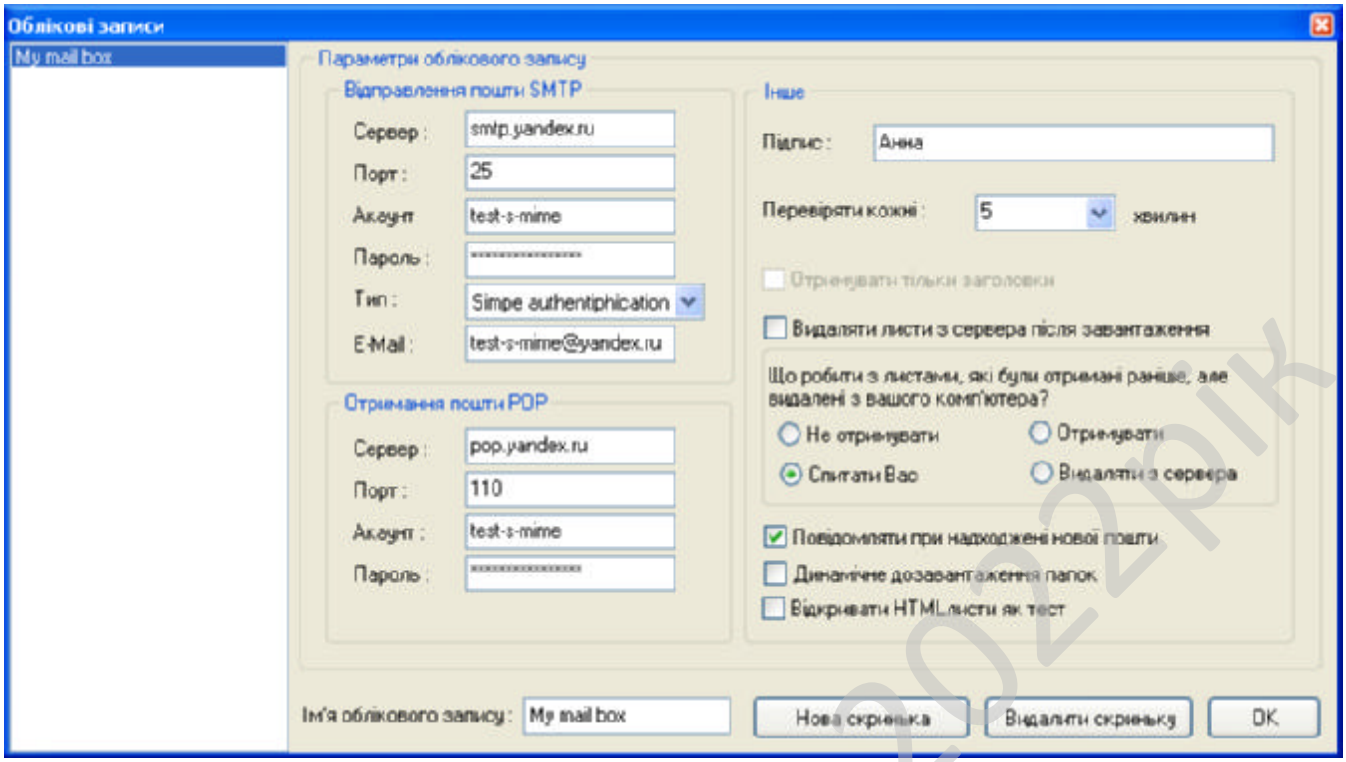

Рисунок 5.10 – Параметри облікових записів

Для того, щоб змінити параметри програми слід вибрати пункт «Параметри програми» в меню «Файл», після чого з'явиться вікно зображене на рисунку 5.11.

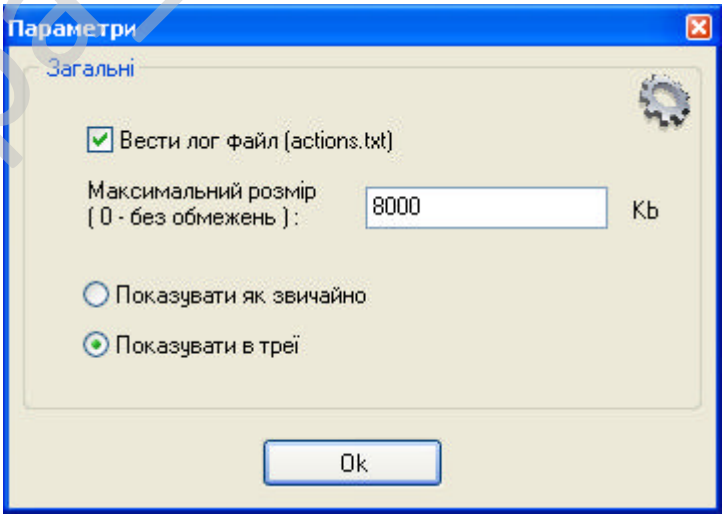

Рисунок 5.11 – Параметри програми

Коротку довідку про розроблену програму можна переглянути вибравши пункт «Про програму…» в меню «Довідка», після чого з'явиться вікно зображене на рисунку 5.12.

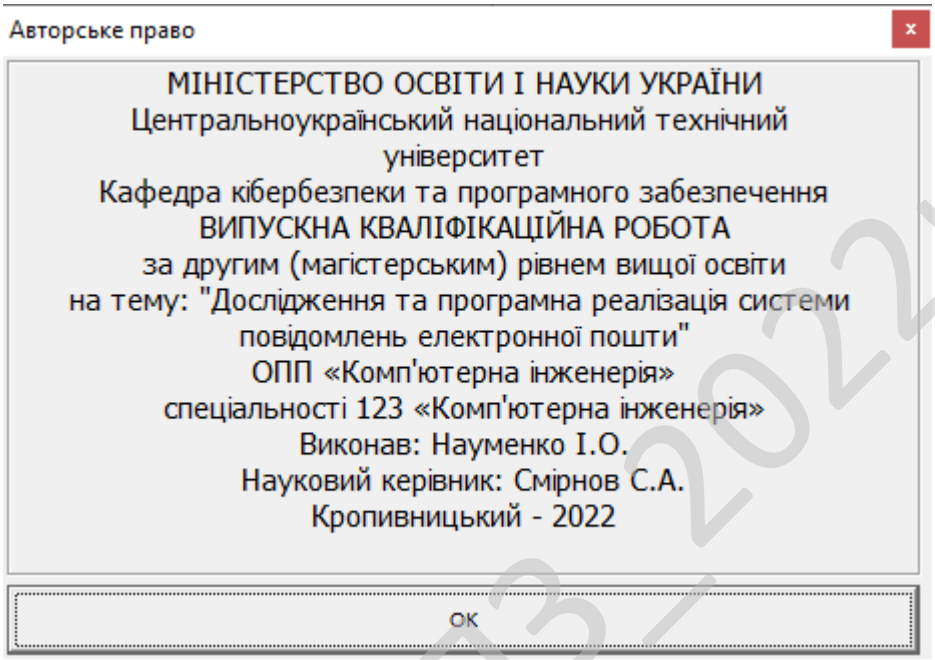

Рисунок 5.12 – Довідка про програму

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи повідомлень електронної пошти.

Метою розробки є дослідження та програмна реалізація системи повідомлень електронної пошти.

Об'єктом дослідження є процес повідомлень електронної пошти.

Предметом дослідження є методи повідомлень електронної пошти.

Методи дослідження базуються на методах теорії комп'ютерних мереж, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод повідомлень електронної пошти.

– Розроблено вітчизняний продукт повідомлень електронної пошти, 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 48 днів (два місяці).

В магістерській роботі проведене дослідження та виконана програмна реалізація системи повідомлень електронної пошти.

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	58
3. Запланований термін розробки, днів	Fpq	48 (2 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2
7. Кількість макетів вхідної інформації	–	3

Продовження таблиці 7.1

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

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн	–	60000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	50
38. Ставка податку на додану вартість, %	Ндв	20

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 А – коефіцієнт Боема, А=2,45;

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

B – показник ступеня, що визначається співвідношенням

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

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

$$B=1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}026$$

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

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

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

де $ПВ$j – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн}= 6{,}78\cdot(0{,}88\cdot0{,}93\cdot0{,}88\cdot0{,}91\cdot0{,}95\cdot1\cdot1\cdot0{,}87\cdot1{,}22\cdot1{,}16\cdot1{,}1\cdot1{,}1\cdot$$

$$\cdot1{,}12\cdot1{,}1\cdot1\cdot1{,}1\cdot1{,}1)=9{,}37 \text{ люд-міс.}$$

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

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

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

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)%

$$T_{РП}=0{,}3\cdot2{,}66\cdot9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)}\cdot55=93 \text{ люд/день}$$

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	9	Д7
Робочий проект	93	Ф 7.1-7.4
Впровадження	13	Д13
Всього	134	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

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

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

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

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

$$Ч = \frac{134 \cdot 1}{48\text{-}5} = 3{,}1 \text{ ставки}$$

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	8	720	12
Монітор	60	8	480	8
Клавіатура	30	8	240	4
Маніпулятор «мишка»	30	8	240	4
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	1	30	0,5
Кабельні господарства ЛОМ на 1 м.п.	2,5	250	625	10,42
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	44,58

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

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

$$Ф_{др}^{с} = \frac{45 \cdot 2}{1,2} = 75 \text{ год}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

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

$Ч_{ел}$=75/(48·8)=0,2 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 2016, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	0,8	0,2
	Налаштування і конфігурування базової станції безпровідного зв’язку (CMTS)	0,2	
	Розробка та впровадження проектів з організації зв’язку між віддаленими об’єктами, ЛОМ	0,2	
	Забезпечення цілодобової роботи зв’язку клієнтів до мережі Інтернет	0,4	
Всього		1,6	

Продовження таблиці 7.4

Посада	Вид роботи	Час	Кількість 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	0,5	0,2
	Створення графічних і стилістичних елементів сайту	0,5	
	Оформлення банерів і промо-сторінок	0,3	
	Розміщення графіки і контенту на Інтернет сторінках	0,3	
Всього		1,6	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,2
	Верстка друкованих видань	0,2	
	Додрукова підготовка макетів	0,2	
	Розміщення графіки і контенту на Інтернет сторінках	0,2	
Всього		1,6	

Складемо штатний розклад виконавців у таблицю 7.5.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньо-місячний оклад, грн.	Всього за період роз-робки, грн.
Керівник (IT-менеджер)	1	23000	46000
Продакт-менеджер	0,5	16900	16900
Інженер-програміст	3,1	20500	127100
Інженер-електронщик	0,2	15000	6000
Інженер-системотехнік	0,2	17000	6800
Адміністратор мережі	0,2	18000	7200
Системний програміст	0,2	17000	6800
Дизайнер WEB	0,2	18000	7200
Інженер-верстальник	0,2	15000	6000
Бухгалтер-економіст	0,2	19000	7600
Всього за період розробки	$R_{сп}$=6	-	$Ф_{роб}$=237600

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рq}}, \tag{7.8}$$

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

$$З_{сд} = \frac{237600}{6 \cdot 48} = 825 \; грн.$$

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 8 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:

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

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

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

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

де Цм – ціна меблів для одного робочого місця, грн.

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

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

Специфікація на обчислювальну техніку наведена в таблиці 7.7. Дані по оптовій ціні на обладнання та комплектуючі вибирались за комерційною пропозицією фірми Brain за 14.11.22 – джерело http://brain.com.ua/

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		11771
Системний блок		7771
Процесор	Intel Core i7-3770 (4 (8) ядра по 3.4 – 3. GHz), 8 MB Smart Cache	-
Системна плата	Huanan B75 (s1155, Intel B75, PCI-Ex16 DVI/VGA/HDMI	-
Жорсткий диск	240 SSD	-
Оперативна пам’ять	8 GB DDR3	-
Відеокарта	вбудована	-
DVD-привод	DVD±RW ASUS DRW-24B5ST Black Bulk	-
Корпус	Logicpower 8702 – 550w 12cm	-
Кардрідер внутрішній	Transcend TS-RDF8K USB 3.0	-
інше	Клавіатура, мишка	-
Монітор	Монітор BenQ GL2450HM Black	2600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USI cable	5500
Сканер	Epson Perfection V37	2800
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965
Пристрій безперебійног живлення	Powercom BNT-600AP USB	1400

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	8	11771	9416,8	103584,8
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	1	2800	280	3080
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	125216,3

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400
Група 4			
3. Обчислювальна техніка	125216	-	-
Всього по групі	125216	50	62608

Продовження таблиці 7.8

1	2	3	4
Група 5,6			
4. Вимірювальні пристрої	5190	-	-
5. Транспортні засоби	143000	-	
6. Господарський інвентар	28000	-	-
Всього по групі	176190	20	35238
7. Нематеріальні активи	60000	10	6000
Разом	$К_р$= 1769406		$А_р$= 174246

Примітка: вартість автомобіля Ford Focus Ghia 2002 взята по даним з автосалону «Авто-РІА», джерело https://auto.ria.com/uk/auto_ford_focus_33565425.html, складає 143000 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.

$$З_о = 825 \cdot 134/58 = 1906 \text{ грн}$$

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, %

$$З_д = 1906 \cdot 10 \cdot 0{,}01 = 191 \text{ грн}$$

Відрахування на соціальні потреби за нормативом Нс=22% від суми основної та додаткової зарплати

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

де H_c – відрахування на соціальні потреби, %

$$C_{оц} = 0{,}01 \cdot 22(1906+191) = 461 \text{ грн}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом *Нг*=15% від основної зарплати

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

де $H_г$ – загальногосподарські витрати, %

$$Г_{осп} = 1906 \cdot 15 \cdot 0{,}01 = 286 \text{ грн}$$

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 0,4 пачки паперу на період розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=200 грн., визначаємо вартість паперу за період розробки:

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

$$З_{М1} = 200 \cdot 0{,}4 = 80 \text{ грн.}$$

Згідно прийнятих норм по комплектації до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків. Їх кількість дорівнює кількості коробочних версій запропонованого продукту (приймаємо 20):

$$З_{М2} = \sum Ц_д, \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR box – 23,7 грн./шт., DVD-R box – 35 грн./шт.

$$З_{М2} = 20 \cdot 23{,}7 = 474 \text{ грн.}$$

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

$$З_{МЗ} = \sum Ц_з., \quad (7.18)$$

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.

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

$$З_М = (80 + 474 + 1702)/58=39 \text{ грн.}$$

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, %

$$О_п = 1906 \cdot 15 \cdot 0{,}01=286 \text{ грн}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 58 прим.)

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

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

$$А_м = 174246 \cdot 2/(58 \cdot 12)=501 \text{ грн}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції

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

$$С_п = 1906+191+461+286+39+286+501=3670 \text{ грн.}$$

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

Для даного програмного забезпечення рівень рентабельності складає 50%

$$П_р = 0{,}01 \cdot Р_п \cdot С_п, \quad (7.22)$$

де $Р_с$ – рівень рентабельності, %

$$П_р = 0{,}01 \cdot 50 \cdot 3670 = 1835 \text{ грн.}$$

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	$З_о$	1906
2. Додаткова зарплата виконавців	$З_д$	191
3. Відрахування на соціальні потреби	$С_{оц}$	461
4. Загальногосподарські витрати	$Г_{осп}$	286
5. Витрати на матеріали	$З_М$	39
6. Освоєння нових операційних систем, мов програмування	$О_п$	286
7. Амортизація основних фондів	$А_м$	501
8. Повна собівартість програмного забезпечення	$С_п$	3670
9. Плановий прибуток	$П_р$	1835
10.Ціна підприємства $Ц_п=С_п+П_р$	$Ц_п$	5505
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	1101
12.Відпускна ціна програмної продукції Ц = Цп + ПДВ	$Ц$	6606

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	–	6606
Всього капітальних витрат	–	6006

7.7 Визначення експлуатаційних витрат

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

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на технічне обслуговування	$З_р$	40260	33550
2. Витрати на електроенергію	$З_{ел}$	0	0
3. Витрати на амортизацію	$З_{ам}$	0	3303
Всього витрат за рік	$І$	40260	36853

Витрати на технічне обслуговування:

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

де $Т_р$ – кількість годин обслуговування системи за рік, *год.*, $З_г$ – заробітна плата обслуговуючого персоналу, *грн / год*

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 250 годин на рік до 160 годин на рік, тому витрати на технічне обслуговування зменшилися з

$$З_{р\ баз} = 250 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 33550\ грн.$$

до

$$З_{р\ нов} = 160 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 21472\ грн.$$

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

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

Витрати на електроенергію при впровадженні нової системи не змінюються.

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	50	–	6606	–	3303
Всього відрахувань	-	–	6606	–	3303

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

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

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

$$Ев = (5505 - 3670) \cdot 58 - (0,05 \cdot 1408000 + 0,5 \cdot 125216 +$$
$$+0,2 \cdot 176190 + 0,1 \cdot 60000) \cdot 2/12 = 77389 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_р^*}{(Ц_п - C_п) \cdot N_е}, \quad (7.26)$$

де: $K_р^*$ – балансова вартість основних фондів розробника без врахування вартості ОФ третьої групи, так як їх строк служби на порядок більший ніж період розробки ПЗ.

$$T_в = \frac{361406}{(5505 - 3670) \cdot 58 \cdot 12 / 2} = 0,62 \text{ років}$$

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

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

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

$$E_{сп} = (40260\text{-}36853)\text{-}0,5 \cdot 6606 = 104 \text{ грн.}$$

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

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

$$T_{сп} = \frac{6606}{40260 - 36853} = 1,9 \text{ року}$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	58
2. Повна собівартість розробленої програми	Грн.	3670
3. Ціна розробленої програми	Грн.	5505
4. Плановий прибуток від реалізації розробленої програми	Грн.	1835
5. Рентабельність програмної продукції	%	50
6. Об’єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1769406
7. Загальний прибуток від реалізації програмної продукції	Грн.	106430
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	77389
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	0,62
10.Об’єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	6606
11.Величина економічного ефекту у користувача програмної продукції	Грн.	104
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	1,9

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

Наявний в даний час в нашій країні комплекс розроблених організаційних заходів та технічних засобів захисту, накопичений передовий досвід роботи ряду обчислювальних центрів показує, що є можливість домогтися значно більших успіхів у справі усунення впливу на працюючих небезпечних і шкідливих виробничих факторів. Проте стан умов праці та його безпеки в ряді обчислювальних центрів (ОЦ) та підприємств ще не задовольняють сучасним вимогам. Оператори ЕОМ, оператори підготовці даних, програмісти та інші працівники ОЦ та підприємств ще стикаються з впливом таких фізично небезпечних і шкідливих виробничих факторів, як підвищений рівень шуму, підвищена температура зовнішнього середовища, відсутність або недостатня освітленість робочої зони, електричний струм, статична електрика і інші.

Багато працівників ОЦ та підприємств пов'язані з впливом таких психофізичних факторів, як розумова перенапруга, перенапруження зорових і слухових аналізаторів, монотонність праці, емоційні перевантаження. Вплив зазначених несприятливих факторів призводить до зниження працездатності, викликане розвиваються втомою. Поява і розвиток втоми пов'язане зі змінами, які виникають під час роботи в центральній нервовій системі, з гальмівними процесами в корі головного мозку. Наприклад сильний шум викликає труднощі з розпізнаванням колірних сигналів, знижує швидкість сприйняття кольору, гостроту зору, зорову адаптацію, порушує сприйняття візуальної інформації, зменшує на 5 – 12% продуктивність праці. Тривала дія шуму з рівнем звукового тиску 90 дБ знижує продуктивність праці на 30 – 60%.

Медичні обстеження працівників ОЦ та підприємств показали, що крім зниження продуктивності праці високі рівні шуму призводять до погіршення

слуху. Тривале перебування людини в зоні комбінованого впливу різних несприятливих факторів може призвести до професійного захворювання. Аналіз травматизму серед працівників ВЦ показує, що в основному нещасні випадки відбуваються від впливу фізично небезпечних виробничих факторів при заправці носія інформації на обертовий барабан при знятому кожусі, під час співробітниками невластивих їм робіт. На другому місці випадки, пов'язані з дією електричного струму.

8.2 Аналіз умов праці на робочому місці ІТ-фахівця

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

– підвищений рівень шуму;

– несприятливі мікрокліматичні умови;

– недостатній рівень освітленості;

– шкідливі речовини;

– підвищений рівень електромагнітних випромінювань радіочастот;

– висока напруга електричної мережі;

– статична електрика та інші.

Робота програміста супроводжується також підвищеним ступенем напруженості трудового процесу. При систематичному впливі виробничих факторів, які не відповідають нормативним показникам, зростає рівень професійно зумовленої захворюваності працюючих та можуть виникнути професійні захворювання органів зору, руху, нервової системи. Таким чином, вивчення умов праці на робочому місці програміста є необхідною умовою запобігання негативних наслідків впливу небезпечних та шкідливих факторів. Робоче місце, добре пристосоване до трудової діяльності інженера, правильно і доцільно організоване, щодо простору, форми, розміру забезпечує йому зручне положення при роботі і високу продуктивність праці при найменшому фізичному

і психічному напруженні.

Нормування параметрів проводиться в залежності від періоду року та категорії важкості виконуваних робіт. Для постійних робочих місць, якими є робочі місця ІТ-фахівців, встановлені оптимальні параметри мікроклімату, а за неможливості їх дотримання використовують допустимі параметри. Робота ІТ-фахівця за важкістю відноситься до Іа (роботи, що виконуються сидячи і не потребують фізичного напруження) та Іб (роботи, що виконуються сидячи, стоячи або пов'язані з ходінням та супроводжуються деяким фізичним напруженням) категорій. В таблиці 8.1. наведені оптимальні параметри мікроклімату в приміщеннях.

Таблиця 8.1 – Параметри мікроклімату для приміщень з ПК

Період року	Параметр мікроклімату	Величина
Холодний	Температура повітря в приміщенні; відносна вологість; швидкість руху повітря	22...24°С; 40... 60%; до 0,1 м/с
Теплий	Температура повітря в приміщенні; відносна вологість; швидкість руху повітря	23...25 °С 40...60% 0,1...0,2 м/с

Виміряні за допомогою приладів температура та вологість у приміщеннях праці ІТ-фахівців повинні відповідати зазначеним у таблиці для теплого періоду року. Слід зазначити, що для нормалізації параметрів мікроклімату слід використовувати у приміщеннях кондиціювання повітря, або забезпечити подачу свіжого повітря системами вентиляції. Норми подачі свіжого повітря наведені у таблиці 8.2.

Таблиця 8.2 – Норми подачі свіжого повітря в приміщення

Характеристика приміщення	Об'ємна витрата свіжого повітря, що подається в приміщення, $м^3$ на одну людину в годину
Об'єм до 20 $м^3$ на людину 20... 40 $м^3$ на людину Більше 40 $м^3$на людину	Не менше 30 Не менше 20 Може біти використана природна вентиляція

Створення сприятливих умов праці і правильне естетичне оформлення робочих місць на виробництві має велике значення як для полегшення праці, так і для підвищення його привабливості, позитивно впливає на продуктивність праці. Забарвлення приміщень і меблів повинні сприяти створенню сприятливих умов для зорового сприйняття, гарного настрою. У службових приміщеннях, у яких виконується одноманітна розумова робота, що вимагає значної нервової напруги і великого зосередження, забарвлення повинно бути спокійних тонів – малонасичені відтінки холодного зеленого або блакитного кольорів.

При розробці оптимальних умов праці програміста необхідно враховувати освітленість. Раціональне освітлення робочого місця є одним з найважливіших факторів, що впливають на ефективність трудової діяльності людини, що попереджають травматизм і професійні захворювання. Правильно організоване освітлення створює сприятливі умови праці, підвищує працездатність і продуктивність праці. Освітлення на робочому місці програміста повинно бути таким, щоб працівник міг без напруги зору виконувати свою роботу. Стомлюваність органів зору залежить від ряду причин: недостатність освітленості; надмірна освітленість; неправильний напрям світла. Недостатність освітлення приводить до напруги зору, ослабляє увагу, приводить до настання передчасної стомленості. Надмірно яскраве освітлення викликає засліплення, роздратування і різь в очах. Неправильний напрямок світла на робочому місці

може створювати різкі тіні, відблиски, дезорієнтувати працюючого. Всі ці причини можуть призвести до нещасного випадку або профзахворювань. [2].

8.3 Пропозиції щодо підвищення працездатності ІТ-фахівців

Поява та впровадження нових інформаційно-комунікаційних технологій зумовлює необхідність подальшого вдосконалення охорони праці фахівців it-індустрії. Все це потребує розробки нових нормативно-правових актів з регламентації праці та відпочинку фахівців it-індустрії і стандартів підприємств, центрів комп'ютерної техніки, центрів інформаційних технологій, сучасних комп'ютерних класів. Для підвищення розумової працездатності то зорової роботи повинна здійснюватися ергономічна оптимізація в рамках системи «оператор-термінал», яка сприятиме результативній фізичній та інтелектуальний працездатності і відновленню психосоматичного здоров'я фахівців it-індустрії. Всі наведені заходи щодо вдосконалення охорони праці фахівців it-індустрії повинні контролюватися службою охорони праці та комісією з охорони праці підприємства. Особливе значення у соціальному захисті цієї категорії працівників належить прийняття комплексного договору, який може забезпечити фахівців додатковими пільгами та компенсаціями.

Для більшого розуміння, пропозиції щодо підвищення працездатності it-фахівців, розіб'ємо на декілька категорій:

1. Середовище і розпорядок праці. Для мінімізації негативних ефектів, що пов'язані з перевтомленням it-фахівців, потрібно чітко прописати і реалізувати графік періодів праці-відпочинку, щоб фахівець міг можливість переключити увагу, дати можливість відпочити очам, мозку, елементарно, встати розім'яти ноги. Також потрібно зробити максимально комфортними умови мікроклімату у офісному приміщенні, де працюють it-фахівці. Мається на увазі встановлення і експлуатація, коли виникає необхідність, кондиціонерів, опалення, та системи вентиляції, задля попередження перегрівання,

переохолодження it-фахівців, і подальшої неможливості ними виконувати свої функції. Також, за можливості, нами пропонується введення практики віддаленої праці it-фахівцями, якщо роботодавець не може забезпечити оптимальні і безпечні умови в офісному приміщенні, або якщо фахівця вони не влаштовують із певних причин.

2. Фізичні і психоемоційні чинники. Першим і найважливішим чинником, що впливає на працездатність it-фахівців є робоче місце, і саме тому, роботодавець має забезпечити максимальний його комфорт і безпеку. Гарантією цих факторів може слугувати сертифікація меблів, що використовуються на підприємстві it-галузі. Тому нами пропонується закупівля тільки меблів , які пошли сертифікацію на відповідність. Під психоемоційними чинниками ми розуміємо гарне самопочуття фахівців, позитивний настій, гарний психологічний клімат у колективі, тощо. Задля того, щоб психоемоційні чинники мали максимально позитивний ефект, керівництву слід поводити заходи, які сприятимуть укріпленню і покращенню міжособистісних стосунків у колективі, таких як психологічні тренінги, тімбілдінг, спортивні змагання і естафети. Також, сюди можна віднести розробку і впровадження системи мотивації працівників, як фінансової, так моральної і адміністративної.

8.4 Розробка заходів з умов поліпшення охорони праці

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

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

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значень.

Таким чином можна припустити, що основною причиною можливого зниження працездатності програміста є психофізіологічний фактор, тому основна

пропозиція буде така: дотримання позитивної психологічної атмосфери в колективі та регламентованого режиму праці та відпочинку, організація робочого місця з урахуванням ергономічних вимог.

Рекомендовані заходи: регулярні періодичні наочні огляди персоналом шляхів для евакуації людей із приміщення, відповідно до плану евакуації (який повинен розташовуватись на видному місці у приміщенні), включення до колективного договору мінімально можливого вмісту аптечок з обов'язковою наявністю масок-клапанів, або іншого спорядження для штучного дихання. Регулярна періодична перевірка параметрів заземлення та занулення (вимірювання опору ланцюга) [9].

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

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

8.5 Розрахункова частина

Система освітлення робочого місця користувача ПК має відповідати наступним вимогам (рис. 8.1).

Проведемо розрахунок штучного освітлення за методом коефіцієнта використання світлового потоку для приміщення ширина якого складає 3 м, довжина – 4,4 м, висота – 3 м.

У зазначеному приміщенні працює 2 людей.

Рисунок 8.1 – Вимоги до системи освітлення робочого місця користувача ПК

Для того, щоб визначити потрібну кількість світильників, які повинні забезпечити нормований рівень освітленості, визначимо світловий потік, що падає на робочу поверхню за формулою:

$$F=ESKZ/n,$$

де:

F – світловий потік, що розраховується, Лм;

E – нормована мінімальна освітленість, Лк; *E = 300 Лк;*

S – площа освітлюваного приміщення.

K – коефіцієнт запасу, що враховує зменшення світлового потоку лампи в результаті забруднення світильників в процесі експлуатації (його значення залежить від типу приміщення і характеру робіт, що проводяться в ньому, в нашому випадку *K = 1,5);*

Z – відношення середньої освітленості до мінімальної (зазвичай приймається рівним 1.1... 1.2, в нашому випадку *Z = 1,1*);

n – коефіцієнт використання світлового потоку, (відношення світлового потоку, що падає на розрахункову поверхню, до сумарного потоку всіх ламп і обчислюється в долях одиниці; залежить від характеристик світильника, розмірів приміщення, забарвлення стін і стелі, що характеризуються коефіцієнтами відбиття від стін **($\rho_{стін.}$)** і стелі **($\rho_{стелі}$),** значення коефіцієнтів дорівнюють $\rho_{стін}$ = 50% і $\rho_{стелі}$ = 50%.

Обчислимо індекс приміщення за формулою:

$$i=S/(h(A+B)),$$

де:

S – площа приміщення, *S = 13,2 м²;*

h – розрахункова висота підвісу, *h = 3 м. (співпадає з висотою* стелі, т.я. лампи освітлення закріплюються на стелі)*;*

A – ширина приміщення, *A = 3 м;*

B – довжина приміщення, *B = 4,4 м.*

Підставимо всі значення у формулу та визначимо індекса приміщення:

$$i=1,1.$$

Знаючи індекс приміщення, за знаходимо *n = 0,46 (з табличних даних коефіцієнтів використання світлового потоку (n) світильників з відповідним типом лампам) [8].* Підставимо всі значення у формулу, визначимо світловий потік: *F=14204 Лм.*

Для розрахунку будемо використовувати стельові *світлодіодні панелі Призма-72 6400К*, світловий потік яких $F_л$ = *7200* Лм.

Число ламп визначається по формулі:

$$N=F/F_л$$

де:

F – світловий потік,

$F_л$ – світловий потік однієї лампи.

Підставимо всі значення у формулу та визначимо індекса приміщення:

$$N= 14204 / 7200=1,9 \text{ шт.}$$

Приймаємо необхідну кількість *світлодіодних світильників* 2 шт.

8.6 Висновки до розділу

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

З цих міркувань було здійснено аналіз умов праці, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Виконано розрахунок штучного освітлення, як одного з ключових факторів впливу на працездатність та здоров'я програміста. Розроблено заходи з охорони праці.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи повідомлень електронної пошти.

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 повідомлень електронної пошти.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем повідомлень електронної пошти.

– Досліджена система повідомлень електронної пошти.

– На основі отриманих результатів досліджень створена програмна реалізація системи повідомлень електронної пошти.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання повідомлень електронної пошти.

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 10.4. Дана мова програмування дозволяє найбільш ефективно обробляти дані. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм MARS.

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 104 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 1,9 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Науменко І.О. Дослідження та програмна реалізація системи повідомлень електронної пошти // Збірник праць молодих науковців ЦНТУ. – Вип. 13. – Кропивницький: ЦНТУ, 2022.

2. Смірнов О.А. Дисперсійний аналіз мережного трафіку для забезпечення інформаційної безпеки телекомунікаційних систем / О.О. Кузнецов, О.А. Смірнов, Д.О. Даниленко // Інформаційна та економічна безпека: сучасний стан та тенденції розвитку : монографія за заг. ред. – Х.: ХІБС УБС НБУ – 2014 – С. 82-100.

3. Смірнов О.А. Дослідження методів виявлення вторгнень в телекомунікаційні системи та мережі / Д.О. Даниленко, О.А. Смірнов, Є.В. Мелешко // Системи озброєння і військова техніка. – Випуск 1(29) – Х.: ХУПС – 2012. – С. 92-100

4. Смирнов А.А. Метод обнаружения вредоносного программного обеспечения. Часть 1. Корреляционный анализ сетевого трафика // А.А.Смирнов, Д.А. Даниленко, Е.В.Мелешко // Науково-технічний журнал «Інформаційно-керуючі системи на залізничному транспорті» – Випуск 4(95). – Х.: УкрДАЗТ – 2012. – С. 8-14.

5. Смирнов А.А. Методы обнаружения вредоносного программного обеспечения в телекоммуникационных системах и сетях / Д.А. Даниленко // Збірник наукових праць "Системи обробки інформації". – Випуск 3(101) том 2. – Х.: ХУПС – 2012. – С. 152-155.

6. Смирнов А.А. Системы обнаружения и предотвращения вторжений для защиты телекоммуникационных сетей от вредоносного программного обеспечения / Д.А. Даниленко, А.А. Смирнов, А.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 1 (21) том 2. – Київ: ДП «ЦНДІНУ». – 2012. – С. 183-186.

7. Смирнов А.А. Системы обнаружения и предотвращения вторжений для защиты компьютерных сетей от вредоносного программного обеспечения / Д.А. Даниленко, А.А. Смирнов, И.Г. Кирилов // Збірник тез доповідей науково-практичної конференції «Застосування інформаційних технологій у підготовці та діяльності сил охорони правопорядку». м. Харків. 21-22 березня 2012 р. – Харків. АВВ МВС. – 2012. – С. 70-71.

8. Смірнов О.А. Дослідження методів виявлення вторгнень в телекомунікаційні мережі для підвищення інформаційної безпеки // Д.О. Даниленко // Збірник тез науково-практичної конференції «Захист інформації в інформаційно-комунікаційних системах». м. Київ. 24-27 квітня 2012 р. – Київ: НАУ. – 2012. – С. 22-25.

9. Смирнов А.А. Исследование систем обнаружения и предотвращения вторжений для защиты телекоммуникационных сетей от вредоносного программного обеспечения / Д.А. Даниленко // Збірник тез доповідей VIII наукової конференції «Новітні технології – для захисту повітряного простору». Харків. 18-19 квітня 2012 р. – м. Харків. ХУПС. – 2012. – С. 45.

10. Смирнов А.А. Исследование методов сигнатурного обнаружения вредоносного программного обеспечения в телекоммуникационных системах и сетях // Д.А. Даниленко // Збірник тез XIII міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування». м. Кіровоград. 13-14 квітня 2012 р. – Кіровоград: КНТУ. – 2012. – С. 43-45.

11. Смирнов А.А. Исследование методов проактивной защиты от вредоносного программного обеспечения в телекоммуникационных системах и сетях / Д.А. Даниленко // Збірник тез V міжнародної науково-практичної конференції «інтегровані інтелектуальні робототехнічні комплекси» (ІІРТК-2012). м. Київ. 15-16 травня 2012 р. – Київ: НАУ. – 2012. – С. 314-315.

12. Смирнов А.А. Метод обнаружения вредоносного программного обеспечения на основе корреляционного анализа сетевого трафика / Д.А. Даниленко // Матеріали XII всеукраїнської наукової інтернет-конференції

«Наукові дослідження: зв'язок теорії і практики». м. Тернопіль. 29-30 квітня 2012 р. – Тернопіль: ТНЕУ. – 2012. – С. 9-10.

13. Смирнов А.А. Метод детектирования вредоносного трафика в телекоммуникационных сетях на основе использования bds-тестирования / Д.А. Даниленко // Збірник тез V міжнародної науково-практичної конференції «Комп'ютерні системи та мережні технології» (CSNT-2012). м. Київ. 13-15 червня 2012 р. – Київ: НАУ. – 2012. – С. 121.

14. Смирнов А.А. Обнаружение и предотвращение вторжений в компьютерных сетях на основе статистического анализа сетевого трафика / А.А. Смирнов, Д.А. Даниленко // Збірник тез доповідей науково-практичної конференції «Застосування інформаційних технологій у підготовці та діяльності сил охорони правопорядку». м. Харків. 12-13 березня 2014 р. – Харків. АВВ МВС. – 2014. – С. 13-14.

15. Смірнов О.А. дисперсійний аналіз мережного трафіку для забезпечення інформаційної безпеки телекомунікаційних систем та мереж / О.А. Смірнов, Д.О. Даниленко // Збірник тез V Всеукраїнської науково-практичної конференції "Інформатика та системні науки". м. Полтава. 13-15 березня 2014 р. – Полтава: ПУЕТ. – 2014. – С. 289-291.

16. Смирнов А.А. Метод дисперсионного анализа сетевого трафика для обнаружения и предотвращения вторжений в телекоммуникационных системах и сетях/ А.А. Смирнов, Д.А. Даниленко // Збірник тез VI міжнародної науково-практичної конференції "Проблеми і перспективи розвитку IT-індустрії". м. Харків. 17-18 квітня 2014 р. – Харків: ХНЕУ. – 2014. – С. 258.

17. Смірнов О.А. метод забезпечення інформаційної безпеки телекомунікаційних систем з використанням дисперсійного аналізу мережного трафіку / О.А. Смірнов, Д.О. Даниленко // Збірник тез міжнародної науково-практичної конференції «Інформаційна та економічна безпека» (INFECO-2014)». м. Харків. 15-16 травня 2014 р. – Харків: ХІБС УБС НБУ. – 2014. – С. 135-139.

18. Девянин П.Н. Модели безопасности компьютерных систем /

П.Н. Девянин. – М.:Издательский центр «Академия», 2005. – 144 с.

19. Домарев В.В. Безопасность информационных технологий. Методология создания систем защиты / В.В. Домарев. – К.: ООО "ТИД "ДС", 2002 – 688 с.

20. ДСТУ ISO/IEC TR 13243-2003 Інформаційні технології. Посібник із методів та механізмів якості послуг / [Электронный ресурс]. – Режим доступа к ресурсу: http://document.ua/informaciini-tehnologiyi.-posibnik-iz-metodiv-ta-mehanizmiv--nor2718.html

21. ДСТУ В 3265 – 95. Зв'язок військовий. Терміни та визначення. – К.: УкрНДІССІ, 1995. – 23 с.

22. ДСТУ ISO 9000:2007 Системи управління якістю. Основні положення та словник термінів [Электронный ресурс]. – Режим доступа к ресурсу: http://document.ua/docs/tdoc14237.php

23. Ершов В.А.. Мультисервисные телекоммуникационные сети / В.А. Ершов, Н.А. Кузнецов. – М.: МГТУ им. Н.Э. Баумана, 2003. – 432 с.

24. Закон України «Про захист інформації в інформаційно-телекомунікаційних системах» [Электронный ресурс]. – Режим доступа к ресурсу: http://zakon4.rada.gov.ua/laws/show/2594-15

25. Информационная война и защита информации. Словарь основных терминов и определений [Электронный ресурс]. – Режим доступа к ресурсу: http://www.csef.ru/files/csef/articles/2176/2176.pdf

26. Казарин О.В. Безопасность программного обеспечения компьютерных систем / О.В. Казарин. – М.:МГУЛ, 2003. – 212 с.

27. Касперский Е. Компьютерное зловредство / Е. Касперский. – СПб.: Питер, 2007. – 208 с.

i. Касперский К. Техника сетевых атак. [Електронний ресурс]. – Режим доступу до ресурсу: http://rghost.ru/download/43730077/

28. 8e48b6263ce45c7dc2a65a7453383dc33b22486d/Крис%20Касперски%20-%20Техника%20сетевых%20атак.pdf

29. Касперский К. Техника и философия хакерских атак. / К. Касперский. – М.: Солон-Пресс, 2004 – 272 с.

30. Концепция (основы государственной политики) Национальной безопасности Украины, утвержденная постановлением Верховной Рады Украины от 16 января 1997 г. № 3/97-ВР. [Электронный ресурс]. – Режим доступа к ресурсу: http://search.ligazakon.ua/l_doc2.nsf/link1/ F970003.html

31. Конри-Мюррей Эндрю Выявление вторжений / «Журнал сетевых решений/LAN», № 01, 2001 [Электронный ресурс]. – Режим доступа к ресурсу: http://www.osp.ru/lan/2001/01/131532/

32. Кузнецов О.О. Протоколи захисту інформації у комп'ютерних системах та мережах / О.О. Кузнецов, С.Г. Семенов. – Х.: ХНУРЕ, 2009. – 184 с.

33. Кузнецов О.О. Методи обробки сигналів даних та зображень / О.О. Кузнецов, Г.А. Кучук, С.Г. Семенов. –Х.: НТУ «ХПІ», 2011. – 292 с.

34. Кузнецов А.А. Дисперсионный анализ сетевого трафика для обнаружения и предотвращения вторжений в телекоммуникационных системах и сетях / А.А. Кузнецов, А.А. Смирнов, Д.А. Даниленко // Збірник наукових праць "Системи обробки інформації". – Випуск 2(118). – Х.: ХУПС – 2014. – С. 81-85.

35. Лоскутов А.Ю. Основы теории сложных систем / А.Ю. Лоскутов, А.С. Михайлов. – М.-Ижевск: Институт компьютерных исследований, 2007. – 620 с.

36. Лукацкий А.В. Обнаружение атак / А.В. Лукацкий. – С.Пб.:BHV – Санкт – Петербург, 2003. – 596 с.

37. Методи підвищення оперативності передачі даних та захисту інформації у телекомунікаційній мережі: звіт про НДР (проміжний) / Наук. кер. О.А. Смірнов. – К.:КНТУ, 2013 № ДР 0112U006631

38. Методология функционального моделирования IDEF0. Руководящий документ [Электронный ресурс]. – Режим доступа к ресурсу: http://www.scribd.com/doc/76782197/МЕТОДОЛОГИЯ-ФУНКЦИОНАЛЬНОГО-МОДЕЛИРОВАНИЯ-IDEF0

39. Моделирование технических систем с AllFusion Process Modeler (ранее BPwin) [Электронный ресурс]. – Режим доступа к ресурсу:: http://www.sdteam.com/t5506

40. Назаров А.Н. Модели и методы расчета структурно-сетевых параметров сети АТМ / А.Н. Назаров. – М.: Горячая линия – Телеком, 2002. –255 с.

41. Наиболее опасная страна. Попытка заражения ПК. [Электронный ресурс]. – Режим доступа к ресурсу:: http://sooweb.ru/news/naibolee_opasnaja_strana_popytka_zarazhenija_pk/ 2012-01-23-526

42. НД ТЗІ Вимоги до захисту інформації WEB-сторінки від несанкціонованого доступу [Электронный ресурс]. – Режим доступа к ресурсу: http://www.csk.kfc.in.ua/documents/nakaz-web.doc

43. НД ТЗІ 2.5-004-99 Критерії оцінки захищеності інформації в комп'ютерних системах від несанкціонованого доступу. Затверджено наказом ДСТСЗІ СБ України від 28.04.1999р., №22. [Электронный ресурс]. – Режим доступа к ресурсу: http://do.gendocs.ru/docs/index-27508.html?page=8

44. Олифер В.Г. Компьютерные сети. Принципы, технологии, протоколы. 4-е изд. / В.Г. Олифер, Н.А. Олифер. – СПб.: Питер, 2012. – 943 с.

45. Поповский В.В. Защита информации в телекоммуникационных системах / В.В. Поповский, А.В. Персиков. – Х.: ООО "Компания СМИТ", 2006. Т2 – 292 с.

46. Постанова Кабінета Міністрів України від 29.03.2006 №373 «Про затвердження Правил забезпечення захисту інформації в інформаційних, телекомунікаційних та інформаційно-телекомунікаційних системах» [Электронный ресурс]. – Режим доступа к ресурсу: http://zakon2.rada.gov.ua/laws/show/373-2006-п.

47. Розробка методів підвищення оперативності передачі та захисту інформації у телекомунікаційних системах: звіт про НДР (проміжний) / Наук.

кер. О.А. Смірнов. – К.:КНТУ, 2013 № ДР 0113U003086

48. Розробка стеганографічних засобів вбудовування інформації в нерухливі та рухливі зображення: звіт про НДР (проміжний) / Наук. кер. О.А. Смірнов. – К.:КНТУ, 2013 № ДР 0112U002599

49. Розробка методів підвищення безпеки телекомунікаційних мереж: звіт про НДР (проміжний) / Наук. кер. О.А. Смірнов. – К.:КНТУ, 2013 № ДР 0112U006630

50. Связь военная. Термины и определения. ГОСТ В 23609-86 (СТ В СЭВ 0217-86). – Издание официальное. – М.: Изд-во стандартов, 1987.– 12 с.

51. Семенов С.Г. Модели и методы управления сетевыми ресурсами в информационно-телекоммуникационных системах: монография / С.Г. Семенов, О.А. Смирнов, Є.В. Мелешко. Х.: НТУ «ХПИ». – 2012. – 212 с.

52. Семенов С.Г. Безопасность операционных систем реального времени в автоматизированных системах управления технологическим процессом / С.Г. Семенов, С.Ю. Гавриленко, В.В. Давыдов // Авіаційно- космічна техніка і технологія. – Х.:НАКУ «ХАІ». – 2011. – Вип. 8(85). – С. 222-225

53. Семенов С.Г. Модели и методы распределения доступа и защиты данных в компьютеризированных информационно-измерительных управляючих системах критического применения: монография / С.Г. Семенов. – Х.:НТУ «ХПИ», 2013. – 360 с.

54. Семенов С.Г. Защита данных в компьютеризированных управляющих системах: монография / С.Г. Семенов, В.В. Давыдов, С.Ю. Гавриленко. – LAP Lambert Academic Publishing, 2014. – 237 с.

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-123.22.0017.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Науменко І.О.			Дослідження та програмна реалізація системи повідомлень електронної пошти	Літ.	Аркуш 1 / Аркушів 6
Перевірив		Смірнов С.А.				М	
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 повідомлень електронної пошти.

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 повідомлень електронної пошти.

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 повідомлень електронної пошти;

– 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 10.4.

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 повинні бути розглянуті пропозиції щодо підвищення працездатності ІТ-фахівців.

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 – 128 аркушів.

10 Етапи розробки

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

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

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

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 10.12.2022 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 21.12.2022 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

_____ Смірнов С.А.

Дослідження та програмна реалізація системи повідомлень електронної пошти

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 57

Літера: РП

Кропивницький – 2022 року

Основна програма

Файл SecurityMail.dpr основної програми

```
program SecurityMail;

uses
  Forms,
  frmMain in 'frmMain.pas' {frm_Main},
  DataModule in 'DataModule.pas' {DM: TDataModule},
  frmMailSettings in 'frmMailSettings.pas' {frm_MailSettings},
  frmSettings in 'frmSettings.pas' {frm_Settings},
  LetterStorage in 'LetterStorage.pas',
  frmNewLetter in 'frmNewLetter.pas' {frm_NewLetter},
  frmAskOnDuplicate in 'frmAskOnDuplicate.pas' {frm_AskOnDuplicate},
  about in 'about.pas' {frm_about},
  param_keys in 'param_keys.pas' {keys1},
  sha1 in 'sha1.pas',
  DES in 'DES.pas';

{$R *.res}

begin
  Application.Initialize;
  Application.Title := 'SecurityMail by SerSEr';
  Application.CreateForm(Tfrm_Main, frm_Main);
  Application.CreateForm(TDM, DM);
  Application.CreateForm(Tfrm_MailSettings, frm_MailSettings);
  Application.CreateForm(Tfrm_Settings, frm_Settings);
  Application.CreateForm(Tfrm_NewLetter, frm_NewLetter);
  Application.CreateForm(Tfrm_AskOnDuplicate, frm_AskOnDuplicate);
  Application.CreateForm(Tfrm_about, frm_about);
  Application.CreateForm(Tkeys1, keys1);
  Application.HintPause:=200;
  Application.Run;
end.
```

Файл frmMain.pas - головні вікно програми

```
unit frmMain;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Menus, ToolWin, ComCtrls, ExtCtrls, StdCtrls, XPMan,
  IdBaseComponent, IdComponent, IdTCPConnection, IdTCPClient,
  IdMessageClient, IdPOP3, ActnList, OleCtrls, SHDocVw, about, param_keys,
  CoolTrayIcon,  LetterStorage,frmMailSettings;
  {HTTPApp, {HTTPProd,}


type
  Tfrm_Main = class(TForm)
    MainMenu: TMainMenu;
    Splitter1: TSplitter;
    Splitter2: TSplitter;
    N1: TMenuItem;
    N2: TMenuItem;
    N3: TMenuItem;
    N4: TMenuItem;
    N5: TMenuItem;
    CoolBar: TCoolBar;
    Tree_MailBoxes: TTreeView;
    ListView_LettersHeaders: TListView;
    ToolBar1: TToolBar;
    btn_CheckMail: TToolButton;
    ToolButton2: TToolButton;
    ToolButton3: TToolButton;
    XPManifest1: TXPManifest;
    Panel_Content: TPanel;
    N6: TMenuItem;
    N7: TMenuItem;
    N8: TMenuItem;
    N9: TMenuItem;
    N10: TMenuItem;
    ActionList1: TActionList;
    aCheckMail: TAction;
    aSendMail: TAction;
    aMailSettings: TAction;
    aExit: TAction;
    aNewLetter: TAction;
    ToolButton7: TToolButton;
    MemoLog: TMemo;
    Splitter3: TSplitter;
    aSettings: TAction;
    Web_LetterViewer: TWebBrowser;
    Splitter4: TSplitter;
    List_Attachmant: TListView;
    Memo_LetterBody: TMemo;
    Popup_Attachment: TPopupMenu;
    mnuSaveAtt: TMenuItem;
    SaveDialog_Att: TSaveDialog;
    PopupMenuHeaders: TPopupMenu;
    mnu_DeleteLetter: TMenuItem;
    mnu_Answer: TMenuItem;
    mnu_Resend: TMenuItem;
    N12: TMenuItem;
    aAnswer: TAction;
    aResend: TAction;
    Tray: TCoolTrayIcon;
    PopupMenu_MailBoxes: TPopupMenu;
    N13: TMenuItem;
```

```
    Jnghfdbnmgjxne1: TMenuItem;
    N14: TMenuItem;
    About1: TMenuItem;
    N15: TMenuItem;
    SMIME1: TMenuItem;
    N16: TMenuItem;
    N17: TMenuItem;
    N18: TMenuItem;
    N19: TMenuItem;
    ToolButton1: TToolButton;
    ToolButton10: TToolButton;
    ToolButton12: TToolButton;
    ToolButton13: TToolButton;
    ToolButton14: TToolButton;
    ToolButton4: TToolButton;
    ToolButton5: TToolButton;
    ToolButton6: TToolButton;
    N11: TMenuItem;
    N20: TMenuItem;
    procedure FormShow(Sender: TObject);
    procedure aNewLetterExecute(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Tree_MailBoxesClick(Sender: TObject);
    procedure aCheckMailExecute(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure List_AttachmantClick(Sender: TObject);
    procedure Web_LetterViewerBeforeNavigate2(Sender: TObject; const pDisp:
IDispatch; var URL, Flags, TargetFrameName, PostData, Headers: OleVariant; var
Cancel: WordBool);
    procedure ListView_LettersHeadersSelectItem(Sender: TObject; Item:
TListItem; Selected: Boolean);
    procedure List_AttachmantMouseDown(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; X, Y: Integer);
    procedure mnuSaveAttClick(Sender: TObject);
    procedure Popup_AttachmentPopup(Sender: TObject);
    procedure mnu_DeleteLetterClick(Sender: TObject);
    procedure PopupMenuHeadersPopup(Sender: TObject);
    procedure aAnswerExecute(Sender: TObject);
    procedure aResendExecute(Sender: TObject);
    procedure Tree_MailBoxesDragDrop(Sender, Source: TObject; X, Y: Integer);
    procedure Tree_MailBoxesDragOver(Sender, Source: TObject; X, Y: Integer;
State: TDragState; var Accept: Boolean);
    procedure TrayClick(Sender: TObject);
    procedure TrayMinimizeToTray(Sender: TObject);
    procedure N13Click(Sender: TObject);
    procedure PopupMenu_MailBoxesPopup(Sender: TObject);
    procedure Jnghfdbnmgjxne1Click(Sender: TObject);
    procedure N15Click(Sender: TObject);
    procedure N19Click(Sender: TObject);
    procedure ToolButton10Click(Sender: TObject);
    procedure ToolButton14Click(Sender: TObject);
    procedure ToolButton12Click(Sender: TObject);
    procedure ToolButton6Click(Sender: TObject);
    procedure ToolButton4Click(Sender: TObject);
    procedure N18Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
    procedure ShowMailBoxesInTree;
    procedure LoadLetters;
    procedure AddToLog(s : shortstring);
    procedure HideLetterBody;
    procedure ShowInHtmlViewer(body : TStrings);
  end;


const
    LogFileName   = 'Actions.log';
```

```
var
  frm_Main: Tfrm_Main;

  MailInfoLoaded  : boolean=false;

  { список поштових повідомлень }
  MailBoxes       : TMailBoxList;


function  ExecuteFile(const FileName, Params, DefaultDir: string; ShowCmd:
Integer): THandle;
procedure FillPOPwithMailBoxSettings(mbSettings : PMailBoxSettings);

implementation

uses shellapi, frmSettings, IdMessage,
     DataModule, frmNewLetter,
     ComObj, frmAskOnDuplicate, IdEMailAddress;

{$R *.dfm}

procedure Tfrm_Main.FormShow(Sender: TObject);
{var
   IE: Variant;
begin
IE := CreateOleObject('InternetExplorer.Application');
        IE.Navigate('C:\MyHTML.html');
        While IE.Busy do begin end;
        Memo1.Text:= IE.Document.Body.innerText;
{end; }
begin
  if not MailInfoLoaded then
  begin
    { показує поштову скриньку }

    ShowMailBoxesInTree;
    LoadLetters;
    MailInfoLoaded:=true;

    Memo_LetterBody.Visible:=false;
    Web_LetterViewer.Visible:=false;
    if Settings.ShowInTray then
       begin
       Tray.IconVisible:=true;
       Tray.HideTaskbarIcon;
       frm_Main.Tray.MinimizeToTray:=true;
       end
       else
       begin
       Tray.IconVisible:=false;
       Tray.ShowMainForm;
       frm_Main.Tray.MinimizeToTray:=false;
       end;
  end;
  Application.MessageBox('','');
end;

procedure Tfrm_Main.ShowMailBoxesInTree;
var
    i     : word;
    node  : TTreeNode;
begin
  Tree_MailBoxes.Items.Clear;
  for i:=1 to TotalMailBoxes do
  begin
    node:=Tree_MailBoxes.Items.Add(nil,MailBoxesSettings[i].Name);
    node.ImageIndex:=0;
```

```
    Tree_MailBoxes.Items.AddChild(node,'Вхідні').ImageIndex:=1;
    Tree_MailBoxes.Items.AddChild(node,'Вихідні').ImageIndex:=2;
    Tree_MailBoxes.Items.AddChild(node,'Відправлені').ImageIndex:=2;
    Tree_MailBoxes.Items.AddChild(node,'Видалені').ImageIndex:=3;

    AddToLog('Завантажені параметри скриньки <'+MailBoxesSettings[i].Name+'>
');
  end;
  AddToLog('Всього облікових записів : '+inttostr(TotalMailBoxes)+'   ');
end;


procedure Tfrm_Main.LoadLetters;
var
    i : word;
Begin
  MailBoxes.TotalBoxes:=0;
  for i:=1 to TotalMailBoxes do
  begin
    MailBoxes.Add(MailBoxesSettings[i].Name);
    AddToLog('Завантажені листи для скриньки <'+MailBoxesSettings[i].Name+'>
');
  end;
End;


procedure Tfrm_Main.aNewLetterExecute(Sender: TObject);
var
  MailBoxName : shortstring;
begin
  frm_NewLetter.ClearForm;
  frm_NewLetter.Show;

  if Tree_MailBoxes.Selected=nil then
    begin
      Application.MessageBox('Ви повинні виділити якусь скриньку, щоб відправити
з неї листа','З якої скриньки відправляти?');
      exit;
    end;

  if Tree_MailBoxes.Selected.Parent = nil then
MailBoxName:=Tree_MailBoxes.Selected.Text
     else
     MailBoxName:=Tree_MailBoxes.Selected.Parent.Text;
  frm_NewLetter.Caption:='Новий лист  <'+MailBoxName+'>';
end;


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


procedure Tfrm_Main.AddToLog(s  : shortstring);
var
    st    :   TFileStream;
    SLog  :   shortstring;
    Mess  :   shortstring;
    mem   :   TMemoryStream;
Begin
  if (s[1]=#10)and(s[2]=#13) then
     begin
     delete(s,1,2);
```

```
     Mess:=#10#13+DateTimeToStr(Now)+' --> '+s+#10#13;
     end
     else Mess:=DateTimeToStr(Now)+' --> '+s+#10#13;

  MemoLog.Lines.Add(Mess);

  if Settings.AllowLog then
  begin
    SLog:=ExtractFilePath(Application.ExeName)+LogFileName;
    try
      st:=TFileStream.Create(SLog,fmOpenReadWrite or fmShareDenyNone);
    except
      try
        st:=TFileStream.Create(SLog,fmCreate);
      except
        Application.MessageBox( pchar('Не можу відкрити файл "'+SLog+'" для
запису !'),'Помилка збереження логу файлу');
        System.exit;
      end;
    end;

    if (Settings.MaxLogSize<>0)and(St.Size > Settings.MaxLogSize * 1024) then
        try
          mem:=TMemoryStream.Create;
          mem.Size:=Settings.MaxLogSize*1024;
          mem.seek(0,soFromBeginning);

          st.Seek(-Settings.MaxLogSize*1024,soFromEnd);
          mem.CopyFrom(st,Settings.MaxLogSize*1024);
          st.Seek(0,soFromBeginning);
          mem.Seek(0,soFromBeginning);
          st.CopyFrom(mem,Settings.MaxLogSize*1024);
          st.Size:=Settings.MaxLogSize*1024;
        finally mem.Free; end;

    st.Seek(0,soFromEnd);
    st.Write(Mess[1],length(Mess));
    st.Free;
  end;
End;


procedure DeleteFiles(s : shortstring);
var
  sr  : TSearchRec;
Begin
  try
    findfirst(s, faAnyFile, sr);
    deletefile( IncludeTrailingPathDelimiter(ExtractFilePath(s))+sr.Name );
    while findnext(sr)=0 do deletefile(
IncludeTrailingPathDelimiter(ExtractFilePath(s))+sr.Name );
  except end;
End;


procedure Tfrm_Main.FormCreate(Sender: TObject);
begin
frm_Settings.LoadGlobalSettingsFromFile;
DeleteFiles(
IncludeTrailingPathDelimiter(ExtractFilePath(Application.ExeName))+'temp_*.html'
);

AddToLog('Запуск системи обміну електронними повідомленнями (S/MIME) на основі
технології PKI');
end;

procedure Tfrm_Main.Tree_MailBoxesClick(Sender: TObject);
var
    MailBoxName     : shortstring;
```

```
    MailBoxFolder   : shortstring;
    LetterList      : PLetters;
    i               : word;
begin
  if Tree_MailBoxes.Selected = nil then
  begin
    HideLetterBody;
    exit;
  end;
  List_Attachmant.Clear;
  List_Attachmant.Visible:=false;
  Memo_LetterBody.Clear;
  if Tree_MailBoxes.Selected.Parent = nil then
  begin
    if MailBoxes.FindBox(Tree_MailBoxes.Selected.Text).IsBuisyNow then
aCheckMail.Enabled:=false
                                                                  else
aCheckMail.Enabled:=true;
    ListView_LettersHeaders.Clear;
    HideLetterBody;
    findMailBoxAndSelectIt(Tree_MailBoxes.Selected.Text);
    exit;
  end;

  MailBoxName   :=  Tree_MailBoxes.Selected.Parent.Text;
  MailBoxFolder :=  Tree_MailBoxes.Selected.Text;

  LetterList:=MailBoxes.FindLetters(MailBoxName,MailBoxFolder);

  ListView_LettersHeaders.Items.Clear;
  if LetterList = nil then
  begin
    Panel_Content.Caption:='Немає такої папки.'+
                           ' Або вона була завантажена з помилкою. ';
    exit;
  end;

  if MailBoxes.FindBox(MailBoxName).IsBuisyNow then aCheckMail.Enabled:=false
                                             else aCheckMail.Enabled:=true;
  findMailBoxAndSelectIt(MailBoxName);

  for i:=1 to LetterList^.Count do
  with ListView_LettersHeaders.Items.Add do
  begin
    Caption:=LetterList^.Letters[i]^.Subject;
    if LetterList^.Letters[i]^.Name='' then
SubItems.Add(LetterList^.Letters[i]^.From.Address)
                                       else
SubItems.Add(LetterList^.Letters[i]^.From.Name);
    SubItems.Add(DateTimeToStr(LetterList^.Letters[i]^.Date));
    //розмір
    SubItems.Add( inttostr(LetterList^.LettersSizes[i]) );
    case LetterList^.Letters[i]^.Priority of
      mpHighest : SubItems.Add('Дуже високий');
      mpHigh    : SubItems.Add('Високий');
      mpNormal  : SubItems.Add('Звичайний');
      mpLow     : SubItems.Add('Низький');
      mpLowest  : SubItems.Add('Дуже низький');
      end;
  end;

end;


procedure FillPOPwithMailBoxSettings(mbSettings : PMailBoxSettings);
Begin
  DM.POP.Host :=mbSettings.POPServer;
  DM.POP.Port :=mbSettings.POPPort;
  DM.POP.Username :=mbSettings.POPAccount;
```

```
  DM.POP.Password :=mbSettings.POPPass;
End;

(*
procedure RetrieveMail(let_count  : word);
var
    mes : TIdMessage;
    i   : word;
    LettersNot2beDownloaded : word;
Begin
  try
  LettersNot2beDownloaded:=0;
  for i:=1 to let_count do
    if CurrentMailBoxSettings^.RetrieveOnlyHeaders then
    // приймає тільки заголовки
    begin
    mes:=TIdMessage.Create(frm_Main);
    DM.POP.RetrieveHeader(i,mes);
    end
    else
    // приймає наступні листи
    begin
      mes:=TIdMessage.Create(frm_Main);
      DM.POP.RetrieveHeader(i,mes);
      if
MailBoxes.GetByName(CurrentMailBoxSettings^.Name).HasLetterWithID(mes.MsgId)
then
      begin
        inc(LettersNot2beDownloaded);
        continue;
      end;
      if
MailBoxes.GetByName(CurrentMailBoxSettings^.Name).hadLetterWithId(mes.msgId)
then
      begin
        case CurrentMailBoxSettings.ifDuplicate of
          Ignore  :  continue;
          Recieve :  ;
          Ask     :  begin
                     frm_AskOnDuplicate.FillInInfo(@mes);
                     frm_AskOnDuplicate.ShowModal;
                     case frm_AskOnDuplicate.ModalResult of
                      mrIgnore : continue;   // не приймає
                      mrOk     : ;           // приймає
                      mrAbort  : begin       // видаляє з серверу
                                 if not DM.POP.Delete(i) then
frm_Main.AddToLog('Помилка видалення листа як отриманого раніше, алі вилученого
з комп'ютера(див. настроювання) Від: : "'+mes.From.Address+'" Тема:
"'+mes.Subject+'"');
                                 continue;
                                 end;
                     end;

                   end;
          DeleteFromServer  : begin
                              if not DM.POP.Delete(i) then
frm_Main.AddToLog('Помилка видалення листа як отриманого раніше, алі вилученого
з комп'ютера(див. настроювання) Від: "'+mes.From.Address+'" Тема:
"'+mes.Subject+'"');
                              continue;
                              end;
          end;
      end;

      frm_Main.AddToLog('Отримання листа
('+inttostr(i)+'/'+inttostr(let_count)+') від '+mes.From.Text+' по Темі
:"'+mes.Subject+'"');
      DM.POP.Retrieve(i,mes);
```

```
      MailBoxes.GetByName(CurrentMailBoxSettings^.Name).Inbox.Add(@mes,false,
DM.POP.RetrieveMsgSize(i));
      if CurrentMailBoxSettings^.DeleteFromServer then
         if not DM.POP.Delete(i) then frm_Main.AddToLog(' Помилка видалення
листа із сервера. Від: "'+mes.From.Address+'" Тема: "'+mes.Subject+'"');
    end;
  finally {mes.Free;} end;

  if LettersNot2beDownloaded<>0 then
     frm_Main.AddToLog('На сервері лежить(ат)
'+inttostr(LettersNot2beDownloaded)+' лист(и), які не були завантажені тому що
вони були завантажені раніше й лежать у папках поточного ящика');

End;
*)

procedure Tfrm_Main.aCheckMailExecute(Sender: TObject);
var
    CurMailBox    :   PMailBox;
begin
  // перевірка поганих листів (фішинг)
  Tree_MailBoxesClick(self);
  if CurrentMailBoxSettings =nil then exit;

  aCheckMail.Enabled:=false;
  CurMailBox:=MailBoxes.GetByName(CurrentMailBoxSettings^.Name);
  if CurMailBox.IsBuisyNow then exit;
  try
    //ініціюється процес перевірки листів
    CurMailBox.onTimer_Check(self);
  except
    on e:exception do Application.MessageBox(pchar(E.Message),'Помилка при
запуску створення перевірки скриньки');
  end;

end;

procedure Tfrm_Main.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  MailBoxes.SaveMailBoxes;
end;

procedure Tfrm_Main.HideLetterBody;
begin
  Memo_LetterBody.Hide;
  Web_LetterViewer.Hide;
end;

procedure Tfrm_Main.List_AttachmantClick(Sender: TObject);
var
    MailBoxName     : shortstring;
    MailBoxFolder   : shortstring;
    LetterList      : PLetters;
begin
  if Tree_MailBoxes.Selected = nil then exit;
  if Tree_MailBoxes.Selected.Parent = nil then
  begin
    findMailBoxAndSelectIt(Tree_MailBoxes.Selected.Text);
    exit;
  end;
  if (ListView_LettersHeaders.Selected=nil)or
     (List_Attachmant.Selected=nil)  then exit;

  MailBoxName   :=  Tree_MailBoxes.Selected.Parent.Text;
  MailBoxFolder :=  Tree_MailBoxes.Selected.Text;

  LetterList:=MailBoxes.FindLetters(MailBoxName,MailBoxFolder);
```

```
  if
LetterList.Letters[ListView_LettersHeaders.ItemIndex+1]^.MessageParts.Items[List
_Attachmant.ItemIndex].ContentType = 'text/plain' then
     // text/plain
     begin
     Web_LetterViewer.Hide;
     Memo_LetterBody.Clear;

Memo_LetterBody.Lines.AddStrings(TIdText(LetterList^.Letters[ListView_LettersHea
ders.ItemIndex+1]^.MessageParts.Items[List_Attachmant.ItemIndex]).Body);
     Memo_LetterBody.Show;
     end;
  if
LetterList.Letters[ListView_LettersHeaders.ItemIndex+1]^.MessageParts.Items[List
_Attachmant.ItemIndex].ContentType = 'text/html' then
     // text/html
     begin
     Memo_LetterBody.Hide;

ShowInHtmlViewer(TIdText(LetterList^.Letters[ListView_LettersHeaders.ItemIndex+1
]^.MessageParts.Items[List_Attachmant.ItemIndex]).Body);
     Web_LetterViewer.Show;
     end;
end;


// Вхід: рядок у HTML коді   Вихід: в Web_LetterViewer
procedure Tfrm_Main.ShowInHtmlViewer(body: TStrings);
var
    tempFileName  : shortstring;
    st  :  TFileStream;
    i   :  integer;
    s   :  string;
begin
  if not CurrentMailBoxSettings^.OpenHTMLasTEXT then
  BEGIN
    try
       randomize;

tempFileName:=IncludeTrailingPathDelimiter(ExtractFilePath(Application.ExeName))
+'temp_'+inttostr(random(999))+'.html';
       deletefile(tempFileName);
       st:=TFileStream.Create(tempFileName,fmCreate);
    except
       on E:Exception do Application.MessageBox(pchar('Тимчасовий файл
"'+tempFileName+'" не може бути створений :=> '+e.message),'Помилка створення
тимчасового файлу');
    end;

    st.Seek(0,soFromBeginning);
    for i:=0 to body.Count-1 do
    begin
    s:=body.Strings[i]+#10#13;
    st.Write(s[1], length(s) );
    end;
    st.Free;

    Web_LetterViewer.Navigate('файл://'+tempFileName);
  END
  ELSE
  BEGIN
     Web_LetterViewer.Hide;
     Memo_LetterBody.Clear;
     Memo_LetterBody.Lines.AddStrings(Body);
     Memo_LetterBody.Show;
  END;
end;
```

```
procedure Tfrm_Main.Web_LetterViewerBeforeNavigate2(Sender: TObject;
  const pDisp: IDispatch; var URL, Flags, TargetFrameName, PostData,
  Headers: OleVariant; var Cancel: WordBool);
begin
  try
    if (pos('temp_',URL)=0)and(URL[2]<>':') then
    Cancel:=true;
  except end;
end;


procedure Tfrm_Main.ListView_LettersHeadersSelectItem(Sender: TObject;
  Item: TListItem; Selected: Boolean);
var
    MailBoxName     : shortstring;
    MailBoxFolder   : shortstring;
    LetterList      : PLetters;
    intIndex        : integer;
    BodyPart        : word;
begin
  if Tree_MailBoxes.Selected = nil then exit;
  if Tree_MailBoxes.Selected.Parent = nil then
  begin
    findMailBoxAndSelectIt(Tree_MailBoxes.Selected.Text);
    exit;
  end;
  if ListView_LettersHeaders.Selected=nil then exit;

  MailBoxName   :=  Tree_MailBoxes.Selected.Parent.Text;
  MailBoxFolder :=  Tree_MailBoxes.Selected.Text;

  LetterList:=MailBoxes.FindLetters(MailBoxName,MailBoxFolder);
  //перелік листів.

  if pos('TEXT/HTML' ,
ANSIUPPERCASE(LetterList^.Letters[ListView_LettersHeaders.ItemIndex+1]^.ContentT
ype))<>0 then
  begin
  Memo_LetterBody.Hide;

ShowInHtmlViewer(LetterList^.Letters[ListView_LettersHeaders.ItemIndex+1]^.Body)
;
  Web_LetterViewer.Show;
  end
  else
  begin

Memo_LetterBody.Lines:=LetterList^.Letters[ListView_LettersHeaders.ItemIndex+1]^
.Body;
  Memo_LetterBody.visible:=true;
  end;

  BodyPart:=0;
  List_Attachmant.Items.Clear;
    for intIndex := 0 to
Pred(LetterList^.Letters[ListView_LettersHeaders.ItemIndex+1]^.MessageParts.Coun
t) do
      begin
         if
(LetterList^.Letters[ListView_LettersHeaders.ItemIndex+1]^.MessageParts.Items[in
tIndex] is TIdAttachment) then
            begin //головне приєднання
               List_Attachmant.visible := true;
               with List_Attachmant.Items.Add do
               begin
                StateIndex:=0;
                Caption :=
TIdAttachment(LetterList^.Letters[ListView_LettersHeaders.ItemIndex+1]^.MessageP
arts.Items[intIndex]).Filename;
```

```
SubItems.Add(TIdAttachment(LetterList^.Letters[ListView_LettersHeaders.ItemIndex
+1]^.MessageParts.Items[intIndex]).ContentType);
               end;
            end
         else
            begin //body text
               if
LetterList^.Letters[ListView_LettersHeaders.ItemIndex+1]^.MessageParts.Items[int
Index] is TIdText then
                  begin
                    List_Attachmant.visible := true;
                    inc(BodyPart);
                    with List_Attachmant.Items.Add do
                    begin
                      StateIndex:=1;
                      if
AnsiUppercase(LetterList^.Letters[ListView_LettersHeaders.ItemIndex+1]^.MessageP
arts.Items[intIndex].ContentType) = 'TEXT/PLAIN'
                         then Caption:='Частина #'+inttostr(BodyPart)
                         else
                         If
AnsiUppercase(LetterList^.Letters[ListView_LettersHeaders.ItemIndex+1]^.MessageP
arts.Items[intIndex].ContentType) = 'TEXT/HTML'
                            then Caption:='HTML #'+inttostr(Bodypart)
                            else
Caption:=LetterList^.Letters[ListView_LettersHeaders.ItemIndex+1]^.MessageParts.
Items[intIndex].ContentType;
                         if
LetterList^.Letters[ListView_LettersHeaders.ItemIndex+1]^.Body.Count>0 then
                         if
(AnsiUppercase(LetterList^.Letters[ListView_LettersHeaders.ItemIndex+1]^.Body.St
rings[0])=AnsiUppercase('Це повідомлення у форматі S-MIME') )
                         and

(AnsiUppercase(LetterList^.Letters[ListView_LettersHeaders.ItemIndex+1]^.Message
Parts.Items[intIndex].ContentType) = 'TEXT/HTML')
                         then begin
                              Selected:=true;
                              List_AttachmantClick(self);
                              end;
                    end;
                    //Memo_LetterBody.Visible:=false;
                    //Memo_LetterBody.Lines.Add('');
                    //Memo_LetterBody.Lines.Add('Лист. Частина
#'+inttostr(BodyPart));
                    //Memo_LetterBody.Lines.Add('');
                    //Memo_LetterBody.Lines.AddStrings (
TIdText(LetterList^.Letters[ListView_LettersHeaders.ItemIndex+1]^.MessageParts.I
tems[intIndex]).Body );
                    //Memo_LetterBody.Visible:=true;
                  end
            end;
      end;
      if List_Attachmant.Items.Count=0 then List_Attachmant.Visible:=false;
end;

procedure Tfrm_Main.List_AttachmantMouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var
    it  : TListItem;
begin
if Button = mbRight then
  begin
  it:=List_Attachmant.GetItemAt(x,y);
  if it=nil then exit;
  it.Selected:=true;
  end;
end;
```

```
const
  invalid :set of char=['\','/',':','?','*','"','<','>','|'];

function isValidFileName(s:string):boolean;
var
    i : word;
Begin
  isValidFileName:=false;
  for i:=1 to length(s) do
      if s[i] in invalid then exit;
  isValidFileName:=true;
End;


procedure Tfrm_Main.mnuSaveAttClick(Sender: TObject);
var
    MailBoxName     : shortstring;
    MailBoxFolder   : shortstring;
    LetterList      : PLetters;
Begin
  if Tree_MailBoxes.Selected = nil then exit;
  if Tree_MailBoxes.Selected.Parent = nil then
  begin
    findMailBoxAndSelectIt(Tree_MailBoxes.Selected.Text);
    exit;
  end;
  if (ListView_LettersHeaders.Selected=nil)or
     (List_Attachmant.Selected=nil)  then exit;


  MailBoxName   :=  Tree_MailBoxes.Selected.Parent.Text;
  MailBoxFolder :=  Tree_MailBoxes.Selected.Text;

  LetterList:=MailBoxes.FindLetters(MailBoxName,MailBoxFolder);

  if
LetterList.Letters[ListView_LettersHeaders.ItemIndex+1]^.MessageParts.Items[List
_Attachmant.ItemIndex] is TIdAttachment then
  begin
    SaveDialog_Att.FileName:=List_Attachmant.selected.Caption;
    if not isValidFileName(SaveDialog_Att.FileName) then
SaveDialog_Att.FileName:='Приєднання '+inttostr(random(99))+'.txt';
    if not SaveDialog_Att.Execute then exit;

TIdAttachment(LetterList.Letters[ListView_LettersHeaders.ItemIndex+1]^.MessagePa
rts.Items[List_Attachmant.ItemIndex]).SaveToFile(SaveDialog_Att.FileName);
  end else Application.MessageBox('Данна частина листа не являється
додатком!','Так неможливо');

End;


procedure Tfrm_Main.Popup_AttachmentPopup(Sender: TObject);
begin
  if List_Attachmant.Selected = nil then mnuSaveAtt.Enabled:=false
                                    else mnuSaveAtt.Enabled:=true;
end;

procedure Tfrm_Main.mnu_DeleteLetterClick(Sender: TObject);
var
    MailBoxName,MailBoxFolder : shortstring;
    LetterList                : PLetters;
begin
  if ( Tree_MailBoxes.Selected=nil ) or
     ( Tree_MailBoxes.Selected.Parent = nil ) then exit;
  if ListView_LettersHeaders.Selected = nil then exit;

  MailBoxName   :=  Tree_MailBoxes.Selected.Parent.Text;
  MailBoxFolder :=  Tree_MailBoxes.Selected.Text;
```

```
  LetterList:=MailBoxes.FindLetters(MailBoxName,MailBoxFolder);

MailBoxes.FindBox(MailBoxName).MoveToGarbage(LetterList,ListView_LettersHeaders.
ItemIndex+1);
  ListView_LettersHeaders.DeleteSelected;
end;

procedure Tfrm_Main.PopupMenuHeadersPopup(Sender: TObject);
var
    i : word;
begin
  if ListView_LettersHeaders.Selected=nil then
     for i:=0 to PopupMenuHeaders.Items.Count-1 do
        PopupMenuHeaders.Items[i].Enabled:=false
  else for i:=0 to PopupMenuHeaders.Items.Count-1 do
        PopupMenuHeaders.Items[i].Enabled:=true;
end;

procedure Tfrm_Main.aAnswerExecute(Sender: TObject);
var
    MailBoxName,MailBoxFolder : shortstring;
    LetterList                : PLetters;
begin
  MailBoxName   :=  Tree_MailBoxes.Selected.Parent.Text;
  MailBoxFolder :=  Tree_MailBoxes.Selected.Text;

  LetterList:=MailBoxes.FindLetters(MailBoxName,MailBoxFolder);

  frm_NewLetter.ClearForm;

frm_NewLetter.Ed_Reciever.Text:=LetterList.Letters[ListView_LettersHeaders.itemi
ndex+1].From.Text;
  frm_NewLetter.Ed_Subject.Text:='Re:
'+LetterList.Letters[ListView_LettersHeaders.itemindex+1].Subject;
  frm_NewLetter.Caption:='Новий лист з <'+MailBoxName+'> [відповідь]';
  frm_NewLetter.Show;
end;

procedure Tfrm_Main.aResendExecute(Sender: TObject);
var
    MailBoxName,MailBoxFolder : shortstring;
    LetterList                : PLetters;
    i                         : integer;
begin
  MailBoxName   :=  Tree_MailBoxes.Selected.Parent.Text;
  MailBoxFolder :=  Tree_MailBoxes.Selected.Text;

  LetterList:=MailBoxes.FindLetters(MailBoxName,MailBoxFolder);

  frm_NewLetter.ClearForm;
  frm_NewLetter.Ed_Subject.Text:='Fw:
'+LetterList.Letters[ListView_LettersHeaders.itemindex+1].Subject;
  frm_NewLetter.Caption:='Новий лист з <'+MailBoxName+'> [пересільця]';
  frm_NewLetter.Show;

  frm_NewLetter.Memo_Body.Lines.Add('< -- Пересилаємий лист (початок) -->');
  for i:=0 to
LetterList.Letters[ListView_LettersHeaders.itemindex+1].Body.Count-1 do
              frm_NewLetter.Memo_Body.Lines.Add('>
'+LetterList.Letters[ListView_LettersHeaders.itemindex+1]^.Body.Strings[i]);
  frm_NewLetter.Memo_Body.Lines.Add('< -- Пересилаємий лист (кінець) -->');

end;


procedure Tfrm_Main.Tree_MailBoxesDragDrop(Sender, Source: TObject; X,  Y:
Integer);
var
```

```
    node  : TTreeNode;
    SourceMailBoxName : shortstring;
    SourceFolderName  : shortstring;
    SourceLetters     : PLetters;

    DestMailBox,
    DestFolderName    : shortstring;
    DestLetters       : PLetters;

begin
    node:=Tree_MailBoxes.GetNodeAt(x,y);
    if (node = nil)or(node.Parent = nil) then exit;
    if Tree_MailBoxes.Selected.Parent= nil then
SourceMailBoxName:=Tree_MailBoxes.Selected.Text
                                          else
SourceMailBoxName:=Tree_MailBoxes.Selected.Parent.Text;
    SourceFolderName:=Tree_MailBoxes.Selected.Text;
    SourceLetters:=MailBoxes.FindLetters(SourceMailBoxName,SourceFolderName);

    DestFolderName:=node.Text;
    DestMailBox:=node.Parent.Text;
    DestLetters:=MailBoxes.FindLetters(DestMailBox,DestFolderName);
    DestLetters.Add(SourceLetters.Letters[ListView_LettersHeaders.ItemIndex+1],

false,SourceLetters.LettersSizes[ListView_LettersHeaders.ItemIndex+1]);

SourceLetters.DeleteLetterWithoutFreeing(ListView_LettersHeaders.ItemIndex+1);
    Tree_MailBoxesClick(self);
end;

procedure Tfrm_Main.Tree_MailBoxesDragOver(Sender, Source: TObject; X,
  Y: Integer; State: TDragState; var Accept: Boolean);
begin
  if not (Source is TListView) then exit;
  if ListView_LettersHeaders.GetItemAt(x,y) = nil then exit;
  if (Tree_MailBoxes.Selected = nil)or(Tree_MailBoxes.Selected.Parent = nil)
then exit;
  Accept:=true;
end;

procedure Tfrm_Main.TrayClick(Sender: TObject);
begin
  Tray.ShowMainForm;
end;

procedure Tfrm_Main.TrayMinimizeToTray(Sender: TObject);
begin
  if Settings.ShowInTray then
  begin
   frm_Main.Tray.IconVisible:=true;
   frm_Main.Tray.HideTaskbarIcon;
   end
   else
   begin
   frm_Main.Tray.IconVisible:=false;
   frm_Main.Tray.ShowMainForm;
   end;
   //Application.Minimize;
end;

// Видалення усіх аркушів з папки
procedure Tfrm_Main.N13Click(Sender: TObject);
var
    Letters : PLetters;
    i       : word;
begin
  if Tree_MailBoxes.Selected=nil then exit;
  if Tree_MailBoxes.Selected.Parent = nil then exit;
```

```
  Letters:=MailBoxes.FindLetters(Tree_MailBoxes.Selected.Parent.Text,
Tree_MailBoxes.Selected.Text);
  if Letters = nil then exit;
  if MessageDlg('Ви впевнені, що хочете БЕЗПОВОРОТНО видалити ВСІ листи в цій
папці? "',
              mtConfirmation,[mbYes, mbNo], 0)<> mrYes then exit;

  for i:=1 to Letters.Count do Letters.DeleteLetter(1);
  Tree_MailBoxesClick(self);
end;

procedure Tfrm_Main.PopupMenu_MailBoxesPopup(Sender: TObject);
begin
  PopupMenu_MailBoxes.Items[0].Enabled:=false;
  PopupMenu_MailBoxes.Items[1].Enabled:=false;
  PopupMenu_MailBoxes.Items[2].Enabled:=false;
  if Tree_MailBoxes.Selected=nil then exit;

  PopupMenu_MailBoxes.Items[0].Enabled:=true;
  if Tree_MailBoxes.Selected.Parent = nil then exit;
  PopupMenu_MailBoxes.Items[1].Enabled:=true;
  PopupMenu_MailBoxes.Items[2].Enabled:=true;
end;


// Відправлення усіх листів до поштової скриньки
procedure Tfrm_Main.Jnghfdbnmgjxne1Click(Sender: TObject);
var
    BoxName : shortstring;
    MailBox : PMailBox;
    Letters : PLetters;
    i       : word;
begin
  if Tree_MailBoxes.Selected=nil then exit;
  if Tree_MailBoxes.Selected.Parent=nil then
BoxName:=Tree_MailBoxes.Selected.Text
                                        else
BoxName:=Tree_MailBoxes.Selected.Parent.Text;
  MailBox:=MailBoxes.FindBox(BoxName);
  if MailBox=nil then exit;
  Letters:=@MailBox.Outbox;
  if Letters=nil then exit;

  Tree_MailBoxesClick(self);

  try
    try
    FillSMTPwithMailBoxSettings(CurrentMailBoxSettings);
    DM.SMTP.Connect(20000);
    frm_Main.AddToLog('З"єднання з сервером
'+DM.SMTP.host+':'+inttostr(DM.SMTP.port)+' успішне.   ');

    for i:=1 to Letters.Count do
    begin
      frm_Main.AddToLog('Відправлення листа від '+CurrentMailBoxSettings.Name+'
до '+Letters.Letters[1].Recipients.EMailAddresses+' по темі:
'+Letters.Letters[1].Subject+'   ');
      DM.SMTP.Send( Letters.Letters[1]^ ); // Send letter
      MailBox.Sent.Add(Letters.Letters[1],false,Letters.LettersSizes[1]);   //
Add to Sent folder
      MailBox.Outbox.DeleteLetterWithoutFreeing(1);                         //
Delete from Outbox folder
    end;

    finally
      DM.SMTP.Disconnect;
    end;
  except
    on E:Exception do
```

```
        begin
        Application.MessageBox(pchar('Помилка при відправленні пошти зі скриньки
<'+BoxName+'> :'),pchar(E.Message) );
        AddToLog('Помилка при відправленні пошти зі скриньки <'+BoxName+'>
:'+E.Message+'   ');
        end;
  end;

end;

procedure Tfrm_Main.N15Click(Sender: TObject);
begin
frm_about.Show;
end;

procedure Tfrm_Main.N19Click(Sender: TObject);
begin
frm_Settings.show;
end;

procedure Tfrm_Main.ToolButton10Click(Sender: TObject);
begin
frm_MailSettings.Show;
end;

procedure Tfrm_Main.ToolButton14Click(Sender: TObject);
begin
halt;
end;

procedure Tfrm_Main.ToolButton12Click(Sender: TObject);
begin
frm_Settings.show;
end;

procedure Tfrm_Main.ToolButton6Click(Sender: TObject);
begin
frm_about.Show;
end;

procedure Tfrm_Main.ToolButton4Click(Sender: TObject);
begin
keys1.Show;
end;

procedure Tfrm_Main.N18Click(Sender: TObject);
begin
keys1.Show;
end;

end.
```

Файл frmNewLetter.pas - вікно створення захищеного електронного листа

```
unit frmNewLetter;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Buttons, ToolWin, ComCtrls, Menus, HTTPApp, HTTPProd, OleCtrls,
  SHDocVw, StdCtrls, ExtCtrls, ImgList, ActnList,
  IdMessage,  param_keys,   DES,  sha1, FGInt, FGIntRSA,
  frmMailSettings,LetterStorage;

type
  PidMessage =  ^TidMessage;

  Tfrm_NewLetter = class(TForm)
    MainMenu: TMainMenu;
    N1: TMenuItem;
    N2: TMenuItem;
    StatusBar: TStatusBar;
    CoolBar: TCoolBar;
    ToolBar1: TToolBar;
    Panel1: TPanel;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Ed_Reciever: TEdit;
    Ed_Copy: TEdit;
    Ed_Subject: TEdit;
    Memo_Body: TMemo;
    Label4: TLabel;
    Ed_HiddenCopy: TEdit;
    Actions_NewLetter: TActionList;
    SendNewLetter: TAction;
    Images_SendLetter: TImageList;
    ToolButton1: TToolButton;
    N3: TMenuItem;
    N4: TMenuItem;
    N5: TMenuItem;
    MoveNewLetter2Outbox: TAction;
    ToolButton2: TToolButton;
    ToolButton3: TToolButton;
    Splitter1: TSplitter;
    List_NewAttachments: TListView;
    ToolButton4: TToolButton;
    NewAttachment: TAction;
    Dialog_OpenAttachment: TOpenDialog;
    Popup_Attachment: TPopupMenu;
    N6: TMenuItem;
    Ghbrhtgbnmafqk1: TMenuItem;
    Panel_AdditionlSettings: TPanel;
    Group_AdditionalSettings: TGroupBox;
    Label6: TLabel;
    Label7: TLabel;
    Label5: TLabel;
    Combo_Priority: TComboBox;
    Ed_Sender: TEdit;
    Memo_AdditionalHeaders: TMemo;
    ToolButton5: TToolButton;
    Btn_AdvancedPage: TBitBtn;
    Splitter2: TSplitter;
    N7: TMenuItem;
    N8: TMenuItem;
    Label8: TLabel;
    Ed_ReplyTo: TEdit;
```

```
    Label9: TLabel;
    Combo_ContexType: TComboBox;
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
    procedure SendNewLetterExecute(Sender: TObject);
    procedure N4Click(Sender: TObject);
    procedure MoveNewLetter2OutboxExecute(Sender: TObject);
    procedure NewAttachmentExecute(Sender: TObject);
    procedure N6Click(Sender: TObject);
    procedure List_NewAttachmentsMouseDown(Sender: TObject;
      Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
    procedure Btn_AdvancedPageClick(Sender: TObject);
    procedure N8Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure BitBtn2Click(Sender: TObject);
  private
    { Private declarations }
    function CheckIfFormIsFilled : boolean;
  public
    { Public declarations }
    IsToBeResend  : boolean;
    ResendMes     : TidMessage;
    MailBoxName   : shortstring;
    Data: TBitString;
    //CURR_mailbox_SETTINGS : TMailBoxSettings;
    CURR_MAIL_BOX : PMailBox;
    function CreateLetterFromForm:PidMessage;
    procedure UpdateStatusLine;
    procedure ClearForm;
  end;

procedure FillSMTPwithMailBoxSettings(mbSettings : PMailBoxSettings);

var
  frm_NewLetter: Tfrm_NewLetter;
  n,d: TFGInt;
  st, st2: string;

implementation
uses frmMain, DataModule;
{$R *.dfm}


procedure FillSMTPwithMailBoxSettings(mbSettings : PMailBoxSettings);
Begin
  DM.SMTP.Host :=mbSettings.SMTPServer;
  DM.SMTP.Port :=mbSettings.SMTPPort;
  DM.SMTP.Username :=mbSettings.SMTPAccount;
  DM.SMTP.Password :=mbSettings.SMTPPass;
  DM.SMTP.AuthenticationType:=mbSettings.AuthType;
End;



procedure Tfrm_NewLetter.SendNewLetterExecute(Sender: TObject);
var
    mes     : PidMessage;
begin
  if CurrentMailBoxSettings = nil then
  begin
    Application.MessageBox('Ви повинні вибрати скриньку, з якої хочете
відправити листа ','Помилка відправлення');
    exit;
  end;
  if not CheckIfFormIsFilled then exit;
  mes:=CreateLetterFromForm;
  SendNewLetter.Enabled:=false;
```

```
  try
      FillSMTPwithMailBoxSettings(CurrentMailBoxSettings);
      frm_Main.AddToLog('Негайне відправлення листа від
'+CurrentMailBoxSettings.Name+' до '+mes.Recipients.EMailAddresses);
      DM.SMTP.Connect(20000);
      frm_Main.AddToLog('З"єднання з сервером
'+DM.SMTP.host+':'+inttostr(DM.SMTP.port)+' успішне.');
      try
        DM.SMTP.Send(mes^);
      finally
        DM.SMTP.Disconnect;
      end;
      frm_Main.AddToLog('Відключення від сервера, повідомлення відправлене.');
      CURR_MAIL_BOX.Sent.Add(@(mes^),false,0);
      frm_Main.Tree_MailBoxesClick(self);
  except
    on E:Exception do
       begin
       Application.MessageBox(pchar(E.message),'Не вдалося відправити пошту');
       CURR_MAIL_BOX.Outbox.Add(@(mes^),false,0);
       end;
  end;
  hide;
end;


procedure Tfrm_NewLetter.N4Click(Sender: TObject);
begin
  self.Hide;
end;


procedure Tfrm_NewLetter.MoveNewLetter2OutboxExecute(Sender: TObject);
var
    mes : TIdMessage;
begin
  Application.MessageBox('','');
  mes:=CreateLetterFromForm^;
  CURR_MAIL_BOX.Outbox.Add( @mes,false,0);
  frm_Main.Tree_MailBoxesClick(self);
  hide;
end;


procedure Tfrm_NewLetter.NewAttachmentExecute(Sender: TObject);
var
    s : ^shortstring;
    st: TFileStream;
begin
  if not Dialog_OpenAttachment.Execute then exit;
  with List_NewAttachments.Items.Add do
  begin
    caption:=ExtractFileName(Dialog_OpenAttachment.FileName);
    try
      st:=TFileStream.Create(Dialog_OpenAttachment.FileName,fmOpenRead or
fmShareDenyNone);
      SubItems.Add(inttostr(st.Size));
      ImageIndex:=2;
      st.Free;
    except
      Application.MessageBox('Не вдалося отримати розмір файлу!','Помилка');
      exit;
    end;
    new(s);
    s^:=Dialog_OpenAttachment.FileName;
    data:=s;

    Splitter1.Visible:=true;
    List_NewAttachments.Visible:=true;
```

```
    UpdateStatusLine;
  end;
end;


procedure Tfrm_NewLetter.N6Click(Sender: TObject);
begin
  if List_NewAttachments.ItemIndex<0 then exit;
  dispose(List_NewAttachments.Items[List_NewAttachments.ItemIndex].Data);
  List_NewAttachments.DeleteSelected;
  UpdateStatusLine;
  if List_NewAttachments.Items.Count<=0 then
     begin
       Splitter1.Visible:=false;
       List_NewAttachments.Visible:=false;
     end;
end;

procedure Tfrm_NewLetter.List_NewAttachmentsMouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var
  Item  : TListItem;
begin
  Item:=List_NewAttachments.GetItemAt(x,y);
  if Item<>nil then Item.Selected:=true;
end;

function Tfrm_NewLetter.CreateLetterFromForm: PidMessage;
var
    PLet : PidMessage;
    i    : integer;
    s    : ^shortstring;
begin
  frm_Main.Tree_MailBoxesClick(self);
  if CurrentMailBoxSettings =nil then exit;

  new(PLet);
  PLet^:=TIdMessage.Create(self);

  PLet^.Subject:=Ed_Subject.Text;
  PLet^.CCList.EMailAddresses:=Ed_Copy.Text;
  PLet^.BccList.EMailAddresses:=Ed_HiddenCopy.Text;
  PLet^.Body.Assign(Memo_Body.Lines);
  PLet^.From.Text:=CurrentMailBoxSettings^.POPEMail;
  if Ed_ReplyTo.Text='' then
PLet^.ReplyTo.EMailAddresses:=CurrentMailBoxSettings.POPEMail
                    else PLet^.ReplyTo.EMailAddresses:=Ed_ReplyTo.Text;
  PLet^.Recipients.EMailAddresses:=Ed_Reciever.Text;
  case Combo_Priority.ItemIndex of
      0 : PLet^.Priority:=mpHighest;
      1 : PLet^.Priority:=mpHigh;
      2 : PLet^.Priority:=mpNormal;
      3 : PLet^.Priority:=mpLow;
      4 : PLet^.Priority:=mpLowest;
     end;
  PLet^.Sender.Text:=Ed_Sender.Text;
  PLet^.ExtraHeaders.Assign(Memo_AdditionalHeaders.Lines);
  PLet^.ContentType:=Combo_ContexType.Text;

  for i:=1 to List_NewAttachments.Items.Count do
  begin
    s:=List_NewAttachments.Items.Item[ i-1].data;
    TIdAttachment.Create(PLet^.MessageParts, s^);
  end;

  CreateLetterFromForm:=PLet;
end;

procedure Tfrm_NewLetter.UpdateStatusLine;
```

```
var
  AttSize : longint;
  i       : word;
begin
  AttSize:=0;
  for i:=1 to List_NewAttachments.Items.Count do
inc(AttSize,strtointdef(List_NewAttachments.Items[ i-1].SubItems[0],0));
  if AttSize<>0 then StatusBar.Panels.Items[1].Text:='Розмір приєднаних файлів :
'+inttostr(AttSize div 1024)+' Kb'
                else StatusBar.Panels.Items[1].Text:='';
end;

procedure Tfrm_NewLetter.Btn_AdvancedPageClick(Sender: TObject);
begin
  if Btn_AdvancedPage.Caption<>'Прибрати' then
     begin
     Panel_AdditionlSettings.Show;
     Splitter2.Show;
     Btn_AdvancedPage.Caption:='Прибрати';
     Btn_AdvancedPage.Width:=230;
     end
     else
     begin
     Splitter2.Hide;
     Panel_AdditionlSettings.Hide;

     Btn_AdvancedPage.Caption:='Додатково';
     Btn_AdvancedPage.Width:=170;
     end;
end;

procedure Tfrm_NewLetter.ClearForm;
begin
  Ed_Reciever.Clear;
  Ed_Copy.Clear;
  Ed_Subject.Clear;
  Ed_HiddenCopy.Clear;
  Ed_Sender.Clear;
  Memo_Body.Clear;
  Memo_AdditionalHeaders.Clear;
  List_NewAttachments.Clear;
  Ed_ReplyTo.Clear;
  Combo_Priority.ItemIndex:=2;

  IsToBeResend:=false;
  ResendMes.Clear;
  SendNewLetter.Enabled:=true;
end;

procedure Tfrm_NewLetter.N8Click(Sender: TObject);
begin
  ClearForm;
end;

function Tfrm_NewLetter.CheckIfFormIsFilled: boolean;
begin
  CheckIfFormIsFilled:=false;
  if Ed_Reciever.Text='' then
    begin
    Application.MessageBox('Ви забули вказати кому відправляти лист','Помилка');
    exit;
    end;

  if Ed_Subject.Text='' then
    begin
    if Application.MessageBox('Ви дійсно не хочете вказувати тему
листа?','Очікую підтвердження...',MB_YESNO     )=IDNo
       then exit;
    end;
```

```
  CheckIfFormIsFilled:=true;
end;


procedure Tfrm_NewLetter.FormCreate(Sender: TObject);
begin
  frm_NewLetter.ResendMes:=TIdMessage.Create(self);

end;

procedure Tfrm_NewLetter.FormShow(Sender: TObject);
begin

  CURR_MAIL_BOX:=MailBoxes.FindBox(CurrentMailBoxSettings.Name);
end;

procedure Tfrm_NewLetter.BitBtn1Click(Sender: TObject);
var I, x,j:Integer;
S,m:String;
begin
IF (Length(Memo_Body.Text)mod 8 <> 0) Then
  Begin
    x:= 8 - (Length(Memo_Body.Text)mod 8);
   for j:=1 to x do  begin
    m:=Memo_Body.Lines.Strings[Memo_Body.Lines.Count-1];
    Memo_Body.Lines.Delete(Memo_Body.Lines.Count-1);
    Memo_Body.Lines.Add(m+' ');
   end;
  End;

SetLength(Data,0);
I:=1;
While I<=Length(Memo_Body.Text) Do
  Begin
  S:=Copy(Memo_Body.Text,I,8);
  Data:=ConcatBits([Data,DESEncode(S,keys1.Edit_des.Text)]);
  I:=I+8;
  End;
  Memo_Body.Clear;
  Memo_Body.Text:=BinToAnsiStr(Data);

end;

procedure Tfrm_NewLetter.BitBtn2Click(Sender: TObject);
var h:string;
begin

Memo_Body.Lines.Add('');
Memo_Body.Lines.Add(' - - - - - - - - - - - - - - - - -------');
Memo_Body.Lines.Add('цифровий підпис:');
Memo_Body.Lines.Add('');
h:=Work(Memo_Body.Lines);

ConvertBase64to256(keys1.Edit_N.Text,st);
Base256StringToFGInt(st, n);
ConvertBase64to256(keys1.Edit_D.Text,st);
Base256StringToFGInt(st, d);
RSAEncrypt(h, d, n, st);
Memo_Body.Lines.Add(st);

end;

end.
```

Файл frmMailSettings.pas - управління обліковими записами, створення поштових скриньок

```
unit frmMailSettings;

interface

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

type

  PMailBoxSettings = ^TMailBoxSettings;


  Tfrm_MailSettings = class(TForm)
    ListBox_MailBoxes: TListBox;
    GroupBox1: TGroupBox;
    GroupBox3: TGroupBox;
    GroupBox2: TGroupBox;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Combo_SMTPType: TComboBox;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Ed_SMTPServer: TEdit;
    Ed_SMTPPort: TEdit;
    Ed_SMTPAccount: TEdit;
    Ed_SMTPPassword: TEdit;
    Ed_POPServer: TEdit;
    Ed_POPPort: TEdit;
    Ed_POPAccount: TEdit;
    Ed_POPPassword: TEdit;
    btn_Apply: TBitBtn;
    btn_NewMailBox: TBitBtn;
    GroupBox4: TGroupBox;
    Label11: TLabel;
    Ed_Subscript: TEdit;
    Ed_MailBoxName: TEdit;
    Label12: TLabel;
    Label13: TLabel;
    Label14: TLabel;
    Combo_CheckEvery: TComboBox;
    BitBtn1: TBitBtn;
    Check_RetrieveHeadersOnly: TCheckBox;
    Check_DeleteFromServer: TCheckBox;
    Check_DontOpenHTML: TCheckBox;
    GroupBox5: TGroupBox;
    Label15: TLabel;
    Radio_ignore: TRadioButton;
    Radio_Recieve: TRadioButton;
    Radio_Ask: TRadioButton;
    RadioDelete: TRadioButton;
    Label10: TLabel;
    Ed_POPEmail: TEdit;
    Check_DynamicFolders: TCheckBox;
    Check_ShowBalloon: TCheckBox;
    Button1: TButton;

    procedure btn_NewMailBoxClick(Sender: TObject);
```

```
    procedure FormShow(Sender: TObject);
    procedure Combo_CheckEveryExit(Sender: TObject);
    procedure btn_ApplyClick(Sender: TObject);
    procedure ListBox_MailBoxesClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
    procedure LoadSettingsFromRecords;
    procedure LoadFormFromBox(box : PMailBoxSettings);
    procedure SaveFormToBox(box : PMailBoxSettings);
    procedure ClearForm;
  end;

  TDuplicateAction = (Ask,Ignore,Recieve,DeleteFromServer);

  TMailBoxSettings = record
    SMTPServer  :shortstring;
    SMTPPort    :integer;
    SMTPAccount :shortstring;
    SMTPPass    :shortstring;
    AuthType    :TAuthenticationType;

    POPServer   :shortstring;
    POPPort     :integer;
    POPAccount  :shortstring;
    POPPass     :shortString;
    POPEMail    :shortstring;

    Subscript   :shortstring;
    Name        :shortstring;
    CheckEvery  :word;

    RetrieveOnlyHeaders : boolean;
    DeleteFromServer    : boolean;
    OpenHTMLasTEXT      : boolean;
    ifDuplicate         : TDuplicateAction;
    DynamicFolders      : boolean;
    ShowBalloonOnNewMes : boolean;
    end;

    procedure SaveSettingsToFile;
    procedure LoadSettingsFromfile;
    procedure findMailBoxAndSelectIt(name  : shortstring);


const
    MailBoxSettingsFileName   = 'mailboxes.dat';

var
  frm_MailSettings         : Tfrm_MailSettings;
  TotalMailBoxes           : word=0;
  MailBoxesSettings        : array [1..300] of PMailBoxSettings;
  CurrentMailBoxSettings   : PMailBoxSettings=nil;
  PrevMailBoxSettings      : PMailBoxSettings=nil;

implementation

uses frmMain;
{$R *.dfm}

procedure findMailBoxAndSelectIt(name  : shortstring);
var
    i : word;
Begin
```

```
  CurrentMailBoxSettings  :=  nil;
  for i:=1 to TotalMailBoxes do
    if MailBoxesSettings[i]^.Name = name then
    begin
      CurrentMailBoxSettings:=MailBoxesSettings[i];
      exit;
    end;
End;

procedure SaveSettingsToFile;
var
    SFile : shortstring;
    st    : TFileStream;
    i     : word;
Begin
  SFile:=ExtractFilePath(Application.ExeName)+MailBoxSettingsFileName;
  try
    deletefile(SFile);
  except end;
  try
    st:=TFileStream.Create(SFile,fmOpenWrite or fmShareDenyNone);
  except
    try
      st:=TFileStream.Create(SFile,fmCreate);
    except
      Application.MessageBox( pchar('Не можу відкрити файл "'+SFile+'" для
запису !'),'Помилка збереження параметрів програми');
      exit;
    end;
  end;
  st.Write(TotalMailBoxes,sizeof(TotalMailBoxes));
  for i:=1 to TotalMailBoxes do
      st.write(MailBoxesSettings[i]^,sizeof(TMailBoxSettings));
  st.Free;
End;

procedure LoadSettingsFromfile;
var
    SFile : shortstring;
    st    : TFileStream;
    i     : word;
Begin
{  SFile:=ExtractFilePath(Application.ExeName)+MailBoxSettingsFileName;
  try
    st:=TFileStream.Create(SFile,fmOpenRead or  fmShareDenyNone);
  except
    Application.MessageBox( pchar('Can''t open file "'+SFile+'" for reading
!'),'Error loading Mailboxes settings');
    exit;
  end;

  st.Read(TotalMailBoxes,sizeof(TotalMailBoxes));
  for i:=1 to TotalMailBoxes do
      begin
      new(MailBoxesSettings[i]);
      st.Read(MailBoxesSettings[i]^,sizeof(TMailBoxSettings));
      end;
  st.Free;  }
End;

procedure Tfrm_MailSettings.LoadFormFromBox(box : PMailBoxSettings);
Begin
  Ed_SMTPServer.Text  := box.SMTPServer;
  Ed_SMTPPort.Text    := inttostr(box.SMTPPort);
  Ed_SMTPAccount.Text := box.SMTPAccount;
  Ed_SMTPPassword.Text:= box.SMTPPass;

  Ed_POPServer.Text   := box.POPServer;
  Ed_POPPort.Text     := inttostr(box.POPPort);
```

```
  Ed_POPAccount.Text  := box.POPAccount;
  Ed_POPPassword.Text := box.POPPass;
  Ed_POPEmail.Text    := box.POPEMail;

  Ed_Subscript.Text   := box.Subscript;
  case box.AuthType of
       atNone : Combo_SMTPType.ItemIndex:=0;
       atLogin: Combo_SMTPType.ItemIndex:=1;
       end;

  if box.CheckEvery<>0 then Combo_CheckEvery.Text:=inttostr(box.CheckEvery)
                       else Combo_CheckEvery.Text:='5';

  Check_RetrieveHeadersOnly.Checked :=  box.RetrieveOnlyHeaders;
  Check_DeleteFromServer.Checked    :=  box.DeleteFromServer;

  Check_DontOpenHTML.Checked:=box.OpenHTMLasTEXT;
  case box.ifDuplicate of
    Ignore  : Radio_ignore.Checked:=true;
    Recieve : Radio_Recieve.Checked:=true;
    Ask     : Radio_Ask.Checked:=true;
    DeleteFromServer : RadioDelete.Checked:=true;
    end;
  Check_DynamicFolders.Checked := box.DynamicFolders;
  Check_ShowBalloon.Checked    := box.ShowBalloonOnNewMes;
End;

procedure Tfrm_MailSettings.SaveFormToBox(box : PMailBoxSettings);
Begin
  box.SMTPServer  := Ed_SMTPServer.Text;
  box.SMTPPort    := strtoint(Ed_SMTPPort.Text);
  box.SMTPAccount := Ed_SMTPAccount.Text;
  box.SMTPPass:= Ed_SMTPPassword.Text;

  box.POPServer   := Ed_POPServer.Text;
  box.POPPort     := strtoint(Ed_POPPort.Text);
  box.POPAccount  := Ed_POPAccount.Text;
  box.POPPass     := Ed_POPPassword.Text;
  box.POPEMail    := Ed_POPEmail.Text;

  box.Subscript   := Ed_Subscript.Text;
  case Combo_SMTPType.ItemIndex of
       0 : box.AuthType:=atNone;
       1 : box.AuthType:=atLogin;
       end;

  if AnsiUpperCase(Combo_CheckEvery.Text)<>'Ніколи' then
     box.CheckEvery:=strtoint(Combo_CheckEvery.Text)
     else
     box.CheckEvery:=0;

  box.RetrieveOnlyHeaders :=  Check_RetrieveHeadersOnly.Checked;
  box.DeleteFromServer    :=  Check_DeleteFromServer.Checked;
  box.OpenHTMLasTEXT      :=  Check_DontOpenHTML.Checked;

  if Radio_ignore.Checked then box.ifDuplicate:=Ignore;
  if Radio_Recieve.Checked then box.ifDuplicate:=Recieve;
  if Radio_Ask.Checked then box.ifDuplicate:=Ask;
  if RadioDelete.Checked then box.ifDuplicate:=DeleteFromServer;

  box.DynamicFolders      :=  Check_DynamicFolders.Checked;
  box.ShowBalloonOnNewMes :=  Check_ShowBalloon.Checked;

End;

procedure Tfrm_MailSettings.ClearForm;
Begin
Ed_SMTPServer.Text  :='smtp.yandex.ru';
Ed_SMTPPort.Text    :='25';
```

```
Ed_SMTPAccount.Text :=' test-s-mime';
Ed_SMTPPassword.Text:='smimeforanna1718';

Ed_POPServer.Text   :='pop.yandex.ru';
Ed_POPPort.Text     :='110';
Ed_POPAccount.Text  :=' test-s-mime';
Ed_POPPassword.Text :='smimeforanna1718';
Ed_POPEmail.Text    :=' test-s-mime@yandex.ru';
Ed_Subscript.Text   :='Ганна';

Combo_SMTPType.ItemIndex:=1;
Combo_CheckEvery.ItemIndex:=0;
End;

procedure Tfrm_MailSettings.LoadSettingsFromRecords;
Begin
  if TotalMailBoxes <> 0 then
  begin
    if MailBoxesSettings[1]<>nil then
    begin
      CurrentMailBoxSettings:=MailBoxesSettings[1];

      LoadFormFromBox(CurrentMailBoxSettings);
      ListBox_MailBoxes.Selected[0]:=true;
      PrevMailBoxSettings:=CurrentMailBoxSettings;
    end;
  end
  else CurrentMailBoxSettings:=nil;
End;

procedure Tfrm_MailSettings.btn_NewMailBoxClick(Sender: TObject);
var
    i : word;
begin
  if Ed_MailBoxName.Text<>'' then
  begin
     for i:=1 to ListBox_MailBoxes.Count do
        { if AnsiUpperCase(ListBox_MailBoxes.Items[ i-
1])=AnsiUpperCase(Ed_MailBoxName.Text) then
            begin
            Application.MessageBox('Таке ім"я облікового запису вже існує!','Так
не можно');
            exit;
            end; }
     ListBox_MailBoxes.Items.Add(Ed_MailBoxName.Text);
     GroupBox1.Visible:=true;
     Radio_Ask.Checked:=true;
     inc(TotalMailBoxes);
     new(MailBoxesSettings[TotalMailBoxes]);
     CurrentMailBoxSettings:=MailBoxesSettings[TotalMailBoxes];
     CurrentMailBoxSettings.Name:=Ed_MailBoxName.Text;
     if PrevMailBoxSettings<>nil then SaveFormToBox(PrevMailBoxSettings);
     PrevMailBoxSettings:=CurrentMailBoxSettings;
     ClearForm;
     SaveFormToBox(CurrentMailBoxSettings);
     ListBox_MailBoxes.Selected[ListBox_MailBoxes.Count-1]:=true;
     {Now Create and fill New mailbox }
     MailBoxes.Add(Ed_MailBoxName.Text);
     frm_Main.AddToLog('Створена нова скринька <'+Ed_MailBoxName.Text+'>');
     frm_Main.ShowMailBoxesInTree;
  end
  else
  Application.MessageBox('Спочатку введіть ім"я вашої скриньки','Так
заборонено');
end;



procedure Tfrm_MailSettings.FormShow(Sender: TObject);
```

```
var
    i : word;
begin
  if (ListBox_MailBoxes.Count=0)and(TotalMailBoxes<>0) then
     for i:=1 to TotalMailBoxes do
ListBox_MailBoxes.Items.Add(MailBoxesSettings[i].Name);
  LoadSettingsFromRecords;
  if CurrentMailBoxSettings=nil then GroupBox1.Visible:=false
                        else GroupBox1.Visible:=true;
end;

procedure Tfrm_MailSettings.Combo_CheckEveryExit(Sender: TObject);
begin
  if AnsiUpperCase(Combo_CheckEvery.Text)<>'Ніколи' then
  begin
     try
      strtoint(Combo_CheckEvery.Text);
     except
      Application.MessageBox('Невірне число','Помилка введення');
      Combo_CheckEvery.Text:='5';
      exit;
     end;
  end;
end;


procedure Tfrm_MailSettings.btn_ApplyClick(Sender: TObject);
var
    i : word;
begin
  if ListBox_MailBoxes.ItemIndex>=0 then
  begin
  SaveFormToBox(CurrentMailBoxSettings);
 // Application.MessageBox('Створили');
  end;
  SaveSettingsToFile;
  self.Hide;
  for i:=1 to TotalMailBoxes do
      if MailBoxes.GetByIndex(i)<>nil then
MailBoxes.GetByIndex(i).SetTimerInterval(MailBoxesSettings[i].CheckEvery);
end;


function FindMailBox(name : shortstring):PMailBoxSettings;
var
    i : word;
Begin
FindMailBox:=nil;

for i:=1 to TotalMailBoxes do
    if MailBoxesSettings[i].Name = name then
    begin
      FindMailBox:=MailBoxesSettings[i];
      exit;
    end;

End;


procedure Tfrm_MailSettings.ListBox_MailBoxesClick(Sender: TObject);
begin
  if ListBox_MailBoxes.ItemIndex >=0 then
  begin

CurrentMailBoxSettings:=FindMailBox(ListBox_MailBoxes.Items[ListBox_MailBoxes.it
emindex]);
  if CurrentMailBoxSettings=nil then
CurrentMailBoxSettings:=PrevMailBoxSettings;
  if PrevMailBoxSettings<>nil then SaveFormToBox(PrevMailBoxSettings);
```

```
  PrevMailBoxSettings:=CurrentMailBoxSettings;
  LoadFormFromBox(CurrentMailBoxSettings);
  end;
end;

procedure Tfrm_MailSettings.FormCreate(Sender: TObject);
begin
//  LoadSettingsFromfile;
end;


procedure Tfrm_MailSettings.BitBtn1Click(Sender: TObject);
var
    i,j : word;
begin
  if ListBox_MailBoxes.ItemIndex >= 0 then
  begin
    if MessageDlg('Ви впевнені, що хочете видалити обліковий запис
"'+ListBox_MailBoxes.Items[ListBox_MailBoxes.itemindex]+'" ?',
                  mtConfirmation,[mbYes, mbNo], 0)= mrNo then exit;

    for i:=1 to TotalMailBoxes do
        if MailBoxesSettings[i]^.Name =
ListBox_MailBoxes.Items[ListBox_MailBoxes.itemindex] then
        begin
          for j:=i to TotalMailBoxes-1 do
MailBoxesSettings[j]:=MailBoxesSettings[j+1];
          break;
        end;
    frm_Main.AddToLog('Скринька видалена
<'+ListBox_MailBoxes.Items[ListBox_MailBoxes.itemindex]+'>');
    ListBox_MailBoxes.DeleteSelected;
    dec(TotalMailBoxes);
    try
      ListBox_MailBoxes.Selected[0]:=true;
      ListBox_MailBoxesClick(self);
      frm_Main.ShowMailBoxesInTree;
    except end;
  end;
end;

procedure Tfrm_MailSettings.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  btn_ApplyClick(self);
end;

procedure Tfrm_MailSettings.Button1Click(Sender: TObject);
begin
     ListBox_MailBoxes.Items.Add(Ed_MailBoxName.Text);
     GroupBox1.Visible:=true;
     Radio_Ask.Checked:=true;
     inc(TotalMailBoxes);
     new(MailBoxesSettings[TotalMailBoxes]);
     CurrentMailBoxSettings:=MailBoxesSettings[TotalMailBoxes];
     CurrentMailBoxSettings.Name:=Ed_MailBoxName.Text;
     if PrevMailBoxSettings<>nil then SaveFormToBox(PrevMailBoxSettings);
     PrevMailBoxSettings:=CurrentMailBoxSettings;
     ClearForm;
     SaveFormToBox(CurrentMailBoxSettings);
     ListBox_MailBoxes.Selected[ListBox_MailBoxes.Count-1]:=true;
     {Now Create and fill New mailbox }
     MailBoxes.Add(Ed_MailBoxName.Text);
     frm_Main.AddToLog('Створена нова скринька <'+Ed_MailBoxName.Text+'>');
     frm_Main.ShowMailBoxesInTree;
end;
end.
```

Файл param_keys.pas - вікно зміни параметрів захисту S/MIME

```
unit param_keys;

interface

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

type
  Tkeys1 = class(TForm)
    Button1: TButton;
    Label5: TLabel;
    Label6: TLabel;
    Label8: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    Button3: TButton;
    Edit_E: TEdit;
    Edit_N: TEdit;
    Edit_D: TEdit;
    Label2: TLabel;
    Edit1: TEdit;
    Label3: TLabel;
    Image1: TImage;
    Edit_des: TEdit;
    Button2: TButton;
    Label7: TLabel;
    Image2: TImage;
    Bevel1: TBevel;
    Label1: TLabel;
    Bevel2: TBevel;
    Label4: TLabel;
    Button4: TButton;
    Button5: TButton;
    Button6: TButton;
    Button7: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  keys1: Tkeys1;
   n, e, d, dp, dq, p, q, phi, one, two, gcd, temp, nilgint : TFGInt;
   test, signature : String;
   ok : boolean;
   st: string;
   tx1: text;
   tx2: text;
   tx3: text;
implementation

{$R *.dfm}

procedure Tkeys1.Button1Click(Sender: TObject);
begin

   Base256StringToFGInt('3557', p);
   Base256StringToFGInt('2579', q);
```

```
    PrimeSearch(p);
    PrimeSearch(q);
    FGIntToBase256String(p, st);
    FGIntToBase256String(q, st);

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



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

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

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

end;

procedure Tkeys1.Button2Click(Sender: TObject);
begin
 AssignFile(tx3,'DESKey.keys');

  ReWrite(tx3); CloseFile(tx3);
  Append(tx3);
st:=Edit_des.Text;
  WriteLn(tx3,st);
  CloseFile(tx3);

end;

procedure Tkeys1.Button3Click(Sender: TObject);
begin
  AssignFile(tx1,'OpenKey.keys');
  AssignFile(tx2,'CloseKey.keys');

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

```
  ConvertBase256to64(st, st);
  WriteLn(tx1,st);
  FGIntToBase256String(d, st);
  ConvertBase256to64(st, st);
  WriteLn(tx2,st);
  CloseFile(tx1);
  CloseFile(tx2);

end;

procedure Tkeys1.Button4Click(Sender: TObject);
begin
keys1.Close;

end;

procedure Tkeys1.Button5Click(Sender: TObject);
begin
keys1.Close;
end;

end.
```

Файл frmSettings.pas - вікно зміни параметрів програми

```
unit frmSettings;

interface

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

type
  Tfrm_Settings = class(TForm)
    GroupBox1: TGroupBox;
    Check_AllowLog: TCheckBox;
    L_MaxLogSize: TLabel;
    Ed_MaxLogSize: TEdit;
    L_kb: TLabel;
    btn_Close: TBitBtn;
    RadioShowInTaskBar: TRadioButton;
    RadioShowInTray: TRadioButton;
    Image1: TImage;
    procedure Check_AllowLogClick(Sender: TObject);
    procedure Ed_MaxLogSizeChange(Sender: TObject);
    procedure Ed_MaxLogSizeKeyPress(Sender: TObject; var Key: Char);
    procedure FormCreate(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btn_CloseClick(Sender: TObject);
    procedure FormHide(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
    procedure LoadGlobalSettingsToForm;
    procedure LoadGlobalSettingsFromFile;
    procedure SaveGlobalSettingsFromForm;
    procedure SaveGlobalSettingsToFile;
  end;

  TSettings = record
    AllowLog  : boolean;
    MaxLogSize: longint;  // kb
    ShowInTray: boolean;
    end;

const
    GlobalSettingsFileName  = 'globalsettings.dat';

var
  frm_Settings: Tfrm_Settings;
  Settings    : TSettings=(AllowLog:true);

implementation

uses frmMain;

{$R *.dfm}


procedure Tfrm_Settings.FormCreate(Sender: TObject);
begin
  LoadGlobalSettingsFromFile;
end;



procedure Tfrm_Settings.FormShow(Sender: TObject);
```

```
begin
  LoadGlobalSettingsToForm;
end;


procedure Tfrm_Settings.Check_AllowLogClick(Sender: TObject);
var
    b : boolean;
begin
  if Check_AllowLog.Checked then b:=true
                            else b:=false;
  L_MaxLogSize.Visible:=b;
  Ed_MaxLogSize.Visible:=b;
  L_kb.Visible:=b;
  Settings.AllowLog:=b;
  Settings.MaxLogSize:=strtointdef(Ed_MaxLogSize.text,500);
end;



procedure Tfrm_Settings.Ed_MaxLogSizeChange(Sender: TObject);
var
    maxs  : longint;
begin
  try
    if Ed_MaxLogSize.Text<>'' then
       maxs:=strtoint(Ed_MaxLogSize.text);
  except
    Application.MessageBox('Розмір логу файлу може бути від 0 до 2147483647
Kb','Неправильне введення');
    Ed_MaxLogSize.Text:='500';
    exit;
  end;
  Settings.MaxLogSize:=maxs;
end;


const
    Digits :set of char =['1','2','3','4','5','6','7','8','9','0',#8];

procedure Tfrm_Settings.Ed_MaxLogSizeKeyPress(Sender: TObject;  var Key: Char);
begin
if not (key in Digits) then Key:=#0;
end;


procedure Tfrm_Settings.LoadGlobalSettingsToForm;
begin
  Ed_MaxLogSize.Text      :=  inttostr(Settings.MaxLogSize);
  Check_AllowLog.Checked  :=  Settings.AllowLog;
  RadioShowInTray.Checked :=  Settings.ShowInTray;
end;


procedure Tfrm_Settings.SaveGlobalSettingsFromForm;
begin
  Settings.AllowLog   :=  Check_AllowLog.Checked;
  Settings.MaxLogSize :=  strtointdef(Ed_MaxLogSize.text,500);
  Settings.ShowInTray :=  RadioShowInTray.Checked;
end;


procedure Tfrm_Settings.LoadGlobalSettingsFromFile;
var
    st      : TFileStream;
    SFName  : shortstring;
begin
  SFName:=ExtractFilePath(Application.ExeName)+GlobalSettingsFileName;
  try
```

```
    st:=TFileStream.Create(SFName,fmOpenRead or fmShareDenyNone);
  except
    try
      st:=TFileStream.Create(SFName,fmCreate or fmShareDenyNone);
    except
      Application.MessageBox(pchar('Неможливо відкрити/створити файл
"'+SFName+'"'),'Помилка завантаження глобальних параметрів');
      exit;
    end;
    fillchar(Settings,sizeof(Settings),0);
    LoadGlobalSettingsToForm;
    st.Free;
    exit;
  end;
  fillchar(Settings,sizeof(Settings),0);
  st.Read(Settings,sizeof(Settings));
  st.Free;
end;



procedure Tfrm_Settings.SaveGlobalSettingsToFile;
var
    st      : TFileStream;
    SFName  : shortstring;
begin
  SFName:=ExtractFilePath(Application.ExeName)+GlobalSettingsFileName;
  try
    st:=TFileStream.Create(SFName,fmOpenWrite or     fmShareDenyNone);
  except
    try
      st:=TFileStream.Create(SFName,fmCreate or fmShareDenyNone);
    except
      Application.MessageBox(pchar('Неможливо відкрити/створити файл
"'+SFName+'"'),'Помилка завантаження глобальних параметрів');
      exit;
    end;
  end;

  st.Write(Settings,sizeof(Settings));
  st.Free;
end;


procedure Tfrm_Settings.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  SaveGlobalSettingsFromForm;
  SaveGlobalSettingsToFile;
end;

procedure Tfrm_Settings.btn_CloseClick(Sender: TObject);
begin
  self.Hide;
end;

procedure Tfrm_Settings.FormHide(Sender: TObject);
begin
  SaveGlobalSettingsFromForm;
  SaveGlobalSettingsToFile;
  if Settings.ShowInTray then
   begin
   frm_Main.Tray.IconVisible:=true;
   frm_Main.Tray.HideTaskbarIcon;
   frm_Main.Tray.MinimizeToTray:=true;
   end
   else
   begin
   frm_Main.Tray.IconVisible:=false;
   frm_Main.Tray.ShowMainForm;
```

```
    frm_Main.Tray.ShowTaskbarIcon;
    frm_Main.Tray.MinimizeToTray:=false;
    end;
end;

end.
```

Файл sha1.pas - реалізація алгоритму хешування sha-1

```
unit sha1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,  StdCtrls,
  Dialogs, FGInt, FGIntRSA;

var
  tx: text;
  n,e: TFGInt;
  d, Nilgint: TFGInt;
  st, st2: string;

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
const
  HC0=$67452301;
  HC1=$EFCDAB89;
  HC2=$98BADCFE;
  HC3=$10325476;
  HC4=$C3D2E1F0;

  K1=$5A827999;
  K2=$6ED9EBA1;
  K3=$8F1BBCDC;
  K4=$CA62C1D6;

  dstr = 'BjZVQXx=';
  nstr = 'auPkf0q4JKga3a8Lyz09u3OSfQJUN=TK7tgUiMq0qw';

var H0,H1,H2,H3,H4:integer;  Hout:string;  //Hout - результат
    StopScan:boolean;

function Work(Z:TStrings):string;

implementation
{$R *.DFM}

function rol(const x:integer;const y:byte):integer ;     //зрушення числа x на y
біт уліво
begin
  asm
    mov  eax,x
    mov  cl, y
    rol  eax,cl
    mov  x, eax
  end;
  result:=x;
end;

procedure INIT;        //Ініціалізація - привласнити пересінним значення
констант
begin
  H0:=HC0;//$67452301;
  H1:=HC1;//$EFCDAB89;
  H2:=HC2;//$98BADCFE;
  H3:=HC3;//$10325476;
  H4:=HC4;//$C3D2E1F0;
  Hout:='';
end;

function PADDING(s:string;FS:integer):string;     //додавання одного біта
(1000000=128) і додавання нулів до кратності 64 байтам
var size,i:integer;
begin
```

```
size:=Length(s)*8;   //size -вхідний розмір у бітах
s:=s+char(128);    //додавання одного біта  (1000000=128)

while (Length(s) mod 64) <>0 do s:=s+#0;     //додавання нулів до кратності 64
байтам


IF  ((size mod 512) >= 448) then         // якщо хвіст перевищує 48 байт те
додати порожній блок з 64 нулів
                    begin
                      s:=s+#0;          //додавання нулів до кратності 64
                      while (Length(s) mod 64) <>0 do s:=s+#0;
                    end;

      i:=Length(s);size:=FS*8;
      while size > 0 do             //запис у кінець рядка її розмір
      begin
      s[i]:=char(byte(size));      //одержання молодшого байта
      size:=size shr 8;            //зрушення вправо на 8 біт - перенос старшого
байта на місце молодшого
      i:= i-1;
      end;
Result:=s;
end;


Procedure START(const S_IN:string);
var    A,B,C,D,E,TEMP:integer;    t,i:byte;    W:array[0..79] of integer;
begin

  t:=1;
  for i:=1 to ((Length(S_IN)) div 4) do
  begin

    W[ i-1]:=(ord(S_IN[t]) shl 24) +(ord(S_IN[t+1]) shl 16)+(ord(S_IN[t+2]) shl
8)+ord(S_IN[t+3]);
    t:=t+4;
  end;


  For t:=16 to 79 do W[t]:=ROL(W[ t-3] XOR W[ t-8] XOR W[ t-14] XOR W[ t-16],1);

  A:=H0;B:=H1;C:=H2;D:=H3;E:=H4;

   for t:=0 to 19 do
   begin
      TEMP:=ROL(A,5)+((B AND C) OR ((NOT B) AND D))       +E+K1+W[t];
      E:=D;  D:=C;  C:=ROL(B,30);  B:=A;  A:=TEMP;
   end;
   for t:=20 to 39 do
   begin
      TEMP:=ROL(A,5)+(B XOR C XOR D)                      +E+K2+W[t];
      E:=D;  D:=C;  C:=ROL(B,30);  B:=A;  A:=TEMP;
   end;
   for t:=40 to 59 do
   begin
      TEMP:=ROL(A,5)+((B AND C) OR (B AND D) OR (C AND D))+E+K3+W[t];
      E:=D;  D:=C;  C:=ROL(B,30);  B:=A;  A:=TEMP;
   end;
   for t:=60 to 79 do
   begin
      TEMP:=ROL(A,5)+(B XOR C XOR D)                      +E+K4+W[t];
      E:=D;  D:=C;  C:=ROL(B,30);  B:=A;  A:=TEMP;
   end;

   H0:=A+H0; H1:=B+H1; H2:=C+H2; H3:=D+H3; H4:=E+H4;

end;
function Work(Z:TStrings):string;
```

```
var s,s1:string;    i,L:integer;

n, nn:integer;

begin

INIT;

n:=0;
nn:=Z.Count;
   repeat

s1:=Z.Strings[n];
inc(n);
s:=s1;
     L:=length(s1);
     IF ((L<65536) and (L>0)) then
     begin
          s1:= PADDING(s,0) ;
                 i:=1;
                 L:=length(s1);
                 while i<L do
                 begin
                 START(copy(s1,i,64));
                 i:=i+64;
                 end;
     end;

     IF L =65536  then begin
                 i:=1;
                 L:=length(s1);
                 while i<L do
                 begin
                 START(copy(s1,i,64));
                 i:=i+64;
                 end;

                 end;

          until n=nn;


Hout:=inttohex(H0,8)+' '+inttohex(H1,8)+' '+inttohex(H2,8)+' '+inttohex(H3,8)+'
'+inttohex(H4,8);
s1:=Hout;
Result:=s1;

end;

Function ScanDir(Dir:string):string;
var   SearchRec:TSearchRec; //scan_result :string;
begin

IF StopScan then exit;
if Dir<>'' then if Dir[length(Dir)]<>'\' then Dir:=Dir+'\';

if FindFirst(Dir+'*.*', faAnyFile, SearchRec)=0   then
repeat
  if (SearchRec.name='.') or (SearchRec.name='..')   then continue;

until FindNext(SearchRec)<>0;
FindClose(SearchRec);

end;

end.
```

Файл FGIntRSA.PAS - реалізація алгоритму RSA

```
Unit FGIntRSA;

Interface

Uses Windows, SysUtils, Controls, FGInt;

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


Implementation


{$H+}

// шифруємо строку по алгоритму RSA, P^exp mod modb = E

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

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

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


// Дешифруємо строку по алгоритму RSA  E^exp mod modb = D

Procedure RSADecrypt(E : String; Var exp, modb, d_p, d_q, p, q : TFGInt; Var D :
String);
Var
```

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

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

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



// підписуємо строку по алгоритму RSA, M^d mod n = S
```

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


// Перевіряємо цифровий підпис
// If M = S^e mod n then ok:=true else ok:=false

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

End.
```

Файл FGIntRSA.PAS - генерація сертифікату та ключа для цифрового підпису RSA

```
Unit FGIntPrimeGeneration;

Interface

Uses Windows, SysUtils, Controls, FGInt;

Procedure FGIntAdd(Const FGInt1, FGInt2 : TFGInt; Var Sum : TFGInt);
Procedure FGIntCopy(Const FGInt1 : TFGInt; Var FGInt2 : TFGInt);
Procedure FGIntTrialDiv9999(Const FGInt : TFGInt; Var ok : boolean);
Procedure FGIntPrimetest(Var FGIntp : TFGInt; nrRMtests : integer; Var ok :
boolean);
Procedure FGIntDestroy(Var FGInt : TFGInt);
Procedure PrimeSearch(Var GInt : TFGInt);

Implementation

Var   primes : Array[1..1228] Of integer =
      (3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71,
73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127,
      131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199,
211, 223, 227, 229, 233, 239, 241, 251,
      257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347,
349, 353, 359, 367, 373, 379, 383, 389,
      397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479,
487, 491, 499, 503, 509, 521, 523, 541,
      547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631,
641, 643, 647, 653, 659, 661, 673, 677,
      683, 691, 701, 709, 719, 727, 733, 739, 743, 751, 757, 761, 769, 773, 787,
797, 809, 811, 821, 823, 827, 829, 839,
      853, 857, 859, 863, 877, 881, 883, 887, 907, 911, 919, 929, 937, 941, 947,
953, 967, 971, 977, 983, 991, 997, 1009,
      1013, 1019, 1021, 1031, 1033, 1039, 1049, 1051, 1061, 1063, 1069, 1087,
1091, 1093, 1097, 1103, 1109, 1117, 1123,
      1129, 1151, 1153, 1163, 1171, 1181, 1187, 1193, 1201, 1213, 1217, 1223,
1229, 1231, 1237, 1249, 1259, 1277, 1279,
      1283, 1289, 1291, 1297, 1301, 1303, 1307, 1319, 1321, 1327, 1361, 1367,
1373, 1381, 1399, 1409, 1423, 1427, 1429,
      1433, 1439, 1447, 1451, 1453, 1459, 1471, 1481, 1483, 1487, 1489, 1493,
1499, 1511, 1523, 1531, 1543, 1549, 1553,
      1559, 1567, 1571, 1579, 1583, 1597, 1601, 1607, 1609, 1613, 1619, 1621,
1627, 1637, 1657, 1663, 1667, 1669, 1693,
      1697, 1699, 1709, 1721, 1723, 1733, 1741, 1747, 1753, 1759, 1777, 1783,
1787, 1789, 1801, 1811, 1823, 1831, 1847,
      1861, 1867, 1871, 1873, 1877, 1879, 1889, 1901, 1907, 1913, 1931, 1933,
1949, 1951, 1973, 1979, 1987, 1993, 1997,
      1999, 2003, 2011, 2017, 2027, 2029, 2039, 2053, 2063, 2069, 2081, 2083,
2087, 2089, 2099, 2111, 2113, 2129, 2131,
      2137, 2141, 2143, 2153, 2161, 2179, 2203, 2207, 2213, 2221, 2237, 2239,
2243, 2251, 2267, 2269, 2273, 2281, 2287,
      2293, 2297, 2309, 2311, 2333, 2339, 2341, 2347, 2351, 2357, 2371, 2377,
2381, 2383, 2389, 2393, 2399, 2411, 2417,
      2423, 2437, 2441, 2447, 2459, 2467, 2473, 2477, 2503, 2521, 2531, 2539,
2543, 2549, 2551, 2557, 2579, 2591, 2593,
      2609, 2617, 2621, 2633, 2647, 2657, 2659, 2663, 2671, 2677, 2683, 2687,
2689, 2693, 2699, 2707, 2711, 2713, 2719,
      2729, 2731, 2741, 2749, 2753, 2767, 2777, 2789, 2791, 2797, 2801, 2803,
2819, 2833, 2837, 2843, 2851, 2857, 2861,
      2879, 2887, 2897, 2903, 2909, 2917, 2927, 2939, 2953, 2957, 2963, 2969,
2971, 2999, 3001, 3011, 3019, 3023, 3037,
      3041, 3049, 3061, 3067, 3079, 3083, 3089, 3109, 3119, 3121, 3137, 3163,
3167, 3169, 3181, 3187, 3191, 3203, 3209,
      3217, 3221, 3229, 3251, 3253, 3257, 3259, 3271, 3299, 3301, 3307, 3313,
3319, 3323, 3329, 3331, 3343, 3347, 3359,
      3361, 3371, 3373, 3389, 3391, 3407, 3413, 3433, 3449, 3457, 3461, 3463,
3467, 3469, 3491, 3499, 3511, 3517, 3527,
```

3529, 3533, 3539, 3541, 3547, 3557, 3559, 3571, 3581, 3583, 3593, 3607,
3613, 3617, 3623, 3631, 3637, 3643, 3659,
3671, 3673, 3677, 3691, 3697, 3701, 3709, 3719, 3727, 3733, 3739, 3761,
3767, 3769, 3779, 3793, 3797, 3803, 3821,
3823, 3833, 3847, 3851, 3853, 3863, 3877, 3881, 3889, 3907, 3911, 3917,
3919, 3923, 3929, 3931, 3943, 3947, 3967,
3989, 4001, 4003, 4007, 4013, 4019, 4021, 4027, 4049, 4051, 4057, 4073,
4079, 4091, 4093, 4099, 4111, 4127, 4129,
4133, 4139, 4153, 4157, 4159, 4177, 4201, 4211, 4217, 4219, 4229, 4231,
4241, 4243, 4253, 4259, 4261, 4271, 4273,
4283, 4289, 4297, 4327, 4337, 4339, 4349, 4357, 4363, 4373, 4391, 4397,
4409, 4421, 4423, 4441, 4447, 4451, 4457,
4463, 4481, 4483, 4493, 4507, 4513, 4517, 4519, 4523, 4547, 4549, 4561,
4567, 4583, 4591, 4597, 4603, 4621, 4637,
4639, 4643, 4649, 4651, 4657, 4663, 4673, 4679, 4691, 4703, 4721, 4723,
4729, 4733, 4751, 4759, 4783, 4787, 4789,
4793, 4799, 4801, 4813, 4817, 4831, 4861, 4871, 4877, 4889, 4903, 4909,
4919, 4931, 4933, 4937, 4943, 4951, 4957,
4967, 4969, 4973, 4987, 4993, 4999, 5003, 5009, 5011, 5021, 5023, 5039,
5051, 5059, 5077, 5081, 5087, 5099, 5101,
5107, 5113, 5119, 5147, 5153, 5167, 5171, 5179, 5189, 5197, 5209, 5227,
5231, 5233, 5237, 5261, 5273, 5279, 5281,
5297, 5303, 5309, 5323, 5333, 5347, 5351, 5381, 5387, 5393, 5399, 5407,
5413, 5417, 5419, 5431, 5437, 5441, 5443,
5449, 5471, 5477, 5479, 5483, 5501, 5503, 5507, 5519, 5521, 5527, 5531,
5557, 5563, 5569, 5573, 5581, 5591, 5623,
5639, 5641, 5647, 5651, 5653, 5657, 5659, 5669, 5683, 5689, 5693, 5701,
5711, 5717, 5737, 5741, 5743, 5749, 5779,
5783, 5791, 5801, 5807, 5813, 5821, 5827, 5839, 5843, 5849, 5851, 5857,
5861, 5867, 5869, 5879, 5881, 5897, 5903,
5923, 5927, 5939, 5953, 5981, 5987, 6007, 6011, 6029, 6037, 6043, 6047,
6053, 6067, 6073, 6079, 6089, 6091, 6101,
6113, 6121, 6131, 6133, 6143, 6151, 6163, 6173, 6197, 6199, 6203, 6211,
6217, 6221, 6229, 6247, 6257, 6263, 6269,
6271, 6277, 6287, 6299, 6301, 6311, 6317, 6323, 6329, 6337, 6343, 6353,
6359, 6361, 6367, 6373, 6379, 6389, 6397,
6421, 6427, 6449, 6451, 6469, 6473, 6481, 6491, 6521, 6529, 6547, 6551,
6553, 6563, 6569, 6571, 6577, 6581, 6599,
6607, 6619, 6637, 6653, 6659, 6661, 6673, 6679, 6689, 6691, 6701, 6703,
6709, 6719, 6733, 6737, 6761, 6763, 6779,
6781, 6791, 6793, 6803, 6823, 6827, 6829, 6833, 6841, 6857, 6863, 6869,
6871, 6883, 6899, 6907, 6911, 6917, 6947,
6949, 6959, 6961, 6967, 6971, 6977, 6983, 6991, 6997, 7001, 7013, 7019,
7027, 7039, 7043, 7057, 7069, 7079, 7103,
7109, 7121, 7127, 7129, 7151, 7159, 7177, 7187, 7193, 7207, 7211, 7213,
7219, 7229, 7237, 7243, 7247, 7253, 7283,
7297, 7307, 7309, 7321, 7331, 7333, 7349, 7351, 7369, 7393, 7411, 7417,
7433, 7451, 7457, 7459, 7477, 7481, 7487,
7489, 7499, 7507, 7517, 7523, 7529, 7537, 7541, 7547, 7549, 7559, 7561,
7573, 7577, 7583, 7589, 7591, 7603, 7607,
7621, 7639, 7643, 7649, 7669, 7673, 7681, 7687, 7691, 7699, 7703, 7717,
7723, 7727, 7741, 7753, 7757, 7759, 7789,
7793, 7817, 7823, 7829, 7841, 7853, 7867, 7873, 7877, 7879, 7883, 7901,
7907, 7919, 7927, 7933, 7937, 7949, 7951,
7963, 7993, 8009, 8011, 8017, 8039, 8053, 8059, 8069, 8081, 8087, 8089,
8093, 8101, 8111, 8117, 8123, 8147, 8161,
8167, 8171, 8179, 8191, 8209, 8219, 8221, 8231, 8233, 8237, 8243, 8263,
8269, 8273, 8287, 8291, 8293, 8297, 8311,
8317, 8329, 8353, 8363, 8369, 8377, 8387, 8389, 8419, 8423, 8429, 8431,
8443, 8447, 8461, 8467, 8501, 8513, 8521,
8527, 8537, 8539, 8543, 8563, 8573, 8581, 8597, 8599, 8609, 8623, 8627,
8629, 8641, 8647, 8663, 8669, 8677, 8681,
8689, 8693, 8699, 8707, 8713, 8719, 8731, 8737, 8741, 8747, 8753, 8761,
8779, 8783, 8803, 8807, 8819, 8821, 8831,
8837, 8839, 8849, 8861, 8863, 8867, 8887, 8893, 8923, 8929, 8933, 8941,
8951, 8963, 8969, 8971, 8999, 9001, 9007,
9011, 9013, 9029, 9041, 9043, 9049, 9059, 9067, 9091, 9103, 9109, 9127,
9133, 9137, 9151, 9157, 9161, 9173, 9181,

```
      9187, 9199, 9203, 9209, 9221, 9227, 9239, 9241, 9257, 9277, 9281, 9283,
9293, 9311, 9319, 9323, 9337, 9341, 9343,
      9349, 9371, 9377, 9391, 9397, 9403, 9413, 9419, 9421, 9431, 9433, 9437,
9439, 9461, 9463, 9467, 9473, 9479, 9491,
      9497, 9511, 9521, 9533, 9539, 9547, 9551, 9587, 9601, 9613, 9619, 9623,
9629, 9631, 9643, 9649, 9661, 9677, 9679,
      9689, 9697, 9719, 9721, 9733, 9739, 9743, 9749, 9767, 9769, 9781, 9787,
9791, 9803, 9811, 9817, 9829, 9833, 9839,
      9851, 9857, 9859, 9871, 9883, 9887, 9901, 9907, 9923, 9929, 9931, 9941,
9949, 9967, 9973);

{$H+}

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

```
            If size < size1 Then
            Begin
               SetLength(Sum.Number, size + 1);
            End;
            Sum.Number[0] := size;
            Sum.Sign := FGInt1.Sign;
         End;
      End;
   End;
End;

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

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

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

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

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

End.
```

Файл DES.pas - реалізація алгоритму шифрування DES

```
unit DES;

interface

Uses Windows, Classes, SysUtils, Math, Dialogs;

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

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

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

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

Const
DES_PC1:Array[0..55] Of Byte = (57,49,41,33,25,17,9,
                                1,58,50,42,34,26,18,
                                10,2,59,51,43,35,27,
                                19,11,3,60,52,44,36,
                                63,55,47,39,31,23,15,
                                7,62,54,46,38,30,22,
                                14,6,61,53,45,37,29,
                                21,13,5,28,20,12,4);

DES_PC2:Array[0..47] Of Byte = (14,17,11,24,1,5,
                                3,28,15,6,21,10,
                                23,19,12,4,26,8,
                                16,7,27,20,13,2,
                                41,52,31,37,47,55,
                                30,40,51,45,33,48,
                                44,49,39,56,34,53,
                                46,42,50,36,29,32);

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Procedure CopyBits(Var Dest:TBitString; Source:TBitString; NBits:Integer);
```

```
Function ConcatBits(Bits:Array Of TBitString):TBitString;

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

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

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

implementation

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

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

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

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

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

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

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

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

Function BinToAnsiStr(Bits:TBitString):String;
Var
I:Integer;
B:TBitString;
Begin
Result:='';
SetLength(B,8);
I:=0;
While I<=Length(Bits)-8 Do
```

```
  Begin
  CopyMemory(B,Ptr(Integer(Bits)+I),8);
  Result:=Result+Char(BinToInt(B));
  Inc(I,8);
  End;
End;

Function GetPermutedKey(Key:TBitString):TBitString;
Var
I:Integer;
Begin
SetLength(Result,Length(DES_PC1));
For I:=0 To Length(DES_PC1)-1 Do
  Result[I]:=Key[DES_PC1[I]-1];
End;

Function GetPermutedKey2(Key:TBitString):TBitString;
Var
I:Integer;
Begin
SetLength(Result,Length(DES_PC2));
For I:=0 To Length(DES_PC2)-1 Do
  Result[I]:=Key[DES_PC2[I]-1];
End;

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

Function GetConcatKey(Key:TSplitKey):TConcatKey;
Var
I:Integer;
Temp:TBitString;
Begin
For I:=0 To 15 Do
  Begin
  SetLength(Result[I],56);
```

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

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

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

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

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

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

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

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

Function GetSBox(Index:Integer; T:TBitString):TBitString;
Var
Val,Row,Col:Integer;
Temp:TBitString;
Begin
```

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

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

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

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

Function DESDecode(S,Key:String):TBitString;
Var
I:Integer;
K:TBitString;
M:TBitString;
```

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

end.
```

Файл about.pas - довідка про програму

```
unit about;

interface

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

type
  Tfrm_about = class(TForm)
    Image1: TImage;
    Memo1: TMemo;
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frm_about: Tfrm_about;

implementation

{$R *.dfm}

procedure Tfrm_about.Button1Click(Sender: TObject);
begin
frm_about.Close;
end;

procedure Tfrm_about.FormCreate(Sender: TObject);
begin
Memo1.Clear;
Memo1.Lines.Add('МАГІСТЕРСЬКА РОБОТА');
Memo1.Lines.Add('');
Memo1.Lines.Add('на  тему:');
Memo1.Lines.Add('');
Memo1.Lines.Add('Дослідження та програмна реалізація системи повідомлень
електронної пошти');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
Memo1.Lines.Add('Керівник: Смірнов С.А.');
Memo1.Lines.Add('');
Memo1.Lines.Add('Розробив: студент Науменко Ігор Олегович ');
Memo1.Lines.Add('                    гр. КІ-21М-1,4');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
Memo1.Lines.Add('м. Кропивницький 2022');
Memo1.Lines.Add('');
Memo1.Lines.Add('');

end;

end.
```