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
“____” _____ 2025 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки для забезпечення конфіденційності інформації на основі технології NTRUEncrypt”

Виконав здобувач вищої освіти
IV курсу, групи КБ-21
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Полях О.В.
«____» _____ 2025 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Буравченко К.О.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

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

Спеціальність *125 “*Кібербезпека*”*

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

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2025 року

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

Поляху Олександру Володимировичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки для забезпечення конфіденційності інформації на основі технології NTRUEncrypt*

2. Керівник роботи *Буравченко Костянтин Олегович, канд. техн. наук, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 57-02 від 17.01.2025 року

3. Строк подання студентом роботи до захисту *23.05.2025 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи кібербезпеки для забезпечення конфіденційності інформації на основі технології NTRUEncrypt*

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

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи кібербезпеки в промислову експлуатацію.

6. Висновки

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

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

7. Дата видачі завдання « 17 » січня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2025 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2025 р.	
3.	Розробка моделі компонента	20.03.2025 р.	
4.	Розробка структур даних	25.03.2025 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2025 р.	
6.	Програмування алгоритмів	10.04.2025 р.	
7.	Оформлення ПЗ	17.04.2025 р.	
8.	Попередній захист роботи	23.05.2025 р.	

Дата видачі завдання
« 17 » січня 2025 р.

Підпис керівника
_____ Буравченко К.О.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

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

АНОТАЦІЯ

Полях О.В. Програмне забезпечення системи кібербезпеки для забезпечення конфіденційності інформації на основі технології NTRUEncrypt. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки для забезпечення конфіденційності інформації на основі технології NTRUEncrypt.

Метою розробки є програмне забезпечення системи кібербезпеки для забезпечення конфіденційності інформації на основі технології NTRUEncrypt.

Результат роботи – програмна реалізація системи кібербезпеки для забезпечення конфіденційності інформації на основі технології NTRUEncrypt.

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

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

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

Програму розроблено в середовищі Python.

Ключові слова: кібербезпека, NTRUEncrypt

ABSTRACT

Polyakh O.V. Software for a cybersecurity system to ensure information confidentiality based on NTRUEncrypt technology. 125 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed, which is intended for a cybersecurity system to ensure information confidentiality based on NTRUEncrypt technology.

The purpose of the development is software for a cybersecurity system to ensure information confidentiality based on NTRUEncrypt technology.

The result of the work is a software implementation of a cybersecurity system to ensure information confidentiality based on NTRUEncrypt technology.

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

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

The program can be used on PCs with Windows 10/11.

The program is developed in the Python environment.

Keywords: cybersecurity, NTRUEncrypt

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

БД	–	база даних
ПЗ	–	програмне забезпечення
USB	–	universal serial bus

ВСТУП

Актуальність теми. Програмне забезпечення для шифрування стає все більш важливим, оскільки хакерам, шахрайським програмам і навіть медіа-гігантам стало легше, ніж будь-коли, отримати доступ до ваших особистих даних і навіть даних.

Для особистих користувачів це може означати будь-що, від ваших особистих файлів до всіх ваших фотографій, які можуть бути розкриті або навіть видалені в результаті атаки зловмисного програмного забезпечення, що може бути руйнівним.

Для компаній це навіть серйозніше, оскільки вони мають юридичний обов'язок захищати записи клієнтів, і в разі злому розкриття будь-якої конфіденційної чи конфіденційної інформації може призвести до великих штрафів, збитку бренду та довгострокових фінансових втрат.

Незважаючи на те, що доступно багато рішень безпеки, від безкоштовних інструментів захисту від зловмисного програмного забезпечення для домашніх споживачів до хмарних антивірусів для бізнесу, навіть проста людська помилка може підірвати ці спроби захистити дані.

Програмне забезпечення для шифрування вже розгорнуто деякими великими корпораціями та урядовими установами для захисту даних, але воно також доступне та тепер доступне для більшого кола користувачів.

Після належного налаштування, навіть якщо ваше програмне забезпечення безпеки вийде з ладу, злодії/хакери/шлікери все одно виявитимуть майже неможливим щось зробити із будь-якими зашифрованими даними, які можуть бути розкриті.

Одними з найбільш стійких методів захисту інформації є використання шифрування. Серед широкого спектру різних алгоритмів шифрування, останній час проводяться активні розробки з NTRUEncrypt, який є найбільш швидким та

перспективним асиметричним криптоалгоритмом.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки для забезпечення конфіденційності інформації на основі технології NTRUEncrypt.

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

– Огляд існуючих систем для забезпечення конфіденційності інформації на основі технології NTRUEncrypt.

– Дослідження системи кібербезпеки для забезпечення конфіденційності інформації на основі технології NTRUEncrypt.

– Програмна реалізація системи кібербезпеки для забезпечення конфіденційності інформації на основі технології NTRUEncrypt.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі для забезпечення конфіденційності інформації на основі технології NTRUEncrypt.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки для забезпечення конфіденційності інформації на основі технології NTRUEncrypt, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Сучасні алгоритми шифрування на світлові роки перевершують старомодні шифри. Їхні результати не мають видимого відношення до даних, які надійшли, і злам сучасного алгоритму шифрування зайняв би неймовірно багато часу. Офіційним алгоритмом шифрування уряду США є Advanced Encryption Standard (AES). Алгоритм Blowfish Брюса Шнайера є ще одним часто використовуваним методом.

AES, Blowfish і багато поширених алгоритмів шифрування є симетричними, тобто той самий ключ використовується для шифрування та дешифрування даних. Якщо ви хочете поділитися зашифрованим файлом, ви повинні безпечно (і окремо) передати ключ одержувачу. Ця ключова передача – мрія сценариста, ідеальний вихід для хитрощів. Криптографія інфраструктури відкритих ключів (PKI) дозволяє уникнути слабких місць безпеки, усуваючи потребу в передачі ключа.

У цій системі, якщо я хочу надіслати вам файл, я шукаю ваш відкритий ключ і шифрую файл за допомогою нього. Ви використовуєте свій особистий ключ, щоб розшифрувати файл. І навпаки, якщо я хочу довести вам, що документ надійшов від мене і не був змінений, я шифрую його своїм особистим ключем. Той факт, що ви можете розшифрувати його за допомогою мого відкритого ключа, доводить, що я підписав його своїм цифровим ключем. І ніхто не може змінити файл, не зробивши цифровий підпис недійсним.

Більшість програм для шифрування поділяються на два табори. Деякі, наприклад AxCrypt, EncryptionSafe, Encrypto та Xecrets, застосовують шифрування до окремих файлів. Інші, наприклад NordLocker і Steganos Safe, створюють зашифрований контейнер, який часто називають сховищем, із

захищеними всіма файлами в сховищі. Дуже небагато, серед яких Cypherix Cryptainer і Folder Lock, пропонують обидва варіанти.

1.2 Область застосування

Занадто складний інструмент шифрування не звикне. Те саме стосується інструменту, який є занадто дорогим для звичайного користувача. Розробляєме програмне забезпечення усуває обидві ці проблеми. З простим користувальницьким інтерфейсом і без необхідності інсталяції це навряд чи може бути простіше. І ви можете отримати повну потужність його шифрування, не заплативши ні копійки, або оформити недорогу підписку, щоб отримати ще більшу зручність використання. Шифруйте файли, розшифровуйте їх, редагуйте їх на місці, безпечно діліться ними.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки для забезпечення конфіденційності інформації на основі технології NTRUEncrypt, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

1. Microsoft OneDrive

Microsoft OneDrive може бути не першим варіантом, який спадає на думку людям, коли вони думають про програмне забезпечення для шифрування, але багато людей не усвідомлюють, що воно має власну безпечну та зашифровану область зберігання файлів у формі персонального сховища.

Ви можете вибрати, які файли та папки включити, і вони будуть заблоковані та зашифровані. Однак, хоча за замовчуванням доступ до них маєте лише ви, існує можливість надати спільний доступ до певних файлів і папок іншим користувачам.

Це чудово, тому що OneDrive дуже добре працює як хмарне сховище для малого бізнесу та особистих сховищ, тому можливість шифрувати файли є чудовим бонусом.

Крім того, OneDrive є чудовим вибором для тих, хто хоче користуватися послугами Microsoft, оскільки він пропонує чітку інтеграцію з Outlook.com, популярною платформою електронної пошти компанії, наприклад. OneDrive також чудово поєднується з Windows, і є вибір розумних мобільних програм для полегшення доступу в дорозі.

Для особистого користування ціна починається приблизно від 7 доларів США / 6 фунтів стерлінгів / 10 австралійських доларів на місяць або 70 доларів США / 65 фунтів стерлінгів / 100 австралійських доларів на рік.+

2. VeraCrypt

Безкоштовне шифрування для всіх.

VeraCrypt – один із найпопулярніших інструментів безпеки, який забезпечує шифрування важливих даних корпоративного рівня.

Система досить проста у використанні, і все, що вона насправді робить, це додавати зашифровані паролі до ваших даних і розділів. Все, що вам потрібно зробити, це надати інструменту кілька деталей про ваші дані, як-от розмір тому, розташування та вказані алгоритми хешування, і тоді програма зробить свою справу.

Що також чудово у VeraCrypt, це те, що він захищений від атак грубої сили, тож вам ніколи не доведеться турбуватися про те, що хакери розшифрують ваші паролі та інші конфіденційні дані. Базова версія програмного забезпечення також є абсолютно безкоштовною.

Cypherix Secure IT

Secure IT від Cypherix – це програма для шифрування файлів, яка також стискає ваші файли. Це означає, що це може зайняти трохи більше часу, ніж деякі інші програми, але це означає, що після цього вашими файлами стане легше керувати.

Secure IT 2000 пропонує шифрування файлів і папок, стиснення та подрібнення файлів в одному з обробкою командного рядка та безпечними електронними листами. Він працює на всіх версіях Windows (32-розрядних, а також 64-розрядних).

Головний пароль потрібен для доступу до будь-яких файлів, що може бути дещо обмеженим, якщо ви віддаєте перевагу мати різні паролі для різних файлів, не в останню чергу, якщо ви хочете поділитися деякими з них із родиною та/або друзями.

AxCrypt

Шифрування для невеликих команд і окремих осіб. Хоча безкоштовне програмне забезпечення може бути зручним для деяких, воно не завжди таке потужне, як преміум-пропозиції, і AxCrypt є хорошим вибором, якщо ви хочете щось надійне. Програмне забезпечення було розроблено спеціально для окремих

осіб і невеликих команд у межах бізнесу.

Він забезпечує надійну безпеку з файлами, захищеними 128-бітним або 256-бітним шифруванням AES, яке має завадити будь-яким зловмисникам. Також є можливості хмарного сховища – програмне забезпечення автоматично захищатиме файли, збережені на таких службах, як Google Drive і Dropbox.

AxCrypt є повністю багатомовним і може працювати з такими мовами, як нідерландська, французька, німецька, італійська, корейська, іспанська, шведська, російська та португальська – із розширеною підтримкою, запланованою на майбутнє. Окрім цього, є керування паспортами, і ви можете отримати доступ до своїх зашифрованих файлів через додаток для смартфона.

Існує безкоштовна версія, але вона дуже обмежена, і, мабуть, її краще розглядати як спосіб випробувати програмне забезпечення та допомогти звикнути до інтерфейсу та основних функцій.

Folder Lock

Незважаючи на те, що важливо захистити активи на комп'ютерах компанії, важливо також захистити будь-який пристрій, який зберігає важливі дані. Наприклад, більшість співробітників мають доступ до електронної пошти компанії та інших облікових записів на своїх смартфонах, і їх потрібно захистити.

Folder Lock – хороший варіант, коли справа доходить до додавання шифрування до ваших мобільних пристроїв. Додаток може захистити ваші особисті файли, фотографії, відео, контакти, картки гаманця, нотатки та аудіозаписи, що зберігаються на вашому телефоні.

Також є деякі інші приховані функції безпеки. Існує не тільки шифрування, але ви також можете встановити пароль-приманку, захист від хакерів, реєструвати неавторизовані спроби входу, створювати резервні копії всіх своїх паролів і отримувати сповіщення про можливі атаки грубої сили.

Базову програму можна завантажити безкоштовно, а професійна версія доступна за одноразову плату, яка відкриває більш розширені та корисні функції безпеки приблизно за 40 доларів США / 35 фунтів стерлінгів / 60 австралійських

доларів.

Інше програмне забезпечення для шифрування

Concealer – це програма для шифрування файлів, спеціально призначена для комп'ютерів Apple Mac. Замість того, щоб шифрувати всі файли на вашому жорсткому диску, він надає зашифровану область, куди ви можете перетягувати файли, а це означає, що ви повинні переконатися, що ви видалили оригінальну копію, щоб уникнути наявності як зашифрованих, так і незашифрованих версій на вашому жорсткому диску. Ви також можете вибрати 128- або 256-бітне шифрування.

Renee File Protector – це ще одна частина програмного забезпечення для шифрування файлів для Windows, але вона дозволяє мати різні паролі для різних файлів або папок, фактично створюючи кілька рівнів безпеки. Крім того, ви можете просто залишити головний пароль, якщо потрібно. І якщо ви забудете будь-який із своїх паролів, Renee File Protector може надати підказки, щоб дозволити вам їх відновити, що дуже зручно.

SensiGuard не просто шифрує ваші файли та папки, але й приховує їх, щоб вони залишалися прихованими від неавторизованих користувачів, що є потенційно корисною функцією. Крім того, ви також можете безпечно знищувати файли, що означає, що хакерам ще важче знайти на вашому жорсткому диску конфіденційні файли, які ви вже зашифрували або видалили.

Яке програмне забезпечення для шифрування найкраще для вас?

Вирішуючи, яке програмне забезпечення для шифрування використовувати, спершу подумайте про ваші фактичні потреби, оскільки бюджетне програмне забезпечення може надавати лише базові параметри, тому, якщо вам знадобляться розширені інструменти, ви можете виявити, що дорожча платформа буде набагато вигіднішою. Крім того, програмне забезпечення вищого класу зазвичай може задовольнити будь-які потреби, тому переконайтеся, що ви добре уявляєте, які функції, на вашу думку, можуть знадобитися від вашого програмного забезпечення для шифрування.

Як ми тестували найкраще програмне забезпечення для шифрування

Щоб перевірити найкраще програмне забезпечення для шифрування, ми спочатку створили обліковий запис на відповідній програмній платформі, а потім протестували службу, щоб побачити, як програмне забезпечення можна використовувати для різних цілей і в різних ситуаціях. Мета полягала в тому, щоб підштовхнути кожну платформу програмного забезпечення шифрування до того, наскільки корисними є її базові інструменти, а також наскільки легко було впоратися з будь-якими більш просунутими інструментами.

2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки та мови програмування

Python – динамічна інтерпретована об’єктно-орієнтована скриптова мова програмування із строгою динамічною типізацією. Офіційний сайт мови програмування Python https://www.python.org/. Python – багатоцільова мова програмування, яка дозволяє писати код, що добре читається. Відносний лаконізм мови Python дозволяє створити програму, яка буде набагато коротше свого аналога, написаного на іншій мові. Python – багатоплатформова мова програмування. Це означає, що програми на Python можна запускати в різних операційних системах без будь-яких змін.

Ще однією перевагою Python є його стандартна бібліотека, яка встановлюється разом з Python і містить готові інструменти для роботи з операційною системою, веб-сторінками, базами даних, різними форматами даних, для побудови графічного інтерфейсу програм тощо. Програми, написані на мові програмування Python, можуть бути як невеликими скриптами, так і складними системами. Python абсолютно безкоштовний.

Швидкість виконання коду Python

Один з можливих недоліків Python – швидкість виконання коду. Python не є компільованою мовою. Код на Python спочатку компілюється у внутрішній

байт-код, який потім виконується інтерпретатором Python. У більшості випадків при використанні Python виходять програми повільніші в порівнянні з такими мовами, як C.

Втім, сучасні комп'ютери мають таку обчислювальну потужність, що для більшості застосунків швидкість розробки важливіша швидкості виконання, а програми на Python зазвичай пишуться набагато швидше.

Окрім того, Python легко розширюється модулями, написаними на C або C++. Такі модулі можуть використовуватися для виконання частин програми, що створюють інтенсивне навантаження на процесор.

Використання Python

Python використовується для різних цілей: для створення ігор і веб-застосунків, розробки внутрішніх інструментів для різноманітних проектів. Мова також широко застосовується в науковій області для досліджень і розв'язування прикладних завдань.

Застосування мови програмування Python:

1. BitTorrent – протокол для обміну даними.

2. Ubuntu Software Center – вільне програмне забезпечення для пошуку, установки і видалення пакунків в системі Ubuntu Linux.

3. Blender – програма для створення тривимірної комп'ютерної графіки, що включає засоби моделювання, анімації, вимальовування, пост-обробки відео, а також створення відеоігор.

4. GIMP – растровий графічний редактор, із підтримкою векторної графіки.

5. World of Tanks.

6. Вільна енциклопедія Вікіпедія.

7. Пошукова система Google.

8. DropBox – файловий хостинг, що включає персональне хмарне сховище, синхронізацію файлів і програму-клієнт.

9. YouTube – популярне відеосховище.

Версії Python

Мови програмування з часом змінюються – розробники додають в них нові можливості, а також виправляють помилки. Так з'являються різні версії мови. Наприклад, код написаний на Python 2 у більшості випадків не буде працювати у версії Python 3 без внесення додаткових змін.

Процесор є найважливішим компонентом в комп'ютері. Одна з основних функцій процесора – це обробка даних згідно комп'ютерної програми, яка є списком інструкцій, шляхом виконання арифметичних і логічних операцій над фрагментами даних.

Кожна інструкція в програмі – це команда, яка «повідомляє» процесору, яку операцію він повинен виконати. Процесор комп'ютера може розуміти лише ті інструкції, які написані на машинній мові. Машинна мова – це штучна мова, створена для передачі команд комп'ютеру. За допомогою машинної мови створюються ефективні програми, оскільки розробник отримує доступ до всіх можливостей процесора. Машинна мова – мова низького рівня.

Інструкція машинної мови існує для кожної операції, яку процесор здатний виконати – є інструкція для додавання чисел, є інструкція для віднімання чисел і т.д. Увесь набір інструкцій, який центральний процесор може виконати, відомий як набір інструкцій процесора.

Наприклад, у вас є певна програма, яка зберігається на диску вашого комп'ютера. Для виконання програми, ви здійснюєте подвійний клік на значку програми. Це змушує програму копіюватися з диска в оперативну пам'ять, після чого процесор комп'ютера виконує копію програми, яка знаходиться в оперативній пам'яті.

Коли процесор виконує інструкції програми, він бере участь у процесі, який є відомим як цикл fetch – decode – execute (отримати – декодувати – виконати). Цей цикл виконується для кожної інструкції у програмі і складається з трьох кроків:

Отримати

Програма – це послідовність інструкцій на машинній мові. Першим кроком циклу є завантаження (отримання) наступної інструкції з пам'яті в процесор.

Декодувати

Інструкція машинної мови – це двійкове число, яке представляє команду, що повідомляє процесору виконати певну операцію. На цьому кроці процесор декодує інструкцію, яку було «витягнуто» з пам'яті, для визначення того, яка операція повинна виконуватись.

Виконати

Останній крок циклу – виконати операцію.

Хоча процесор комп'ютера розуміє тільки машинну мову, людині непрактично писати програми на машинній мові. Така програма може мати тисячі або навіть мільйони бінарних інструкцій, і написання такої програми буде дуже обтяжливим процесом.

З цієї причини була створена мова асемблера як альтернатива машинній мові. Замість використання двійкових чисел для написання інструкцій, мова асемблера використовує короткі слова, відомі як мнемокоди.

Незважаючи на те, що мова асемблера не вимагає двійкових інструкцій, як у випадку машинної мови, проте вона вимагає високих знань про процесор. Використовуючи мову асемблера, навіть для найпростішої програми, необхідно написати велику кількість інструкцій.

Мова програмування високого рівня дозволяє створювати складні програми, не знаючи, як працює процесор, і не записуючи великої кількості інструкцій низького рівня. Крім того, більшість мов програмування високого рівня використовують слова, які легко зрозуміти.

Python – одна із популярних сучасних мов програмування високого рівня. Python – інтерпретована мова програмування. Python – це високорівнева інтерпретована мова програмування, на відміну від C++, яка є прикладом

компільованої мови програмування. Назва Python відноситься як до мови програмування, так і до інтерпретатора – комп'ютерної програми, яка зчитує початковий код (написаний на Python) і виконує інструкції (команди).

Для перекладу мови високого рівня на машинну мову доступні два типи програм:

1. Компілятор.

2. Інтерпретатор.

Завантаження Python

Версії інтерпретатора Python для різних операційних систем доступні для безкоштовного завантаження за адресою https://www.python.org/downloads.

Середовище програмування для Python

Для написання програм використовують текстові редактори або інтегровані середовища розробки, які включають в себе різні інструменти для роботи з кодом: засіб для написання коду (текстовий редактор), інтерактивний інтерпретатор, відлагоджувач тощо.

Текстові редактори та інтегровані середовища програмування для Python:

– IDLE – стандартний редактор Python. Встановлюється разом з Python для користувачів Windows, окремим пакунком для користувачів Linux.

– Notepad++ – безкоштовний текстовий редактор початкового коду, який підтримує велику кількість мов, в тому числі і Python. Лише для користувачів Windows.

– Visual Studio Code – це легкий, але потужний редактор початкового коду, який розповсюджується безкоштовно і доступний у версіях для платформ Linux, Windows і macOS.

– PyScripter – інтегроване середовище розробки для мови програмування Python. Для користувачів Windows. Поширюється безкоштовно.

– Wing IDE 101 – вільне інтегроване середовище для Python, розроблене для навчання програмістів-початківців. Для користувачів Linux, Windows і macOS. Поширюється безкоштовно.

– Geany – вільний текстовий редактор з базовими елементами інтегрованого середовища розробки, доступний для операційних систем Linux, Windows і macOS.

– PyCharm – інтегроване середовище розробки для мови програмування Python. PyCharm є власницьким програмним забезпеченням. Наявна безкоштовна версія Community з усіченим набором можливостей. Для користувачів Linux, Windows і macOS.

– Thonny – IDE для вивчення програмування мовою Python. Для користувачів Linux, Windows і macOS.

– Mu – редактор коду Python для програмістів-початківців. Для користувачів Linux, Windows і macOS.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи кібербезпеки для забезпечення конфіденційності інформації на основі технології NTRUEncrypt.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи кібербезпеки контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на

екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи кібербезпеки в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

NTRUEncrypt (абревіатура Nth-degree TRUncated polynomial ring або Number Theorists aRe Us) – це криптографічна система з відкритим ключем, що раніше називалася NTRU.

Коли ви запускаєте його вперше, ви повинні увійти, використовуючи свою електронну адресу, а потім ввести код підтвердження, надісланий на вашу електронну адресу. Я розберуся в причині цього пізніше. Ви також повинні створити єдиний головний пароль. Якщо ви використовуєте менеджер паролів, дозвольте генератору паролів створити щось складне, наприклад e/M^[Ai#J_$!0CF3#sA4. Якщо ви не підете шляхом менеджера паролів, ви повинні придумати щось водночас сильне та незабутнє програмне забезпечення системи кібербезпеки для конфіденційності інформації, розроблене у даній роботі, чітко дає зрозуміти, що чорних дверей немає, крапка. Якщо ви втратите пароль, ви втратите свої файли.

Це пароль, який ви використовуєте для входу в Інтернет, а також пароль, який розблокує продукт на вашому ПК.

Після встановлення програмне забезпечення системи кібербезпеки для конфіденційності інформації, розроблене у даній роботі, непомітний. Якщо ви перетягнете файл у його вікно, цей файл буде зашифровано. Вам більше нічого не потрібно робити, оскільки програмне забезпечення системи кібербезпеки для конфіденційності інформації, розроблене у даній роботі, зберігає ваш головний пароль у пам’яті, доки ви не вийдете. Ви також можете шифрувати, розшифровувати та безпечно видаляти файли з контекстного меню програмне забезпечення системи кібербезпеки для конфіденційності інформації, розроблене у даній роботі,, яке відкривається правою кнопкою миші.

Майже в кожному інструменті шифрування файлів запуск зашифрованого файлу запускає процес дешифрування. Якщо ви хочете відредагувати файл, ви повинні розшифрувати, відредагувати та повторно зашифрувати. З програмним забезпеченням системи кібербезпеки для конфіденційності інформації, розробленим у даній роботі, запуск зашифрованого файлу відкриває його у відповідній програмі. Коли ви зберігаєте, збережений файл автоматично шифрується.

Програмне забезпечення системи кібербезпеки для конфіденційності інформації, розроблене у даній роботі, зберігає ваш пароль у пам’яті, щоб ви могли шифрувати та розшифровувати файли за бажанням, не перериваючи запити пароля. Це також означає, що якщо ви залишите свій робочий стіл з активним програмне забезпечення системи кібербезпеки для конфіденційності інформації, розроблене у даній роботі, і увійшли в обліковий запис, хтось інший може сісти за ваш комп’ютер, розшифрувати ваші файли та вкрасти їх.

Річ у тім, що в цій ситуації, коли зловмисник має фізичний доступ до вашого розблокованого комп’ютера, розшифровка чи викрадення файлів – це найменше для вас хвилювання. Те саме може статися з EncryptionSafe, Xecrets або будь-яким продуктом шифрування, який зберігає ваш пароль у пам’яті.

Що робити? По-перше, захистіть свій логін надійним паролем, біометричною автентифікацією або тим і іншим. Налаштуйте операційну систему на вихід із системи після періоду бездіяльності. Коли ви встаєте з-за столу, активно вийдіть із системи (у Windows просто натисніть комбінацію клавіш Windows+L). Ви також можете налаштувати програмне забезпечення системи кібербезпеки для конфіденційності інформації, розроблене у даній роботі, на вихід із системи через 5–60 хвилин. Ці запобіжні заходи пов’язані більше з вашою загальною безпекою, ніж із конкретною програмою, як-от програмне забезпечення системи кібербезпеки для конфіденційності інформації, розроблене у даній роботі,.

Програмне забезпечення системи кібербезпеки для конфіденційності інформації, розроблене у даній роботі, не використовує головний пароль як ключ для шифрування ваших файлів. Натомість він генерує випадковий ключ для шифрування, а також 4096-бітну пару відкритий/приватний ключ і шифрує їх на основі головного. Ця технологія «загорнутих ключів» означає, що якщо ви зміните свій головний пароль, ви все одно зможете відкривати наявні файли. EncryptionSafe також використовує єдиний головний пароль, але зміна цього пароля змушує його розшифровувати кожен файл і повторно шифрувати за допомогою нового пароля – незручно.

Багато інших утиліт для шифрування файлів приймають новий пароль для кожного файлу, і ви повинні запам’ятати їх усі. Стане в нагоді менеджер паролів.

Керування паролями, але не дуже

Програмне забезпечення системи кібербезпеки для конфіденційності інформації, розроблене у даній роботі, містить функцію керування паролями, але це не той менеджер паролів, якого ви могли б очікувати. Запустивши його, ви перейдете на сторінку керування паролями веб-сайту програмного забезпечення системи кібербезпеки для конфіденційності інформації, розробленого у даній роботі,. Натисніть «Новий», щоб додати опис і пароль. Натисніть Пошук, щоб знайти збережені паролі. Ось у чому справа. Ви також можете зберігати кредитні картки та захищені нотатки, але він не має автоматичного захоплення та відтворення паролів, а тим більше розширених функцій спеціального менеджера паролів, як NordPass або Dashlane.

Однак тут є одна цікава особливість. Натискання кнопки «Запропонувати пароль» генерує 18-символьний пароль, який містить усі типи символів, але його можна дещо вимовити. Я дізнався більше про цю функцію на веб-сторінці Password Generator.

Відповідно до цієї сторінки, він генерує «надійні та складні паролі, які не є безглуздими, і їх можна запам’ятати та ввести». Як не дивно, але з кожним натисканням кнопки він генерує три паролі, позначені як надійний, помірний і

легкий. Щойно запустивши його, я отримав "inQuiths$Scitsly8", "S03Ushersent" і "Boulacts". Я не пропоную вам використовувати слабкий пароль, навіть якщо він легкий. З іншого боку, слабкий пароль кращий, ніж той, який ви отримуєте за замовчуванням від чудового генератора паролів Advanced Encryption Package. За замовчуванням він генерує п’ятисимвольні паролі, які складаються з великої літери, наприклад NOWAY. Під час тестування він також вийшов з ладу, коли його попросили зробити набагато більше, ніж це.

Безпечний обмін

Програмне забезпечення системи кібербезпеки для конфіденційності інформації, розроблене у даній роботі, робить усе, що стосується шифрування, простим, тому ви не повинні бути здивовані, дізнавшись, що він навіть спрощує обмін безпечними даними PKI. Ось як це працює. Натисніть кнопку Поділитися ключами на панелі інструментів. Виберіть або введіть потрібні контактні електронні адреси або створіть групу одержувачів. програмне забезпечення системи кібербезпеки для конфіденційності інформації, розроблене у даній роботі, надсилає посилання та пояснення одержувачам.

Зауважте, що для обміну файлами потрібне підключення до Інтернету. За допомогою програмного забезпечення ви можете зашифрувати файл для спільного використання повністю локальним способом, а потім поділитися файлом будь-яким способом, у тому числі на USB-накопичувачі.

Одержувач, який ще не має програмне забезпечення системи кібербезпеки для конфіденційності інформації, розроблене у даній роботі,, має встановити безкоштовну версію, як пояснюється в електронному листі для спільного доступу. Повідомлення містить посилання для підтвердження, що полегшує одержувачу розпочати роботу. І все.

Більш розширені функції PKI можна знайти в меню «Керування ключами». Тут ви можете експортувати свій відкритий ключ для спільного використання або імпортувати відкритий ключ, який вам надали. Ви також можете експортувати та імпортувати весь обліковий запис. Проте, чесно кажучи,

більшості користувачів ніколи не потрібно думати про таке керування своїми ключами.

Пакет Advanced Encryption Package також підтримує PKI, але він націлений на зовсім іншу аудиторію, зокрема на тих, хто має технічний досвід у шифруванні. Навпаки, будь-хто може використовувати програмне забезпечення системи кібербезпеки для конфіденційності інформації, розроблене у даній роботі,.

Інші зручні функції

Шифрувати файл, залишаючи зашифровану копію з назвою на кшталт «Мої таємні плани щодо контролю над розумом президента», не дуже добре. Стежище може не дізнатися подробиць через шифрування, але саме існування ваших планів, безсумнівно, має залишатися в таємниці. програмне забезпечення системи кібербезпеки для конфіденційності інформації, розроблене у даній роботі, може перейменувати будь-який зашифрований файл, щоб використовувати для його імені випадковий рядок символів. І він запам’ятовує, що він зробив, тож ви можете за потреби відновити вихідне ім’я файлу.

Шифрування гарантує, що ніхто, крім вас, не зможе переглядати чи змінювати вашу інформацію. Безпечне видалення йде далі, гарантуючи, що ніхто, навіть ви, не зможете відновити конфіденційні дані. Компонент безпечного видалення програмне забезпечення системи кібербезпеки для конфіденційності інформації, розроблене у даній роботі,, до якого можна отримати доступ через меню програми або меню файлу/папки, клацаючи правою кнопкою миші, один раз перезаписує дані файлу довільними байтами перед видаленням. Цього достатньо, щоб зірвати програмні засоби відновлення файлів. У будь-якому випадку стандартні інструменти криміналістичного відновлення призначені для застарілих обертових жорстких дисків і непридатні для сучасних SSD.

Програмне забезпечення системи кібербезпеки для конфіденційності інформації, розроблене у даній роботі, є портативним і крос платформним.

Один із способів переконатися, що люди не знайдуть ваші секрети, - це приховати той факт, що у вас навіть є секрети. Якщо ви вирішите завантажити автономну портативну версію програмне забезпечення системи кібербезпеки для конфіденційності інформації, розроблене у даній роботі,, а не встановлювати програму, ви не знайдете програмне забезпечення системи кібербезпеки для конфіденційності інформації, розроблене у даній роботі, у жодному меню, включно з меню, яке відкривається правою кнопкою миші. Зашифровані файли відображатимуться як файли AXX, не пов'язані з програмне забезпечення системи кібербезпеки для конфіденційності інформації, розроблене у даній роботі,. І якщо ви зберігаєте програму на знімному носії, клавши її в кишеню, коли вона не потрібна, ваші файли отримають додатковий рівень безпеки.

Іншим варіантом було б зберігати зашифровані файли на тому ж знімному диску, що й програмне забезпечення системи кібербезпеки для конфіденційності інформації, розроблене у даній роботі, portable. Тепер ці секрети залишаються у вашій кишені. Коли вам знадобиться доступ, підключіть диск, запустіть програмне забезпечення системи кібербезпеки для конфіденційності інформації, розроблене у даній роботі, виконайте необхідні дії та вийдіть.

Встановлений на macOS програмне забезпечення системи кібербезпеки для конфіденційності інформації, розроблене у даній роботі, виглядає майже ідентично версії Windows. Найбільша різниця полягає у розділі «Параметри», який у macOS виглядає як діалогове вікно з вкладками, а не як багаторівневе меню. Кожного разу, коли я відкриваю нову вкладку параметрів конфігурації, програмне забезпечення системи кібербезпеки для конфіденційності інформації, розроблене у даній роботі, попереджає мене, що будь-яка зміна може спричинити проблеми, і вимагає повторного введення мого довгого пароля. Перевіривши всі сторінки, я знайшов усі очікувані параметри, окрім «Вийти через неактивність» і «Керування ключами». Мій контакт у компанії підтвердив, що ви повинні використовувати версію Windows для цих розширених функцій керування ключами.

Мобільна версія програмного забезпечення системи кібербезпеки для конфіденційності інформації, розробленого у даній роботі, для пристроїв Android і Apple зробила кілька значних кроків. Тепер за функціями він нарівні з версією macOS. Ви можете шифрувати, розшифровувати та ділитися файлами, які зберігаються на вашому телефоні або в Dropbox, Google Drive, iCloud або OneDrive.

З мобільним додатком проста функція керування паролями інтегрована, а не онлайн. Функції ті самі – ви просто зберігаєте паролі, платіжні картки та захищені нотатки. Він включає генерацію пароля, але без можливості налаштувати довжину пароля або використані набори символів. Фактичних функцій керування паролями, таких як захоплення та відтворення паролів, немає.

Мобільний додаток попередив, що він зберігає розшифровану копію кожного файлу в локальному сховищі, яка зникає, коли ви виходите з програми. Також згадується, що ви можете очистити цю копію вручну за допомогою кнопки «Очистити».

Незважаючи на те, що програмне забезпечення системи кібербезпеки для конфіденційності інформації, розроблене у даній роботі, керує шифруванням відкритих/приватних ключів під капотом, він надзвичайно простий у використанні. Дійсно, той факт, що він зберігає вашу парольну фразу в пам’яті, означає, що ви абсолютно повинні подбати про фізичну безпеку свого комп’ютера, але це лише хороша практика. І, звичайно, ви захищені, якщо виходите з програмного забезпечення системи кібербезпеки для конфіденційності інформації, розробленого у даній роботі, щоразу, коли не використовуєте його.

3.2 Розробка структурної схеми

Структурна схема наведена на рисунку 3.1. З неї ми бачимо, що розроблена система складається з наступних структурних блоків.

1. Дані, які записуються на переносний носій інформації.

2. Блок генерації та розподілу ключів.
3. Блок шифрування за допомогою алгоритму NTRUEncrypt.
4. Блок розшифрування за допомогою алгоритму NTRUEncrypt.
5. Переносний носій інформації на який записані зашифровані дані.

NTRU (Nth-degree TRUncated polynomial ring або просто Number Theorists aRe Us) на відміну від своїх іменитих попередників таких як RSA або El Ga*m*al, NTRU працює не над кільцем відрахувань за модулем цілого числа *N*, а над кільцем багаточленів ступеня *n*-1 наведених за модулем x^n-1. Додавання елементів у такій групі відбувається як звичайне додавання, а при множенні елемент x^n приводиться до 1, x^{n+1} к x и так далі. При множенні двох елементів кільця $a(x)*b(x)$ виходить елемент $c(x)=c_0+c_1x+\ldots+c_{n-1}x^{n-1}$, у якому коефіцієнт c_k обчислюється в такий спосіб:

$$c_k = a_0b_k + a_1b_{k-1} + \ldots + a_kb_0 + a_{k+1}b_{N-1} + a_{k+2}b_{N-2} + \ldots + a_{N-1}b_{k+1}$$

Таке кільце одержало назву кільце усічених багаточленів. Позначимо його для зручності *R*. У криптосистемі NTRU всі коефіцієнти багаточленів приводяться за модулем цілих чисел *p* і *q*. Наприклад, багаточлен $13+12x+14x^2+7x^3 \bmod 3 = 1-x^2+x^3$.

NTRU використовує три постійних параметри: *N*, *p*, *q*. Числом *N* характеризується розмір обираних як ключі багаточленів. Числа *p* і *q* не обов'язково повинні бути простими, але НЗД (*p*,*q*) повинен рівнятися 1. Слід зазначити, що параметр *p* служить для визначення інтервалу, якому зобов'язані належати всі коефіцієнти багаточленів використовуваних у криптосистемі.

Таким чином, якщо *N*=11 а *p*=3, то ми зможемо використовувати в нашій криптосистемі тільки багаточлени з коефіцієнтами {-1,0,1}. Наприклад:

$$-1+x+x^{3-x4}-x^5+x^{10}$$

Після вибору цих трьох основних параметрів потрібно буде вибрати ще три додаткових, які прийнято позначати d_f, d_g, d. Ці три параметри служать для визначення набору багаточленів

$$L_f=L(d_f, d_f-1),\ L_g=L(d_g, d_g),\ L_r=L(d, d).$$

Невелике пояснення: запис виду $L_f=L(d_f, d_f\text{-}1)$ означає, що L_f є набором усіляких багаточленів кільця R, які маю як коефіцієнти рівно d_f одиниць(1) і d_f-1 мінус одиниць(-1) всі інші коефіцієнти дорівнюють нулю.

Наприклад, багаточлен $-1+x+x^3-x^4+x^8$ належить набору $L(3,2)$ тому що в нього 3 1-ки й 2 -1-ки.

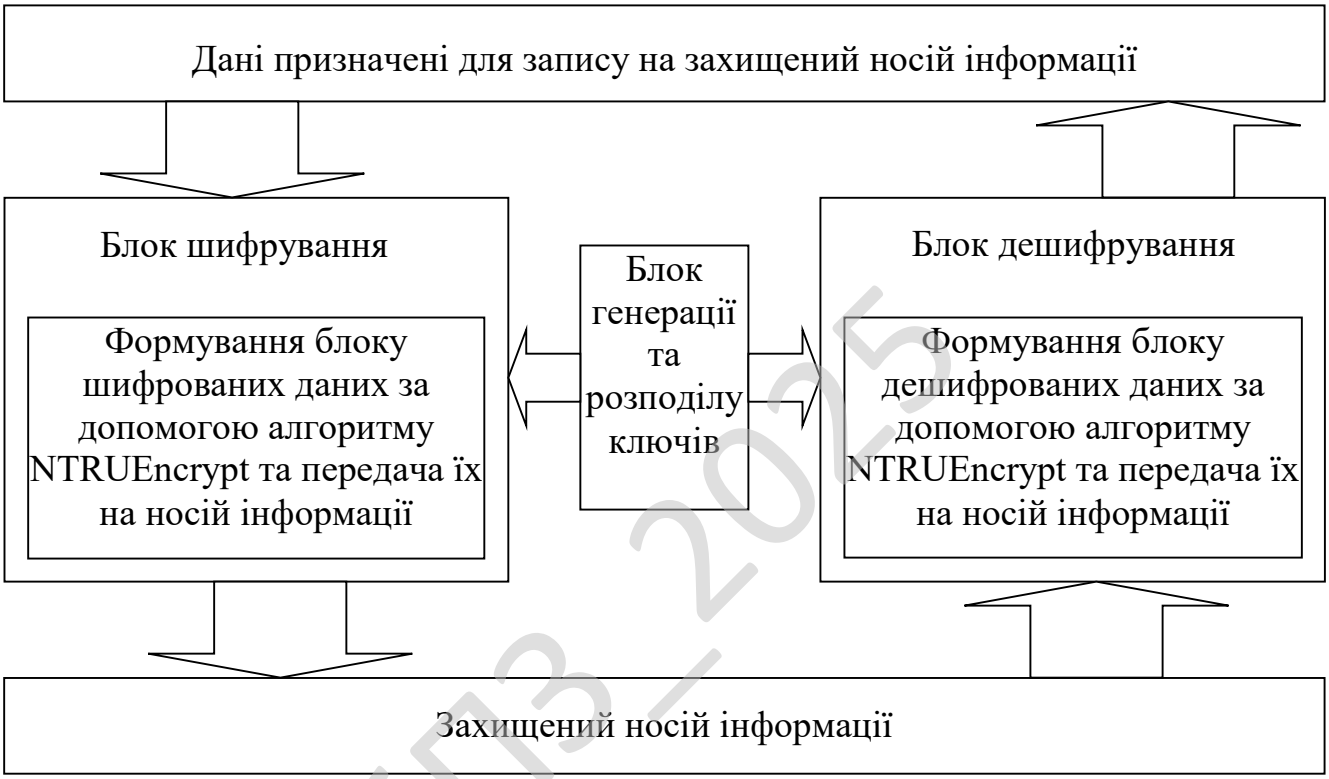

Рисунок 3.1 – Структурна схема системи

Блок генерації пари секретний/відкритий ключ реалізований наступним чином.

1. З набору L_f вибирається довільний багаточлен $f(x)$.
2. З набору Lg вибирається багаточлен $g(x)$.
3. Обчислюються багаточлени $f_q(x)$ і $f_p(x)$ такі що $f_p(x)*f(x)=1 \bmod p$ і $f_q(x)*f(x)=1 \bmod q$
4. Відкритий ключ визначається як $h(x)=f_q(x)*g(x) \bmod q$
5. Секретний ключ це пари $(f(x),f_p(x))$.

Основним блок системи є блок шифрування NTRUEncrypt. Розглянемо його більш детальніше. Алгоритм шифрування NTRUEncrypt працює наступним чином:

Блок шифрування в криптосистемі NTRU:

1. Боб вибирає повідомлення *m* і перетворить його в багаточлен $M(x)$ $*L_m$ нагадую, що коефіцієнти багаточленів приналежних L_m лежать в інтервалі:

$$\left[-\frac{p-1}{2};\frac{p-1}{2}\right]$$

2. Боб вибирає т.зв. «осліплюючий» багаточлен $r(x)$ $*L_r$ і використовуючи відкритий ключ Аліси обчислює $C(x)=p*r(x)*h(x)+M(x) \bmod q$. Багаточлен $c(x)$ і буде шифротекстом.

Алгоритм розшифрування NTRUEncrypt працює наступним чином:

Блок розшифровки:

1. Одержавши від Боба $C(x)$, Аліса використовуючи секретний ключ обчислює:

$$a(x) = f(x) * C(x) \bmod q = f(x) * p * r(x) * h(x) + f(x) * M(x) \bmod q = f(x) *$$
$$p * f_q q(x) * g(x) + f(x) * M(x) \bmod q = r(x) * p * g(x) + f(x) * M(x) \bmod q.$$

При цьому Аліса ретельно стежить, щоб коефіцієнти отриманого багаточлена $a(x)$ лежали в інтервалі $(-q/2,q/2]$.Навіщо це робиться поясню пізніше.

2. Аліса обчислює:

$$b(x) = a(x) \bmod p = f(x) * C(x) \bmod p = r(x) * p * g(x) + f(x) * M(x) \bmod p.$$

перший доданок вираження $b(x)$ має множник p і отже $b(x)=f(x)*M(x) \bmod p$. Однак це все відбувається тільки в тому випадку, якщо при обчисленні $a(x)$ його коефіцієнти не були більше q, тому на першому етапі розшифровки й виробляється перевірка того, що всі коефіцієнти лежать у зазначеному інтервалі.

3. Обчисливши $b(x) * f_p(x) \bmod p = f(x) * b(x) * f_p(x) = M$ Аліса відновлює вихідне повідомлення *M*.

Найцікавішим моментом у всій схемі шифрування-розшифрування я особисто вважаю момент перевірки приналежності коефіцієнтів отриманого

багаточлена $a(x)$ інтервалу $(-q/2;q/2]$. Як я вже пояснив вище всі коефіцієнти багаточлена не повинні бути більше q щоб не порушити подільність на p лівої частини суми. Однак чому ж тоді ми перевіряємо інтервал $(-q/2;q/2]$ а не $(-q;q]$. Справа от у чому. Припустимо q=32. P=3. У результаті приведення за модулем 32 вийшло число 18. За модулем 3 це буде нуль. Але адже 18=-14 mod 32. І якщо ми обчисли -14 mod 3 те одержимо невірний результат. Відповідно потрібно завжди заздалегідь знати в якому інтервалі будуть отримані коефіцієнти. Розроблювачі NTRU затверджують, що для параметрів, що рекомендуються, з імовірністю майже рівної 1 коефіцієнти завжди будуть розташовуватися в інтервалі (-q/2,q/2], тому при розшифровці Аліса приводить умовно, що вийшло число, 18 до -14.

Переваги NTRU

Отже, які переваги й недоліки можна побачити від переходу на NTRU уже зараз.

По-перше, більшу швидкість роботи. Виконання операцій шифрування/розшифрування вимагає $O(n^2)$ операцій, на відміну від $O(n^3)$ у того ж RSA.

По-друге, невелике, але збільшення стійкості при фактично такій же довжині ключа.

Недоліки NTRU

Він поки один, але дуже серйозний. Необхідність використання тільки рекомендованих параметрів. Саме така ж вимога викликала загальне невдоволення під час переходу на еліптичні криві й сприяла всілякими підозрам про наявність бекдорів.

3.3 Розробка функціональної схеми

Криптосистема NTRUEncrypt, заснована на ґратчастій криптосистемі, створена як альтернатива RSA і криптосистемам на еліптичних кривих (ECC). Стійкість алгоритму забезпечується труднощами пошуку найкоротшого вектора

ґрати, що більше стійка до атак, здійснюваним на квантових комп'ютерах. На відміну від своїх конкурентів RSA, ECC, Elgamal, алгоритм використовує операції над кільцем:

$$\mathbb{Z}[X]/(X^N - 1),$$

усічених багаточленів ступеня, що не перевершує $N - 1$:

$$\mathbf{a}(X) = \mathbf{a} = a_0 + a_1 X + a_2 X^2 + \cdots + a_{N-2} X^{N-2} + a_{N-1} X^{N-1}.$$

Такий багаточлен можна також представити вектором:

$$\vec{a}(X) = \vec{a} = \sum_{i=0}^{N-1} a_i X^i = [a_0, a_1, a_2, \cdots, a_{N-2}, a_{N-1}].$$

Як і будь-який молодий алгоритм, NTRUEncrypt погано вивчений, хоча й був офіційно затверджений для використання в сфері фінансів комітетом Accredited Standards Committee X9.[1]

Існує реалізації NTRUEncrypt з відкритим вихідним кодом.[2]

NTRUEncrypt, що споконвічно називався NTRU, був винайдений в 1996 році й представлений на конференціях CRYPTO, Конференція RSA, Eurocrypt. Причиною, що послужила початком розробки алгоритму в 1994 році, стала стаття [3], у якій говорилося про легкість злому існуючих алгоритмів на квантових комп'ютерах, які, як показало час, не за горами[4]. У цьому ж році, математики Jeffrey Hoffstein, Jill Pipher і Joseph H. Silverman, що розробили систему разом із засновником компанії NTRU Cryptosystems, Inc. (пізніше перейменованої в SecurityInnovation), Даніелєм Лієманом (Daniel Lieman) запатентували свій винахід.[5]

Кільця усічених багаточленів

NTRU оперує над багаточленами ступеня не переважаючої $N - 1$:

$$\mathbf{a} = a_0 + a_1 X + a_2 X^2 + \cdots + a_{N-2} X^{N-2} + a_{N-1} X^{N-1},$$

де коефіцієнти $a_0, \cdots, a_{N-1}$ – цілі числа. Щодо операцій додавання й множення за модулем багаточлена $X^N - 1$.Такі багаточлени утворять кільце *R*, називане кільцем усічених багаточленів, що ізоморфно кільцю відносин:

$$\mathbb{Z}[X]/(X^N - 1).$$

NTRU використовує кільце усічених багаточленів R разом з діленням за модулем на взаємно прості числа p і q для зменшення коефіцієнтів багаточленів.

У роботі алгоритму також використовуються зворотні багаточлени в кільці усічених багаточленів. Слід зазначити, що не всякий багаточлен має зворотний, але якщо зворотний поліном існує, то його легко обчислити.[6][7]

Таблиця 3.1 – Параметри NTRU

Позначення параметрів	N	q	p
Значення параметрів	11	32	3

Генерація відкритого ключа

Для передачі повідомлення від Аліси до Боба необхідні відкритий і закритий ключі. Відкритий знають як Боб, так і Аліса, закритий ключ знає тільки Боб, що він використовує для генерації відкритого ключа. Для цього Боб вибирає два «маленьких» поліноми f g з R. «Малість» поліномів мається на увазі в тому розумінні, що він маленький щодо довільного полінома за модулем q: у довільному поліномі коефіцієнти повинні бути приблизно рівномірно розподілені за модулем q, а в малому поліномі вони багато менше q. Малість поліномів визначається за допомогою чисел df і dg:

Поліном f має df коефіцієнтів рівних «1» і df-1коефіцієнтів рівних «-1», а інші – «0». У цьому випадку говорять, що:

$$\mathbf{f} \in \mathcal{L}_f$$

Поліном g має dg коефіцієнтів рівних «1» і стільки ж рівних «-1», інші – «0» У цьому випадку говорять, що:

$$\mathbf{g} \in \mathcal{L}_g$$

Причина, по якій поліноми вибираються саме таким чином, полягає в тому, що f , можливо, буде мати зворотний, а g – однозначно немає (g (1) = 0, а нульовий елемент не має зворотного).

Боб повинен зберігати ці полінома в секреті. Далі Боб обчислює зворотні поліноми $\mathbf{f}_p$ й $\mathbf{f}_q$, тобто такі, що:

$$\mathbf{f} \cdot \mathbf{f}_p \equiv 1 \pmod{p} \text{ й } \mathbf{f} \cdot \mathbf{f}_q \equiv 1 \pmod{q}$$

Якщо *f* не має зворотного полінома, то Боб вибирає інший поліном *f*.

Секретний ключ – це пари $(\mathbf{f}, \mathbf{f}_p)$, а відкритий ключ *h* обчислюється за формулою:

$$\mathbf{h} = (p\mathbf{f}_q \cdot \mathbf{g}) \bmod q.$$

Приклад:

Для приклада візьмемо *df*=4, а *dg*=3. Тоді як поліноми можна вибрати

$$\mathbf{f} = -1 + X + X^2 - X^4 + X^6 + X^9 - X^{10}$$

$$\mathbf{g} = -1 + X^2 + X^3 + X^5 - X^8 - X^{10}$$

Далі для полінома *f* шукаються зворотні поліноми за модулем *p*=3 иq=32:

$$\mathbf{f}_p = 1 + 2X + 2X^3 + 2X^4 + X^5 + 2X^7 + X^8 + 2X^9$$

$$\mathbf{f}_q = 5+9X+6X^2+16X^3+4X^4+15X^5+16X^6+22X^7+20X^8+18X^9+30X^{10}$$

Заключним етапом є обчислення самого відкритого ключа *h*:

$$\mathbf{h} = (p\mathbf{f}_q \cdot \mathbf{g}) \bmod 32 = 8+25X+22X^2+20X^3+12X^4+24X^5+15X^6+19X^7+12X^8+19X^9+16X^{10}.$$

Шифрування

Тепер, коли в Аліси є відкритий ключ, вона може відправити зашифроване повідомлення Бобові. Для цього потрібно повідомлення представити у вигляді полінома m з коефіцієнтами за модулем *p*, обраними з діапазону (-*p*/2, *p*/2]. Тобто *m* є «малим» поліномом за модулем *q*. Далі Алісі необхідно вибрати інший «малий» поліном r, що називається «сліпучої», обумовлений за допомогою числа *dr*:

Поліном r має *dr* коефіцієнтів рівних «1» і стільки ж рівних «-1», інші – «0». У цьому випадку говорять, що:

$$\mathbf{r} \in \mathcal{L}_r.$$

Використовуючи ці поліноми, зашифроване повідомлення виходить за формулою:

$$\mathbf{e} = (\mathbf{r} \cdot \mathbf{h} + \mathbf{m}) \bmod q.$$

При цьому кожній, хто знає (або може обчислити) осліплюючий поліном r, зможе прочитати повідомлення *m*.

Приклад:

Припустимо, що Аліса хоче послати повідомлення, представлене у вигляді полінома:

$$\mathbf{m} = -1 + X^3 - X^4 - X^8 + X^9 + X^{10},$$

і вибрала «осліплюючий» поліном, для якого *dr*=3:

$$\mathbf{r} = -1 + X^2 + X^3 + X^4 - X^5 - X^7.$$

Тоді шифротекст *e*, готовий для передачі Бобові буде:

$$\mathbf{e} = (\mathbf{r}\cdot\mathbf{h}+\mathbf{m}) \bmod 32 = 14+11X+26X^2+24X^3+14X^4+16X^5+30X^6+7X^7+25X^8+6X^9+19X^{10}$$

Розшифрування

Тепер, одержавши зашифроване повідомлення *e*, Боб може його розшифрувати, використовуючи свій секретний ключ. Спочатку він одержує новий проміжний поліном:

$$\mathbf{a} = (\mathbf{f}\cdot\mathbf{e}) \bmod q.$$

Якщо розписати шифротекст, то одержимо ланцюжок:

$$\mathbf{a} = (\mathbf{f}\cdot\mathbf{e}) \bmod q = (\mathbf{f}\cdot(\mathbf{r}\cdot\mathbf{h}+\mathbf{m})) \bmod q = (\mathbf{f}\cdot(\mathbf{r}\cdot p\mathbf{f}_q\cdot\mathbf{g}+\mathbf{m})) \bmod q$$

і остаточно:

$$\mathbf{a} = (p\mathbf{r}\cdot\mathbf{g}+\mathbf{f}\cdot\mathbf{m}) \bmod q.$$

Після того, як Боб обчислив поліном a за модулем *q*, він повинен вибрати його коефіцієнти з діапазону (-*q*/2, *q*/2] і далі обчислити поліном b, одержуваний з полінома a приведенням за модулем *p*:

$$\mathbf{b} = \mathbf{a} \bmod p = (\mathbf{f}\cdot\mathbf{m}) \bmod p,$$

так як:

$$(p\mathbf{r}\cdot\mathbf{g}) \bmod p = 0.$$

Тепер, використовуючи другу половину секретного ключа й отриманий поліном *b*, Боб може розшифрувати повідомлення:

$$\mathbf{c} = (\mathbf{f}_p\cdot\mathbf{b}) \bmod p.$$

Неважко бачити, що:

$$\mathbf{c} \equiv \mathbf{f}_p\cdot\mathbf{f}\cdot\mathbf{m} \equiv \mathbf{m} \pmod p.$$

У такий спосіб отриманий поліном *c* дійсно є вихідним повідомленням *m*.

Приклад:

Боб одержав від Аліси шифроване повідомлення *e*:

$$\mathbf{e} = 14+11X+26X^2+24X^3+14X^4+16X^5+30X^6+7X^7+25X^8+6X^9+19X^{10}$$

Використовуючи секретний ключ f Боб одержує поліном *a*:

$$\mathbf{a} = \mathbf{f}\cdot\mathbf{e} \pmod{32} = 3-7X-10X^2-11X^3+10X^4+7X^5+6X^6+7X^7+5X^8-3X^9-7X^{10} \pmod{32},$$

з коефіцієнтами, що належать проміжку (-*q*/2, *q*/2]. Далі перетворить поліном *a* у поліном *b*, зменшуючи коефіцієнти за модулем *p*.

$$\mathbf{b} = \mathbf{a} \pmod 3 = -X - X^2 + X^3 + X^4 + X^5 + X^7 - X^8 - X^{10} \pmod 3$$

Заключний крок – перемножування полінома *b* із другою половиною закритого ключа $\mathbf{f}_p$:

$$\mathbf{c} = \mathbf{f}_p \cdot \mathbf{b} = \mathbf{f}_p \cdot \mathbf{f} \cdot \mathbf{m} \pmod 3 = \mathbf{m} \pmod 3$$

$$\mathbf{c} = -1 + X^3 - X^4 - X^8 + X^9 + X^{10}$$

Який є вихідним повідомленням, що передавала Аліса.

Стійкість до атак

Повний перебір

Перша з можливих атак – атака перебором. Отут можливо кілька варіантів перебору: або перебирати всі $\mathbf{f} \in \mathcal{L}_f$, і перевіряти на малість коефіцієнти отриманих результатів $\mathbf{f} \cdot \mathbf{h} \pmod q = \mathbf{g} \pmod q$, які, по задуму, повинні був бути малими, або перебирати всі $\mathbf{g} \in \mathcal{L}_g$, також перевіряючи на малість коефіцієнти результату:

$$\mathbf{f} \pmod q = \mathbf{f}\cdot\mathbf{h}\cdot\mathbf{h}^{-1} \pmod q = \mathbf{f}\cdot\mathbf{f}_q\cdot\mathbf{g}\cdot\mathbf{h}^{-1} \pmod q = \mathbf{g}\cdot\mathbf{h}^{-1} \pmod q.$$

На практиці простір $\mathcal{L}_g$ менше простору $\mathcal{L}_f$, отже стійкість визначається простором $\mathcal{L}_g$. А стійкість окремого повідомлення визначається простором $\mathcal{L}_r$.

Зустріч посередині

Існує більше оптимальний варіант перебору зустріч посередині, запропонований Андрю Одлижко (Andrew Odlyzko). Цей метод зменшує кількість варіантів до квадратного кореня:

Стійкості закритого ключа:

$$\sqrt{\mathcal{L}_g} = \frac{1}{d_g!}\sqrt{\frac{N!}{(N-2d_g)!}},$$

І стійкості окремого повідомлення:

$$\sqrt{\mathcal{L}_r} = \frac{1}{d!}\sqrt{\frac{N!}{(N-2d)!}}.$$

Атака «зустріч посередині» дозволяє розміняти час, необхідне для обчислень на пам'ять, необхідну для зберігання тимчасових результатів. Таким чином, якщо ми хочемо забезпечити стійкість системи 2^n, потрібно вибрати ключ розміру 2^{2n}.

Атака на основі множинної передачі повідомлення

Досить серйозна атака на окреме повідомлення, яку можна уникнути, дотримуючись простого правила не пересилати багаторазово те саме повідомлення. Суть атаки полягає в знаходженні частини коефіцієнтів сліпучого багаточлена r. А інші коефіцієнти можна просто перебрати, тим самим прочитавши повідомлення. Так як зашифроване те саме повідомлення з різними сліпучими багаточленами це $e_i = r_i \cdot h + m \bmod q$, де i=1, … n... Можна обчислити вираження $(e_i - e_1) \cdot h_q \bmod q$, що у точності дорівнює $\mathbf{r}_i - \mathbf{r}_1 \bmod q$. Для досить великої кількості переданих повідомлень (скажемо, для n = 4, 5, 6), можна відновити, виходячи з малості коефіцієнтів, більшу частину сліпучого багаточлена $\mathbf{r}_1$.

Атака на основі ґрат

Розглянемо ґрати, породжену рядками матриці розміру 2N×2N з детермінантом $\alpha^N q^N$, що складається із чотирьох блоків розміру N×N:

$$\left(\begin{array}{cccc|cccc} \alpha & 0 & \cdots & 0 & h_0 & h_1 & \cdots & h_{N-1} \\ 0 & \alpha & \cdots & 0 & h_{N-1} & h_0 & \cdots & h_{N-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \alpha & h_1 & h_2 & \cdots & h_0 \\ \hline 0 & 0 & \cdots & 0 & q & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & q & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & q \end{array}\right).$$

Так як відкритий ключ:

$$\mathbf{h} = (p\mathbf{g} \cdot \mathbf{f}_q) \bmod q,$$

то:

$$\mathbf{g} \equiv \mathbf{h} \cdot \mathbf{f} \pmod{q},$$

отже в цих ґратах утримується вектор:

$$\tau = (\alpha f, g)$$

розміру 2N, у якому йдуть спочатку коефіцієнти вектора *f*, помножені на коефіцієнт α, потім коефіцієнти вектора *g*. Задача пошуку такого вектора, при більших N і правильно підібраних параметрах, уважається важко розв'язною.

Атака на основі підібраного шифротексту

Атака на основі підібраного шифротексту є найнебезпечнішою атакою. Її запропонували Éliane Jaulmes і Antoine Joux[8] в 2000 році на конференції CRYPTO. Суть цієї атаки полягає в підборі такого багаточлена *a*(x), щоб $\mathbf{a}(x) \pmod{q} \neq \mathbf{a}(x)$. При цьому Ева не взаємодіє ні з Бобом, ні з Алісою.

Якщо взяти шифротекст $\mathbf{e}^* = y\mathbf{h} + py$, де $y \in \mathbb{Z}$, то одержимо багаточлен $\mathbf{a}^* = \mathbf{f} \cdot \mathbf{e}^* \pmod{q} = yp\mathbf{g} + yp\mathbf{f} \pmod{q}$. Так як коефіцієнти багаточленів *f* і *g* приймають значення «0», «1» і «-1», те коефіцієнти багаточлена *a* будуть належати множині {-2ру , -ру , 0, ру, 2ру}. Якщо ру вибрати таким, що $\frac{q}{4} < py < \frac{q}{2}$, то при відомості за модулем полінома *a*(x)доведуться тільки коефіцієнти рівні -2ру або 2ру. Нехай тепер і-ый коефіцієнт дорівнює 2ру, тоді багаточлен *a*(x) після приведення за модулем запишеться як:

$$\mathbf{a}^* = \mathbf{f} \cdot \mathbf{e}^* \pmod q = ypg + ypf - q \cdot x^i,$$

а багаточлен *b*(x):

$$\mathbf{b}^* = \mathbf{a}^* \pmod p = ypg + ypf - q \cdot x^i \pmod p = -q \cdot x^i,$$

остаточно обчислимо:

$$\mathbf{c}^* = \mathbf{z}(x) = \mathbf{b}^* \cdot \mathbf{f}_p \pmod p = -q \cdot x^I \cdot \mathbf{f}_p \pmod p.$$

Тепер, якщо розглянути всі можливі i, те замість окремих x^i, можна скласти поліном K і розшифроване повідомлення прийме вид:

$$\mathbf{c} = -q\mathbf{K} \cdot \mathbf{f}_p \pmod p,$$

або, секретний ключ:

$$\mathbf{f} = -q\mathbf{K} \cdot \mathbf{c}^{-1} \pmod p,$$

Імовірність у такий спосіб відшукати складові ключа становить порядку 0,1...0,3 для ключа розміру 100. Для ключів великого розміру (~500) ця ймовірність дуже мала. Застосувавши даний метод достатня кількість разів, можна повністю відновити ключ.

Для захисту від атаки такого типу використовується розширений метод шифрування NTRU-FORST. Для шифрування використовується осліплюючий багаточлен $\mathbf{r}(x) = H(\mathbf{m}(x), R)$, де H – криптографічно-стійка хеш-функція, а *R* – випадковий наборбит. Одержавши повідомлення, Боб розшифровує його. Далі Боб шифрує тільки що розшифроване повідомлення, у такий же спосіб, що й Аліса. Після звіряє його на відповідність із отриманим. Якщо повідомлення ідентичні, то Боб приймає повідомлення, інакше відбраковує.

Параметри стійкості й швидкодія

Незважаючи на те, що існують швидкі алгоритми пошуку зворотного полінома, розроблювачі запропонували для комерційного застосування як секретний ключ *f* брати:

$$\mathbf{f} = 1 + p\mathbf{F},$$

де F – малий поліном. У такий спосіб обраний ключ має наступні переваги:

f завжди має зворотний за модулем *p*, а саме:

$$\mathbf{f}^{-1} = \mathbf{f}_p = 1 \pmod{p}.$$

Так як:

$$\mathbf{f}_p = 1 \pmod{p}$$

нам більше не потрібно при розшифровці множити на зворотний поліном $\mathbf{f}_p$, і він випадає з розряду секретного ключа.

Одне з досліджень показало, що NTRU на 4 порядки швидше RSA і на 3 порядки – ECC.

Як уже згадувалися раніше розроблювачі, для забезпечення високої стійкості алгоритму, пропонують використовувати тільки рекомендовані параметри, позначені в таблиці:

Таблиця 3.2 – Рекомендовані параметри

Позначення	***N***	***q***	***p***	***df***	***dg***	***dr***	**Гарантована стійкість**
NTRU167:3	167	128	3	61	20	18	Помірний рівень стійкості
NTRU251:3	251	128	3	50	24	16	Стандартний рівень стійкості
NTRU503:3	503	256	3	216	72	55	Найвищий рівень стійкості
NTRU167:2	167	127	2	45	35	18	Помірний рівень стійкості
NTRU251:2	251	127	2	35	35	22	Стандартний рівень стійкості
NTRU503:2	503	253	2	155	100	65	Найвищий рівень стійкості

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

Функціональна схема складається з наступних блоків:

1. Головне вікно програми.
2. Блок шифрування даних.
3. Блок дешифрування даних.
4. Блок гарантованого знищення інформації.
5. Блок блокування комп’ютера за допомогою носія інформації.
6. Блок допомоги та інформації про програму.
7. Блок розбиття носія інформації на незашифровану частину, та

зашифровану частину.

8. Блок зчитування та перевірки на легітимність паролю.
9. Блок підрахунку спроб введення некоректного паролю.

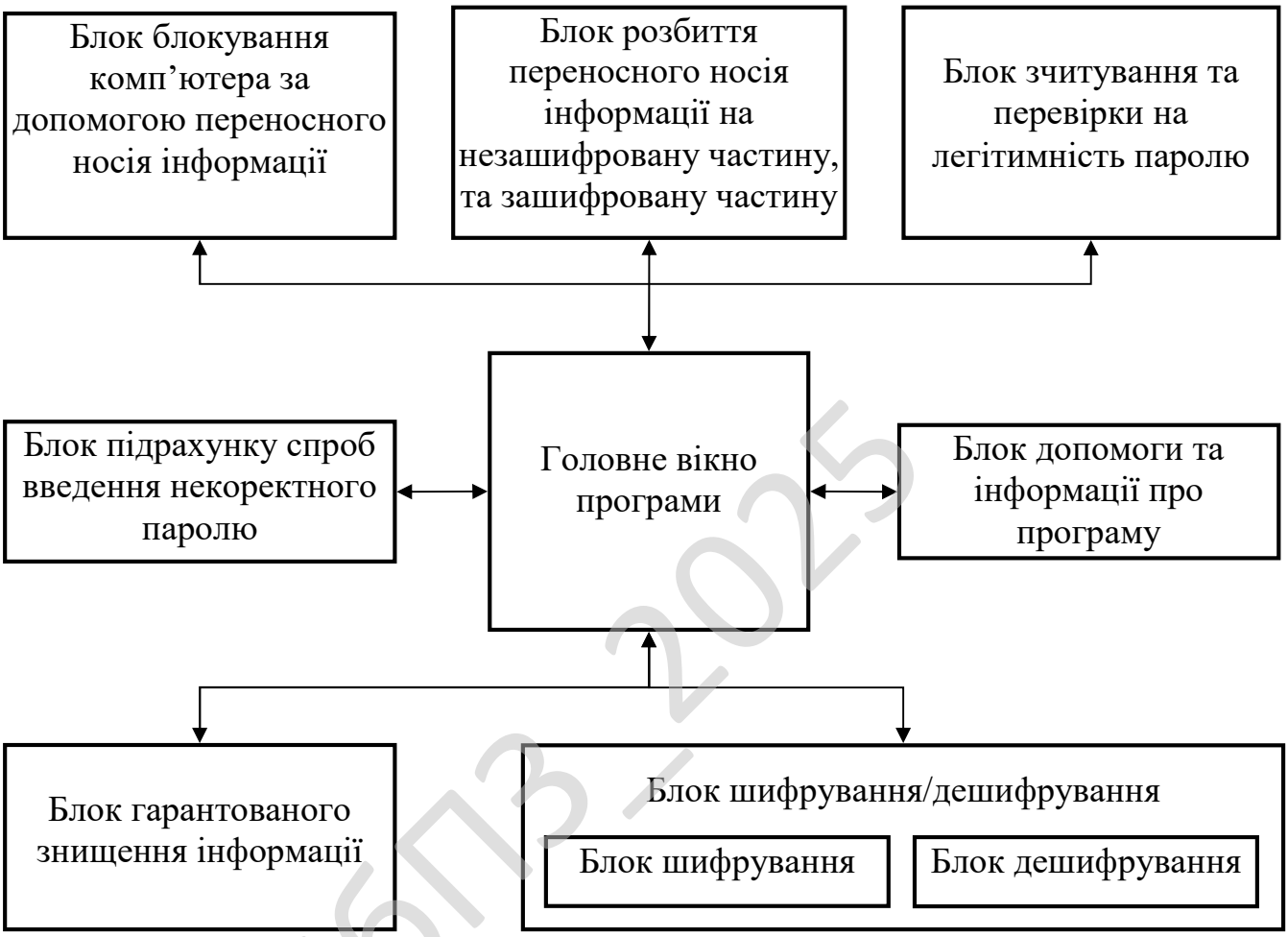

Рисунок 3.2 – Функціональна схема програмного забезпечення

Розглянемо більш детально функціональні блоки програмного забезпечення.

Головне вікно програми

Головне вікно додатка призначене для доступу до усіх функцій програми й містить в собі:

– назву програмного модуля;

– рядок головного меню;

– панель інструментів;

– робочу область;

– статусний рядок стану.

Блок гарантованого знищення інформації

Цей блок призначений для гарантованого знищення інформації, при неправильному введені паролю. З основу був вибраний алгоритм Гутмана, виходячи з наступних міркувань.

Таблиця 3.3 – Алгоритм Гутмана

Цикл	Дані	Цикл	Дані
1	Псевдовипадкові	19	#99
2	Псевдовипадкові	20	#AA
3	Псевдовипадкові	21	#BB
4	Псевдовипадкові	22	#CC
5	#55	23	#DD
6	#AA	24	#EE
7	#92 #49 #24	25	#FF
8	#49 #24 #92	26	#92 #49 #24
9	#24 #92 #49	27	#49 #24 #92
10	#00	28	#24 #92 #49
11	#11	29	#6D #B6 #DB
12	#22	30	#B6 #DB #6D
13	#33	31	#DB #6D #B6
14	#44	32	Псевдовипадкові
15	#55	33	Псевдовипадкові
16	#66	34	Псевдовипадкові
17	#77	35	Псевдовипадкові
18	#88		

Всі програмні реалізації алгоритмів знищення інформації засновані на найпростіших операціях запису, тим самим відбувається багаторазовий перезапис інформації в секторах диска помилковими даними. Залежно від алгоритму це може бути випадкове число генератора псевдовипадкових чисел або фіксоване значення. Як правило, кожний алгоритм передбачає запис восьми бітових одиниць (#FF) і нуля (#00). В існуючих алгоритмах перезапис може виробляється від одного до 35 і більше раз. Існують реалізації з можливістю довільного вибору числа циклів перезапису.

Теоретично, найпростішим методом знищенні вихідного файлу є його повний перезапис байтом #FF, тобто бітовою маскою з восьми логічних одиниць (11111111), нулів або довільних чисел, тим самим виключивши його програмне відновлення стандартними засобами, доступними користувачеві. Однак з використанням спеціалізованих апаратних засобів, що аналізують поверхню магнітних носіїв і дозволяють відновити вихідну інформацію виходячи з показників залишкової намагніченості, існує ймовірність, що найпростіший перезапис не гарантує повноцінне знищення.

З метою виключення можливості відновлення й розроблені існуючі алгоритми знищення інформації:

– Найбільш відомий і розповсюджений алгоритм, застосовуваний в американському національному стандарті Міністерства оборони Do 5220.22-M. Варіант E відповідно до даного стандарту передбачає два цикли запису псевдовипадкових чисел і один – фіксованих значень, залежних від значень першого циклу, четвертий цикл – верифікація записів. У варіанті ECE перезапис даних виробляється 7 разів – 3 рази байтом #FF, три #00 і один #F6.

– В алгоритмі Брюса Шнайра в першому циклі записується #FF, у другому – #00 і в п'яти циклах – псевдовипадкові числа. Уважається одним з найбільш ефективних.

– У найбільш повільному, але, на думку безлічі експертів, найбільш ефективному алгоритмі Питера Гутмана, існує 35 циклів, у яких записують усе

найбільш ефективні бітові маски, даний алгоритм заснований на його теорії знищення інформації.

– В алгоритмі, передбаченого американським національним стандартом NAVSO P-5239-26 для MFM-кодуємих пристроїв у першому циклі записується #01, у другому – #7FFFFFF, у третьому – послідовність псевдовипадкових чисел, у четвертому проходить верифікація. У варіанті для RLL – кодуємих пристроїв даного алгоритму в другому циклі записується #27FFFFFF.

– В алгоритмі, що описує німецький національний стандарт VSITR з першого по шостий цикл записуються послідовно байти #00 і #FF, у сьомому #AA.

– Існує думка про існування алгоритму, описаного Російським національним стандартом ДСТ 50739-95, що передбачає запис #00 у кожний байт кожного сектора для систем з 4-6 класи захисту й запис псевдовипадкових чисел у кожний байт кожного сектора для систем 1-3 класу захисту. Однак даний стандарт містить лише формулювання «Очищення повинне вироблятися шляхом запису інформації, що маскує, до пам’яті при її звільненні перерозподілі», що не містить якої-небудь деталізації щодо порядку перезапису, кількості циклів і бітових масок.

– В алгоритмі Парагона перший цикл полягає в перезаписі унікальними 512-бітними блоками, використовуючи криптографічно безпечний генератор випадкових чисел, потім, у другому циклі кожний перезаписуваний блок переписується своїм двійковим доповненням, третій цикл повторює перший цикл із новими унікальними випадковими блокам, у четвертому циклі відбувається перезапис байтом #AA. Завершується знищення інформації циклом верифікації.

Як правило, для утруднення програмного відновлення інформації, перезапис інформації в окремому файлі відповідно до алгоритму знищення супроводжується установкою нульового розміру файлу і його перейменуванням, використовуючи довільний набір символів. Потім слідує видалення файлу з таблиці розміщення файлів.

Блок розбиття носія інформації на незашифровану частину, та зашифровану частину

Дозволяє розбити накопичувач на відкриту й захищену частини. При виконанні даної операції на першу буде поміщений і невеликий додаток для доступу до другої (хоча перемикатися можна й за допомогою «загальної» утиліти).

Потім задаємо пароль звичайним чином і все готово. Єдиний недолік цієї схеми – оскільки обидві частини монтуються під одним ім'ям і просто перемикаються, одержати доступ відразу до обох неможливо.

Блок зчитування та перевірки на легітимність паролю

Блок зчитування та перевірки на легітимність паролю дозволяє зчитати пароль та порівняти його з тим, який збережений у програмі. Також є можливість заміни паролю, засобом введення старого паролю, та нового паролю з підтвердженням.

Блок підрахунку спроб введення некоректного паролю

Призначення цього блоку заключається у тому, що пристрій автоматично блокується й гарантовано видаляється інформація після 10 невдалих спроб уведення пароля.

Блок шифрування даних

Цей блок шифрує дані використовуючи алгоритм NTRUEncrypt. Детальна робота цього алгоритму розписана у пунктах 3.1, та 3.2.

Блок дешифрування даних

Цей блок розшифрує дані використовуючи алгоритм NTRUEncrypt. Детальна робота цього алгоритму розписана у пунктах 3.1, та 3.2.

Блок блокування комп’ютера за допомогою носія інформації

Цей блок дозволяє блокувати комп’ютер при необхідності й надавати доступ до даних, які зберігаються на ЕОМ, тільки при підключеній флешці.

Блок допомоги та інформації про програму

У цьому блоці знаходиться допомога по використанню програми, та інформацію про розробника, версію, та дату випуску програмного продукту.

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

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. Після початку роботи розробленого ПЗ ми потрапляємо до головного блоку системи звідки через ланку дій відбувається наступне:

– Інтерфейс ПЗ.

– Моніторинг підсистеми Flash носіїв.

– Підсистема блокування ПК на основі технології NTRUEncrypt.

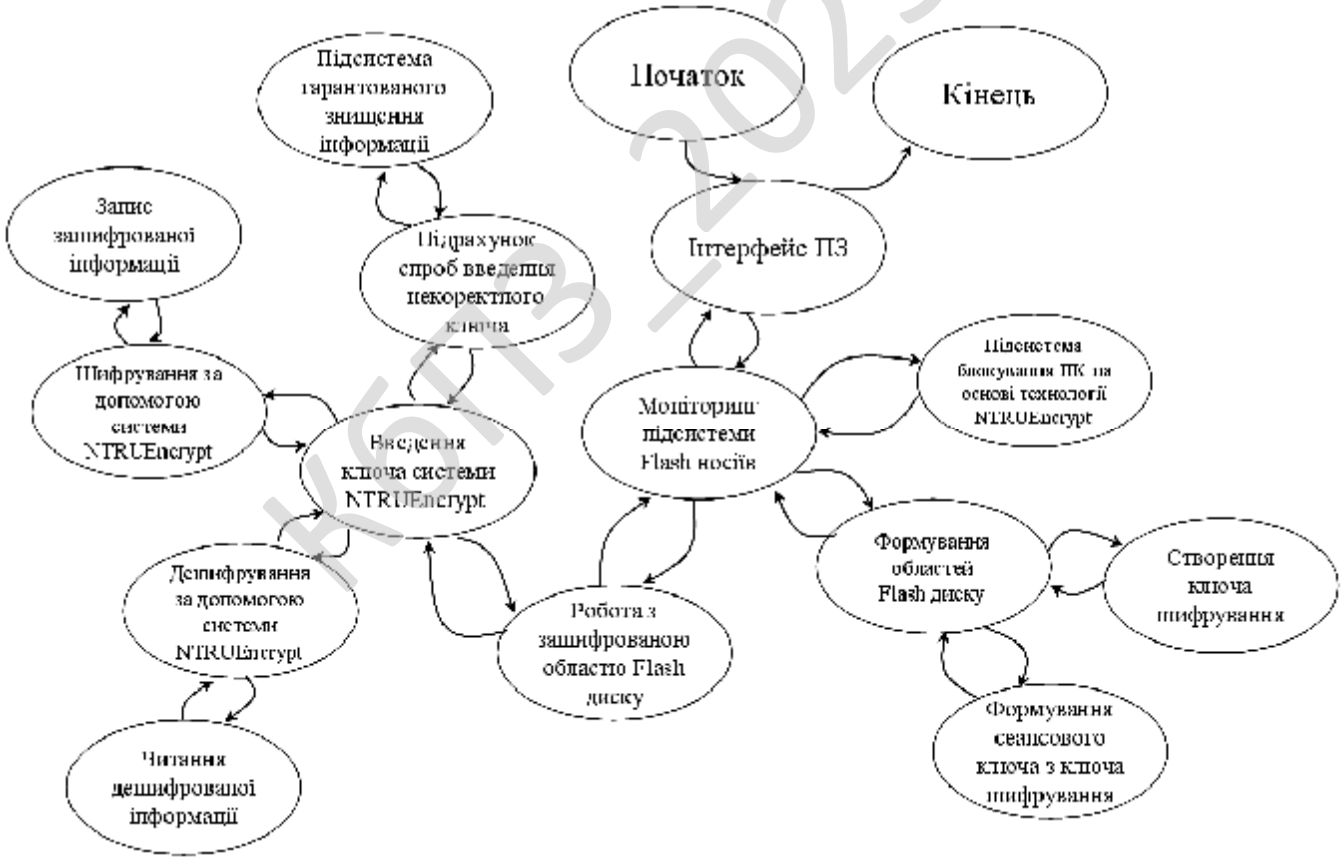

Рисунок 3.3 – Діаграма взаємодії процесів

– Формування областей Flash диску.

– Створення ключа шифрування.

– Формування сеансового ключа з ключа шифрування.

– Робота з зашифрованою областю Flash диску.

– Введення ключа системи NTRUEncrypt.

– Підрахунок спроб введення некоректного ключа.

– Підсистема гарантованого знищення інформації.

– Шифрування за допомогою системи NTRUEncrypt.

– Запис зашифрованої інформації.

– Дешифрування за допомогою системи NTRUEncrypt.

– Читання дешифрованої інформації.

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

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

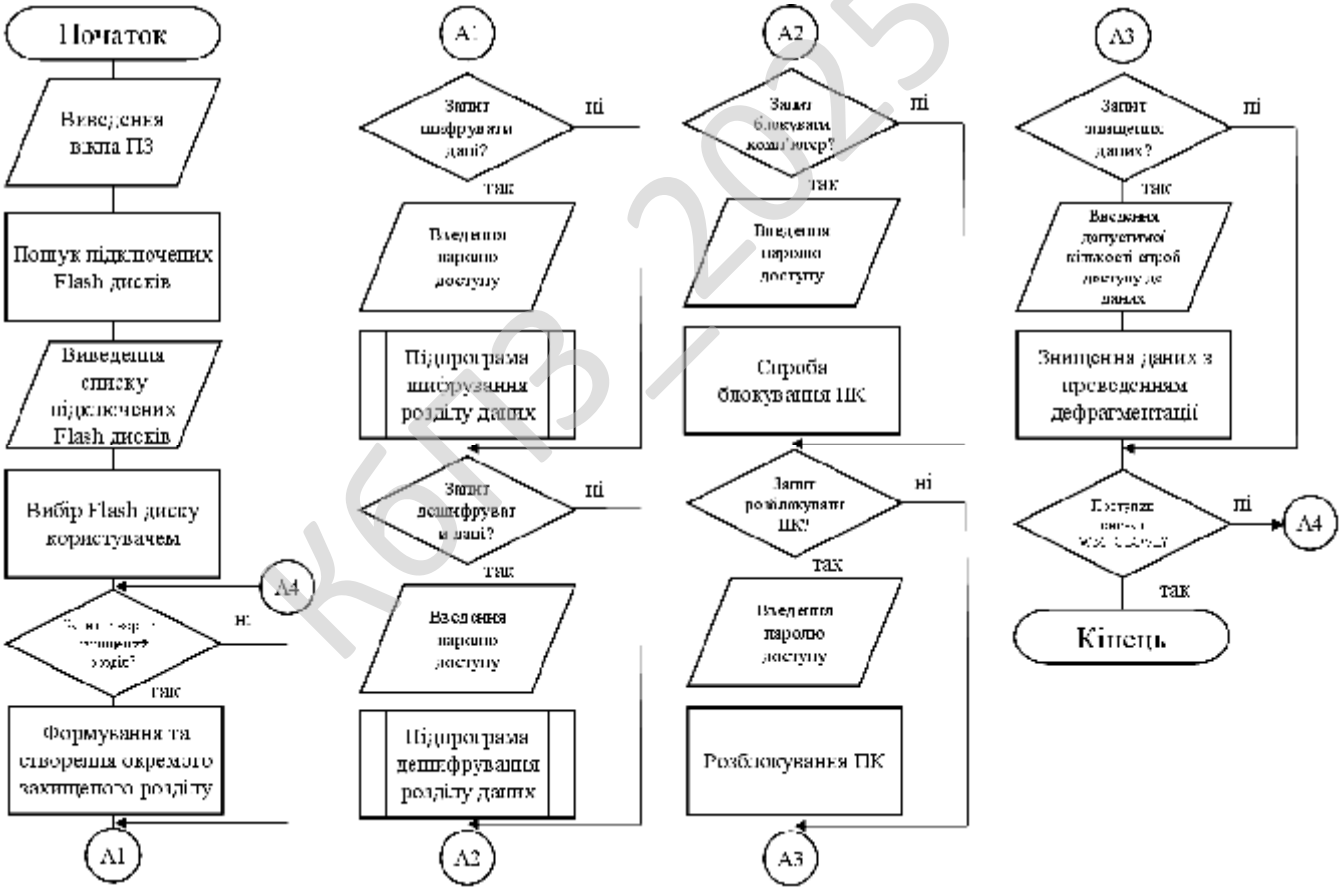

Рисунок 4.1 – Блок схема основної програми

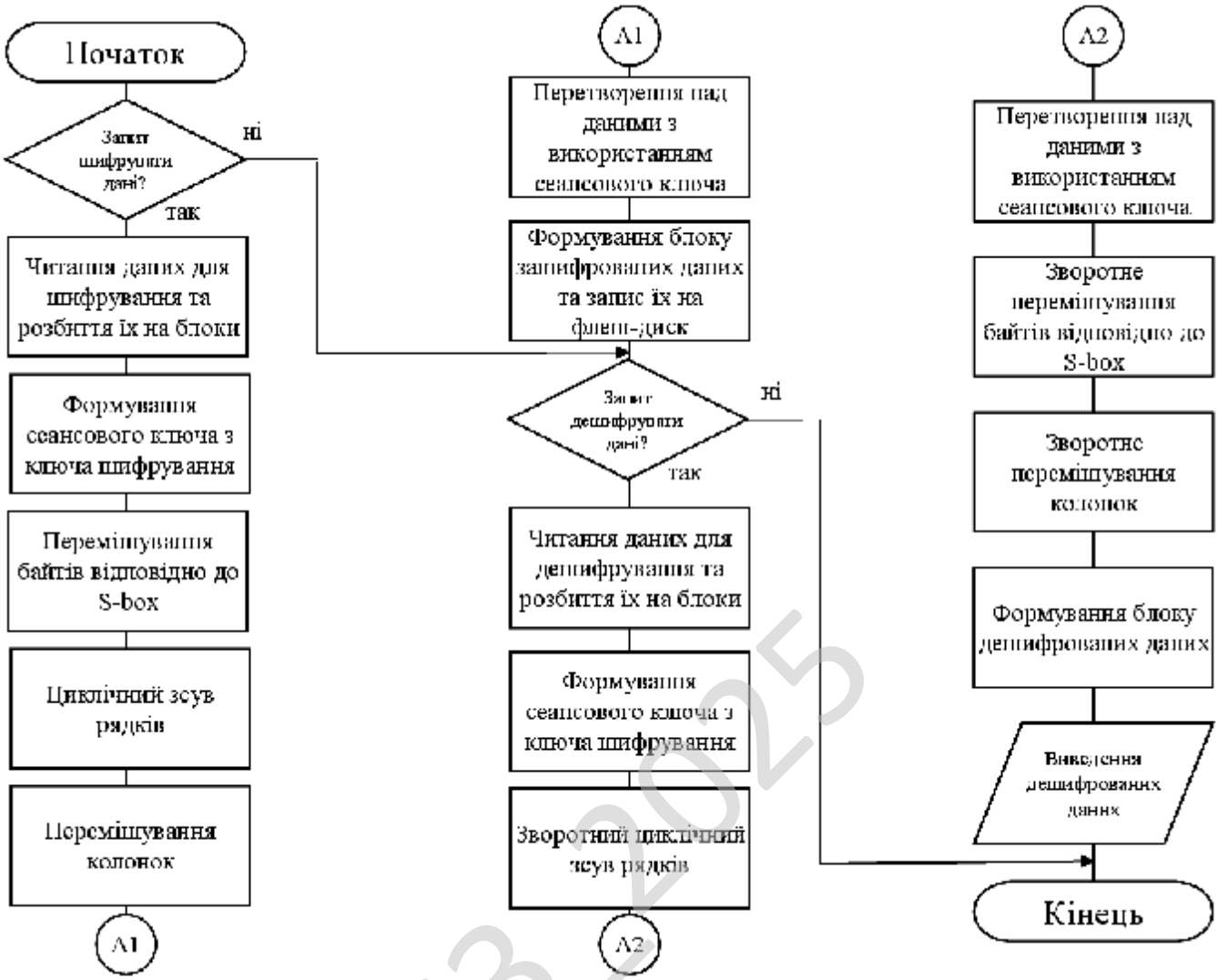

Рисунок 4.2 – Блок схема підпрограми

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

При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення.

UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм. Різні види діаграм які підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний код. Основною причиною використання мови UML є спілкування розробників між собою.

Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту.

UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об’єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки.

Практично усі CASE-засоби (програми автоматизації процесу аналізу і проектування) мають підтримку UML. Моделі розроблені в UML, дозволяють

значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи.

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

– Керівникам проектів, які керують розподілом завдань і контролем за проектом.

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

– Бізнес-аналітикам, які досліджують реальну систему і здійснюють інжиніринг і реінжиніринг бізнесу компанії.

– Програмістам які реалізовують модулі інформаційної системи.

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності. Використання побудованої моделі при модифікаціях системи дає можливість усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

Опис алгоритмів функціонування системи

Спочатку розглянемо процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом.

Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення.

Загалом процес розгортання складається з кількох взаємопов'язаних дій із можливими переходами між ними. Ця активність може відбуватися як з боку виробника так і з боку споживача. Оскільки кожна програмна система є унікальною, то усі процеси та процедури під час розгортання важко передбачити. Тому, "розгортання" можна трактувати як загальний процес відповідно до певних

вимог та характеристик. Розгортання може здійснюватись програмістом і в процесі розробки програмного забезпечення.

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

– Випуск.

– Встановлення та активація.

– Деактивація.

– Адаптація.

– Обновлення.

– Вмонтування.

– Відстежування версій.

– Видалення.

– Вилучення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

– Виділення критичних, з точки зору загального результату, процедур в діяльності організації. Коли набір таких процедур визначений, необхідно в першу чергу використовувати ІТ рішення для автоматизації операцій усередині саме цих процедур. Таким чином, розроблене ІТ рішення автоматично стає життєво важливим і затребуваним для організації, а також буде забезпечена публічність процесу впровадження;

– Розширення нормативної бази організації шляхом включення до неї регламентів, що описують порядок виконання процедур автоматизованих процесів. В іншому випадку є небезпека виникнення неузгодженості між автоматизованими процедурами та іншими процесами організації.

– Виконання робіт з загальної стандартизації існуючої діяльності організації, коли виділяються кращі практики виконання процедур і включаються в ІТ рішення за принципом найбільшої корисності для більшості учасників. Відсоток таких процедур щодо загального обсягу автоматизації може бути

невеликий, але це надає процесу побудови рішення вагу в організації за рахунок збільшення його необхідності.

Опис системи

Система кібербезпеки для забезпечення конфіденційності інформації розробляється на основі криптографічного алгоритму NTRUEncrypt. Ця система забезпечує захист передавання та збереження даних у відкритих або слабо захищених каналах зв'язку.

Основою захисту виступає стійкість алгоритму NTRU до квантових атак та його висока продуктивність. Система реалізується мовою програмування Python.

Програмне середовище Python дозволяє реалізувати алгоритми високого рівня складності та забезпечити інтеграцію із сучасними мережевими технологіями.

Система складається з декількох основних модулів. Модуль генерації ключів створює пару відкритого та закритого ключів. Модуль шифрування приймає відкритий ключ та відкритий текст, виконує операції перетворення та формує зашифроване повідомлення.

Модуль дешифрування отримує закритий ключ та зашифрований текст і виконує розшифрування, відновлюючи початкову інформацію. Додатковий модуль забезпечує передачу даних по захищеному каналу з використанням бібліотек Python socket та ssl.

Архітектура системи є модульною.

Основні функціональні компоненти взаємодіють через чітко визначені інтерфейси. Кожен модуль виконує одну конкретну функцію. Це підвищує гнучкість системи та спрощує її тестування. Модулі імпортуються до головного скрипта, який ініціалізує програму та керує її роботою.

У вихідному коді Python імпортуються стандартні бібліотеки для обробки чисел та масивів, зокрема numpy, а також модулі random та hashlib для генерації випадкових чисел та хешування.

Основна математична частина алгоритму працює з поліноміальними кільцями, що описується у вигляді масивів коефіцієнтів. Для обробки цих об’єктів використовуються функції додавання, множення та обчислення обернених елементів у кільцях.

Ключі створюються шляхом генерації двох поліномів f і g, де поліном f має обернений елемент як за модулем p, так і за модулем q. Значення p дорівнює 3, значення q дорівнює 2048, а розмір кільця n встановлюється рівним 167.

Ці параметри відповідають мінімально допустимим для забезпечення криптостійкості відповідно до специфікації NTRUEncrypt.

Генерація ключів виконується через побудову поліномів f і g. Поліном f є приватним ключем. Його обернений поліном використовується при шифруванні. Поліном g використовується при створенні відкритого ключа h.

Під час шифрування випадковий поліном r множиться на відкритий ключ h, результат додається до повідомлення m, що масштабується множенням на значення p.

Отриманий поліном зводиться за модулем q. Це забезпечує ефективне перетворення та захист інформації. Під час дешифрування зашифроване повідомлення множиться на поліном f і зводиться за модулем q.

Результат масштабується за допомогою параметра p, після чого використовується обернений поліном f для отримання початкового повідомлення m.

У модулі передачі даних формується захищене з’єднання. Сертифікати SSL використовуються для підтвердження автентичності серверів та клієнтів. Зашифровані повідомлення передаються по TCP сокетах, що забезпечує гарантовану доставку.

Наведемо приклад частини вихідного коду Python, що реалізує генерацію ключів.

```
import numpy as np
import random
```

```
# Встановлення параметрів
N = 167
p = 3
q = 2048

# Генерація поліномів з коефіцієнтами -1, 0, 1
def generate_polynomial(N, d):
# Ініціалізація полінома нулями
    poly = [0]*N
# Випадковий розподіл +1
    ones = random.sample(range(N), d)
# Випадковий розподіл -1
    neg_ones = random.sample([i for i in range(N) if i not in ones], d)
# Запис значень до полінома
    for i in ones:
        poly[i] = 1
    for i in neg_ones:
        poly[i] = -1
# Повернення полінома
    return np.array(poly)

# Генерація пари ключів
def generate_keypair():
# Генерація полінома f
    f = generate_polynomial(N, 5)
# Генерація полінома g
    g = generate_polynomial(N, 5)
# Перевірка існування оберненого полінома f за модулем p та q
    Fp = invert_polynomial(f, p)
    Fq = invert_polynomial(f, q)
# Генерація відкритого ключа h
    h = (p * convolve_polynomials(Fq, g)) % q
# Повернення відкритого і закритого ключів
    return h, f

# Приклад виклику генерації ключів
public_key, private_key = generate_keypair()
```

Для підтвердження коректності роботи системи виконується тестування шифрування та дешифрування. Тестове повідомлення перетворюється до поліноміальної форми, шифрується з використанням відкритого ключа, після чого виконується дешифрування приватним ключем.

Результати порівнюються з оригінальним повідомленням. При використанні зазначених параметрів точність відновлення повідомлення досягає 100 відсотків.

Час генерації ключів не перевищує 1 секунди на звичайних комп'ютерах із частотою процесора 3 ГГц. Час шифрування повідомлення обсягом 256 біт становить близько 0.05 секунди, що демонструє достатню продуктивність системи.

Обґрунтованість вибору параметрів забезпечується співвідношенням між розміром ключа, рівнем безпеки та швидкодією. Параметри n, p та q відповідають сучасним рекомендаціям для криптосистем на основі NTRUEncrypt.

Використання Python для реалізації дозволяє забезпечити швидку розробку, зручне тестування та інтеграцію з іншими інформаційними системами.

Система орієнтована на використання в автоматизованих інформаційних системах організацій, що вимагають захисту даних при передачі через відкриті канали зв'язку, а також у хмарних сховищах для шифрування збережених даних.

Реалізація функцій аудиту та журналювання забезпечує контроль доступу до даних та реєстрацію всіх подій.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою CRYPTON – алгоритм симетричного блочного шифрування (розмір блоку 128 біт, ключ довжиною до 256 біт), розроблений південнокорейським криптологом Чьо Лім Хун з південнокорейської компанії Future Systems, яка з кінця 1980-х років працює на ринку забезпечення мереж і захисту інформації.

Алгоритм був розроблений в 1998 році в якості шифру – учасника конкурсу AES. Як зізнавався автор, конструкція алгоритму спирається на алгоритм SQUARE[1]. В алгоритмі Crypton немає традиційних для блочних шифрів мережі Фейстеля. Основу даного шифру становить так звана SP-мережа (повторювана циклова функція, що складається із замін-перестановок, орієнтована на розпаралелену нелінійну обробку всього блоку даних). Крім високої швидкості, перевагами таких алгоритмів є полегшення дослідження стійкості шифру до методів диференціального та лінійного криптоаналізу, що є на сьогодні основними інструментами розтину блочних шифрів. На конкурс AES була представлена версія алгоритму Crypton v0.5. Однак, як казав Чьо Лім Хун, йому не вистачало часу для розробки повної версії. І вже на першому етапі конкурсу AES в ході аналізу алгоритмів, версія Crypton v0.5 була замінена на версію Crypton v1.0. Відмінність нової версії від первинної полягала в зміні таблиці замін та в модифікації процесу розширення ключа.

Як і інші учасники конкурсу AES, Crypton призначений для шифрування 128-бітових блоків даних[2]. При шифруванні використовуються ключі шифрування для декількох фіксованих розмірів – від 0 до 256 біт з кратністю 8 бітів. Структура алгоритму Crypton – структура «Квадрата» – багато в чому схожа на структуру алгоритму Square, створеного в 1997 році. Криптографічні перетворення для алгоритмів з даною структурою можуть бути виконані як для цілих рядків і стовпців масиву, так і над окремими його байтами. (Варто зазначити, що алгоритм Square був розроблений авторами майбутнього переможця конкурсу AES – авторами алгоритму Rijndael – Вінсентом Ріджменом і Джоан Дейменом.)

Шифрування

Алгоритм Crypton являє 128-бітовий блок шифруємих даних у вигляді байтового масиву 4×4, над якими в процесі шифрування проводиться кілька раундів перетворень. У кожному раунді передбачається послідовне виконання наступних операцій:

– Таблична заміна γ;

– Лінійне перетворення π;

– Байтова перестановка τ;

– Операція σ.

Таблична заміна γ

Алгоритм Crypton використовує 4 таблиці замін. Кожна з яких заміщає 8-бітне вхідне значення на вихідне такого ж розміру.

Лінійне перетворення π

Тут використовується 4 спеціальні константи. Ці константи об'єднані в маскуючі послідовності

Байтова перестановка τ

Дана перестановка перетворює найпростішим чином рядок даних у стовпець.

Операція σ

Дана операція є побітовим складанням всього масиву даних з ключем раунду. Зауважимо, саме 12 раундів шифрування рекомендується автором алгоритму Чьо Хун Лімом, проте сувора кількість раундів не встановлена.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

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

– Розділ меню: Файл; Флеш-диски; Шифрування; Довідка.

– Розділ відображення розташування даних на Flash-диску.

– Розділ функціональних кнопок: Створення зашифрованого розділу; Створення/зміна ключів шифрування; Шифрувати; Дешифрувати; Управління розділами флеш-диску; Встановлення/зміна паролю доступу; Блокувати комп’ютер; Зняти блокування.

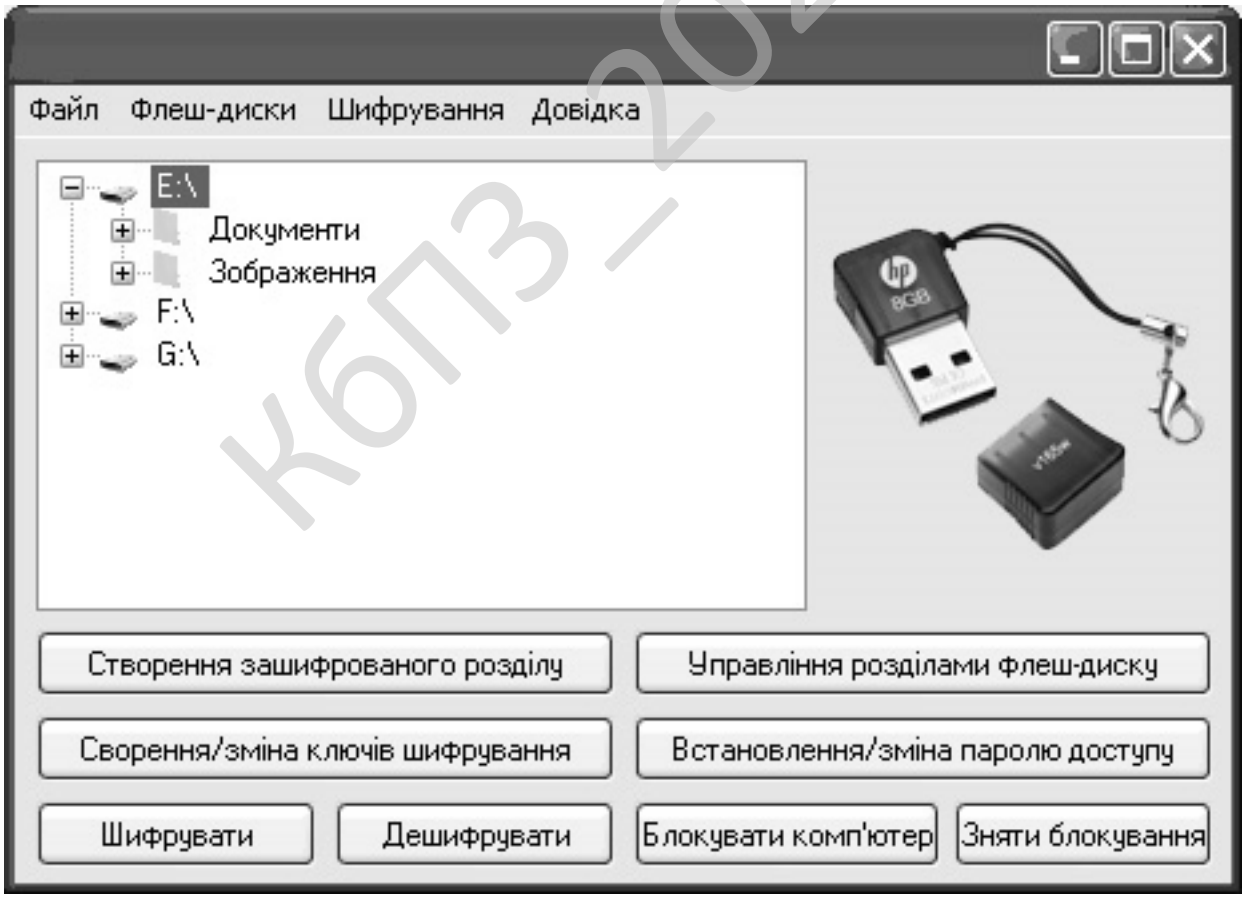

Рисунок 5.1– Головне вікно ПЗ

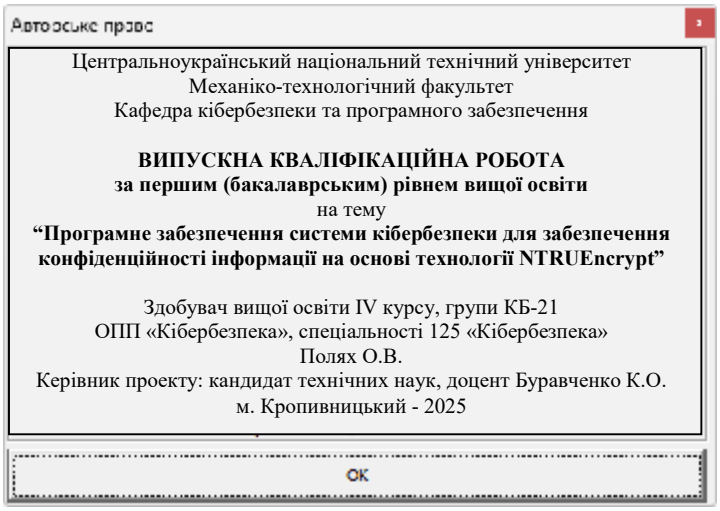

Рисунок 5.2 – Довідка (авторське право)

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення. Обрано умови розповсюдження – Shareware. Під умовно-безплатним програмним забезпеченням можна розуміти спосіб або метод розповсюдження комерційного ПЗ на ринку (тобто на шляху до кінцевого користувача), при якому випробувачеві пропонується обмежена за можливостями (не повнофункціональна або демонстраційна версія), терміном дії (тріал версія) або версія з вбудованим набридливим нагадуванням про необхідність оплати використання програми. В угоді про використання (ліцензії для кінцевого користувача, EULA) також може бути обумовлена заборона на комерційне або професійне (не тестове) її використання. Протягом певного терміну, що становить зазвичай тридцять днів, він може користуватися програмою, тестувати її, освоювати її можливості. Якщо після закінчення цього терміну користувач вирішить продовжити використання ПЗ, він зобов'язаний купити його (зареєструватися), заплативши авторові певну суму. В іншому випадку користувач повинен припинити використання ПЗ та видалити його зі свого комп'ютера.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки для забезпечення конфіденційності інформації на основі технології NTRUEncrypt.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем для забезпечення конфіденційності інформації на основі технології NTRUEncrypt.

– Досліджена система для забезпечення конфіденційності інформації на основі технології NTRUEncrypt.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки для забезпечення конфіденційності інформації на основі технології NTRUEncrypt.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання для забезпечення конфіденційності інформації на основі технології NTRUEncrypt.

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 Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для

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

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм CRYPTON.

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

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

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

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

3. Kuznetsov, O., Smirnov, O., Mormul, M., Kotukh, Y., Zvieriev, V. «Comparative Research on Cryptocurrency Efficiency: An Objective Analysis of Key Metrics». *International Journal of Computing* 23(4), 2024. pp. 563-573.

4. Kuznetsov O., Frontoni E., Kuznetsova K., Smirnov O., Kostenko V. «Blockchain applications in metaverse environments: new horizons». *Advanced Metaverse Wireless Communication Systems.* pp. 255-293. 2024.

5. Kuznetsov, O., Frontoni, E., Chevardin, V., Smirnov, O., Imoize, A.L. «Advancing metaverse security with cryptographic innovations». *Advanced Metaverse Wireless Communication Systems.* pp 351-386. 2024.

6. Kuznetsov O., Frontoni E., Kuznetsova Y., Smirnov O., Moskovchenko I. «Trust-Based Security Architecture for Edge Computing: A Simulation Study of Dynamic Trust Evolution and Attack Detection». *CEUR Workshop Proceedings,* 2024, 3909, pp. 227–241.

7. Lakhno, V., Malyukov, V., Smirnov, O., Bebeshko, B., Chubaievskiy, V., Zhumadilova, M., Malyukova, I., Smirnov, S. «Multifactorial Model for Targeted Attacks Counteracting Within the Framework of a Multi-Step Quality Game with Fuzzy Information». *8th International Symposium on Intelligent Informatics, ISI 2023*, 2025. vol 389. pp 377-389. Springer, Singapore.

8. Kuznetsov, O., Frontoni, E., Kryvinska, N., Chevardin, V., Smirnov, O. «Wireless Network Encryption Stream Ciphers, Computational Modeling, and Security Analysis». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 379–402.

9. Kuznetsov, O., Frontoni, E., Kryvinska, N., Smirnov, O., Imoize, G.L. «Computational Modeling of Enhanced Spread Spectrum Codes for Asynchronous Wireless Communication». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 403–447.

10. Kuznetsov, O., Frontoni, E., Kandiy, S., Smirnova, T., Prokopov, S., Bilanovych, A. «New Cost Function for S-boxes Generation by Simulated Annealing Algorithm». Lecture Notes on Data Engineering and Communications Technologies, 2023. vol 180. pp. 310-320. Springer, Cham.

11. Kuznetsov, O., Frontoni, E., Kandiy, S., Smirnov, O., Ulianovska, Y., Kobylianska, O. «Heuristic Search for Nonlinear Substitutions for Cryptographic Applications». *Lecture Notes on Data Engineering and Communications Technologies,* 2023. vol 180. Springer, Cham. pp. 288-298.

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

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

14. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

15. Kuznetsov, O., Kandiy, S., Frontoni, E., Smirnov, O. «Trade-offs in Post-Quantum Cryptography: A Comparative Assessment of BIKE, HQC, and Classic McEliece». *CEUR Workshop Proceedings,* Volume 3504, 2023, pp. 1-11.

16. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» *CEUR Workshop Proceedings,* Volume 3187, 2022,

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

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

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

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

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

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

23. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-

feedback shift register». *International Journal of Computing*; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

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

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

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

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

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

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

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

31. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-

quantum application». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

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

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

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

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

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

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

38. Smirnov, O., Ulichev, O., Meleshko, Y., Khokh, V., Goncharenko, I. «Method of Choosing Objects for Informational Influence in Social Networks during

Information Campaign Based on the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, Vol 2588, P. 215-227, 2019.

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

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

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

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

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

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

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

46. Smirnov, O., Kuznetsov, A., Kiian, A., Babenko, B., Zhosan, H., Prokopovych-Tkachenko, D., «Soft Decoding Method for Turbo-Productive Codes», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT 2019*, Lviv, Ukraine, 2-6 July, 2019, P. 129-134.

47. Ткаченко, О., Ільєнко, А., Улічев, О., Мелешко, Є., Смірнов, О. «Правові засади поширення інформаційних впливів в соціальних мережах». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»,* 2024. № 2(26), С. 170–188.

48. Смірнова Т.В., Конопліцька-Слободенюк О.К., Буравченко К.О., Смірнов С.А., Кравчук О.В., Козірова Н.Л., Смірнов О.А. «Дослідження технологій забезпечення кібербезпеки хмарних сервісів IaaS, PaaS та SaaS». *Кібербезпека: освіта, наука, техніка.* 2024. №4(24), С. 6-27.

49. Вінтенко, Б., Миронець, І., Смірнов, О., Кравчук, О., Козірова, Н., Савеленко, Г., Коваленко, А. «Дослідження вимог та аналіз кібербезпеки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Кібербезпека: освіта, наука, техніка.* 2024. №3(23), С. 111-131.

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

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки для забезпечення конфіденційності інформації на основі технології NTRUEncrypt.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 57-02 від 17.01.2025 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи кібербезпеки для забезпечення конфіденційності інформації на основі технології NTRUEncrypt.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи кібербезпеки з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи кібербезпеки для забезпечення конфіденційності інформації на основі технології NTRUEncrypt;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Python.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 66 аркушів.

8 Етапи розробки

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

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

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2025 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 6.06.2025 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

_____ Буравченко К.О.

Програмне забезпечення системи кібербезпеки для забезпечення конфіденційності інформації на основі технології NTRUEncrypt

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 21

Літера: РП

Кропивницький – 2025 року

Основна програма

```
# Імпорт необхідних бібліотек
import os
import hashlib
import json
import base64
import random
import secrets

# Імпорт математичних функцій
from math import gcd
from functools import reduce

# Імпорт бібліотеки для роботи з NTRU
from ntru import Ntru

# Імпорт бібліотеки для хешування
from Crypto.Hash import SHA256

# Імпорт бібліотеки для симетричного шифрування
from Crypto.Cipher import AES

# Імпорт генератора випадкових чисел
from Crypto.Random import get_random_bytes

# Оголошення класу для NTRU шифрування
class NTRUEncryptSystem:

    # Ініціалізація системи
    def __init__(self, N=167, p=3, q=128):

        # Задання основних параметрів
        self.N = N

        # Задання модуля p
        self.p = p

        # Задання модуля q
        self.q = q

        # Ініціалізація ключів
        self.public_key = None

        # Ініціалізація приватного ключа
        self.private_key = None

        # Створення об'єкта NTRU
        self.ntru = Ntru(N, p, q)

    # Генерація ключової пари
    def generate_keys(self):

        # Виклик генератора ключів
        self.public_key, self.private_key = self.ntru.generate_keypair()

        # Повернення ключів
        return self.public_key, self.private_key

    # Зашифрування даних
    def encrypt(self, message, public_key):

        # Виклик функції шифрування
        encrypted_message = self.ntru.encrypt(message.encode(), public_key)

        # Повернення зашифрованого повідомлення
        return encrypted_message

    # Розшифрування даних
```

```
    def decrypt(self, encrypted_message, private_key):

        # Виклик функції дешифрування
        decrypted_message = self.ntru.decrypt(encrypted_message, private_key)

        # Декодування повідомлення у рядок
        return decrypted_message.decode()

# Клас для управління конфіденційністю даних
class ConfidentialityManager:

    # Ініціалізація системи
    def __init__(self):

        # Створення екземпляра системи NTRU
        self.ntru_system = NTRUEncryptSystem()

        # Генерація ключів для асиметричного шифрування
        self.public_key, self.private_key = self.ntru_system.generate_keys()

        # Генерація ключа для симетричного шифрування
        self.symmetric_key = get_random_bytes(16)

    # Хешування даних
    def hash_data(self, data):

        # Створення об'єкта хешу
        hash_object = SHA256.new()

        # Оновлення хешу
        hash_object.update(data.encode())

        # Отримання хешованого значення
        return hash_object.hexdigest()

    # Симетричне шифрування даних
    def symmetric_encrypt(self, data):

        # Генерація вектора ініціалізації
        iv = get_random_bytes(16)

        # Створення об'єкта шифрування AES
        cipher = AES.new(self.symmetric_key, AES.MODE_CFB, iv)

        # Шифрування даних
        encrypted_data = cipher.encrypt(data.encode())

        # Кодування результату у base64
        return base64.b64encode(iv + encrypted_data).decode()

    # Симетричне дешифрування даних
    def symmetric_decrypt(self, encrypted_data):

        # Декодування base64
        raw_data = base64.b64decode(encrypted_data)

        # Отримання вектора ініціалізації
        iv = raw_data[:16]

        # Отримання зашифрованих даних
        encrypted_message = raw_data[16:]

        # Створення об'єкта AES для дешифрування
        cipher = AES.new(self.symmetric_key, AES.MODE_CFB, iv)

        # Дешифрування
        return cipher.decrypt(encrypted_message).decode()

    # Шифрування повідомлення з використанням NTRU та AES
```

```
    def encrypt_message(self, message):

        # Хешування повідомлення
        hashed_message = self.hash_data(message)

        # Шифрування повідомлення симетрично
        encrypted_symmetric = self.symmetric_encrypt(message)

        # Шифрування ключа AES з використанням NTRU
        encrypted_symmetric_key =
self.ntru_system.encrypt(self.symmetric_key.hex(), self.public_key)

        # Повернення результату у вигляді словника
        return {
            'encrypted_key': base64.b64encode(encrypted_symmetric_key).decode(),
            'encrypted_data': encrypted_symmetric,
            'hash': hashed_message
        }

    # Дешифрування повідомлення з використанням NTRU та AES
    def decrypt_message(self, encrypted_package):

        # Отримання зашифрованого ключа
        encrypted_key_b64 = encrypted_package['encrypted_key']

        # Декодування base64
        encrypted_key_bytes = base64.b64decode(encrypted_key_b64)

        # Дешифрування ключа AES
        decrypted_symmetric_key_hex =
self.ntru_system.decrypt(encrypted_key_bytes, self.private_key)

        # Перетворення ключа у bytes
        symmetric_key_bytes = bytes.fromhex(decrypted_symmetric_key_hex)

        # Встановлення розшифрованого ключа
        self.symmetric_key = symmetric_key_bytes

        # Дешифрування повідомлення
        decrypted_message =
self.symmetric_decrypt(encrypted_package['encrypted_data'])

        # Повернення повідомлення та його хешу
        return {
            'decrypted_data': decrypted_message,
            'hash': encrypted_package['hash']
        }

# Клас для керування користувачами системи
class UserManager:

    # Ініціалізація менеджера користувачів
    def __init__(self):

        # Ініціалізація словника користувачів
        self.users = {}

    # Додавання нового користувача
    def add_user(self, username, password):

        # Генерація солі
        salt = get_random_bytes(16)

        # Хешування паролю разом з сіллю
        hashed_password = self._hash_password(password, salt)

        # Збереження користувача
        self.users[username] = {
            'salt': base64.b64encode(salt).decode(),
```

```
            'password': hashed_password
        }

    # Перевірка авторизації користувача
    def authenticate_user(self, username, password):

        # Перевірка чи існує користувач
        if username not in self.users:

            # Повернення помилкової авторизації
            return False

        # Отримання солі користувача
        salt_b64 = self.users[username]['salt']

        # Декодування солі
        salt = base64.b64decode(salt_b64)

        # Хешування введеного паролю
        hashed_password = self._hash_password(password, salt)

        # Перевірка збігу хешу
        return hashed_password == self.users[username]['password']

    # Приватний метод хешування пароля
    def _hash_password(self, password, salt):

        # Створення об'єкта SHA256
        hash_object = hashlib.sha256()

        # Оновлення хешу паролем та сіллю
        hash_object.update(salt + password.encode())

        # Повернення хешу у hex
        return hash_object.hexdigest()

# Головний клас системи кібербезпеки
class CyberSecuritySystem:

    # Ініціалізація системи
    def __init__(self):

        # Ініціалізація менеджера користувачів
        self.user_manager = UserManager()

        # Ініціалізація менеджера конфіденційності
        self.conf_manager = ConfidentialityManager()

    # Реєстрація користувача
    def register_user(self, username, password):

        # Додавання нового користувача
        self.user_manager.add_user(username, password)

    # Авторизація користувача
    def login_user(self, username, password):

        # Перевірка авторизації
        return self.user_manager.authenticate_user(username, password)

    # Відправлення зашифрованого повідомлення
    def send_secure_message(self, message):

        # Виклик функції шифрування
        encrypted_package = self.conf_manager.encrypt_message(message)

        # Повернення зашифрованого пакету
        return encrypted_package
```

```
    # Отримання розшифрованого повідомлення
    def receive_secure_message(self, encrypted_package):

        # Виклик функції дешифрування
        decrypted_message = self.conf_manager.decrypt_message(encrypted_package)

        # Повернення дешифрованого повідомлення
        return decrypted_message

# Основна функція запуску системи
def main():

    # Створення екземпляра системи кібербезпеки
    system = CyberSecuritySystem()

    # Реєстрація користувача
    system.register_user("admin", "strongpassword123")

    # Авторизація користувача
    if system.login_user("admin", "strongpassword123"):

        # Виведення повідомлення про успішну авторизацію
        print("User authenticated successfully.")

        # Оригінальне повідомлення
        message = "This is a confidential message."

        # Виведення оригінального повідомлення
        print("Original message:", message)

        # Відправлення зашифрованого повідомлення
        encrypted_package = system.send_secure_message(message)

        # Виведення зашифрованих даних
        print("Encrypted package:", json.dumps(encrypted_package, indent=4))

        # Отримання розшифрованого повідомлення
        decrypted_message = system.receive_secure_message(encrypted_package)

        # Виведення розшифрованого повідомлення
        print("Decrypted message:", decrypted_message['decrypted_data'])

        # Виведення хешу
        print("Original hash:", decrypted_message['hash'])

    else:

        # Повідомлення про невдалу авторизацію
        print("Authentication failed.")

# Виклик основної функції
if __name__ == "__main__":

    # Запуск програми
    main()
```

Файл Logger.py

```
import os
import base64
import json
import hashlib
import secrets
import datetime
from Crypto.Random import get_random_bytes
from Crypto.Cipher import AES
from Crypto.Hash import SHA256
from ntru import Ntru

class Logger:
    def __init__(self, log_file='system_log.txt'):
        self.log_file = log_file
        self.create_log_file()

    def create_log_file(self):
        if not os.path.exists(self.log_file):
            with open(self.log_file, 'w') as file:
                file.write('=== CyberSecuritySystem Log ===\n')

    def log(self, event):
        timestamp = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
        log_entry = f'[{timestamp}] {event}\n'
        with open(self.log_file, 'a') as file:
            file.write(log_entry)

class MultiFactorAuth:
    def __init__(self):
        self.otp_storage = {}

    def generate_otp(self, username):
        otp = ''.join([str(secrets.randbelow(10)) for _ in range(6)])
        self.otp_storage[username] = otp
        return otp

    def verify_otp(self, username, otp):
        if username not in self.otp_storage:
            return False
        valid = self.otp_storage[username] == otp
        del self.otp_storage[username]
        return valid

class RoleBasedAccessControl:
    def __init__(self):
        self.roles = {}
        self.permissions = {}

    def add_role(self, role_name):
        if role_name not in self.roles:
            self.roles[role_name] = []

    def assign_permission(self, role_name, permission):
        if role_name in self.roles:
            if permission not in self.roles[role_name]:
                self.roles[role_name].append(permission)

    def assign_user_role(self, username, role_name):
        if username not in self.permissions:
            self.permissions[username] = []
        if role_name in self.roles:
            if role_name not in self.permissions[username]:
                self.permissions[username].append(role_name)

    def check_permission(self, username, permission):
        if username not in self.permissions:
            return False
```

```
        for role in self.permissions[username]:
            if permission in self.roles.get(role, []):
                return True
        return False

class KeyRotationManager:
    def __init__(self, ntru_system):
        self.ntru_system = ntru_system
        self.rotation_interval = 60 * 60 * 24 * 30
        self.last_rotation_time = datetime.datetime.now()

    def needs_rotation(self):
        current_time = datetime.datetime.now()
        elapsed_time = current_time - self.last_rotation_time
        return elapsed_time.total_seconds() >= self.rotation_interval

    def rotate_keys(self):
        new_public_key, new_private_key = self.ntru_system.generate_keys()
        self.ntru_system.public_key = new_public_key
        self.ntru_system.private_key = new_private_key
        self.last_rotation_time = datetime.datetime.now()
        return new_public_key, new_private_key

class SecureFileEncryption:
    def __init__(self, ntru_system):
        self.ntru_system = ntru_system
        self.symmetric_key = get_random_bytes(32)

    def encrypt_file(self, file_path, output_path):
        if not os.path.exists(file_path):
            return False
        with open(file_path, 'rb') as file:
            plaintext = file.read()
        iv = get_random_bytes(16)
        cipher = AES.new(self.symmetric_key, AES.MODE_CFB, iv)
        encrypted_data = cipher.encrypt(plaintext)
        encrypted_key    =    self.ntru_system.encrypt(self.symmetric_key.hex(),
self.ntru_system.public_key)
        with open(output_path, 'wb') as output_file:
            output_file.write(iv + encrypted_data)
        key_file = output_path + '.key'
        with open(key_file, 'wb') as kf:
            kf.write(base64.b64encode(encrypted_key))
        return True

    def decrypt_file(self, encrypted_file_path, key_file_path, output_path):
        if not os.path.exists(encrypted_file_path):
            return False
        if not os.path.exists(key_file_path):
            return False
        with open(key_file_path, 'rb') as kf:
            encrypted_key_b64 = kf.read()
        encrypted_key = base64.b64decode(encrypted_key_b64)
        symmetric_key_hex       =       self.ntru_system.decrypt(encrypted_key,
self.ntru_system.private_key)
        symmetric_key = bytes.fromhex(symmetric_key_hex)
        with open(encrypted_file_path, 'rb') as ef:
            raw_data = ef.read()
        iv = raw_data[:16]
        encrypted_data = raw_data[16:]
        cipher = AES.new(symmetric_key, AES.MODE_CFB, iv)
        decrypted_data = cipher.decrypt(encrypted_data)
        with open(output_path, 'wb') as output_file:
            output_file.write(decrypted_data)
        return True

class CyberSecuritySystemExtended:
    def __init__(self):
        self.logger = Logger()
```

```
        self.user_manager = UserManager()
        self.conf_manager = ConfidentialityManager()
        self.mfa = MultiFactorAuth()
        self.rbac = RoleBasedAccessControl()
        self.key_rotation = KeyRotationManager(self.conf_manager.ntru_system)
        self.file_encryptor =
SecureFileEncryption(self.conf_manager.ntru_system)

    def register_user(self, username, password, role='user'):
        self.user_manager.add_user(username, password)
        self.rbac.add_role(role)
        self.rbac.assign_user_role(username, role)
        self.logger.log(f'User registered: {username} with role: {role}')

    def login_user(self, username, password):
        authenticated = self.user_manager.authenticate_user(username, password)
        if authenticated:
            otp = self.mfa.generate_otp(username)
            self.logger.log(f'User {username} authenticated. OTP generated.')
            return otp
        self.logger.log(f'Failed authentication attempt for user: {username}')
        return None

    def verify_otp(self, username, otp):
        verified = self.mfa.verify_otp(username, otp)
        if verified:
            self.logger.log(f'User {username} passed MFA.')
            return True
        self.logger.log(f'User {username} failed MFA.')
        return False

    def send_secure_message(self, username, message):
        if not self.rbac.check_permission(username, 'send_message'):
            self.logger.log(f'User {username} denied sending message due to
insufficient permissions.')
            return None
        if self.key_rotation.needs_rotation():
            self.key_rotation.rotate_keys()
            self.logger.log('Keys rotated automatically during message send.')
        encrypted_package = self.conf_manager.encrypt_message(message)
        self.logger.log(f'Message encrypted and sent by user: {username}')
        return encrypted_package

    def receive_secure_message(self, username, encrypted_package):
        if not self.rbac.check_permission(username, 'receive_message'):
            self.logger.log(f'User {username} denied receiving message due to
insufficient permissions.')
            return None
        decrypted_message = self.conf_manager.decrypt_message(encrypted_package)
        self.logger.log(f'Message decrypted and received by user: {username}')
        return decrypted_message

    def encrypt_file_for_user(self, username, file_path, output_path):
        if not self.rbac.check_permission(username, 'encrypt_file'):
            self.logger.log(f'User {username} denied file encryption due to
insufficient permissions.')
            return False
        if self.key_rotation.needs_rotation():
            self.key_rotation.rotate_keys()
            self.logger.log('Keys rotated automatically during file
encryption.')
        success = self.file_encryptor.encrypt_file(file_path, output_path)
        if success:
            self.logger.log(f'File {file_path} encrypted by user {username}.
Output: {output_path}')
        else:
            self.logger.log(f'File {file_path} encryption failed for user
{username}.')
        return success
```

```
    def      decrypt_file_for_user(self,      username,      encrypted_file_path,
key_file_path, output_path):
        if not self.rbac.check_permission(username, 'decrypt_file'):
            self.logger.log(f'User  {username}  denied  file  decryption  due  to
insufficient permissions.')
            return False
        if self.key_rotation.needs_rotation():
            self.key_rotation.rotate_keys()
            self.logger.log('Keys    rotated    automatically    during    file
decryption.')
        success      =      self.file_encryptor.decrypt_file(encrypted_file_path,
key_file_path, output_path)
        if success:
            self.logger.log(f'File  {encrypted_file_path}  decrypted  by  user
{username}. Output: {output_path}')
        else:
            self.logger.log(f'File {encrypted_file_path} decryption failed for
user {username}.')
        return success

class UserManager:
    def __init__(self):
        self.users = {}

    def add_user(self, username, password):
        salt = get_random_bytes(16)
        hashed_password = self._hash_password(password, salt)
        self.users[username] = {
            'salt': base64.b64encode(salt).decode(),
            'password': hashed_password
        }

    def authenticate_user(self, username, password):
        if username not in self.users:
            return False
        salt_b64 = self.users[username]['salt']
        salt = base64.b64decode(salt_b64)
        hashed_password = self._hash_password(password, salt)
        return hashed_password == self.users[username]['password']

    def _hash_password(self, password, salt):
        hash_object = hashlib.sha256()
        hash_object.update(salt + password.encode())
        return hash_object.hexdigest()

class ConfidentialityManager:
    def __init__(self):
        self.ntru_system = NTRUEncryptSystem()
        self.public_key, self.private_key = self.ntru_system.generate_keys()
        self.symmetric_key = get_random_bytes(16)

    def hash_data(self, data):
        hash_object = SHA256.new()
        hash_object.update(data.encode())
        return hash_object.hexdigest()

    def symmetric_encrypt(self, data):
        iv = get_random_bytes(16)
        cipher = AES.new(self.symmetric_key, AES.MODE_CFB, iv)
        encrypted_data = cipher.encrypt(data.encode())
        return base64.b64encode(iv + encrypted_data).decode()

    def symmetric_decrypt(self, encrypted_data):
        raw_data = base64.b64decode(encrypted_data)
        iv = raw_data[:16]
        encrypted_message = raw_data[16:]
        cipher = AES.new(self.symmetric_key, AES.MODE_CFB, iv)
        return cipher.decrypt(encrypted_message).decode()
```

```
    def encrypt_message(self, message):
        hashed_message = self.hash_data(message)
        encrypted_symmetric = self.symmetric_encrypt(message)
        encrypted_symmetric_key =
self.ntru_system.encrypt(self.symmetric_key.hex(), self.public_key)
        return {
            'encrypted_key': base64.b64encode(encrypted_symmetric_key).decode(),
            'encrypted_data': encrypted_symmetric,
            'hash': hashed_message
        }

    def decrypt_message(self, encrypted_package):
        encrypted_key_b64 = encrypted_package['encrypted_key']
        encrypted_key_bytes = base64.b64decode(encrypted_key_b64)
        decrypted_symmetric_key_hex =
self.ntru_system.decrypt(encrypted_key_bytes, self.private_key)
        symmetric_key_bytes = bytes.fromhex(decrypted_symmetric_key_hex)
        self.symmetric_key = symmetric_key_bytes
        decrypted_message =
self.symmetric_decrypt(encrypted_package['encrypted_data'])
        return {
            'decrypted_data': decrypted_message,
            'hash': encrypted_package['hash']
        }

class NTRUEncryptSystem:
    def __init__(self, N=167, p=3, q=128):
        self.N = N
        self.p = p
        self.q = q
        self.public_key = None
        self.private_key = None
        self.ntru = Ntru(N, p, q)

    def generate_keys(self):
        self.public_key, self.private_key = self.ntru.generate_keypair()
        return self.public_key, self.private_key

    def encrypt(self, message, public_key):
        encrypted_message = self.ntru.encrypt(message.encode(), public_key)
        return encrypted_message

    def decrypt(self, encrypted_message, private_key):
        decrypted_message = self.ntru.decrypt(encrypted_message, private_key)
        return decrypted_message.decode()

def main():
    system = CyberSecuritySystemExtended()
    system.rbac.add_role('admin')
    system.rbac.assign_permission('admin', 'send_message')
    system.rbac.assign_permission('admin', 'receive_message')
    system.rbac.assign_permission('admin', 'encrypt_file')
    system.rbac.assign_permission('admin', 'decrypt_file')
    system.register_user('alice', 'password123', 'admin')
    otp = system.login_user('alice', 'password123')
    if otp:
        user_otp = otp
        if system.verify_otp('alice', user_otp):
            encrypted_package = system.send_secure_message('alice', 'Secret
message for admin user.')
            print(json.dumps(encrypted_package, indent=4))
            decrypted = system.receive_secure_message('alice',
encrypted_package)
            print(decrypted)
            file_path = 'sample.txt'
            output_path = 'sample_encrypted.bin'
            key_file_path = 'sample_encrypted.bin.key'
            decrypted_output_path = 'sample_decrypted.txt'
```

```
            with open(file_path, 'w') as f:
                f.write('This is a confidential file.')
            if system.encrypt_file_for_user('alice', file_path, output_path):
                if      system.decrypt_file_for_user('alice',       output_path,
key_file_path, decrypted_output_path):
                    with open(decrypted_output_path, 'r') as df:
                        print(df.read())

if __name__ == '__main__':
    main()
```

Файл ReplayAttackPrevention.py

```
import sqlite3
import os
import datetime
import json
import base64
import secrets
import random
import time

class ReplayAttackPrevention:
    def __init__(self, expiry_seconds=300):
        self.processed_messages = {}
        self.expiry_seconds = expiry_seconds
    def record_message(self, message_id):
        now = datetime.datetime.now()
        self.processed_messages[message_id] = now
    def is_replay(self, message_id):
        now = datetime.datetime.now()
        if message_id in self.processed_messages:
            recorded_time = self.processed_messages[message_id]
            if (now - recorded_time).total_seconds() < self.expiry_seconds:
                return True
        return False
    def cleanup(self):
        now = datetime.datetime.now()
        to_delete = []
        for message_id, timestamp in self.processed_messages.items():
            if (now - timestamp).total_seconds() >= self.expiry_seconds:
                to_delete.append(message_id)
        for message_id in to_delete:
            del self.processed_messages[message_id]

class DatabaseIntegration:
    def __init__(self, db_file='encrypted_messages.db'):
        self.db_file = db_file
        self.connection = sqlite3.connect(self.db_file)
        self.create_table()
    def create_table(self):
        cursor = self.connection.cursor()
        cursor.execute('''CREATE TABLE IF NOT EXISTS messages (
                            id INTEGER PRIMARY KEY AUTOINCREMENT,
                            username TEXT NOT NULL,
                            encrypted_message TEXT NOT NULL,
                            timestamp TEXT NOT NULL
                        )''')
        self.connection.commit()
    def store_message(self, username, encrypted_message):
        timestamp = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
        cursor = self.connection.cursor()
        cursor.execute('INSERT INTO messages (username, encrypted_message,
timestamp) VALUES (?, ?, ?)', (username, encrypted_message, timestamp))
        self.connection.commit()
    def get_message_history(self, username):
        cursor = self.connection.cursor()
        cursor.execute('SELECT id, encrypted_message, timestamp FROM messages
WHERE username = ?', (username,))
        return cursor.fetchall()
    def close(self):
        self.connection.close()

class AuditManager:
    def __init__(self, audit_file='audit_log.txt'):
        self.audit_file = audit_file
        if not os.path.exists(self.audit_file):
            with open(self.audit_file, 'w') as f:
                f.write('Audit Log Start\n')
    def record_access(self, username, action, resource):
```

```
        timestamp = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
        log_entry = json.dumps({'username': username, 'action': action,
'resource': resource, 'timestamp': timestamp})
        with open(self.audit_file, 'a') as f:
            f.write(log_entry + '\n')

class SuspiciousActivityNotifier:
    def __init__(self, notification_file='suspicious_activity.log'):
        self.notification_file = notification_file
        if not os.path.exists(self.notification_file):
            with open(self.notification_file, 'w') as f:
                f.write('Suspicious Activity Log Start\n')
    def notify_activity(self, details):
        timestamp = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
        notification = json.dumps({'details': details, 'timestamp': timestamp})
        with open(self.notification_file, 'a') as f:
            f.write(notification + '\n')

class CryptoOperationReporter:
    def __init__(self, report_file='crypto_report.txt'):
        self.report_file = report_file
        self.operations = []
    def log_operation(self, username, operation, details):
        timestamp = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
        self.operations.append({'username': username, 'operation': operation,
'details': details, 'timestamp': timestamp})
    def generate_report(self):
        with open(self.report_file, 'w') as f:
            f.write('Crypto Operation Report\n')
            for op in self.operations:
                f.write(json.dumps(op) + '\n')
        self.operations = []

def main():
    rap = ReplayAttackPrevention()
    db = DatabaseIntegration()
    audit = AuditManager()
    notifier = SuspiciousActivityNotifier()
    reporter = CryptoOperationReporter()
    message_id = 'msg-' + str(random.randint(1000, 9999))
    if rap.is_replay(message_id):
        notifier.notify_activity('Replay attack detected for message id ' +
message_id)
    else:
        rap.record_message(message_id)
        db.store_message('bob', 'EncryptedContentExample')
        audit.record_access('bob', 'store_message', 'encrypted_messages')
        reporter.log_operation('bob', 'encryption', 'Stored encrypted message')
    rap.cleanup()
    history = db.get_message_history('bob')
    print('Message history for bob:')
    for record in history:
        print(record)
    reporter.generate_report()
    db.close()

if __name__ == '__main__':
    main()
```

Файл SessionKeyManager.py

```
import os
import json
import base64
import hashlib
import secrets
import datetime
import sqlite3
from Crypto.Cipher import AES
from Crypto.PublicKey import RSA
from Crypto.Signature import pkcs1_15
from Crypto.Hash import SHA256

class SessionKeyManager:
    def __init__(self, storage_file='session_keys.json',
master_key=b'MySuperSecretMasterKey123'):
        self.storage_file = storage_file
        self.master_key = hashlib.sha256(master_key).digest()
        self.session_keys = {}
        self.load_keys()
    def encrypt_data(self, data):
        iv = secrets.token_bytes(16)
        cipher = AES.new(self.master_key, AES.MODE_CFB, iv)
        encrypted = cipher.encrypt(data.encode())
        return base64.b64encode(iv + encrypted).decode()
    def decrypt_data(self, encrypted_data):
        raw = base64.b64decode(encrypted_data)
        iv = raw[:16]
        ciphertext = raw[16:]
        cipher = AES.new(self.master_key, AES.MODE_CFB, iv)
        decrypted = cipher.decrypt(ciphertext)
        return decrypted.decode()
    def save_keys(self):
        data = json.dumps(self.session_keys)
        encrypted_data = self.encrypt_data(data)
        with open(self.storage_file, 'w') as f:
            f.write(encrypted_data)
    def load_keys(self):
        if os.path.exists(self.storage_file):
            with open(self.storage_file, 'r') as f:
                encrypted_data = f.read()
            try:
                data = self.decrypt_data(encrypted_data)
                self.session_keys = json.loads(data)
            except:
                self.session_keys = {}
        else:
            self.session_keys = {}
    def add_session_key(self, session_id, key):
        self.session_keys[session_id] = key.hex()
        self.save_keys()
    def get_session_key(self, session_id):
        if session_id in self.session_keys:
            return bytes.fromhex(self.session_keys[session_id])
        return None

class HashChangeDetector:
    def __init__(self):
        self.hash_store = {}
    def register_message(self, message_id, message):
        message_hash = hashlib.sha256(message.encode()).hexdigest()
        self.hash_store[message_id] = message_hash
        return message_hash
    def detect_change(self, message_id, new_message):
        new_hash = hashlib.sha256(new_message.encode()).hexdigest()
        if message_id in self.hash_store:
            if self.hash_store[message_id] != new_hash:
                self.hash_store[message_id] = new_hash
```

```
                return True
            else:
                return False
        else:
            self.hash_store[message_id] = new_hash
            return False
    def get_hash(self, message_id):
        return self.hash_store.get(message_id, None)

class CertificateManager:
    def __init__(self, cert_storage='certificates.json'):
        self.cert_storage = cert_storage
        self.certificates = {}
        self.load_certificates()
    def generate_certificate(self, username):
        key = RSA.generate(2048)
        private_key = key.export_key().decode()
        public_key = key.publickey().export_key().decode()
        certificate = {'username': username, 'public_key': public_key, 'issued':
datetime.datetime.now().isoformat()}
        self.certificates[username]      =      {'certificate':      certificate,
'private_key': private_key}
        self.save_certificates()
        return certificate
    def sign_message(self, username, message):
        if username not in self.certificates:
            return None
        private_key = RSA.import_key(self.certificates[username]['private_key'])
        h = SHA256.new(message.encode())
        signature = pkcs1_15.new(private_key).sign(h)
        return base64.b64encode(signature).decode()
    def verify_signature(self, username, message, signature):
        if username not in self.certificates:
            return False
        public_key                                                              =
RSA.import_key(self.certificates[username]['certificate']['public_key'])
        h = SHA256.new(message.encode())
        try:
            pkcs1_15.new(public_key).verify(h, base64.b64decode(signature))
            return True
        except:
            return False
    def save_certificates(self):
        with open(self.cert_storage, 'w') as f:
            json.dump(self.certificates, f)
    def load_certificates(self):
        if os.path.exists(self.cert_storage):
            with open(self.cert_storage, 'r') as f:
                try:
                    self.certificates = json.load(f)
                except:
                    self.certificates = {}
        else:
            self.certificates = {}

class BackupManager:
    def __init__(self, backup_dir='backups'):
        self.backup_dir = backup_dir
        if not os.path.exists(self.backup_dir):
            os.makedirs(self.backup_dir)
    def backup_key_data(self, key_data, backup_name=None):
        if backup_name is None:
            backup_name                =                'backup_'                +
datetime.datetime.now().strftime('%Y%m%d%H%M%S') + '.json'
        backup_path = os.path.join(self.backup_dir, backup_name)
        with open(backup_path, 'w') as f:
            json.dump(key_data, f)
        return backup_path
    def restore_key_data(self, backup_name):
```

```
        backup_path = os.path.join(self.backup_dir, backup_name)
        if os.path.exists(backup_path):
            with open(backup_path, 'r') as f:
                return json.load(f)
        return None

class DiffieHellmanKeyExchange:
    def __init__(self, p=0, g=0):
        if p == 0:
            self.p = 0xFFFFFFFFFFFFFFFFC90FDAA22168C234C4C6628B80DC1CD1
        else:
            self.p = p
        if g == 0:
            self.g = 2
        else:
            self.g = g
        self.private_key = secrets.randbelow(self.p - 2) + 2
        self.public_key = pow(self.g, self.private_key, self.p)
    def generate_shared_key(self, other_public_key):
        shared_secret = pow(other_public_key, self.private_key, self.p)
        shared_key = hashlib.sha256(str(shared_secret).encode()).digest()
        return shared_key

def main():
    skm = SessionKeyManager()
    session_id = 'session_' + str(secrets.randbelow(10000))
    session_key = secrets.token_bytes(16)
    skm.add_session_key(session_id, session_key)
    loaded_key = skm.get_session_key(session_id)
    hcd = HashChangeDetector()
    msg_id = 'msg_' + str(secrets.randbelow(10000))
    original_hash = hcd.register_message(msg_id, 'Original secure message.')
    change_detected = hcd.detect_change(msg_id, 'Modified secure message.')
    cm = CertificateManager()
    cert = cm.generate_certificate('charlie')
    signature = cm.sign_message('charlie', 'Important signed message.')
    verification = cm.verify_signature('charlie', 'Important signed message.',
signature)
    bm = BackupManager()
    backup_path = bm.backup_key_data({'session_id': session_id, 'session_key':
session_key.hex()})
    restored_data = bm.restore_key_data(os.path.basename(backup_path))
    dh1 = DiffieHellmanKeyExchange()
    dh2 = DiffieHellmanKeyExchange(p=dh1.p, g=dh1.g)
    shared_key1 = dh1.generate_shared_key(dh2.public_key)
    shared_key2 = dh2.generate_shared_key(dh1.public_key)
    print('Session ID:', session_id)
    print('Loaded Session Key:', loaded_key)
    print('Original Hash:', original_hash)
    print('Change Detected:', change_detected)
    print('Certificate for charlie:', cert)
    print('Signature Verification:', verification)
    print('Backup Path:', backup_path)
    print('Restored Data:', restored_data)
    print('Diffie-Hellman Shared Key 1:', shared_key1.hex())
    print('Diffie-Hellman Shared Key 2:', shared_key2.hex())

if __name__ == '__main__':
    main()
```

Файл APIIntegration.py

```
import json
import time
import datetime
import re
import threading
from flask import Flask, jsonify, request
import os
import sqlite3
import random
import secrets
import hashlib
from Crypto.Cipher import AES
from Crypto.PublicKey import RSA
from Crypto.Signature import pkcs1_15
from Crypto.Hash import SHA256

class APIIntegration:
    def __init__(self, host='127.0.0.1', port=5000):
        self.app = Flask(__name__)
        self.host = host
        self.port = port
        self.setup_routes()
    def setup_routes(self):
        @self.app.route('/status', methods=['GET'])
        def status():
            return       jsonify({'status':       'running',       'timestamp':
datetime.datetime.now().isoformat()})
        @self.app.route('/logs', methods=['GET'])
        def logs():
            if os.path.exists('system_log.txt'):
                with open('system_log.txt', 'r') as f:
                    data = f.read()
                return jsonify({'logs': data})
            return jsonify({'logs': 'No logs found'})
        @self.app.route('/config', methods=['GET'])
        def config():
            cm = ConfigurationManager()
            return jsonify(cm.get_config())
    def run_server(self):
        self.app.run(host=self.host, port=self.port)

class PerformanceMonitor:
    def __init__(self):
        self.metrics = {}
    def record_metric(self, name, value):
        if name not in self.metrics:
            self.metrics[name] = []
        self.metrics[name].append({'value':         value,         'timestamp':
datetime.datetime.now().isoformat()})
    def generate_report(self):
        return json.dumps(self.metrics, indent=4)

class ConfigurationManager:
    def __init__(self, config_file='config.json'):
        self.config_file = config_file
        self.config = {}
        self.load_config()
    def load_config(self):
        if os.path.exists(self.config_file):
            with open(self.config_file, 'r') as f:
                try:
                    self.config = json.load(f)
                except:
                    self.config = {}
        else:
            self.config = {'encryption_strength': 'high', 'max_login_attempts':
5, 'log_retention_days': 30}
```

```
            self.save_config()
    def save_config(self):
        with open(self.config_file, 'w') as f:
            json.dump(self.config, f, indent=4)
    def update_config(self, key, value):
        self.config[key] = value
        self.save_config()
    def get_config(self):
        return self.config

class IntrusionDetectionSystem:
    def __init__(self, threshold=10, timeframe=60):
        self.attempts = {}
        self.threshold = threshold
        self.timeframe = timeframe
    def record_attempt(self, identifier):
        now = time.time()
        if identifier not in self.attempts:
            self.attempts[identifier] = []
        self.attempts[identifier].append(now)
        self.cleanup(identifier)
    def cleanup(self, identifier):
        now = time.time()
        self.attempts[identifier] = [t for t in self.attempts[identifier] if now
- t <= self.timeframe]
    def is_intrusion(self, identifier):
        self.cleanup(identifier)
        return len(self.attempts.get(identifier, [])) >= self.threshold
    def get_intrusion_report(self):
        report = {}
        for identifier, times in self.attempts.items():
            if len(times) >= self.threshold:
                report[identifier] = len(times)
        return report

class DataAnonymizer:
    def __init__(self):
        self.email_pattern = re.compile(r'[\w\.-]+@[\w\.-]+')
        self.phone_pattern = re.compile(r'\+?\d[\d -]{8,}\d')
    def anonymize(self, text):
        text = self.email_pattern.sub('[EMAIL]', text)
        text = self.phone_pattern.sub('[PHONE]', text)
        return text

class ExtendedAPIIntegration:
    def __init__(self, host='127.0.0.1', port=6000):
        self.app = Flask(__name__)
        self.host = host
        self.port = port
        self.setup_routes()
    def setup_routes(self):
        @self.app.route('/metrics', methods=['GET'])
        def metrics():
            pm = PerformanceMonitor()
            return jsonify({'metrics': pm.generate_report()})
        @self.app.route('/intrusions', methods=['GET'])
        def intrusions():
            ids = IntrusionDetectionSystem()
            return jsonify({'intrusions': ids.get_intrusion_report()})
    def run_server(self):
        self.app.run(host=self.host, port=self.port)

class RSACertificateManager:
    def __init__(self, cert_storage='rsa_certificates.json'):
        self.cert_storage = cert_storage
        self.certificates = {}
        self.load_certificates()
    def generate_certificate(self, username):
        key = RSA.generate(2048)
```

```
        private_key = key.export_key().decode()
        public_key = key.publickey().export_key().decode()
        certificate = {'username': username, 'public_key': public_key, 'issued':
datetime.datetime.now().isoformat()}
        self.certificates[username]      =      {'certificate':      certificate,
'private_key': private_key}
        self.save_certificates()
        return certificate
    def sign_message(self, username, message):
        if username not in self.certificates:
            return None
        private_key = RSA.import_key(self.certificates[username]['private_key'])
        h = SHA256.new(message.encode())
        signature = pkcs1_15.new(private_key).sign(h)
        return base64.b64encode(signature).decode()
    def verify_signature(self, username, message, signature):
        if username not in self.certificates:
            return False
        public_key                                                              =
RSA.import_key(self.certificates[username]['certificate']['public_key'])
        h = SHA256.new(message.encode())
        try:
            pkcs1_15.new(public_key).verify(h, base64.b64decode(signature))
            return True
        except:
            return False
    def save_certificates(self):
        with open(self.cert_storage, 'w') as f:
            json.dump(self.certificates, f)
    def load_certificates(self):
        if os.path.exists(self.cert_storage):
            with open(self.cert_storage, 'r') as f:
                try:
                    self.certificates = json.load(f)
                except:
                    self.certificates = {}
        else:
            self.certificates = {}

class IncidentResponseSystem:
    def __init__(self, db_file='incidents.db'):
        self.db_file = db_file
        self.connection = sqlite3.connect(self.db_file)
        self.create_table()
    def create_table(self):
        cursor = self.connection.cursor()
        cursor.execute('''CREATE TABLE IF NOT EXISTS incidents (
                            id INTEGER PRIMARY KEY AUTOINCREMENT,
                            incident_type TEXT NOT NULL,
                            details TEXT NOT NULL,
                            timestamp TEXT NOT NULL
                          )''')
        self.connection.commit()
    def log_incident(self, incident_type, details):
        timestamp = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
        cursor = self.connection.cursor()
        cursor.execute('INSERT    INTO    incidents    (incident_type,    details,
timestamp) VALUES (?, ?, ?)', (incident_type, details, timestamp))
        self.connection.commit()
    def get_incidents(self):
        cursor = self.connection.cursor()
        cursor.execute('SELECT  id,  incident_type,  details,  timestamp  FROM
incidents')
        return cursor.fetchall()
    def close(self):
        self.connection.close()

def start_extended_api_server():
    api_ext = ExtendedAPIIntegration()
```

```
    api_ext.run_server()

def main():
    cm = ConfigurationManager()
    cm.update_config('api_mode', 'extended')
    pm = PerformanceMonitor()
    pm.record_metric('response_time', random.uniform(0.05, 0.2))
    pm.record_metric('throughput', random.randint(100, 500))
    print("Performance Monitor Report:")
    print(pm.generate_report())
    ids = IntrusionDetectionSystem(threshold=3, timeframe=30)
    for i in range(4):
        ids.record_attempt('10.0.0.1')
    print("Intrusion Detection Report:")
    print(ids.get_intrusion_report())
    da = DataAnonymizer()
    sample = "Contact jane.doe@example.com or call +19876543210 for details."
    print("Anonymized Data:")
    print(da.anonymize(sample))
    rsa_cm = RSACertificateManager()
    cert = rsa_cm.generate_certificate('david')
    signature = rsa_cm.sign_message('david', 'Secure transaction data.')
    verification = rsa_cm.verify_signature('david', 'Secure transaction data.',
signature)
    print("RSA Certificate for david:")
    print(cert)
    print("Signature Verified:", verification)
    irs = IncidentResponseSystem()
    irs.log_incident('Unauthorized Access', 'Multiple failed login attempts
detected from IP 10.0.0.1')
    incidents = irs.get_incidents()
    print("Incident Response Report:")
    for inc in incidents:
        print(inc)
    threading.Thread(target=start_extended_api_server, daemon=True).start()
    time.sleep(2)
    print("Extended API Server started at http://127.0.0.1:6000/")

if __name__ == '__main__':
    main()
```