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 захисту конфіденційності інформації з використанням інструкцій Intel Core i7”

Виконав здобувач вищої освіти
IV курсу, групи КІ-21-1
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Курченко В.В.
«____» _____ 2025 р.

Керівник проекту
доктор філософії (PhD)
_____ Усік П.С.
«____» _____ 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 *123 “*Комп’ютерна інженерія*”*

Освітньо-професійна (освітньо-наукова) програма _*“*Комп’ютерна інженерія*”*_

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

Курченку Віталію Валерійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи захисту конфіденційності інформації з використанням інструкцій Intel Core i7*

2. Керівник роботи *Усік Павло Сергійович, доктор філософії (PhD)*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 46-02 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 захисту конфіденційності інформації з використанням інструкцій Intel Core i7*

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 в промислову експлуатацію.
6. Висновки

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

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*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
_____ Усік П.С.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Курченко В.В.
(прізвище та ініціали)

АНОТАЦІЯ

Курченко В.В. Програмне забезпечення системи захисту конфіденційності інформації з використанням інструкцій Intel Core i7. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи захисту конфіденційності інформації з використанням інструкцій Intel Core i7.

Метою розробки є програмне забезпечення системи захисту конфіденційності інформації з використанням інструкцій Intel Core i7.

Результат роботи – програмна реалізація системи захисту конфіденційності інформації з використанням інструкцій Intel Core i7.

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: комп'ютерна інженерія, Intel Core i7

ABSTRACT

Kurchenko V.V. Software for the information confidentiality protection system using Intel Core i7 instructions. 123 Computer Engineering. 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 the information confidentiality protection system using Intel Core i7 instructions.

The purpose of the development is software for the information confidentiality protection system using Intel Core i7 instructions.

The result of the work is a software implementation of the information confidentiality protection system using Intel Core i7 instructions.

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 Python.

Keywords: computer engineering, Intel Core i7

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

КСЗ	–	комплексна система захисту
ПЕОМ	–	персональна електронно-обчислювальна машина
СУБД	–	система управління базами даних
PGP	–	Pretty Good Privacy

ВСТУП

Актуальність теми. Відповідно до досліджень ІТ-фахівців, вартість інформації, що зберігається на середньостатистичному користувальницькому комп'ютері, починає перевищувати вартість самого комп'ютера вже протягом перших двох місяців використання. Звичайно, не зовсім коректно говорити про середньостатистичний комп'ютер, адже не кожний користувач зберігає на своїй машині дані, що представляють цінність. Проте, для тих користувачів, які входять у цю категорію, гостро стоїть питання захисту й шифрування даних. Комерційні угоди, нові розробки, фінансова інформація, особисті дані – все це може становити інтерес для зловмисників.

Якщо для більшості користувачів цілком вистачає стандартних мір безпеки – використання антивірусів і хоча б більш-менш надійних паролів, то власники важливих даних все частіше вдаються до допомоги спеціальних програм, що забезпечують їхній захист. Вибір такого програмного забезпечення досить широкий і включає як безкоштовні вільно розповсюджувані програми, так і досить дорогі продукти, що коштують кілька сотень доларів за копію.

Розрізняються програми для шифрування даних і своїми функціями. Найпростіші продукти звичайно вміють тільки приховувати й зашифровувати окремі файли й папки – завдання саме популярне, але не єдине. До речі, можливість захистити паролем текстовий файл уже убудований в Microsoft Word і особою цінності не представляє. Для всебічного захисту важливих даних потрібні також функції захисту електронної пошти, диска (у тому числі й зовнішніх пристроях), хмарних сервісів, які ви використовуєте.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи захисту конфіденційності інформації з використанням інструкцій Intel Core i7.

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

– Огляд існуючих систем захисту конфіденційності інформації з використанням інструкцій Intel Core i7.

– Дослідження системи захисту конфіденційності інформації з використанням інструкцій Intel Core i7.

– Програмна реалізація системи захисту конфіденційності інформації з використанням інструкцій Intel Core i7.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі захисту конфіденційності інформації з використанням інструкцій Intel Core i7.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи захисту конфіденційності інформації з використанням інструкцій Intel Core i7, 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

Система призначена для захисту конфіденційності інформації з використанням інструкцій Intel Core i7. Інформація, безсумнівно, є одним з найцінніших і в той же час уразливих активів будь-якої компанії. Чим менша кількість людей має до неї доступ, тим більшою цінністю володіє інформація. У спробі забезпечити належний захист даних від яких-небудь погроз компаніям варто вживати всі необхідні міри, які дозволять забезпечити їхня цілісність і таємність. Одна із ключових цілей у цьому зв'язку – запобігання несанкціонованого доступу й незаконного розголошення інформації нинішніми або колишніми співробітниками.

Серед інформації, використовуваної компаніями у своїй діяльності, захисту підлягає тільки інформація з обмеженим доступом. Як правило, така інформація ставиться або до конфіденційної інформації, або до комерційної таємниці.

Відповідно до закону, конфіденційною є інформація про фізичну особу (персональні дані), а також інформація, доступ до якої обмежений фізичною або юридичною особою і яка може поширюватися за бажанням (згоді) відповідної особи в визначеному нею порядку й на встановлених нею умовах. Українське законодавство також визначає перелік даних, які не можуть розглядатися як інформація з обмеженим доступом, наприклад, відомості про стан навколишнього середовища, якості продуктів живлення й побутових товарів, аваріях і обставинах непереборної сили, охороні здоров'я й рівні життя, а також про інші питання, що представляють суспільний інтерес.

Що стосується комерційної таємниці, відповідно до законодавства України, така інформація не повинна бути відомою й легкодоступною, але в той

же час повинна мати комерційну цінність і бути предметом адекватних існуючим обставинам мір по збереженню її таємності, початих особою, що законно володіє такою інформацією. Постановою Кабінету Міністрів "Про перелік відомостей, не складових комерційної таємниці" ("Постанова") установлений перелік інформації, що не може становити комерційну таємницю. Такі відомості, серед іншого, включають:

− інформацію, що стосується звітності, і дані, необхідні для перевірки вирахування й сплати податків і інших обов'язкових платежів;

− відомості про чисельність і состав працівників, їхній заробітній платі, а також наявності вакансій;

− документи про платоспроможність.

Даний перелік доцільно брати до уваги при визначенні інформації, що компанія збирається віднести до комерційної таємниці. Додатковим аргументом на користь цього може також виступати відсутність якого-небудь офіційного тлумачення, а також стабільні й послідовної судової практики по даному питанню.

1.2 Область застосування

Областю застосування є системи конфіденційної інформації. У цей час завдання захисту конфіденційних даних стає усе більше актуальною.

Запобігання витоків (Data Leak Prevention, DLP) – технології запобігання витоків конфіденційної інформації з інформаційної системи зовні, а також технічні пристрої (програмні або програмно-апаратні) для такого запобігання витоків. DLP-системи будуються на аналізі потоків даних, що перетинають периметр інформаційної системи, яка захищається. При детектуванні в цьому потоці конфіденційної інформації спрацьовує активний компонент системи, і передача повідомлення (пакета, потоку, сесії) блокується.

Важливим доповненням до визначення є також і те, що DLP-система повинна охоплювати всі основні канали витоку конфіденційної інформації. Саме такої позиції дотримується сьогодні більшість експертів у цій області. Крім того, DLP-система повинна бути чутливою стосовно змісту, що перевіряється (контенту) і забезпечувати автоматизований механізм відстеження порушень заданих правил, тобто без залучення істотного числа співробітників-контролерів.

Системи захисту від витоків даних – автоматизований засіб, що дозволяє розпізнавати й/або блокувати переміщення конфіденційних даних за межі інформаційної системи, яка захищається, по всіх каналах, використовуваним у повсякденній роботі.

Основне завдання технічної системи захисту від витоків:

– одержати опис конфіденційних даних;

– після чого вміти розпізнавати їх у потоці, що виходить із внутрішнього інформаційного поля організації зовні;

– реагувати на виявлені спроби. Саме цей функціонал становить ядро будь-якого DLP-рішення.

Можна виділити 3 основних сценарії, що приводять до виведення інформації за межі інформаційного середовища компанії:

− мережний;

− локальний;

− у зв'язку із втратою носія.

Мережний сценарій припускає відправлення інформації за «периметр» контрольованого інформаційного поля засобами електронної пошти, через системи передачі миттєвих повідомлень (ICQ, MSN, AOL), за допомогою веб-пошти (ukr.net, gmail.com), через використання ftp-з'єднання, шляхом друку документа на мережному принтері. Для виявлення конфіденційної інформації, переданої мережними засобами, потрібні механізми перехоплення поштового й інтернет-трафіка, а також контроль за мережними принт-серверами.

Локальний шлях виводу інформації включає застосування зовнішніх USB-накопичувачів і знімних жорстких дисків, запис на CD/DVD і локальний друк.

Очевидно, єдиним способом відстеження такого роду дій є установка на комп'ютері користувача програми-агента, здатної відслідковувати потенційно небезпечні дії й реагувати на них відповідно до централізовано керованих політиків.

Незаконне заволодіння носієм (переносним комп'ютером, смартфоном) у реальності є найпоширенішим випадком, коли конфіденційна інформація стає доступної третім особам. Ноутбуки губляться й викрадаються – майже кожна компанія зіштовхується із цим ризиком, і звести його до нуля неможливо. Практично єдиний діючий спосіб боротьби в цьому випадку – шифрування всього диска або окремих файлів.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи захисту конфіденційності інформації з використанням інструкцій Intel Core i7, 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

Tor Browser

Браузер для анонімного web-серфінгу. Являє собою web-оглядач Firefox з інтегрованим розширенням Torbutton, блокувальником скриптів і адд-оном для входу на сайти за протоколом HTTPS.

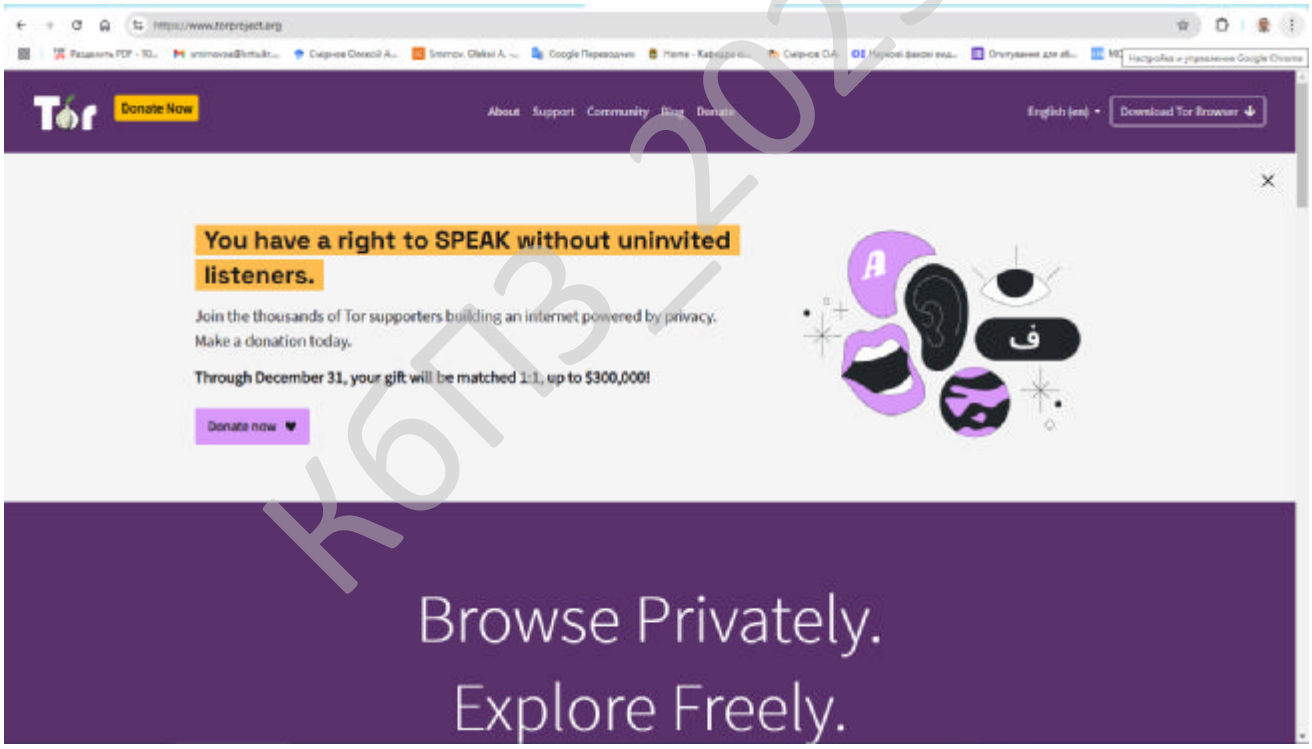

Рисунок 2.1 – Інтерфейс користувача Tor Browser

Програма дозволяє відвідувати заборонені адміністратором або державою сайти, а також вести переписку в web-чатах, робити замовлення й завантажувати файли, приховуючи свою IP-адресу й інші елементи ідентифікації.

Hotspot Shield

Програма для створення приватної віртуальної мережі (VPN) і захисту від вірусів і хакерських атак під час підключення до Wi-Fi. Буде особливо корисна користувачам, які часто заходять в Інтернет з ноутбука через крапки суспільного доступу.

Рисунок 2.2 – Інтерфейс користувача Hotspot Shield

Hotspot Shield для Windows – це поширювана програма, яку напевно оцінять ті, хто часто працює з ноутбуком у кафі, готелях або торгових центрах. Підключаючись до Wi-Fi ви, швидше за все, не особливо замислюєтеся про те, що в такий спосіб хакерам або добре написаним вірусам одержати доступ до ваших даних набагато простіше.

TrueCrypt

Безкоштовне рішення шифрування дисків. TrueCrypt містить всі необхідні функції й набір інструментів для безпечного шифрування важливої інформації.

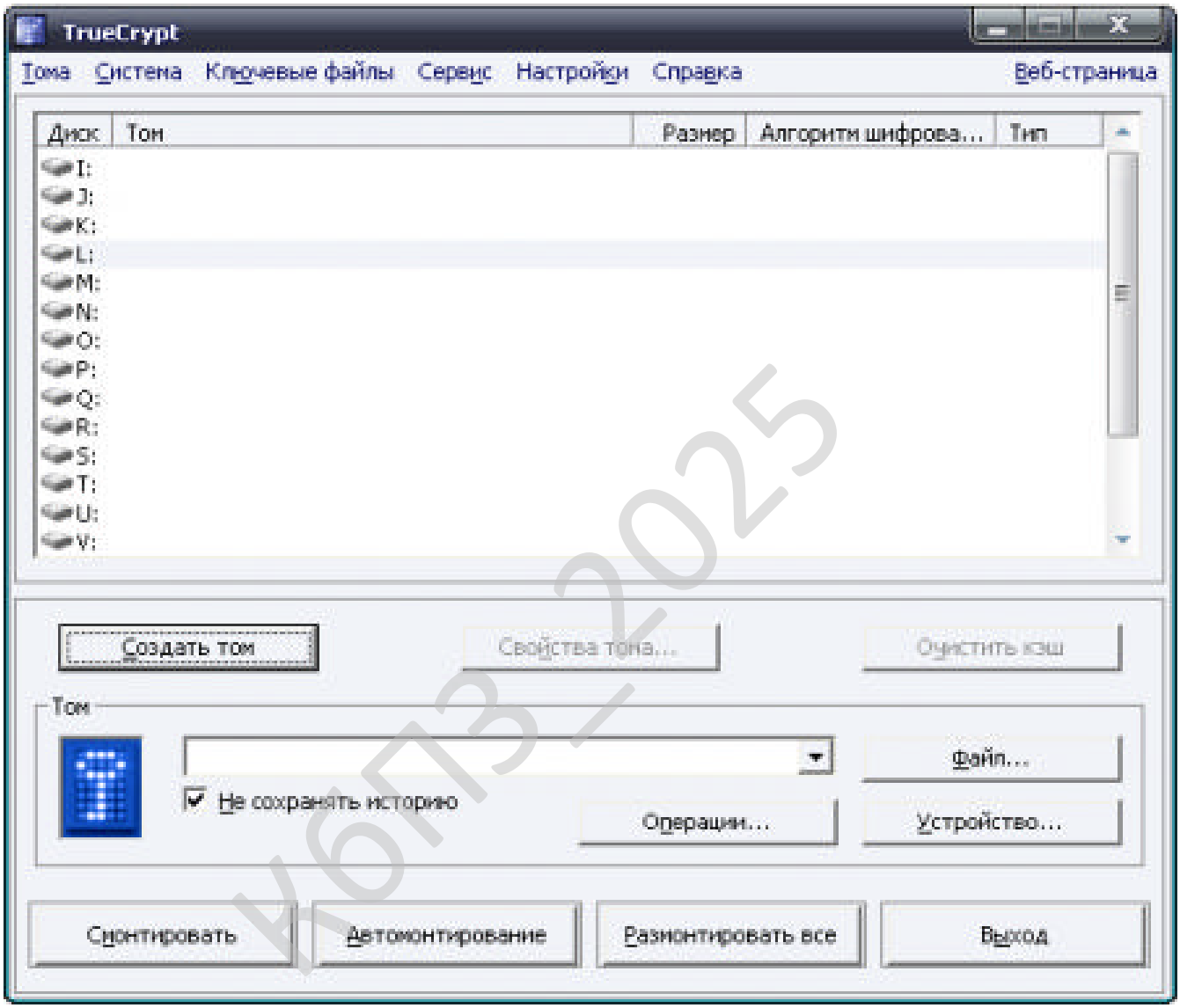

Рисунок 2.3 – Інтерфейс користувача TrueCrypt

Деякі особливості:

– Створює віртуальний зашифрований диск у файл і монтує його як реальний диск.

– Шифрування всього жорсткого диска або пристрою зберігання, такі, як USB флеш-диск.

– Шифрування здійснюється автоматично, у реальному масштабі часу (на лету).

– Алгоритми шифрування: AES-256 і Twofish.

Системи криптографічного захисту інформації (СКЗІ)

Виробники СКЗІ пропонуть різні механізми для інтеграції криптозасобів в інформаційні системи. Існують рішення, орієнтовані на підтримку систем з Web-інтерфейсом, мобільних і десктопних додатків, серверних компонентів. СКЗІ інтегруються в додатки Microsoft і в продукти Open Source, забезпечують підтримку різних прикладних протоколів і форматів електронного підпису.

З урахуванням зростаючої кількості проектів із застосуванням ЕЦП і появи масових проектів для фізичних осіб, розроблювачам подібних проектів потрібно добре орієнтуватися в пропонованих виробниками рішеннях по ЕЦП для того, щоб зробити систему зручною в експлуатації й недорогою у плані техпідтримки. Таким чином, якщо ще років 5 назад головним фактором вибору криптозасобу була його повна відповідність вимогам регуляторів, то при сьогоднішній розмаїтості важливими критеріями можуть виступати охват підтримуваних платформ, можливість інтеграції із браузером, підтримка мобільних користувачів, можливість установки без прав системного адміністратора й т.п.

Класифікація побудована на основі:

– технологій інтеграції (Crypto API, Active-X, NPAPI і ін.), які підтримують СКЗІ для вбудовування в додатки й прикладні системи;

– інтерфейсів, які надають СКЗІ для вбудовування в додатки й прикладні системи.

Крім того, показані способи інтеграції СКЗІ з Web-додатками й можливість його використання на мобільних платформах

Криптопровайдери

Де-факто стандартом галузі є клас криптозасобів, відомих як криптопровайдери. Криптопровайдер – бібліотека, яка надає спеціальний API і спеціальним образом зареєстрована в ОС, що дозволяє розширити список підтримуваних в ОС криптоалгоритмів.

Слід зазначити, що недосконаліть пропонованих MS Windows механізмів розширення змушує розроблювачів криптопровайдерів додатково модифікувати високорівневі криптобібліотеки й додатка MS Windows у процесі їхнього виконання для того, щоб «навчити» їх використовувати українські криптоалгоритми.

Варто розуміти, що не всі СКЗІ одного виду реалізують повний обсяг функціональності, наведений у таблицях. Для уточнення можливостей криптозасобів випливають звернутися до виробника.

Проблеми:

– Відсутність нормальної кроссплатформеності.

– Установка із правами адміністратора, настроювання.

– Установка відновлення Windows може зажадати відновлення провайдеру.

– Необхідність вбудовування в додатки за допомогою модифікації коду «на лету».

– CSP – нерідний інтерфейс для не-Windows-додатків.

Плюси:

– Широкий охват Windows-додатків.

– Багатий інструментарій для розроблювачів захищених систем.

– Апробована на великій кількості проектів технологія.

Нативні бібліотеки

OpenSSL-style

Open Source бібліотека OpenSSL має широкі криптографічні можливості й зручний механізм її розширення іншими криптоалгоритмами. OpenSSL є основним криптоядром для широкого спектра додатків Open Source.

Після того, як у цю бібліотеку компанією Криптоком були додані ДСТ, з'явилися патчи для «ДСТфікації» багатьох популярні додатки, що використовують OpenSSL. На базі OpenSSL деякі вендори розробили й сертифікували СКЗІ, крім того в ряд продуктів OpenSSL входить «неявним» образом.

Проблеми:

– OpenSSL і його аналоги не підтримується Windows-додатками.

– Необхідність патчити СПЗ, що підтримує OpenSSL, для включення ДСТ.

Плюси:

– Кроссплатформеність.

– Використання у величезній кількості проектів, відкриті вихідні коди більшої частини проекту – виявлення й усунення уразливостей (як приклад, недавнє виявлення heartbleed).

– Поширюється копіюванням – можна робити додатки, що не вимагають інсталяції.

– Широкий охват додатків СПЗ, на базі яких можна робити захищені сертифіковані продукти.

– Широка інтеграція у фреймворки, але при цьому проблеми із ДСТ.

PKCS#11

Бібліотека PKCS#11 надає універсальний кроссплатформенний програмний інтерфейс до USB-токенів і смарт-карт.

Функції діляться на:

– Функції доступу до пристрою.

– Функції запису/читання довільних даних.

– Функції роботи із ключами (пошук, створення, видалення, імпорт, експорт).

– Функції роботи із сертифікатами (пошук, імпорт, експорт).

– Функції ЕЦП.

– Функції хешування.

– Функції шифрування.

– Функції обчислення імітовставки.

– Функції вироблення ключа узгодження.

– Функції експорту/імпорту сесійного ключа.

Таким чином, стандарт PKCS#11 підтримує повний набір криптопримітивів, придатний для реалізації криптографічних форматів (PKCS#7/CMS/CADES, PKCS#10, X.509 і ін.) і протоколів (TLS, IPSEC, openVPN і ін.).

Для забезпечення швидкодії частина криптопримітивів може бути реалізована програмно.

У стандарті PKCS#11, починаючи з версії 2.30, підтримуються ДСТ Р 34.10-2001, ДСТ Р 34.11-94, ДСТ 28147-89.

Використання бібліотеки PKCS#11 забезпечує сумісність ПЗ різних вендорів при роботі з токенами. Через PKCS#11 інтерфейс уміють працювати додатка, написані на на базі Crypto API, NSS, OpenSSL.

Приклад сумісності додатків наведений на схемі. Таким чином, можливе використання підходящого додатка у відповідному місці інфосистеми.

PKCS#11 бувають також без підтримки апаратних пристроїв із програмною реалізацією криптоалгоритмів і зберіганням об'єктів у файловій системі.

Приклади – PKCS#11 інтегрований в NSS (Mozilla), проект aToken, бібліотека Агава-про.

У компанії Крипто-про є бібліотека PKCS#11, реалізована на базі MS Crypto API.

Існують PKCS# 11-бібліотеки для мобільних платоформ. Прикладом подібної бібліотеки служить бібліотека для Рутокен ЕЦП Bluetooth, що дозволяє використовувати пристрій на iOS і Android.

NSS

NSS являє собою криптографічну бібліотеку від співтовариства Mozilla. NSS використовується такими додатками, як браузер Mozilla Firefox, поштовим клієнтом Mozilla Thunderbird.

У цей момент існують два проекти по «ДСТифікації» NSS:

Компанія Ліссі періодично публікує на своєму сайті доступні для завантаження актуальні версії Mozilla Firefox і Mozilla Thunderbird, перезібрані з підтримкою української криптографії. Крім того, існує ряд продуктів цієї компанії, побудований на базі модифікованої бібліотеки NSS – високорівнева бібліотека NSSCryptoWrapper, плагін LCSignPlugin, десктопний додаток для ЕЦП під Android SignMaker-A.

Слід зазначити, що модифікований фахівцями цієї компанії NSS дозволяє використовувати як програмні PKCS# 11-токени, так і апаратні (Рутокен ЕЦП, eToken ДСТ, JaCarta ДСТ, MS_KEY).

Atoken – це open source проект компанії R-альфа. У рамках проекту створений програмний PKCS# 11-токен з українською криптографією й викладені патчі для певної версії NSS і компонента Security Manager, що дозволяють використовувати в продуктах Mozilla українську криптографію (TLS, ЕЦП, PKI). Крім того R-альфа пропонує реалізацію програмного PKCS# 11-токена з підтримкою сертифікованої бібліотеки Агава-із за назвою Агава-про.

Бібліотеки с власним інтерфейсом

Пропріетарні бібліотеки надають власний API для вбудовування в додатки. У даний список можна внести:

- Агава-с.
- Крипто-С.
- Крипто-про.

Локальні проксі

Основним принципом дії локального проксі є прийом незахищеного з'єднання від додатка, установка TLS-тунелю з віддаленим сервером і передача «прикладного рівня» між додатком і віддаленим сервером по цьому тунелі.

Деякі локальні проксі крім того доповнені механізмом ЕЦП спеціальним образом промартільних WEB-форм (Inter-PRO, Магпро Криптотуннель). Існують локальні проксі, які надають для браузера WEB API ЕЦП (систему HTTP-Запитів і відповідей, аналогічних програмному API криптобібліотеки).

Проблеми:

− проксі повинен бути запущений;

− додаток повинне працювати через проксі, потрібно «навчити» його цьому;

− можуть використовуватися нестандартні порти, звідси проблеми у файрволом;

− якщо додаток «ходить» через localhost, то, наприклад, в адресному рядку браузера прописане localhost…– нестандартно;

− додаткові обмеження на розробку web-сайту – у ряді випадків використання тільки відносних посилань, щоб не «вилетіти» з тунелю;

− проксі зконфігурован на проксіровані кінцевої групи сайтів, розширення групи – це відновлення клієнтського конфігу;

− робота через зовнішній проксі вимагає додаткового конфігурування локального проксі, при цьому можуть бути проблеми з автентифікацією користувача на зовнішньому проксі.

Плюси:

− рішення використовує універсальну технологію, тому можна не боятися його старіння;

− рішення може застосовуватися на великій кількості платформ, у тому числі на мобільних платформах;

− кроссбраузерність, підтримка всіх WEB-серверів без модифікації;

− не вимагає інсталяції;

− підтримка різних прикладних протоколів.

Браузерні плагіни

Для того, щоб зі скриптів WEB-сторінки викликати нативну бібліотеку більшість браузерів підтримують спеціальні розширення – Active для IE і NPAPI-плагін для GH, MF, Opera, Safari і ін. У цей момент на ринку існує широкий спектр продуктів, що ставляться до браузерним плагінам. Архітектурно дані плагіни можуть бути виконані по-різному. Деякі працюють на базі Crypto API і вимагають додаткової установки криптопровайдера, інші використовують у якості криптоядра PKCS# 11-сумісні пристрої й не вимагають установки додаткових СКЗІ на робоче місце клієнта. Є універсальні плагіни, які підтримують як всі основні криптопровайдери, так і широкий спектр апаратних СКЗІ.

Кроссбраузерні плагіни

Проблеми:

− відсутність TLS;

− видалення NPAPI з Chromium;

− браузери на мобільних платформах не підтримують плагіни;

− настроювання безпеки IE можуть блокувати виконання плагіну.

Плюси:

− кроссплатформеність для плагінов на базі PKCS#11;

− кроссбраузерність;

− плагіни на базі PKCS#11 не вимагають установки СКЗІ;

− прозоре використання для користувача.

ActiveX

Компанія Microsoft розробила два основних клієнтських Active-компоненти, які транслюють функціонала Crypto API у скрипти, у браузер. Для генерації ключа й створення PKCS# 10-запиту застосуються компонент XEnroll/CertEnroll, а для ЕЦП/шифрування й роботи із сертифікатами компонентів CAPICOM.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Python – це об’єктноорієнтована мова програмування високого рівня загального призначення з відкритим кодом. Це визначення може бути важким для новачків, тому розглянемо кожну характеристику окремо, щоб зрозуміти, що вона означає:

– Відкритий вихідний код: це безкоштовно та доступно для подальших покращень, таких як додавання корисних функцій або виправлення помилок.

– Об’єктноорієнтована: заснована не на функціях, але в об’єктах з певними атрибутами й методами.

– Високий рівень: зручний для людини, а не для комп’ютера.

– Загальне призначення: можна використовувати для створення будь-яких програм.

Ця мова використовується в будь-якому програмному забезпеченні, про яке ви тільки можете подумати. Ви можете використовувати його для створення вебсайтів, штучного інтелекту, серверів, програмного забезпечення для бізнесу та багато іншого. Також застосовується в науці про дані, аналізі даних, машинному навчанні, інженерії даних, веброзробці, розробці програмного забезпечення та інших галузях.

Переваги та недоліки Python

Переваги:

– Її легко читати, вчити та писати. Це мова програмування високого рівня з англійським синтаксисом. Це полегшує читання та розуміння коду. Її дійсно легко зрозуміти і вивчити, тому багато людей рекомендують Python новачкам. Вам потрібно менше рядків коду для виконання того ж завдання в порівнянні з іншими основними мовами, такими як C/C++ та Java.

– Підвищує продуктивність. Це дуже продуктивна мова. Завдяки її простоті розробники можуть зосередитися на розв’язанні проблеми. Їм не

потрібно витрачати багато часу на розуміння синтаксису або поведінку мови програмування. Ви пишете менше коду та виконуєте більше завдань.

– Інтерпретована мова. Python мова, що інтерпретується, а це означає, що вона безпосередньо виконує код по рядку. Якщо сталася помилка, вона зупиняє подальше виконання та повідомляє про її виникнення. Вона показує лише одну помилку, навіть якщо у програмі їх кілька. Це спрощує налагодження.

– Динамічно типізована. Python не визначає тип змінної, доки ми не запустимо код. Вона автоматично надає тип даних, коли відбувається процес виконання. Фахівець може не турбуватися про оголошення змінних та типи даних.

– Безкоштовна та з відкритим вихідним кодом. Ця мова постачається під схваленою OSI ліцензією з відкритим вихідним кодом. Це робить його безкоштовним для використання та розповсюдження. Ви можете завантажити вихідний код, змінити його та навіть розповсюджувати свою версію. Це корисно для організацій, які хочуть використати свою версію для розробки.

– Підтримка великих бібліотек. Стандартна бібліотека Python є величезною, ви можете знайти майже всі функції, необхідні для вашого завдання. Таким чином ви не залежите від зовнішніх бібліотек.

– Портативність .У багатьох мовах, таких як C/C++, потрібно змінити свій код, щоб запустити програму на різних платформах. З Python все інакше. Ви тільки пишете один раз і запускаєте її будь-де.

Недоліки:

– Низька швидкість. Вище ми обговорювали, що це інтерпретована мова з динамічною типізацією. Порядкове виконання коду часто призводить до повільного виконання. Динамічна природа Python також є причиною її низької швидкості, оскільки їй доводиться виконувати додаткову роботу при виконанні коду. Тому вона не підходить для цілей, де швидкість важливий аспект проєкту.

– Неефективна для пам’яті. Ця мова програмування використовує великий обсяг пам’яті, це може бути недоліком при створенні програм, коли віддають перевагу оптимізації пам’яті.

– Слабка у мобільних обчисленнях. Python зазвичай використовується у серверному програмуванні. Ми не бачимо – її на стороні клієнта або в мобільних програмах з таких причин: вона не заощаджує пам’ять і має повільну обчислювальну потужність у порівнянні з іншими мовами.

– Доступ до бази даних. Програмувати на цій мові легко, але коли ми взаємодіємо з базою даних, її не вистачає. Рівень доступу до бази даних у Python примітивний та недостатньо розвинений у порівнянні з іншими популярними технологіями.

– Помилки виконання. Це мова з динамічною типізацією, тому тип даних змінної може змінюватись у будь-який час. Змінна, що містить ціле число, у майбутньому може містити рядок, що може призвести до помилок виконання.

Застосування Python:

– Для аналізу даних. Дані стали цінним активом у будь-якій сучасній галузі, і більшість компаній зацікавлені у збиранні, обробці та аналізі релевантних даних, щоб витягти з них цінну інформацію для бізнесу. І тут Python виходить за межі будь-якої конкуренції. Python особливо цінна тим, що крім великої стандартної бібліотеки надає величезний набір додаткових модулів, розроблених спеціально для аналітичних цілей. Найвідоміші бібліотеки Python для аналізу даних – це pandas і NumPy . Ці інструменти дозволяють робити з вашими даними майже все, наприклад, очищати і аналізувати їх, вивчати статистику або візуалізувати приховані тенденції у ваших даних.

– Для візуалізації даних. Візуалізація даних – це окрема частина аналізу даних, яка допомагає нам подавати інформацію, необроблену чи очищену, у більш змістовній формі. Тут Python знову входить у гру, пропонуючи широкий спектр інструментів візуалізації даних. Найпопулярніші з них – matplotlib і

заснований на ній seaborn. Використовуючи їх, ми можемо створювати буквально всі види візуалізації: від найпростіших до складніших.

– Для машинного навчання. Машинне навчання (ML) є основою більшості завдань науки даних. Він є областю штучного інтелекту, пов'язаною з використанням алгоритмів, що дозволяють машинам вивчати закономірності та тенденції на основі історичних даних, щоб робити прогнози на основі невідомих даних. – Використовуючи методи ML, ми можемо створювати моделі, які можуть точно передбачити швидкість відтоку клієнтів компанії, оцінити ризик виникнення у людини певного захворювання, визначити оптимальне розташування автомобілів таксі й т.д. За допомогою Python ми можемо побудувати модель ML, використовуючи лише три рядки коду.

– Для розробки програмного забезпечення. Крім свого багатостороннього застосування в галузях науки про дані, Python використовується на кожному етапі розробки програмного забезпечення, включаючи контроль складання, автоматичну безперервну компіляцію, прототипування, відстеження помилок, тестування та обслуговування програмного забезпечення. За допомогою цієї мови можемо створювати аудіо- або відеопрограми на основі методів штучного інтелекту, машинного навчання, API (інтерфейсів прикладного програмування), GUI (графічних інтерфейсів) або будь-якого іншого типу програмного забезпечення.

– Для веброзробки. У той час як для створення візуальної частини вебсайту ми переважно будемо використовувати такі мови, як HTML, CSS та JavaScript, для його невидимої частини ми часто вибираємо Python. Серед масштабних вебсайтів та програм, створених за допомогою цієї мови, варто згадати Google, Facebook, Instagram, YouTube, Dropbox та Reddit.

– Для автоматизації задач/скриптингу. Це відмінний інструмент для написання програм для автоматизації різних завдань, що повторюються. Цей процес називається скриптингом. Зокрема, можна робити скрипти для роботи з файлами та папками. Наприклад, можна створювати, перейменовувати,

перетворювати, розділяти, об’єднувати або видаляти файли, перевіряти їх на наявність помилок. Ви також можете використовувати автоматизацію Python для пошуку та завантаження інформації з Інтернету, заповнення та надсилання онлайн-форм та надсилання регулярних повідомлень або електронних листів.

Яким фахівцям потрібно володіти Python:

– Фахівець з даних.

– Аналітик даних.

– Інженер даних.

– Інженер з машинного навчання.

– Журналіст даних.

– Архітектор даних.

– Повний стек веброзробника.

– Backend-розробник.

– DevOps-інженер.

– Інженер-програміст

Можемо зробити висновок, що Python ще довго буде популярною мовою, хоч і має низку недоліків. Цю мову використовують для створення вебсайтів, штучного інтелекту, серверів, програмного забезпечення для бізнесу, аналізу даних, машинного навчання, інженерії даних та для багатьох інших областей. Це перспективна і затребувана навичка, яка необхідна у всіх галузях.

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 захисту конфіденційності інформації з використанням інструкцій Intel Core i7.

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 контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

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

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Intel Core i7 – сімейство процесорів Intel з архітектурою AMD64. Це перше сімейство, у якому з'явилася мікроархітектура Intel Nehalem, пізніше також використовувалися мікроархітектури Sandy Bridge, Ivy Bridge, Haswell. Також є спадкоємцем сімейства Intel Core 2. Ідентифікатор Core i7 застосовується й до первісного сімейства процесорів[1][2] з робочою назвою Bloomfield,[3] запущених в 2008 [2]. Назва Core i7 не показує покоління процесора, вона лише продовжує використовувати успішну серію брендів Core.[4]

Дана мікроархітектура містить ряд нових можливостей. От лише деякі з них, у порівнянні з Core 2:

– У процесорів для роз’ємів LGA 1366, FSB замінена на QPI (QuickPath Interconnect). Це означає, що материнська плата повинна використовувати чипсет, що підтримує QuickPath Interconnect. Цю технологію підтримують чипсети Intel X58 і Intel X79.

– Core i7 не призначений для багатопроцесорних материнських плат, тому є тільки один інтерфейс QPI.

– Процесори Core іх для роз’ємів LGA 1156 не використовують зовнішню шину QPI. Вона не потрібно у зв'язку з повною відсутністю північного мосту (повністю інтегрований у процесор і пов'язаний з ядрами по внутрішній шині QPI на швидкості 2,5 гігатранзакції в секунду).

– Контролер пам'яті в Core i7 9xx підтримує до 3 каналів пам'яті, і в кожному може бути один або два блоки пам'яті DDR3DIMMs. Тому материнські плати на s1366 підтримують до 6 планок пам'яті, а не 4, як Core 2. Контролер пам'яті в Core i7, i5 і i3 на сокеті 1156 як і раніше двоканальний.

– Підтримка тільки пам'яті стандарту DDR 3-800/1066 MHz (для Intel Core i7 980 підтримка пам'яті DDR 3-1066 MHz).

– Однокристальний пристрій: всі ядра, контролер пам'яті (а в Core i7 8xx і контролер PCI-E) і кеш перебувають на одному кристалі.

– Підтримка Hyper-threading, з яким виходить до 12 (залежно від моделі CPU) віртуальних ядер. Ця можливість була представлена в архітектурі NetBurst, але від її відмовилися в Core.

– 8 (Або 12 у шестиядерних моделях) мегабайт кеша L3.

– Підтримка Turbo Boost, з яким процесор автоматично збільшує продуктивність тоді, коли це необхідно.

– Починаючи з Sandy Bridge – підтримка DRM технології «Intel Insider» для стримінгу відео високої чіткості [5].

Процесори Intel Core i7 975 Extreme Edition і 950 випущені на зміну процесорам 965 Extreme Edition і 940 відповідно.

У процесорах i7 серії 800 відсутня зовнішня шина QPI, це пов'язане з тим, що процесор повністю поглинув північний міст, отже ні шина FSB, ні QPI не потрібно.

Шина DMI є присутнім між аналогами північного й південного мосту в системах і із шиною QPI, і із шиною DMI.

Intel Core i7 920 заміщається ледве більш швидким 930.

Продуктивність

Система з одним процесором 2,93 Ггц Core i7 940 була використана для запуску програми випробування продуктивності 3DMark Vantage і дала результат по процесорній підсистемі в 17966 умовних балів.[6] Один 2,66 GHz Core i7 920 дав 16294 бала. А один 2,4 GHz Core 2 Duo E6600 дав 4300 тих же умовних балів.[7]

AnandTech випробувала технологію Intel QuickPath Interconnect (версія 4,8 ГП/с) і оцінила пропускну здатність копіювання за допомогою використання пам'яті DDR3 частотою 1066 Мгц у трьохканальному режимі, в 12,0 ГБ/с. А

система 3,0 Ггц Core 2 Quad, що використовує пам'ять DDR3 1066 Мгц у двоканальном режимі, досягла 6,9 ГБ/с.[8]

Користувальницький розгін буде можливий у всіх моделях, що вийшли, дев'ятисотої серії сукупно з материнськими платами, оснащеними чипсетом X58. [9]

Деякі інтернет-ресурси припускають, що Core i7 менш продуктивно в іграх (через L3 кеша, у якого вище затримки перед L2 кешем). У проведених тестах Core i7 940 і QX9770 спостерігається паритет (в 2 іграх верх одержав Core i7 940, ще у двох – QX9770, при невеликій різниці в результатах)[10].

У тесті Super PI 1M процесор Core i7 920, що працює на частоті 2,93 Ггц, пройшов тест за 14,77 секунд, тоді як QX9770 (3,2 Ггц) пройшов його за 14,42 секунди.

Захисна кришка процесорів складається з нікельованої міді, підложка кремнієва, а контакти виконані з позолоченої міді.

Мінімальна й максимальна температури зберігання Core i7 рівні відповідно -55 °C і 70 °C.

Core i7 здатний витримати до 934 Н статичної й до 1834 Н динамічного навантаження.

Максимальне тепловиділення процесорів Core i7 дорівнює 130 Вт, у режимі бездіяльності воно становить 12-15 Вт.

Ефективність стандартного вентилятора Core i7 різко знижується, якщо температура усередині системного блоку перевищує 40 °C.[11]

AES у Core i7

Сьогодні безпека є важливою темою – однак важливої її вважають, головним чином, тільки професіонали. Втім, якщо безпека стає маркетинговим елементом або перетворюється в характеристику продуктивності, те такі компанії, як Intel, починають активно неї просувати. Стандарт AES або Advanced Encryption Standard сертифікований Керуванням національної безпеки США (NSA) і урядом США, а також багатьма іншими органами. 32-нм двопроцесорне

покоління процесорів Intel обіцяє істотний приріст продуктивності шифрування й розшифровки AES завдяки новим інструкціям.

Шифрування насправді використовується набагато більш інтенсивно, ніж звичайно зауважують користувачі. Усе починається із сайтів в Інтернеті, які містять конфіденційну інформацію, таку як особистих даних користувачів, або із сайтів, де є конфіденційна інформація про транзакції: всі вони використовують шифрування TLS або SSL. Такі сервіси, як VoIP, мессенджери й електронна пошта також можуть захищатися таким же способом. Віртуальні приватні мережі (VPN, Virtual Private Network) – ще один приклад, імовірно, дуже популярний. Шифрування також зачіпає й такі конфіденційні області, як електронні платежі. Втім, TLS/SSL – це криптографічні протоколи зв'язку, а AES, що Intel прискорює, починаючи з нового 32-нм покоління процесорів, є стандартом шифрування загального призначення. Його можна використовувати для шифрування окремих файлів, контейнерів даних і архівів або навіть зашифровувати розділи й диски цілком – будь те USB-брелок або системний жорсткий диск. AES може виконуватися програмно, але є й продукти з апаратним прискоренням, оскільки шифрування й розшифровка є досить серйозним обчислювальним навантаженням. Такі рішення, як TrueCrypt або Microsoft BitLocker, що є частиною Windows 10, здатні шифрувати цілі розділи "на льоту".

Чи вважаєте ви чи ні, що у вашій системі є конфіденційні дані, залежить від того, що ви маєте на увазі під цими даними, а також і від вашого персонального рівня комфорту. Крім того, безпека завжди має на увазі правильну стратегію й акуратність у зберіганні конфіденційних даних. Ніколи не можна залишати без уваги такі дані, як реквізити вашого паспорта або номер і дату закінчення терміну дії банківської карти. Або навіть PIN-код телефону.

Одне можна сказати точно: найкраще бути акуратним і розсудливим, ніж навпаки – тим більше що для цього потрібно не так багато зусиль. Підхід Intel до додавання прискорення AES не охоплює всі додатки шифрування й сценарії, тільки самий популярний стандарт – при цьому ви одержите все це безкоштовно

у всіх майбутніх 32-нм настільних процесорах для масового ринку або для більше дорогих сегментів. Але чи дійсно нові інструкції AES New Instructions забезпечують істотний приріст продуктивності в типових сценаріях шифрування або це, скоріше, плід зусиль відділу маркетингу? Давайте подивимося.

3.2 Розробка структурної схеми

Intel Core i7 з підтримкою AES

Процесори насправді знаменують собою зміну поколінь, оскільки при цьому не тільки відбувається перехід на наступний техпроцес (32 нм у порівнянні з 45 нм), але перед нами й перше покоління CPU з підтримкою декількох інструкцій, що прискорюють шифрування. Intel згадує добавку як AES New Instructions. Вони складаються із:

– інструкцій для шифрування AES (AESENC, AESENCLAST);

– інструкцій для розшифровки (AESDEC, AESDECLAST);

– інструкції для роботи із ключем AES (AESIMC, AESKEYGENASSIST).

Як і раніше, інструкції відносяться до SIMD, тобто до типу "одна інструкція багато даних" (Single Instruction Multiple Data). Підтримуються всі три ключі AES (128, 192 і 256 біт з 10, 12 і 14 проходами підстановки й перестановки).

Оскільки всі інструкції AES мають фіксовану затримку, що не залежить від даних, тобто час фіксований й доступ до пам'яті не потрібно. Крім того, модель програмування така ж, як і у випадку інших інструкцій SSE з первісного стандарту SSE4. Таким чином, всі операційні системи, які підтримують роботу з SSE, зможуть використовувати й інструкції AES New Instructions.

Будьте обережні при виборі процесора із прискоренням AES, оскільки сьогодні лише деякі моделі підтримують нові інструкції. Усе було б набагато простіше, якби Intel додала підтримку нових інструкцій в усі моделі.

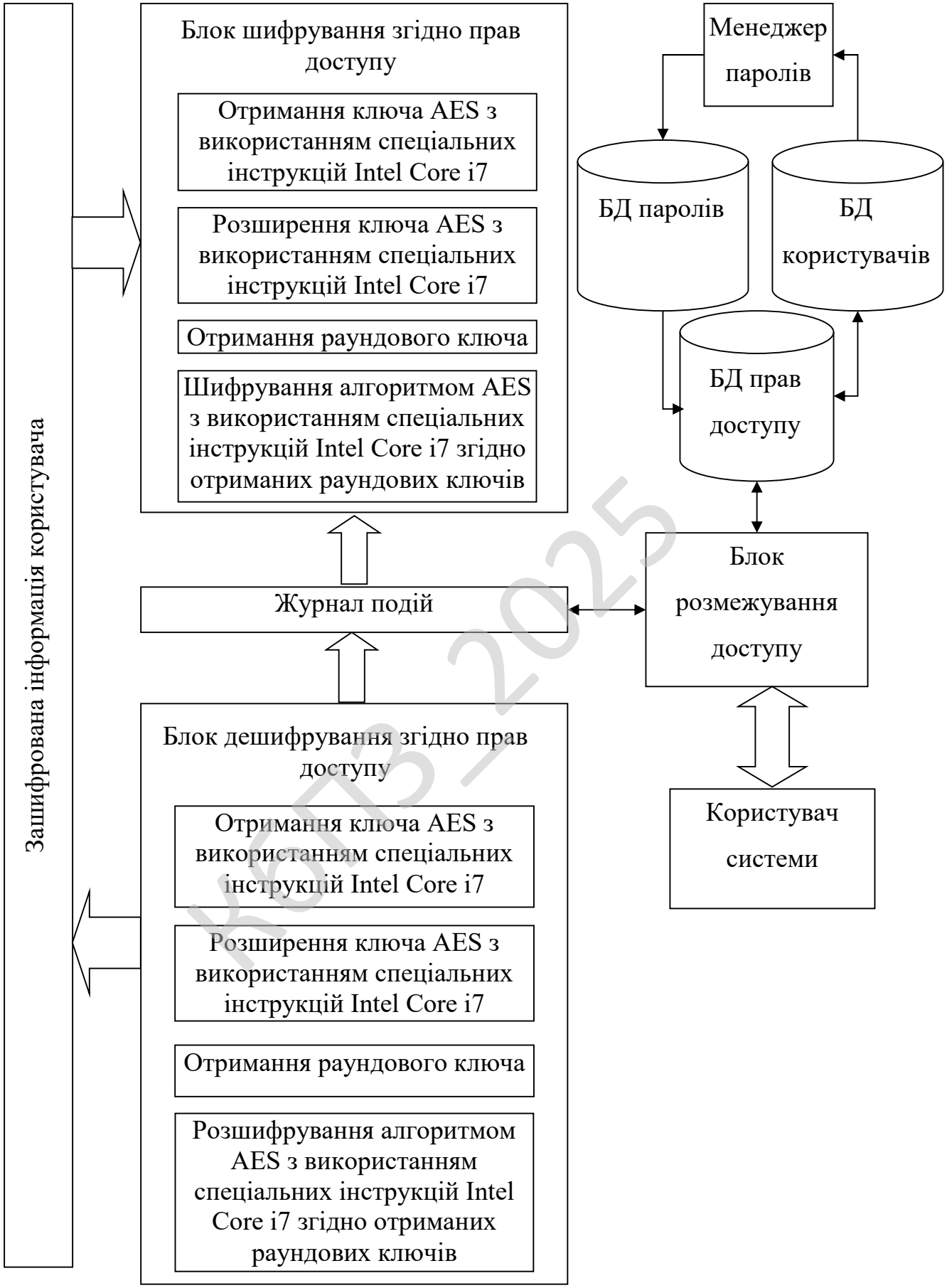

Рисунок 3.1 – Структурна схема системи

3.3 Розробка функціональної схеми

AES

AES розшифровується як "Advanced Encryption Standard" – це найбільш популярний стандарт симетричного шифрування у світі IT. Стандарт працює із блоками розміром 128 біт і підтримує 128-, 192- або 256-бітні ключі (AES-128, AES-192 і AES-256). Багато утиліт шифрування, та ж TrueCrypt, підтримали алгоритм AES на самому початку його існування. Але найбільший фактор успіху AES, звичайно, полягає в його прийнятті урядом США в 2002 році, при цьому в 2003 році він був прийнятий як стандарт для захисту секретних даних.

Шифрування даних за допомогою AES

Шифрування AES базується на системі підстановок з перестановкою, тобто над даними проводиться серія математичних операцій, щоб створити значно модифікований масив даних (зашифрований). Як вихідна інформація виступає текст, а ключ відповідає за виконання математичних операцій. Операції можуть бути як зовсім простими, наприклад, зрушення біт або XOR, так і більше складними. Один прохід можна легко розшифрувати, тому всі сучасні алгоритми шифрування побудовані на декількох проходах. У випадку AES це 10, 12 або 14 проходів для AES-128, AES-192 або AES-256. До речі, ключі AES проходять таку ж процедуру, що й користувальницькі дані, тобто вони являють собою що змінюється раундовий ключ.

Процес працює з масивами 4x4 з одиночних байтів, також називаних боксами: S-box використовуються для підстановок, P-box – для перестановок. Підстановки й перестановки виконуються на різних етапах: підстановки працюють усередині так званих боксів, а перестановки міняють інформацію між боксами. S-box працює по складному принципі, тобто навіть якщо єдиний вхідний біт буде мінятися, то це вплине на декілька вихідних біт, тобто властивості кожного вихідного біта залежать від кожного вхідного біта.

Використання декількох проходів забезпечує гарний рівень шифрування, при цьому необхідно відповідати критеріям розсіювання (diffusion) і

заплутування (confusion). Розсіювання виконується через каскадну комбінацію трансформацій S-box і P-box: при зміні тільки одного біта у вхідному тексті S-box буде модифікувати вихід декількох біт, а P-box буде псевдовипадково поширювати цей ефект по декількох S-box. Коли ми говоримо про те, що мінімальна зміна на вході дає максимальну зміну на виході, ми говоримо про ефект сніжної грудки.

Останнім часом є чимало обговорень так званих зломів, які обходять необхідність запуску розширеного пошуку методом грубої сили для знаходження правильного ключа розшифровки. Технології, такі як атаки XSL і атаки related-key обговорюються досить інтенсивно – але успіх невеликий. Єдиний працюючий спосіб злому шифрування AES полягає в так званій атаці побічного каналу (side-channel). Для її здійснення атака повинна відбуватися тільки на host-системі, на якій виконується шифрування AES, і при цьому вам необхідно знайти спосіб одержання інформації про синхронізацію кеша. У такому випадку можна відстежити число тактів комп'ютера до завершення процесу шифрування.

Звичайно, все це не так легко, оскільки вам потрібен доступ до комп'ютера, причому досить повний доступ для аналізу шифрування й права на виконання коду. Тепер вам напевно зрозуміло, чому "діри" у системі безпеки, які дозволяють зловмисникові одержати такі права, нехай навіть вони звучать зовсім абсурдно, необхідно закривати якнайшвидше. Але не будемо розтікатися думкою по древу: якщо ви одержите доступ до цільового комп'ютера, то добування ключа AES – справа часу, тобто вже не трудомістке завдання для суперкомп'ютерів, що вимагає величезних обчислювальних ресурсів.

AES усередині Intel

На даний момент інтегровані в CPU інструкції AES починають мати сенс – незалежно від можливих переваг по продуктивності. З погляду безпеки процесор може обробляти інструкції AES в інкапсульованому виді, тобто йому не потрібні які-небудь таблиці перетворення, необхідні для атаки методом побічного каналу.

Процес розробки нового федерального інформаційного стандарту (FIPS) для шифрування даних Advanced Encryption Standard (AES) був ініційований

Національним Інститутом стандартів і технологій (NIST). На початку січня 1997 року NIST оголосив про початок розробки AES, випустивши документ "Announcing development of a federal information processing standard for advanced encryption standard", що містить первинні вимоги до алгоритму [2]:

– Алгоритм шифрування AES повинен бути відкрито опублікований.

– Алгоритм повинен бути симетричним блоковим шифром.

– AES повинен передбачати можливість збільшення довжини ключа.

– AES повинен бути легко реалізуємо й апаратної й програмно.

– AES повинен поширюватися безкоштовно або бути загальнодоступним у рамках патентів ANSI.

Як відомо, проаналізувавши результати всіх досліджень, експерти вибрали як стандарт AES алгоритм Rijndael. Що не виглядає дивним – у таблиці з характеристиками алгоритмів чітко видно, що практично по всіх характеристиках Rindael, як мінімум, не уступає іншим алгоритмам-фіналістам.

Що стосується алгоритмів Serpent, Twofish, MARS і RC6, те видно, що вони практично рівнозначні по сукупності характеристик, за винятком алгоритму MARS, що має істотно більше недоліків, у тому числі, алгоритм практично не реалізуємий в умовах обмежених ресурсів.

Дамо опис алгоритму AES. У його основі лежить алгоритм Rijndael

Формат блоків даних і число раундів

RIJNDAEL – це симетричний блоковий шифр, що оперує блоками даних розміром 128 і довжиною ключа 128, 192 або 256 біт – назва стандарту відповідно "AES-128", "AES-192", і "AES-256".

Уведемо наступні позначення:

– Nb – число 32-бітних слів які містяться у вхідному блоці, Nb – 4;

– Nk – число 32-бітних слів, що втримуються в ключі шифрування, Nk = 4,6,8;

– Nr – число раундів шифрування, як функція від Nb і Nk, Nr = 10,12,14.

Вхідні (input), проміжні (state) і вихідні (output) результати перетворень, виконуваних у рамках криптоалгоритму, називаються станами (State). Стан можна представити у вигляді матриці 4 x Nb, елементами якої є байти (чотири рядки по Nb байт) у порядку S00, S10, S20, S30, Sm, Sn, S21, S31. Рисунок 3.2 демонструє таке подання, що носить назву архітектури «Квадрат».

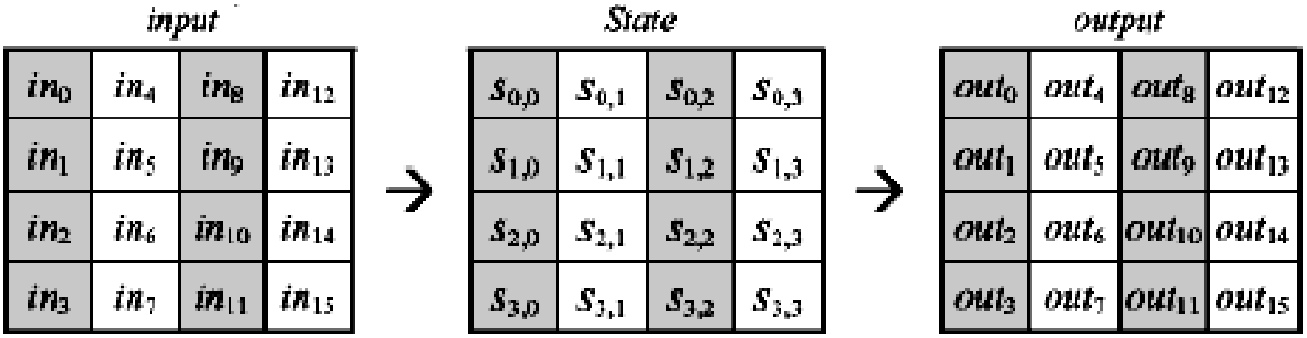

Рисунок 3.2 – Приклад подання блоку у вигляді матриці 4xNb

На старті процесів шифрування й дешифрування масив вхідних даних те, input перетвориться в масив State за правилом:

$$s[r,c] = in[r + 4c], \quad (3.1)$$

де $0 < r < 4$ і $0 < c < Nb$.

Наприкінці дії алгоритму виконується зворотне перетворення:

$$out[r + 4c] = s[r,c], \quad (3.2)$$

де $0 < r < 4$ і $0 < c < Nb$ – вихідні дані виходять із байтів стану в тому же порядку.

Чотири байти в кожному стовпці стану являють собою 32-бітне слово, якщо r – номер рядка в стані, те одночасно він є індексом кожного байта в цьому слові. Отже стан можна представити як одномірний масив 32-бітних слів $w_0,...,w_n$, де номер стовпця стану с є індекс у цьому масиві. Тоді стан можна представити так:

$w_0 = s_{0,0}, s_{1,0}, s_{2,0}, s_{3,0}$	$w_2 = s_{0,2}, s_{1,2}, s_{2,2}, s_{3,2}$
$w_1 = s_{0,1}, s_{1,1}, s_{2,1}, s_{3,1}$	$w_3 = s_{0,3}, s_{1,3}, s_{2,3}, s_{3,3}$

Ключ шифрування також як і масив State представляється у вигляді прямокутного масиву із чотирма рядками. Число стовпців цього масиву дорівнює Nk.

Для алгоритму AES число раундів Nr визначається на старті залежно від значення Nk (таблиця 3.1):

Таблиця 3.1 – Залежність значення Nr від Nk і Nb

	Nk	Nb	Nr
AES– 128	4	4	10
AES– 192	6	4	12
AES – 256	8	4	14

Алгоритм вироблення ключів (Key Schedule)

Введемо наступні позначення:

– Rcon[i] – масив 32-бітних раундових констант;

– RotWord() – операція циклічної перестановки вхідного 4-х байтного слова у вихідне за наступним правилом [a0, ai, a2, a3] —> [ax, a2, a3, a0];

– SubWord() – операція заміни в 4-х байтному слові за допомогою S-Box кожного байта;

⊕ – операція що виключає або – XOR.

```
KeyExpansion(byte key[4*Nk], word w[Nb*(Nr+1)], Nk)
begin
   word temp

   i = 0

   while (i < Nk)
      w[i] = word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
      i = i+1
   end while

   i = Nk

   while (i < Nb * (Nr+1)]
      temp = w[i-1]
      if (i mod Nk = 0)
         temp = SubWord(RotWord(temp)) xor Rcon[i/Nk]
      else if (Nk > 6 and i mod Nk = 4)
         temp = SubWord(temp)
      end if
      w[i] = w[i-Nk] xor temp
      i = i + 1
   end while
end
```

Раундові ключі виходять із ключа шифрування за допомогою алгоритму вироблення ключів. Він містить два компоненти: розширення ключа (Key Expansion) і вибір раундового ключа (Round Key Selection). Основні принципи алгоритму виглядають у такий спосіб:

– загальне число бітів раундових ключів дорівнює довжині блоку, помноженої на число раундів, плюс 1;

– ключ шифрування розширюється в розширений ключ (Expanded Key);

– раундові ключі беруться з розширеного ключа в такий спосіб: перший раундовий ключ містить перші Nb слів, другий – наступні Nb слів і т.д.

Розширення (планування) ключа

Розширений ключ являє собою лінійний масив w[i] складається з Nb*(Nr + 1) 4-х байтових слів, i = Nb*(Nr + 1).

Рисунок 3.3 – Процедури розширення й виборки раундового ключа для Nk = 4

Ясно-сірим кольором виділені слова розширеного ключа, які формуються без використання функцій SubWord() і RotWord().

Темно-сірим кольором, виділені слова розширеного ключа, при обчисленні яких використовуються перетворення SubWord() і RotWord().

Перші Nk слів містять ключ шифрування. Кожне наступне слово w[i] виходить за допомогою XOR попереднього слова w[i-1] і слова на Nk позицій раніше:

$$W[i\text{-}Nk]: w[i]= w[i\text{-}1] \oplus w[i\text{-}Nk]. \quad (3.3)$$

Для слів, позиція яких кратна Nk, перед XOR застосовується перетворення до w[i-1], а потім ще додається раундова константа Rcon[i]. Перетворення реалізується за допомогою двох додаткових функцій: RotWord() і SubWord().

Значення Rcon[j] дорівнює 2j-1. Значення w[i] у цьому випадку визначається вираженням:

$$w[i] = SubWord(RotWord(w[i-1]) \oplus Rcon[i/Nk] \oplus M[i-Nk]. \quad (3.4)$$

Вибір раундового ключа і-тий раундовий ключ вибирається зі слів масиву розширеного ключа в проміжку від W[Nb * i] до W[Nb * (i+1)].

Для функції зашифрування розширений ключ генерується алгоритмом. Для функції зворотного розшифрування використовується цей же ключ, але у зворотній послідовності починаючи з останнього раундового підключа зашифрування.

Для функції прямого розшифрування використовується трохи модифікований алгоритм планування ключа. При формуванні розгорнутого ключа в процедуру планування необхідно додати наприкінці додаткове перетворення, причому розширений ключ використовується при прямому розшифруванні в тій же послідовності, що й при зашифруванні.

Додаткове перетворення розширеного ключа для функції прямого розшифрування:

```
for i = 0 step 1 to (Nr+1)*Nb-1
   dw[i] = w[i]
end for

for round = 1 step 1 to Nr-1
   InvMixColumns(dw[round*Nb, (round+1)*Nb-1])
end for
```

Функція зашифрування

Введемо наступні позначення:

– SubBytes() – заміна байтів – побайтова нелінійна підстановка в State-блоках (S-Box) з використанням фіксованої таблиці замін розмірністю 8x256 (affain map table).

– ShiftRows() – зрушення рядків – циклічне зрушення рядків масиву State на різну кількість байт.

– MixColumns() – перемішування стовпців – множення стовпців стану, розглянутих як багаточлени над GF(2^8).

– AddRoundKey() – додавання з раундовим ключем – поразрядне XOR умісту State з поточним фрагментом розгорнутого ключа.

На псевдокоді операція зашифрування виглядає в такий спосіб:

```
Cipher[byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
   byte  state[4,Nb]

   state = in

   AddRoundKey(state, w[0, Nb-1])

   for round = 1 step 1 to Nr-1
      SubBytes(state)
      ShiftRows(state)
      MixColumns(state)
      AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
   end for

   SubBytes(state)
   ShiftRows(state)
   AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])

   out = state
end
```

Після заповнення масиву State елементами вхідних даних до нього застосовується перетворення AddRoundKey, далі, залежно від величини Nk, масив State піддається раундовій трансформації 10, 12 або 14 разів, причому у фінальний раунд є трохи вкороченим – у ньому відсутнє перетворення MixColumns. Вихідними даними описаної послідовності операцій є шифротекст – результат дії функції зашифрування AES.

Функції розшифрування

У специфікації алгоритму AES пропонуються два види реалізацій функції розшифрування відмінних друг від друга послідовністю додатка перетворень зворотних перетворенням функції зашифрування й послідовністю планування ключів.

Введемо наступні позначення:

– InvSubBytes() – зворотна SubBytes() заміна байтів – побайтова нелінійна підстановка в S-блоках з використанням фіксованої таблиці замін розмірністю 8x256 (inverse affain map).

– InvShiftRows() – зворотне зрушення рядків ShiftRows() – циклічне зрушення рядків масиву State на різну кількість байт.

– InvMixColumns() – відновлення значень стовпців – множення стовпців стану, розглянутих як багаточлени над GF(2^8).

Функція прямого розшифрування

Змінивши певним чином послідовність планування ключа можна побудувати ще один алгоритм – алгоритм прямого розшифрування.

```
EqInvCipher(byte in[4*Nb], byte out[4*Nb], word dw[Nb*(Nr+1)])
begin
   byte  state[4,Nb]

   state = in

   AddRoundKey(state, dw[Nr*Nb, (Nr+1)*Nb-1])

   for round = Nr-1 step -1 downto 1
      InvSubBytes(state)
      InvShiftRows(state)
      InvMixColumns(state)
      AddRoundKey(state, dw[round*Nb, (round+1)*Nb-1])
   end for

   InvSubBytes(state)
   InvShiftRows(state)
   AddRoundKey(state, dw[0, Nb-1])

   out = state
end
```

Дві наступні властивості дозволяють зробити це:

– Порядок додатка функцій SubBytes() і ShiftRows() не грає ролі. Те ж саме вірно й для операцій InvSubBytes() і InvShiftRows(). Це відбувається тому, що функції SubBytes() і InvSubBytes() працюють із байтами, а операції ShiftRows() і InvShiftRows() зрушують цілі байти, не зачіпаючи їхніх значень.

– Операція MixColumns() є лінійною щодо вхідних даних, що означає InvMixColumns(State XOR RoundKey) = InvMixColumns(State) XOR InvMixColumns(RoundKey).

Ці властивості функцій алгоритму шифрування дозволяють змінити порядок застосування функцій InvSubBytes() і InvShiftRows(). Функції AddRounKey() і InvMixColumns() також можуть бути застосовані у зворотному порядку, але за умови, що стовпці (32-бітні слова) розгорнутого ключа розшифрування попередньо пропущені через функцію InvMixColumns().

Таким чином, можна реалізувати більше ефективний спосіб розшифрування з тим же порядком додатка функцій як і в алгоритмі зашифрування.

Функція зворотного розшифрування

Якщо замість SubBytes(), ShiftRows(), MixColumns() і AddRoundKey() у зворотній послідовності виконати інверсні їм перетворення, можна побудувати функцію зворотного розшифрування. При цьому порядок використання раундових ключів є зворотним стосовно тому, що використовується при зашифруванні. На псевдокоді вона виглядає так:

```
InvCipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
   byte  state[4,Nb]

   state = in

   AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])

   for round = Nr-1 step -1 downto 1
      InvShiftRows(state)
      InvSubBytes(state)
      AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
      InvMixColumns(state)
   end for

   InvShiftRows(state)
   InvSubBytes(state)
   AddRoundKey(state, w[0, Nb-1])

   out = state
end
```

Алгоритм зворотного розшифрування, описаний вище має порядок додатка операцій – функцій зворотний порядку операцій в алгоритмі прямого зашифрування, але використовує ті ж параметри розгорнутого ключа.

Раундове перетворення

Раундове перетворення складається з послідовного застосування до масиву State ряду трансформацій.. Зараз обговоримо деталі його реалізації.

Нелінійна заміна байтів масиву стану за допомогою трансформації SubBytes має вигляд:

$$\lambda(f) = ((X^4 + X^3 + X^2 + X + 1) \cdot f + X^6 + X^5 + X + 1) \bmod (X^8 + 1) \quad (3.5)$$

Багаторазове обчислення в процесі зашифрування даного вираження робило б невиправдане обчислювальне навантаження на виконуючу систему, тому для практичної реалізації найбільш прийнятним рішенням є використання попереднє обчисленої таблиці заміни S-Box. Логіка роботи S-Box при перетворенні байта (xy} відбита в шестнадцятковому виді на рисунку 3.4:

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Рисунок 3.4 – Таблиця S-Box заміни байт

Її використання зводить операцію SubBytes до найпростішої вибірки байта з масиву λ (f) = Sbox[f].

У функціях розшифрування застосовується операція зворотна InvSubBytes().

Вона реалізується так само просто, як і попередня за допомогою інверсної таблиці S-Box – $\lambda^{-1}(f)$ = InvSbox[f]. Її логіка роботи при перетворенні байта (xy} відбита в шістнадцятковому виді на рисунку 3.5.

Рисунок 3.6 ілюструє застосування перетворення заміни байт до стану у функціях зашифрування й розшифрування.

У перетворенні зрушення рядків (ShiftRows) останні 3 рядка стану циклічно зрушуються ВЛІВО на різне число байтів. Рядок 1 зрушується на С1 байт, рядок 2 – на С2 байт, і рядок 3 – на С3 байт. Значення зрушень С1, С2 і С3 в Rijndael залежать від довжини блоку Nb .

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
	1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
	2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
	3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
	4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
	5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
	6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
	7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
	8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
	9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
	a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
	b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
	c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
	d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
	e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
	f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Рисунок 3.5 – Таблиця S-Box інверсної заміни байт

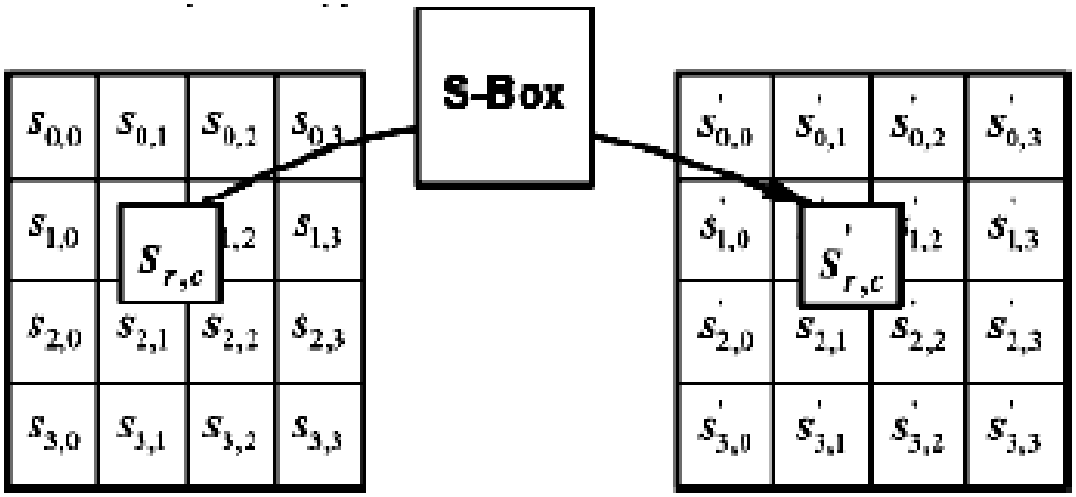

Рисунок 3.6 – Перетворення стану за допомогою таблиці заміни S-Box

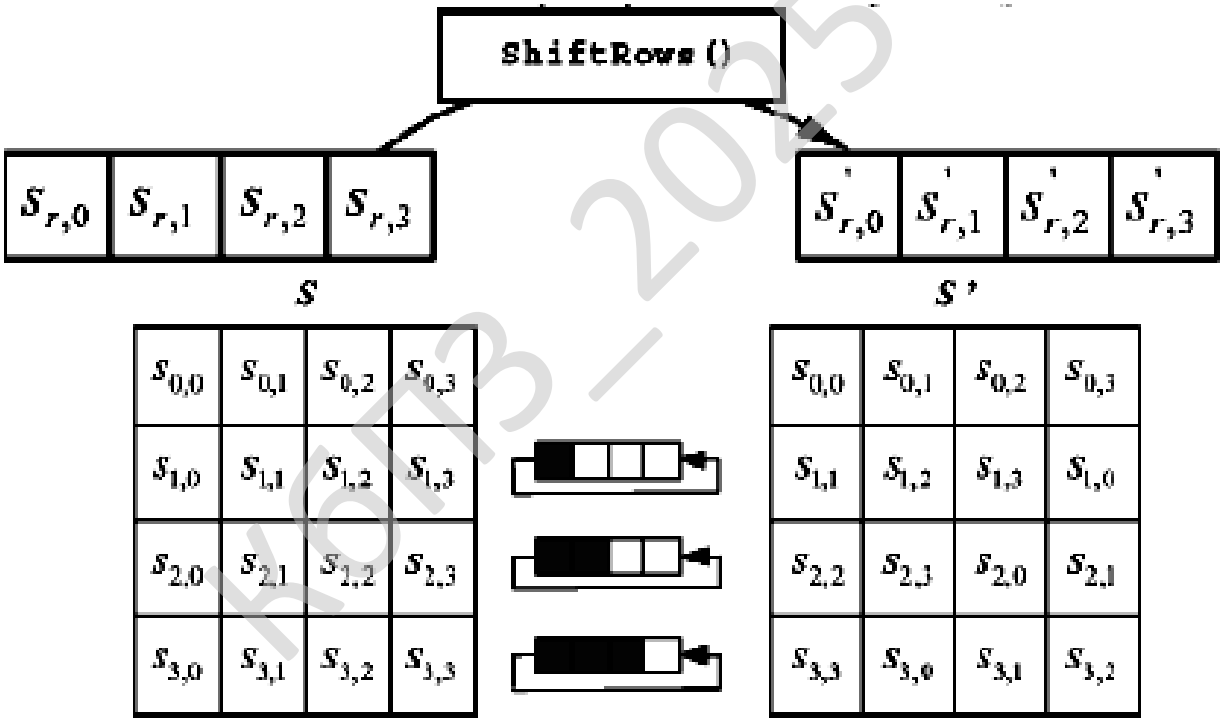

Рисунок 3.7 – Перетворення зрушення рядків у функції зашифрування

У перетворенні зворотного зрушення рядків InvShiftRows останні 3 рядка стану циклічно зрушуються ВПРАВО на різне число байтів. Рядок 1 зрушується на C1 байт, рядок 2 – на C2 байт, і рядок 3 – на C3 байт.

Перемішування стовпців

У перетворенні перемішування стовпців (MixColumns) стовпці стану розглядаються як багаточлени над GF(28) і піддаються перетворенню $c(x) = c * g\mathrm{mod}(Y^4 + 1)$, де c = (X,1,1,X+1), тобто множаться за модулем $x^4 + 1$ на багаточлен c(x), що виглядає, як: $c(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$.

Застосування цієї операції до всім чотирьох стовпцям стану позначається, як MixColumns(State).

Рисунок 3.8 демонструє застосування перетворення MixColumns до стовпця стану.

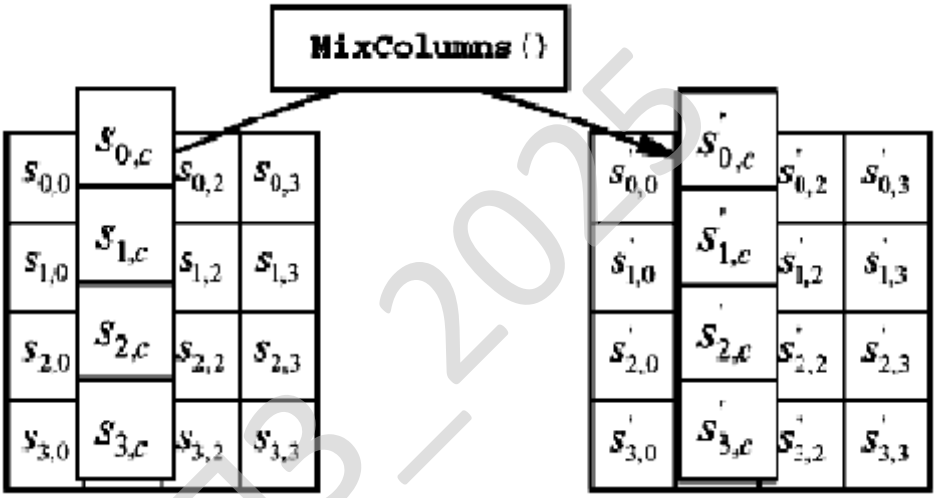

Рисунок 3.8 – Операція перемішування діє на стовпці стану

У зворотному перетворенні InvMixColumns стовпці стану розглядаються як багаточлени над $GF(2^8)$, але, природно, піддаються зворотному перетворенню.

Додавання раундового ключа AddRoundKey()

У даній операції раундовий ключ додається до стану за допомогою поразрядного XOR. Довжина ключа (в 32-розрядних словах) дорівнює довжині блоку Nb . Рисунок 3.9 ілюструє дію даної операції на стан. Це перетворення позначається як AddRoundKey(State, RoundKey).

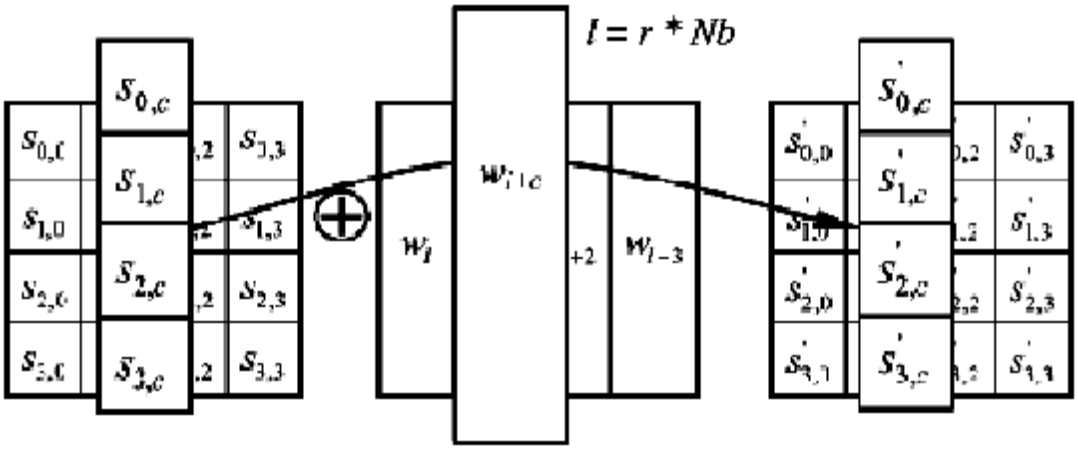

Рисунок 3.9 – AddRoundKey

На рисунку 3.10 зображена функціональна схема системи. Нижче розглянемо її більш докладно.

Функціональна схема складається з наступних блоків:

– Головне вікно програми.

– Блок розмежування доступу.

– Блок менеджеру паролів.

– Блок журналювання подій.

– Блоки шифрування та дешифрування інформації згідно алгоритму AES з використанням спеціальних інструкцій Intel Core i7.

– Допомога.

Розглянемо ці блоки більш детально.

Головне вікно програми

Головне вікно призначене для швидкого доступу до основних функцій програми й меню. Програма складається з головного вікна, розташованого у верхній частині екрана й набору незалежних дочірніх вікон. Розташування й розміри вікон можна змінювати за допомогою миші. Також існує можливість закрити непотрібні дочірні вікна (знову відобразити їх можна шляхом вибору відповідних пунктів у меню натисканням на аналогічні кнопки в головному вікні програми). Всі зроблені зміни збережуться в наступному сеансі роботи. Призначення всіх кнопок у програмі пояснюється спливаючими

підказками: підведіть покажчик миші до будь-якої кнопки й затримаєте його – з'явиться спливаюча підказка із призначенням кнопки. Головне меню надає доступ до основних списків і функцій системи.

Блок розмежування доступу

Призначений для організації безпечного доступу співтовариства користувачів до захищених ресурсів. Члени цього співтовариства, використовуючи програму, одержують визначені переваги. Це дає наступні можливості:

– Надавати користувачам доступ до інформації (наприклад, групи структурних схем, адресні довідники відділів або пошук співробітників) і ресурсам (наприклад, устаткування або облікові записи у внутрішніх системах), у яких вони бідують, буквально з першого дня.

– Синхронізувати кілька паролів з одним ім'ям користувача для всіх систем.

– При необхідності оперативно змінювати або відзивати права на доступ (наприклад, при переході співробітника в іншу групу або при звільненні).

– Підтримувати відповідність урядовим постановам.

У цей блок включені наступні можливості.

Ролі, що дозволяють виконувати наступні дії:

– запитувати призначення ролей і управляти процесом підтвердження запитів на призначення ролей;

– перевіряти стан Ваших запитів ролей;

– визначати ролі і їхні взаємини;

– визначати обмеження поділу обов'язків (SoD) і управляти процесом підтвердження у випадках, коли користувач запитує перевизначення обмеження;

– переглядати довідник ролей;

– переглядати докладні звіти, у яких перераховані ролі й обмеження поділу обов'язків, визначені в довіднику, а також поточний стан призначення ролей, виключення поділу обов'язків і повноваження користувача.

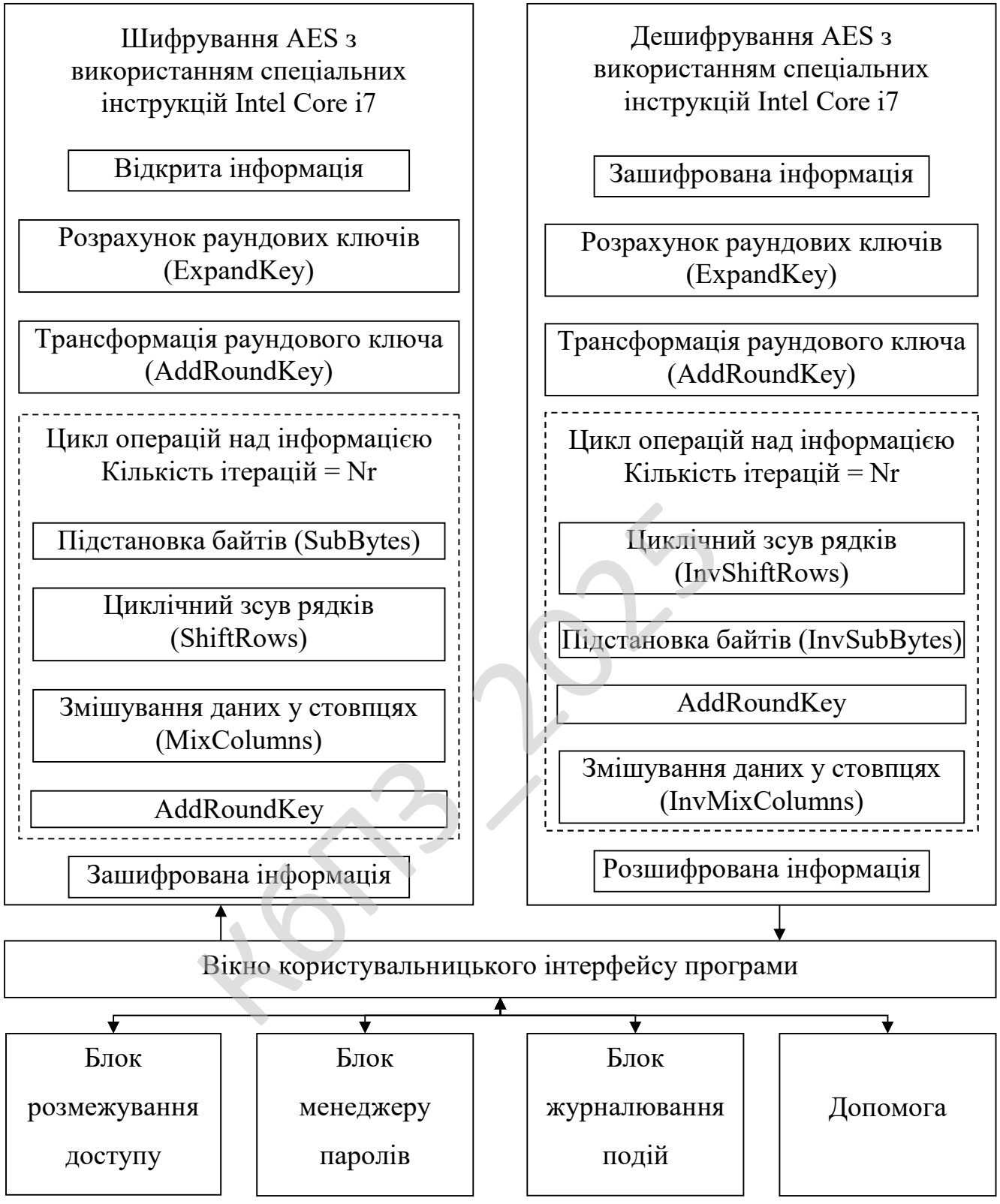

Рисунок 3.10 – Функціональна схема системи

Модуль "Дотримання" дозволяє:

– Запитувати підтвердження профілю користувача.

– Запитувати підтвердження поділу обов'язків (SoD).

– Запитувати підтвердження призначення функцій.

– Запитувати підтвердження призначення користувача.

Самообслуговування облікового запису, що дозволяє:

– відображати структурні схеми;

– повідомляти про додатки, пов'язані з користувачем, для адміністратора;

– змінювати дані профілю;

– виконувати пошук у каталозі;

– змінювати пароль, відповідь на запит-відповідь пароля і його підказку;

– переглядати стан політики й синхронізації пароля;

– створювати облікові записи для нових користувачів і груп (при наявності відповідних повноважень).

Запити й твердження, що дозволяють:

– запитувати ресурси;

– перевіряти підтвердження запитів на ресурси;

– працювати із призначеними завданнями підтвердження інших запитів на ресурси;

– виконувати запити й твердження в якості чиєїсь довіреної особи або делегата;

– призначати кого-небудь ще довіреною особою або делегатом (при наявності відповідних повноважень);

– управляти всіма цими функціями запитів і підтверджень в інтересах Вашої групи (при наявності відповідних повноважень);

– при необхідності для кожного запиту або підтвердження надавати цифровий підпис.

Блок менеджеру паролів

Надає можливості не тільки для простого збереження паролів, але й для повноцінної роботи з ними. Програма підтримує роботу з декількома аккаунтами, і працювати з нею можуть трохи користувачів. При цьому бази даних кожного користувача шифруються.

Додаткові можливості:

– Система пошуку по базі даних.

– Підтримка макросів.

– Можливість резервного копіювання бази даних.

– Можливість швидкого перемикання між користувачами.

– Швидкий доступ до часто використовуваних функцій.

– Генератор паролів.

– Можливість друку паролів.

Допомога

Блок призначений про надання допомоги по роботі з системою, а також для надання інформації про розробників системи, версію та дату випуску.

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

Блок журналювання подій

Призначений для запису у журнал усіх подій, які відбуваються у системі. Журнал дій користувачів містить форму для запуску архівації журналу. Форма архівації являє собою кнопку "Очистити журнал" і поле з датою "по:". Дату можна встановлювати будь-яку, але не раніше, ніж поточна дата мінус 1 місяць, щоб у системі завжди зберігалися дані про дії користувачів як мінімум за місяць.

Після натискання кнопки "Очистити журнал" у Системі генерується текстовий файл із архівом журналу за обраний період. Файл зберігається в зашифрованому виді, а посилання на цей файл показуються адміністраторові. Після створення файлу запису журналу за обраний період

віддаляються з бази даних. У випадку помилки при створенні або збереженні файлу, записи не видаляються.

Блоки шифрування та дешифрування інформації згідно алгоритму AES з використанням спеціальних інструкцій Intel Core i7

Призначені для шифрування та дешифрування інформації, до якої користувач має доступ, згідно прав доступу.

При реалізації шифрування та дешифрування виконуються такі основні операції:

– Key Expansion – процедура використовувана для генерації Round Keys з Cipher Key.

– Cipher Key – секретний, криптографічний ключ, що використовується Key Expansion процедурою, щоб зробити набір ключів для раундів (Round Keys); може бути представлений як прямокутний масив байтів, що має чотири рядки й Nk колонок.

– Round Key – Round Keys виходять із Cipher Key використовуючи процедуру Key Expansion. Вони застосовуються до State при шифруванні й розшифруванні.

– State – проміжний результат шифрування, що може бути представлений як прямокутний масив байтів що має 4 рядки й Nb колонок.

– AddRoundKey() – трансформація при шифруванні й зворотному шифруванні, при якій Round Key XOR’иться с State. Довжина RoundKey дорівнює розміру State (тобто, якщо Nb = 4, то довжина RoundKey дорівнює 128 біт або 16 байт).

– SubBytes() – трансформації при шифруванні які обробляють State використовуючи нелінійну таблицю заміщення байтів (S-box), застосовуючи її незалежно до кожного байта State.

– ShiftRows() – трансформації при шифруванні, які обробляють State, циклічно зміщаючи останні три рядки State на різні величини.

– MixColumns() – трансформація при шифруванні яка бере всі стовпці State і змішує їх дані (незалежно друг від друга), щоб одержати нові стовпці.

– InvShiftRows() – трансформація при розшифруванні яка є зворотною стосовно ShiftRows().

– InvSubBytes() – трансформація при розшифруванні яка є зворотною стосовно SubBytes().

– InvMixColumns() – трансформація при розшифруванні яка є зворотною стосовно MixColumns().

– RotWord() – функція, що використовується в процедурі Key Expansion, що бере 4-х байтне слово й робить над ним циклічну перестановку.

– SubWord() – функція, використовувана в процедурі Key Expansion, що бере на вході 4-х байтне слово й застосовуючи S-box до кожного із чотирьох байтів видає вихідне слово.

– Block – послідовність біт, з яких складається input, output, State і Round Key. Також під Block можна розуміти послідовність байт.

– Ciphertext – вихідні дані алгоритму шифрування.

– S-box – нелінійна таблиця замін, що використовується в декількох трансформаціях заміни байт і в процедурі Key Expansion для взаємнооднозначної заміни значення байта.

– Nb – число стовпців(32-ух бітних слів), що становлять State. Для, AES Nb = 4.

– Nk – число 32-ух бітних слів, що становлять шифроключ. Для AES, Nk = 4,6, або 8.

– Nr – число раундів, що є функцією Nk і Nb. Для AES, Nr = 10, 12, 14.

– Rcon[] – масив, що складається з бітів 32-х розрядного слова і є постійним для даного раунду.

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 3.11. Після початку роботи розробленого ПЗ ми потрапляємо до головного блоку системи звідки через ланку дій відбувається наступне:

– Інтерфейс ПЗ.

– Дешифрування з використанням інструкцій Intel Core i7.

– Відкриття файлу для дешифрування.

– Виведення інформації про файл до дешифрування.

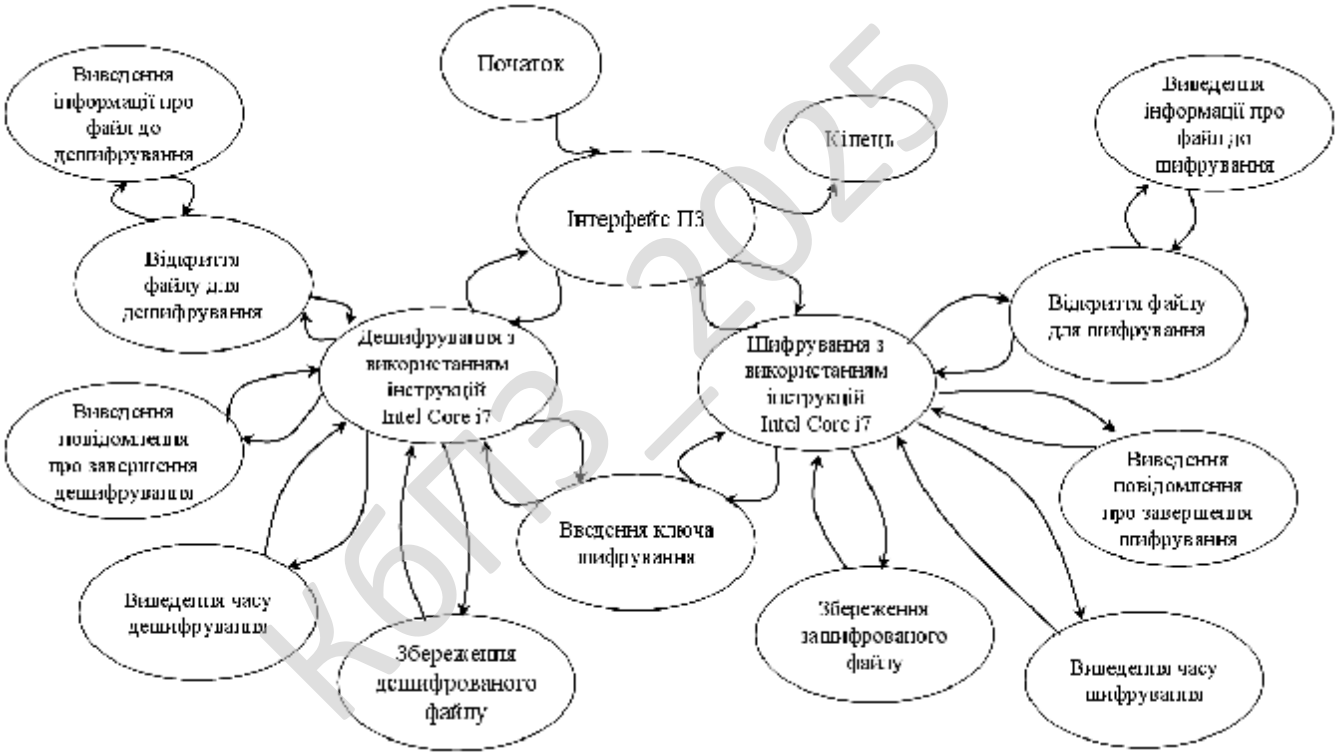

Рисунок 3.11 – Діаграма взаємодії процесів

– Виведення повідомлення про завершення дешифрування.

– Виведення часу дешифрування.

– Збереження дешифрованого файлу.

– Введення ключа шифрування.

– Шифрування з використанням інструкцій Intel Core i7.

– Збереження зашифрованого файлу.

– Виведення часу шифрування.

– Виведення повідомлення про завершення шифрування.

– Відкриття файлу для шифрування.

– Виведення інформації про файл до шифрування.

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. 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.

Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем я враховував, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю захисту конфіденційності інформації з використанням інструкцій Intel Core i7.

При складанні блок-схем програмного забезпечення і напрацювання алгоритмів я зіткнувся з масою проблем, які вимагали напрацювання процедур і функцій над основною проблематикою. Для чого були створені додаткові класи, типи даних і константи, що забезпечило вирішення проблем.

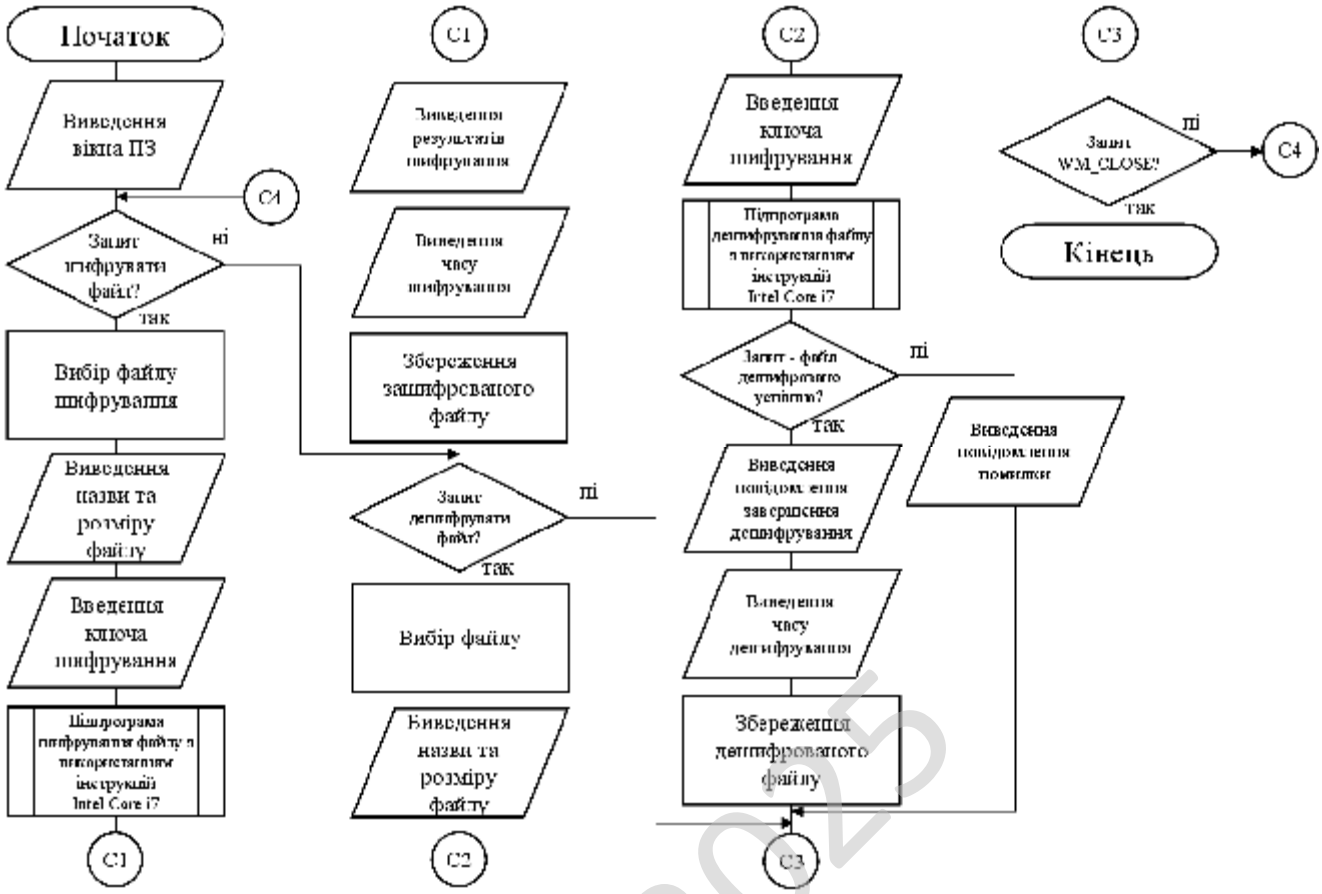

Рисунок 4.1 – Блок схема основної програми

Опис алгоритмів функціонування системи

При розробці використовувались концепції діаграм діяльності. Тобто в UML, візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій.

Це фундаментальна одиниця визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів. Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії

мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності. Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

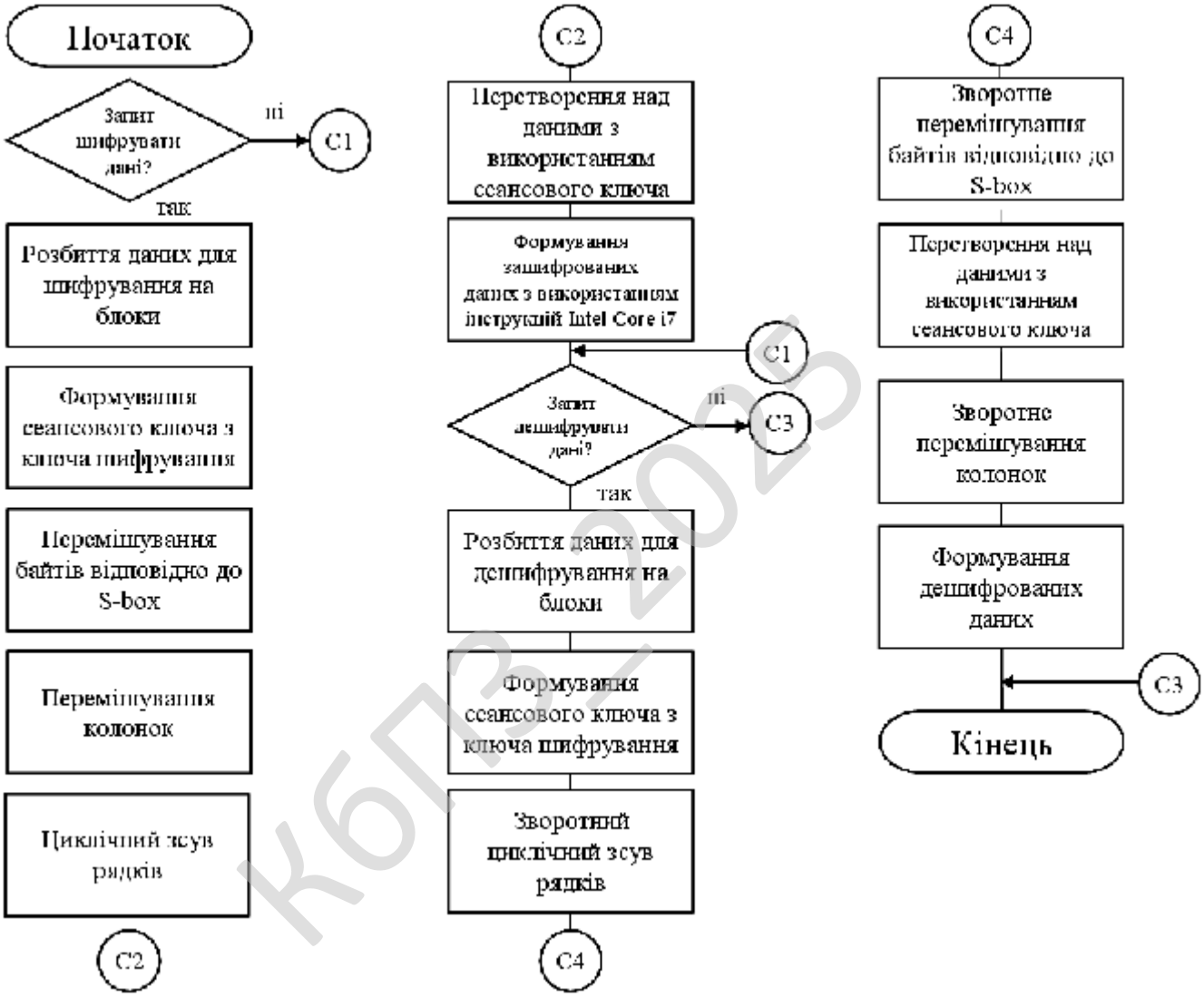

Рисунок 4.2 – Блок схема підпрограми

Основна ідея проєкту

Система захисту конфіденційності інформації будується на принципах шифрування даних та оптимізації швидкодії за допомогою апаратного прискорення, яке забезпечують інструкції Intel Core i7 (наприклад, AES-NI).

Основна мета полягає у забезпеченні конфіденційності даних з мінімальними витратами на обробку і високим рівнем захисту.

Архітектура системи

1. Шифрувальний модуль забезпечує шифрування та дешифрування даних за допомогою алгоритму AES та інших.

2. Модуль керування ключами відповідає за генерацію, зберігання і обробку криптографічних ключів.

3. Модуль автентифікації здійснює перевірку автентичності користувачів.

4. Журнал подій фіксує всі дії у системі з метою виявлення можливих загроз.

5. Інтерфейс користувача надає зручний доступ до функцій системи.

Алгоритм шифрування з використанням AES-NI (Код системи на Python):

```
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes
import base64
import hashlib
import os

class SecuritySystem:
    def __init__(self):
        self.key = None
        self.block_size = 16

    def generate_key(self, password):
        # Генеруємо ключ з пароля користувача
        self.key = hashlib.sha256(password.encode()).digest()

    def encrypt_data(self, plaintext):
        # Генеруємо випадковий вектор ініціалізації
        iv = get_random_bytes(self.block_size)

        # Створюємо об'єкт шифрування AES
        cipher = AES.new(self.key, AES.MODE_CFB, iv)

        # Шифруємо дані
```

```
            encrypted_data = iv + cipher.encrypt(plaintext.encode())

            # Повертаємо зашифровані дані у вигляді base64
            return base64.b64encode(encrypted_data).decode()

        def decrypt_data(self, encrypted_data):
            # Декодуємо дані з base64
            encrypted_data = base64.b64decode(encrypted_data)

            # Витягуємо вектор ініціалізації
            iv = encrypted_data[:self.block_size]

            # Створюємо об'єкт дешифрування AES
            cipher = AES.new(self.key, AES.MODE_CFB, iv)

            # Дешифруємо дані
            decrypted_data = cipher.decrypt(encrypted_data[self.block_size:])

            return decrypted_data.decode()

        def authenticate_user(self, username, password):
            # Автентифікація користувача
            return username == "admin" and password == "securepassword"

    def main():
        # Ініціалізація системи
        system = SecuritySystem()

        # Генеруємо ключ
        system.generate_key("securepassword")

        # Демонстрація шифрування
        plaintext = "Це тестове повідомлення"
        encrypted_text = system.encrypt_data(plaintext)
        print(f"Зашифровані дані: {encrypted_text}")

        # Декодування
        decrypted_text = system.decrypt_data(encrypted_text)
        print(f"Дешифровані дані: {decrypted_text}")

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

Основні функції системи:

1. Модуль генерації ключів забезпечує генерацію криптографічного ключа на основі пароля користувача.

2. Модуль шифрування та дешифрування виконує шифрування тексту за допомогою AES з використанням вектора ініціалізації для забезпечення унікальності шифрованих даних.

3. Модуль автентифікації перевіряє, чи має користувач доступ до системи.

4. Журнал подій фіксує всі шифрувальні операції та спроби входу.

Розрахунки і обґрунтування

Використання AES-NI на процесорах Intel Core i7 дозволяє зменшити час обробки даних до 50% порівняно з програмною реалізацією AES.

Пропускна здатність системи шифрування становить до 1 ГБ/с залежно від розміру даних.

Алгоритм AES забезпечує 128-бітний рівень захисту, що є стійким до сучасних атак.

Підтвердження правильності проектних рішень

Під час тестування системи на файлі розміром 100 МБ вимірюється час шифрування та дешифрування даних з використанням апаратного прискорення AES-NI та без нього.

Час шифрування без апаратного прискорення становить 8.5 секунд, тоді як з AES-NI цей процес виконується за 1.2 секунди. Це дозволяє зменшити час шифрування майже в 7 разів. Аналогічна ситуація спостерігається і при дешифруванні. Без апаратного прискорення дешифрування займає 8.4 секунди, а з використанням AES-NI – лише 1.1 секунди.

Розмір зашифрованих даних практично не збільшується, оскільки алгоритм AES додає лише 16 байт вектора ініціалізації до початку даних. Для файлів розміром до 100 МБ це не має значного впливу на загальний розмір.

Система використовує 256-бітний ключ для шифрування. Сучасні суперкомп'ютери не здатні підібрати ключ у розумний час, оскільки кількість

можливих комбінацій є астрономічно великою. Навіть якщо суперкомп'ютер перевіряє мільярд мільярдів ключів за секунду, на підбір одного ключа знадобиться більше часу, ніж вік Всесвіту.

Завдяки використанню інструкцій Intel AES-NI продуктивність системи значно зростає, що дозволяє застосовувати її у високонавантажених середовищах без втрати ефективності.

Розглянемо алгоритм шифрування AES. Він розшифровується як "Advanced Encryption Standard" – це найбільш популярний стандарт симетричного шифрування у світі IT.

Стандарт працює із блоками розміром 128 біт і підтримує 128-, 192- або 256-бітні ключі (AES-128, AES-192 і AES-256). Багато утиліт шифрування, та ж TrueCrypt, підтримали алгоритм AES на самому початку його існування. Але найбільший фактор успіху AES, звичайно, полягає в його прийнятті урядом США в 2002 році, при цьому в 2003 році він був прийнятий як стандарт для захисту секретних даних.

Надійність шифрування AES

Останнім часом іде чимало обговорень так званих взламів, які обходять необхідність запуску розширеного пошуку методом грубої сили для знаходження правильного ключа розшифровки. Технології, такі як атаки XSL і атаки related-key обговорюються досить інтенсивно – але успіх невеликий.

Єдиний працюючий спосіб злому шифрування AES полягає в так званій атаці побічного каналу (side-channel). Для її здійснення атака повинна відбуватися тільки на host-системі, на якій виконується шифрування AES, і при цьому вам необхідно знайти спосіб одержання інформації про синхронізацію кеша. У такому випадку можна відстежити число тактів комп'ютера до завершення процесу шифрування. Звичайно, все це не так легко, оскільки вам потрібен доступ до комп'ютера, причому досить повний доступ для аналізу шифрування й права на виконання коду. Тепер вам напевно зрозуміло, чому "діри" у системі безпеки, які дозволяють зловмисникові одержати такі права, нехай навіть вони звучать зовсім

абсурдно, необхідно закривати якнайшвидше. Але якщо ви одержите доступ до цільового комп'ютера, то добування ключа AES – справа часу, тобто вже не трудомістке завдання для суперкомп'ютерів, що вимагає величезних обчислювальних ресурсів.

AES усередині Intel

На даний момент інтегровані в CPU інструкції AES починають мати сенс – незалежно від можливих переваг по продуктивності. З погляду безпеки процесор може обробляти інструкції AES в інкапсульованому виді, тобто йому не потрібні які-небудь таблиці перетворення, необхідні для атаки методом побічного каналу.

Процесори насправді знаменують собою зміну поколінь, оскільки при цьому не тільки відбувається перехід на наступний техпроцес (32 нм у порівнянні з 45 нм), але перед нами й перше покоління CPU з підтримкою декількох інструкцій, що прискорюють шифрування. Intel згадує добавку як AES New Instructions. Вони складаються із чотирьох інструкцій для шифрування AES (AESENC, AESENCLAST) і розшифровки (AESDEC, AESDECLAST) плюс ще дві інструкції для роботи із ключем AES (AESIMC, AESKEYGENASSIST). Як і раніше, інструкції відносяться до SIMD, тобто до типу "одна інструкція багато даних" (Single Instruction Multiple Data). Підтримуються всі три ключі AES (128, 192 і 256 бітів з 10, 12 і 14 проходами підстановки й перестановки). Оскільки всі інструкції AES мають фіксовану затримку, що не залежить від даних, тобто час фіксоване й доступ до пам'яті не потрібно. Крім того, модель програмування така ж, як і у випадку інших інструкцій SSE з первісного стандарту SSE4. Таким чином, всі операційні системи, які підтримують роботу з SSE, зможуть використовувати й інструкції AES New Instructions. Будьте обережні при виборі процесора із прискоренням AES, оскільки сьогодні лише деякі моделі підтримують нові інструкції. 32-нм процесори Core i3 на Clarkdale не підтримують інструкції, а дводерна лінійка Core i 5-600 – підтримує. У мобільних процесорів ситуація ледве більше складна: якщо мобільні процесори Core i3 теж

не підтримують прискорення AES, те процесори лінійки Core і 5-500 уже підтримують.

Однак є одна модель Core і 5-400, позбавлена такої підтримки. Усе було б набагато простіше, якби Intel додала підтримку нових інструкцій в усі моделі.

Шифрування даних за допомогою AES

Шифрування AES базується на системі підстановок з перестановкою, тобто над даними проводиться серія математичних операцій, щоб створити значно модифікований масив даних (зашифрований). Як вихідна інформація виступає текст, а ключ відповідає за виконання математичних операцій. Операції можуть бути як зовсім простими, наприклад, зрушення бітів або XOR, так і більше складними. Один прохід можна легко розшифрувати, тому всі сучасні алгоритми шифрування побудовані на декількох проходах. У випадку AES це 10, 12 або 14 проходів для AES-128, AES-192 або AES-256. До речі, ключі AES проходять таку ж процедуру, що й користувальницькі дані, тобто вони являють собою що змінюється раундовий ключ. Процес працює з масивами 4x4 з одиночних байтів, також називаних боксами: S-box використовуються для підстановок, P-box – для перестановок. Підстановки й перестановки виконуються на різних етапах: підстановки працюють усередині так званих боксів, а перестановки міняють інформацію між боксами. S-box працює по складному принципі, тобто навіть якщо єдиний вхідний біт буде мінятися, то це вплине на декілька вихідних бітів, тобто властивості кожного вихідного біта залежать від кожного вхідного біта. Використання декількох проходів забезпечує гарний рівень шифрування, при цьому необхідно відповідати критеріям розсіювання (diffusion) і заплутування (confusion). Розсіювання виконується через каскадну комбінацію трансформацій S-box і P-box: при зміні тільки одного біта у вхідному тексті S-box буде модифікувати вихід декількох біт, а P-box буде псевдовипадково поширювати цей ефект по декількох S-box. Коли ми говоримо про те, що мінімальна зміна на вході дає максимальна зміна на виході, ми говоримо про ефект сніжної грудки.

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 В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

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

– Блок меню: Файл; Шифрування; Дешифрування; Довідка.

Блок кнопок швидкого доступу до елементів програми: Перегляд файлу; Ключ з файлу; Шифрування; Дешифрування.

Блок виведення даних: Розмір файлу; Статус; Час шифрування; Час дешифрування.

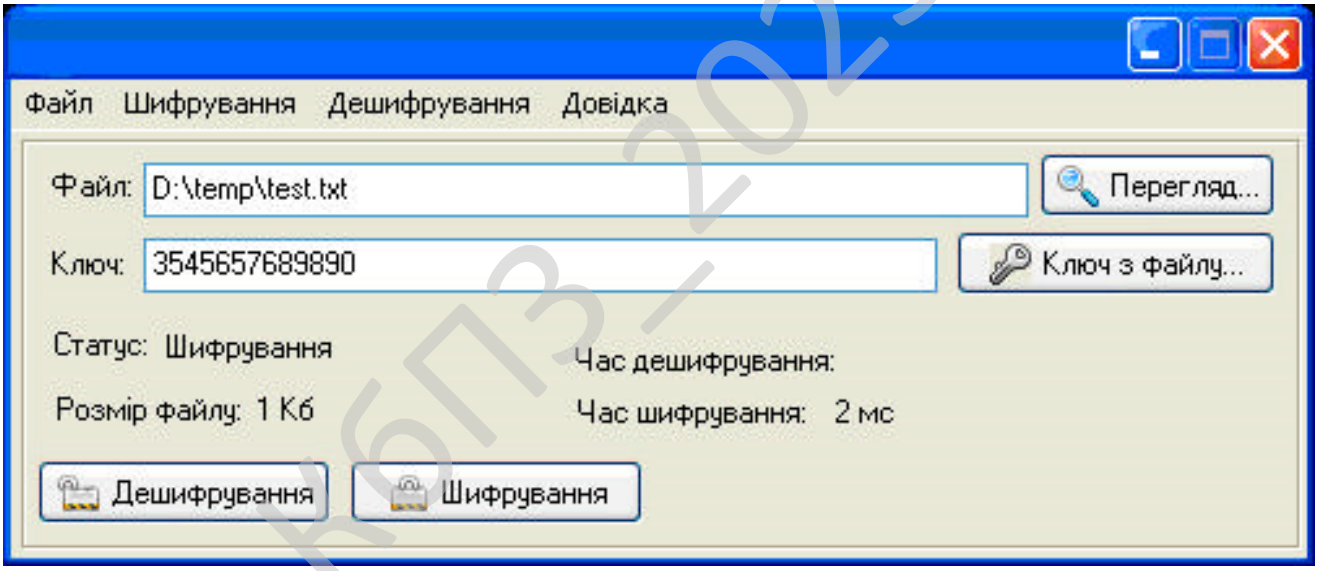

Рисунок 5.1 – Головне вікно програми

Розроблена програма має дуже простий і інтуїтивно зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий. На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

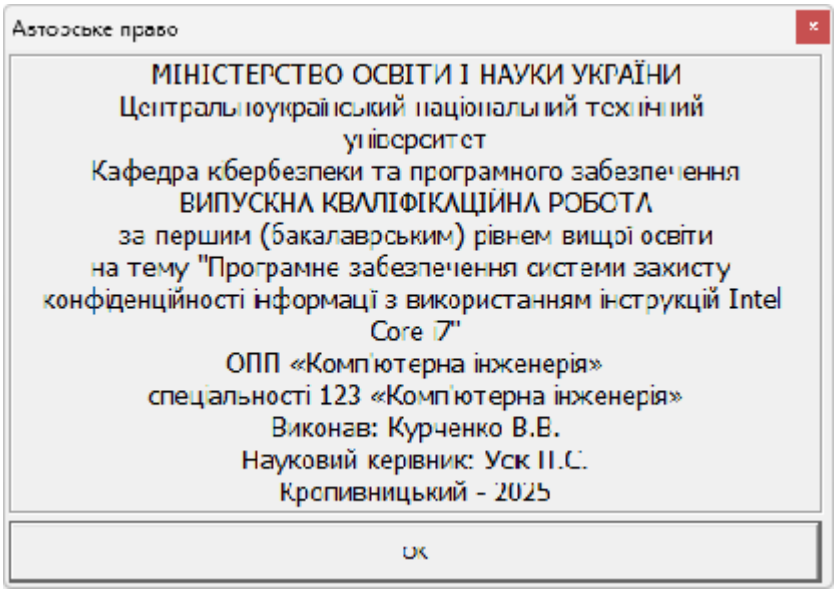

Рисунок 5.2 – Авторське право

Обрано умови розповсюдження – proprietary software. Програмне забезпечення, на яке зберігаються як немайнові, так і майнові авторські права. Отримавши або придбавши таке програмне забезпечення, користувач отримує обмежені права користування ним: може бути заборонено або закрито доступ до коду (вивчення), внесення змін, тиражування, розповсюдження та перепродаж. Програмне забезпечення вважається власницьким, якщо наявне хоча б одне з перелічених обмежень. Власницьке програмне забезпечення та комерційне програмне забезпечення не є синонімами – власницьким може бути і безплатне (тобто, некомерційне) програмне забезпечення. На противагу власницькому ПЗ існує вільне програмне забезпечення, автори і власники якого дозволяють вивчати, модифікувати і поширювати свій продукт. Саме визначення власницького програмного забезпечення виникло в результаті діяльності громадського руху вільного програмного забезпечення (представленого Фондом вільного програмного забезпечення та іншими організаціями) і осмислення умов свободи користування програмами. Сама назва власницьке ПЗ підкреслює визначальне значення власника у способі використання і можливостях розвитку цього програмного забезпечення.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи захисту конфіденційності інформації з використанням інструкцій Intel Core i7.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем захисту конфіденційності інформації з використанням інструкцій Intel Core i7.

– Досліджена система захисту конфіденційності інформації з використанням інструкцій Intel Core i7.

– На основі отриманих результатів досліджень створена програмна реалізація системи захисту конфіденційності інформації з використанням інструкцій Intel Core i7.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання захисту конфіденційності інформації з використанням інструкцій Intel Core i7.

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 захисту конфіденційності інформації з використанням інструкцій Intel Core i7. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм 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. Derek Fisher. Application Security Program Handbook. Manning Publications. 2021. 155 p.

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

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

4. 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.

5. 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.

6. 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.

7. 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.

8. 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.

9. 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.

10. 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.

11. 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.

12. 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.

13. 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,

14. 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.

15. 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.

16. 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

17. 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.

18. 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.

19. 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.*

20. 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.

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

22. 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.

23. 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.

24. 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.

25. 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.

26. 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.

27. 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.

28. 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.

29. 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.

30. 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.

31. 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.

32. 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.

33. 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.

34. 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.

35. 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.

36. 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.

37. 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.

38. 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.

39. 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.

40. 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.

41. 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.

42. 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.

43. 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.

44. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in

Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 353-358.

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

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

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

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

49. 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.

50. 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.

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи захисту конфіденційності інформації з використанням інструкцій Intel Core i7.

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. № 46-02 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 захисту конфіденційності інформації з використанням інструкцій Intel Core i7.

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 з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи захисту конфіденційності інформації з використанням інструкцій Intel Core i7;

– 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 забезпечення.

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 – 75 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 2.06.2025 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

_____ Усік П.С.

Програмне забезпечення системи захисту конфіденційності інформації з використанням інструкцій Intel Core i7

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 28

Літера: РП

Кропивницький – 2025 року

Основна програма

```
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes
import base64
import hashlib
import os
import time

class SecuritySystem:
    def __init__(self):

        # Ініціалізація змінних системи
        self.key = None
        self.block_size = 16
        self.log_file = "system_log.txt"

    def generate_key(self, password):

        # Генеруємо ключ із заданого пароля користувача
        self.key = hashlib.sha256(password.encode()).digest()

    def encrypt_data(self, plaintext):

        # Генеруємо випадковий вектор ініціалізації (IV)
        iv = get_random_bytes(self.block_size)

        # Створюємо об'єкт шифрування AES у режимі CFB
        cipher = AES.new(self.key, AES.MODE_CFB, iv)

        # Шифруємо дані
        encrypted_data = iv + cipher.encrypt(plaintext.encode())

        # Конвертуємо зашифровані дані у base64
        encrypted_base64 = base64.b64encode(encrypted_data).decode()

        # Повертаємо зашифровані дані
        return encrypted_base64

    def decrypt_data(self, encrypted_data):

        # Декодуємо дані з формату base64
        encrypted_data = base64.b64decode(encrypted_data)

        # Витягуємо вектор ініціалізації
        iv = encrypted_data[:self.block_size]

        # Створюємо об'єкт дешифрування AES
        cipher = AES.new(self.key, AES.MODE_CFB, iv)

        # Дешифруємо дані
        decrypted_data = cipher.decrypt(encrypted_data[self.block_size:])

        # Повертаємо дешифровані дані як строку
        return decrypted_data.decode()

    def authenticate_user(self, username, password):

        # Перевірка автентичності користувача
        if username == "admin" and password == "securepassword":
            self.log_event(f"User '{username}' authenticated successfully")
            return True
        else:
```

```
            self.log_event(f"Failed authentication attempt for user
'{username}'")
            return False

    def log_event(self, message):

        # Логування подій у файл
        with open(self.log_file, "a") as log:
            timestamp = time.strftime("%Y-%m-%d %H:%M:%S", time.localtime())
            log.write(f"[{timestamp}] {message}\n")

    def save_encrypted_file(self, filename, data):

        # Збереження зашифрованих даних у файл
        with open(filename, "w") as file:
            file.write(data)

    def read_encrypted_file(self, filename):

        # Читання зашифрованих даних з файлу
        if os.path.exists(filename):
            with open(filename, "r") as file:
                return file.read()
        else:
            return None

    def encrypt_file(self, input_file, output_file):

        # Шифруємо файл
        if os.path.exists(input_file):
            with open(input_file, "r") as file:
                plaintext = file.read()
                encrypted_data = self.encrypt_data(plaintext)
                self.save_encrypted_file(output_file, encrypted_data)
                self.log_event(f"File '{input_file}' encrypted successfully")
        else:
            self.log_event(f"Failed to encrypt file '{input_file}': file not
found")

    def decrypt_file(self, input_file, output_file):

        # Дешифруємо файл
        encrypted_data = self.read_encrypted_file(input_file)
        if encrypted_data:
            decrypted_data = self.decrypt_data(encrypted_data)
            with open(output_file, "w") as file:
                file.write(decrypted_data)
            self.log_event(f"File '{input_file}' decrypted successfully")
        else:
            self.log_event(f"Failed to decrypt file '{input_file}': file not
found or empty")

    def run_system(self):

        # Основний цикл роботи системи
        while True:
            print("1. Authenticate User")
            print("2. Encrypt Text")
            print("3. Decrypt Text")
            print("4. Encrypt File")
            print("5. Decrypt File")
            print("6. Exit")
```

```
            choice = input("Select an option: ")

            if choice == "1":
                username = input("Enter username: ")
                password = input("Enter password: ")
                if self.authenticate_user(username, password):
                    print("Authentication successful")
                else:
                    print("Authentication failed")

            elif choice == "2":
                plaintext = input("Enter text to encrypt: ")
                encrypted = self.encrypt_data(plaintext)
                print(f"Encrypted text: {encrypted}")

            elif choice == "3":
                encrypted = input("Enter text to decrypt: ")
                try:
                    decrypted = self.decrypt_data(encrypted)
                    print(f"Decrypted text: {decrypted}")
                except Exception as e:
                    print(f"Error during decryption: {e}")
            elif choice == "4":
                input_file = input("Enter input file name: ")
                output_file = input("Enter output file name: ")
                self.encrypt_file(input_file, output_file)
                print(f"File '{input_file}' encrypted and saved as
'{output_file}'")

            elif choice == "5":
                input_file = input("Enter input file name: ")
                output_file = input("Enter output file name: ")
                self.decrypt_file(input_file, output_file)
                print(f"File '{input_file}' decrypted and saved as
'{output_file}'")

            elif choice == "6":
                print("Exiting system...")
                break

            else:
                print("Invalid choice. Please try again.")
```

Файл d_crypto71.py

```
import calendar
import datetime
import functools
import sys
import typing
import warnings
from base64 import b16encode
from collections.abc import Iterable, Sequence
from functools import partial
from typing import (
    Any,
    Callable,
    Union,
)

if sys.version_info >= (3, 13):
    from warnings import deprecated
elif sys.version_info < (3, 8):
    _T = typing.TypeVar("T")

    def deprecated(msg: str, **kwargs: object) -> Callable[[_T], _T]:
        return lambda f: f
else:
    from typing_extensions import deprecated

from cryptography import utils, x509
from cryptography.hazmat.primitives.asymmetric import (
    dsa,
    ec,
    ed448,
    ed25519,
    rsa,
)

from OpenSSL._util import StrOrBytesPath
from OpenSSL._util import (
    byte_string as _byte_string,
)
from OpenSSL._util import (
    exception_from_error_queue as _exception_from_error_queue,
)
from OpenSSL._util import (
    ffi as _ffi,
)
from OpenSSL._util import (
    lib as _lib,
)
from OpenSSL._util import (
    make_assert as _make_assert,
)
from OpenSSL._util import (
    path_bytes as _path_bytes,
)

__all__ = [
    "FILETYPE_ASN1",
    "FILETYPE_PEM",
    "FILETYPE_TEXT",
    "TYPE_DSA",
    "TYPE_RSA",
    "X509",
```

```
    "Error",
    "PKey",
    "X509Extension",
    "X509Name",
    "X509Req",
    "X509Store",
    "X509StoreContext",
    "X509StoreContextError",
    "X509StoreFlags",
    "dump_certificate",
    "dump_certificate_request",
    "dump_privatekey",
    "dump_publickey",
    "get_elliptic_curve",
    "get_elliptic_curves",
    "load_certificate",
    "load_certificate_request",
    "load_privatekey",
    "load_publickey",
]


_PrivateKey = Union[
    dsa.DSAPrivateKey,
    ec.EllipticCurvePrivateKey,
    ed25519.Ed25519PrivateKey,
    ed448.Ed448PrivateKey,
    rsa.RSAPrivateKey,
]
_PublicKey = Union[
    dsa.DSAPublicKey,
    ec.EllipticCurvePublicKey,
    ed25519.Ed25519PublicKey,
    ed448.Ed448PublicKey,
    rsa.RSAPublicKey,
]
_Key = Union[_PrivateKey, _PublicKey]
PassphraseCallableT = Union[bytes, Callable[..., bytes]]


FILETYPE_PEM: int = _lib.SSL_FILETYPE_PEM
FILETYPE_ASN1: int = _lib.SSL_FILETYPE_ASN1

FILETYPE_TEXT = 2**16 - 1

TYPE_RSA: int = _lib.EVP_PKEY_RSA
TYPE_DSA: int = _lib.EVP_PKEY_DSA
TYPE_DH: int = _lib.EVP_PKEY_DH
TYPE_EC: int = _lib.EVP_PKEY_EC


class Error(Exception):


_raise_current_error = partial(_exception_from_error_queue, Error)
_openssl_assert = _make_assert(Error)


def _new_mem_buf(buffer: bytes | None = None) -> Any:
    if buffer is None:
        bio = _lib.BIO_new(_lib.BIO_s_mem())
        free = _lib.BIO_free
    else:
```

```
        data = _ffi.new("char[]", buffer)
        bio = _lib.BIO_new_mem_buf(data, len(buffer))

        def free(bio: Any, ref: Any = data) -> Any:
            return _lib.BIO_free(bio)

    _openssl_assert(bio != _ffi.NULL)

    bio = _ffi.gc(bio, free)
    return bio


def _bio_to_string(bio: Any) -> bytes:
    result_buffer = _ffi.new("char**")
    buffer_length = _lib.BIO_get_mem_data(bio, result_buffer)
    return _ffi.buffer(result_buffer[0], buffer_length)[:]


def _set_asn1_time(boundary: Any, when: bytes) -> None:
    if not isinstance(when, bytes):
        raise TypeError("when must be a byte string")
    _openssl_assert(boundary != _ffi.NULL)

    set_result = _lib.ASN1_TIME_set_string(boundary, when)
    if set_result == 0:
        raise ValueError("Invalid string")


def _new_asn1_time(when: bytes) -> Any:
    ret = _lib.ASN1_TIME_new()
    _openssl_assert(ret != _ffi.NULL)
    ret = _ffi.gc(ret, _lib.ASN1_TIME_free)
    _set_asn1_time(ret, when)
    return ret


def _get_asn1_time(timestamp: Any) -> bytes | None:
    string_timestamp = _ffi.cast("ASN1_STRING*", timestamp)
    if _lib.ASN1_STRING_length(string_timestamp) == 0:
        return None
    elif (
        _lib.ASN1_STRING_type(string_timestamp) == _lib.V_ASN1_GENERALIZEDTIME
    ):
        return _ffi.string(_lib.ASN1_STRING_get0_data(string_timestamp))
    else:
        generalized_timestamp = _ffi.new("ASN1_GENERALIZEDTIME**")
        _lib.ASN1_TIME_to_generalizedtime(timestamp, generalized_timestamp)
        _openssl_assert(generalized_timestamp[0] != _ffi.NULL)

        string_timestamp = _ffi.cast("ASN1_STRING*", generalized_timestamp[0])
        string_data = _lib.ASN1_STRING_get0_data(string_timestamp)
        string_result = _ffi.string(string_data)
        _lib.ASN1_GENERALIZEDTIME_free(generalized_timestamp[0])
        return string_result


class _X509NameInvalidator:
    def __init__(self) -> None:
        self._names: list[X509Name] = []

    def add(self, name: X509Name) -> None:
        self._names.append(name)
```

```
    def clear(self) -> None:
        for name in self._names:
            del name._name


class PKey:

    _only_public = False
    _initialized = True

    def __init__(self) -> None:
        pkey = _lib.EVP_PKEY_new()
        self._pkey = _ffi.gc(pkey, _lib.EVP_PKEY_free)
        self._initialized = False

    def to_cryptography_key(self) -> _Key:
        from cryptography.hazmat.primitives.serialization import (
            load_der_private_key,
            load_der_public_key,
        )

        if self._only_public:
            der = dump_publickey(FILETYPE_ASN1, self)
            return typing.cast(_Key, load_der_public_key(der))
        else:
            der = dump_privatekey(FILETYPE_ASN1, self)
            return typing.cast(_Key, load_der_private_key(der, password=None))

    @classmethod
    def from_cryptography_key(cls, crypto_key: _Key) -> PKey:
        if not isinstance(
            crypto_key,
            (
                dsa.DSAPrivateKey,
                dsa.DSAPublicKey,
                ec.EllipticCurvePrivateKey,
                ec.EllipticCurvePublicKey,
                ed25519.Ed25519PrivateKey,
                ed25519.Ed25519PublicKey,
                ed448.Ed448PrivateKey,
                ed448.Ed448PublicKey,
                rsa.RSAPrivateKey,
                rsa.RSAPublicKey,
            ),
        ):
            raise TypeError("Unsupported key type")

        from cryptography.hazmat.primitives.serialization import (
            Encoding,
            NoEncryption,
            PrivateFormat,
            PublicFormat,
        )

        if isinstance(
            crypto_key,
            (
                dsa.DSAPublicKey,
                ec.EllipticCurvePublicKey,
                ed25519.Ed25519PublicKey,
                ed448.Ed448PublicKey,
                rsa.RSAPublicKey,
            ),
```

```
        ):
            return load_publickey(
                FILETYPE_ASN1,
                crypto_key.public_bytes(
                    Encoding.DER, PublicFormat.SubjectPublicKeyInfo
                ),
            )
        else:
            der = crypto_key.private_bytes(
                Encoding.DER, PrivateFormat.PKCS8, NoEncryption()
            )
            return load_privatekey(FILETYPE_ASN1, der)

    def generate_key(self, type: int, bits: int) -> None:
        if not isinstance(type, int):
            raise TypeError("type must be an integer")

        if not isinstance(bits, int):
            raise TypeError("bits must be an integer")

        if type == TYPE_RSA:
            if bits <= 0:
                raise ValueError("Invalid number of bits")

            exponent = _lib.BN_new()
            exponent = _ffi.gc(exponent, _lib.BN_free)
            _lib.BN_set_word(exponent, _lib.RSA_F4)

            rsa = _lib.RSA_new()

            result = _lib.RSA_generate_key_ex(rsa, bits, exponent, _ffi.NULL)
            _openssl_assert(result == 1)

            result = _lib.EVP_PKEY_assign_RSA(self._pkey, rsa)
            _openssl_assert(result == 1)

        elif type == TYPE_DSA:
            dsa = _lib.DSA_new()
            _openssl_assert(dsa != _ffi.NULL)

            dsa = _ffi.gc(dsa, _lib.DSA_free)
            res = _lib.DSA_generate_parameters_ex(
                dsa, bits, _ffi.NULL, 0, _ffi.NULL, _ffi.NULL, _ffi.NULL
            )
            _openssl_assert(res == 1)

            _openssl_assert(_lib.DSA_generate_key(dsa) == 1)
            _openssl_assert(_lib.EVP_PKEY_set1_DSA(self._pkey, dsa) == 1)
        else:
            raise Error("No such key type")

        self._initialized = True

    def check(self) -> bool:
        if self._only_public:
            raise TypeError("public key only")

        if _lib.EVP_PKEY_type(self.type()) != _lib.EVP_PKEY_RSA:
            raise TypeError("Only RSA keys can currently be checked.")

        rsa = _lib.EVP_PKEY_get1_RSA(self._pkey)
        rsa = _ffi.gc(rsa, _lib.RSA_free)
        result = _lib.RSA_check_key(rsa)
```

```
        if result == 1:
            return True
        _raise_current_error()

    def type(self) -> int:
        return _lib.EVP_PKEY_id(self._pkey)

    def bits(self) -> int:
        return _lib.EVP_PKEY_bits(self._pkey)


class _EllipticCurve:

    _curves = None

    def __ne__(self, other: Any) -> bool:
        if isinstance(other, _EllipticCurve):
            return super().__ne__(other)
        return NotImplemented

    @classmethod
    def _load_elliptic_curves(cls, lib: Any) -> set[_EllipticCurve]:
        num_curves = lib.EC_get_builtin_curves(_ffi.NULL, 0)
        builtin_curves = _ffi.new("EC_builtin_curve[]", num_curves)
        lib.EC_get_builtin_curves(builtin_curves, num_curves)
        return set(cls.from_nid(lib, c.nid) for c in builtin_curves)

    @classmethod
    def _get_elliptic_curves(cls, lib: Any) -> set[_EllipticCurve]:
        if cls._curves is None:
            cls._curves = cls._load_elliptic_curves(lib)
        return cls._curves

    @classmethod
    def from_nid(cls, lib: Any, nid: int) -> _EllipticCurve:
        return cls(lib, nid, _ffi.string(lib.OBJ_nid2sn(nid)).decode("ascii"))

    def __init__(self, lib: Any, nid: int, name: str) -> None:
        self._lib = lib
        self._nid = nid
        self.name = name

    def __repr__(self) -> str:
        return f"<Curve {self.name!r}>"

    def _to_EC_KEY(self) -> Any:
        key = self._lib.EC_KEY_new_by_curve_name(self._nid)
        return _ffi.gc(key, _lib.EC_KEY_free)


@deprecated(
    "get_elliptic_curves is deprecated. You should use the APIs in "
    "cryptography instead."
)
def get_elliptic_curves() -> set[_EllipticCurve]:
    return _EllipticCurve._get_elliptic_curves(_lib)


@deprecated(
    "get_elliptic_curve is deprecated. You should use the APIs in "
    "cryptography instead."
)
def get_elliptic_curve(name: str) -> _EllipticCurve:
```

```
    for curve in get_elliptic_curves():
        if curve.name == name:
            return curve
    raise ValueError("unknown curve name", name)


@functools.total_ordering
class X509Name:

    def __init__(self, name: X509Name) -> None:
        name = _lib.X509_NAME_dup(name._name)
        self._name: Any = _ffi.gc(name, _lib.X509_NAME_free)

    def __setattr__(self, name: str, value: Any) -> None:
        if name.startswith("_"):
            return super().__setattr__(name, value)

        if type(name) is not str:
            raise TypeError(
                f"attribute name must be string, not "
                f"'{type(value).__name__:.200}'"
            )

        nid = _lib.OBJ_txt2nid(_byte_string(name))
        if nid == _lib.NID_undef:
            try:
                _raise_current_error()
            except Error:
                pass
            raise AttributeError("No such attribute")

        for i in range(_lib.X509_NAME_entry_count(self._name)):
            ent = _lib.X509_NAME_get_entry(self._name, i)
            ent_obj = _lib.X509_NAME_ENTRY_get_object(ent)
            ent_nid = _lib.OBJ_obj2nid(ent_obj)
            if nid == ent_nid:
                ent = _lib.X509_NAME_delete_entry(self._name, i)
                _lib.X509_NAME_ENTRY_free(ent)
                break

        if isinstance(value, str):
            value = value.encode("utf-8")

        add_result = _lib.X509_NAME_add_entry_by_NID(
            self._name, nid, _lib.MBSTRING_UTF8, value, -1, -1, 0
        )
        if not add_result:
            _raise_current_error()

    def __getattr__(self, name: str) -> str | None:
        nid = _lib.OBJ_txt2nid(_byte_string(name))
        if nid == _lib.NID_undef:
            try:
                _raise_current_error()
            except Error:
                pass
            raise AttributeError("No such attribute")

        entry_index = _lib.X509_NAME_get_index_by_NID(self._name, nid, -1)
        if entry_index == -1:
            return None

        entry = _lib.X509_NAME_get_entry(self._name, entry_index)
```

```
        data = _lib.X509_NAME_ENTRY_get_data(entry)

        result_buffer = _ffi.new("unsigned char**")
        data_length = _lib.ASN1_STRING_to_UTF8(result_buffer, data)
        _openssl_assert(data_length >= 0)

        try:
            result = _ffi.buffer(result_buffer[0], data_length)[:].decode(
                "utf-8"
            )
        finally:
            _lib.OPENSSL_free(result_buffer[0])
        return result

    def __eq__(self, other: Any) -> bool:
        if not isinstance(other, X509Name):
            return NotImplemented

        return _lib.X509_NAME_cmp(self._name, other._name) == 0

    def __lt__(self, other: Any) -> bool:
        if not isinstance(other, X509Name):
            return NotImplemented

        return _lib.X509_NAME_cmp(self._name, other._name) < 0

    def __repr__(self) -> str:
        result_buffer = _ffi.new("char[]", 512)
        format_result = _lib.X509_NAME_oneline(
            self._name, result_buffer, len(result_buffer)
        )
        _openssl_assert(format_result != _ffi.NULL)

        return "<X509Name object '{}'>".format(
            _ffi.string(result_buffer).decode("utf-8"),
        )

    def hash(self) -> int:
        return _lib.X509_NAME_hash(self._name)

    def der(self) -> bytes:
        result_buffer = _ffi.new("unsigned char**")
        encode_result = _lib.i2d_X509_NAME(self._name, result_buffer)
        _openssl_assert(encode_result >= 0)

        string_result = _ffi.buffer(result_buffer[0], encode_result)[:]
        _lib.OPENSSL_free(result_buffer[0])
        return string_result

    def get_components(self) -> list[tuple[bytes, bytes]]:
        result = []
        for i in range(_lib.X509_NAME_entry_count(self._name)):
            ent = _lib.X509_NAME_get_entry(self._name, i)

            fname = _lib.X509_NAME_ENTRY_get_object(ent)
            fval = _lib.X509_NAME_ENTRY_get_data(ent)

            nid = _lib.OBJ_obj2nid(fname)
            name = _lib.OBJ_nid2sn(nid)

            value = _ffi.buffer(
                _lib.ASN1_STRING_get0_data(fval), _lib.ASN1_STRING_length(fval)
            )[:]
```

```
            result.append((_ffi.string(name), value))

        return result


@deprecated(
    "X509Extension support in pyOpenSSL is deprecated. You should use the "
    "APIs in cryptography."
)
class X509Extension:

    def __init__(
        self,
        type_name: bytes,
        critical: bool,
        value: bytes,
        subject: X509 | None = None,
        issuer: X509 | None = None,
    ) -> None:
        ctx = _ffi.new("X509V3_CTX*")

        _lib.X509V3_set_ctx(ctx, _ffi.NULL, _ffi.NULL, _ffi.NULL, _ffi.NULL, 0)

        _lib.X509V3_set_ctx_nodb(ctx)

        if issuer is not None:
            if not isinstance(issuer, X509):
                raise TypeError("issuer must be an X509 instance")
            ctx.issuer_cert = issuer._x509
        if subject is not None:
            if not isinstance(subject, X509):
                raise TypeError("subject must be an X509 instance")
            ctx.subject_cert = subject._x509

        if critical:
            value = b"critical," + value

        extension = _lib.X509V3_EXT_nconf(_ffi.NULL, ctx, type_name, value)
        if extension == _ffi.NULL:
            _raise_current_error()
        self._extension = _ffi.gc(extension, _lib.X509_EXTENSION_free)

    @property
    def _nid(self) -> Any:
        return _lib.OBJ_obj2nid(
            _lib.X509_EXTENSION_get_object(self._extension)
        )

    _prefixes: typing.ClassVar[dict[int, str]] = {
        _lib.GEN_EMAIL: "email",
        _lib.GEN_DNS: "DNS",
        _lib.GEN_URI: "URI",
    }

    def _subjectAltNameString(self) -> str:
        names = _ffi.cast(
            "GENERAL_NAMES*", _lib.X509V3_EXT_d2i(self._extension)
        )

        names = _ffi.gc(names, _lib.GENERAL_NAMES_free)
        parts = []
        for i in range(_lib.sk_GENERAL_NAME_num(names)):
            name = _lib.sk_GENERAL_NAME_value(names, i)
```

```
                try:
                    label = self._prefixes[name.type]
                except KeyError:
                    bio = _new_mem_buf()
                    _lib.GENERAL_NAME_print(bio, name)
                    parts.append(_bio_to_string(bio).decode("utf-8"))
                else:
                    value = _ffi.buffer(name.d.ia5.data, name.d.ia5.length)[
                        :
                    ].decode("utf-8")
                    parts.append(label + ":" + value)
            return ", ".join(parts)

        def __str__(self) -> str:
            if _lib.NID_subject_alt_name == self._nid:
                return self._subjectAltNameString()

            bio = _new_mem_buf()
            print_result = _lib.X509V3_EXT_print(bio, self._extension, 0, 0)
            _openssl_assert(print_result != 0)

            return _bio_to_string(bio).decode("utf-8")

        def get_critical(self) -> bool:
            return _lib.X509_EXTENSION_get_critical(self._extension)

        def get_short_name(self) -> bytes:
            obj = _lib.X509_EXTENSION_get_object(self._extension)
            nid = _lib.OBJ_obj2nid(obj)
            buf = _lib.OBJ_nid2sn(nid)
            if buf != _ffi.NULL:
                return _ffi.string(buf)
            else:
                return b"UNDEF"

        def get_data(self) -> bytes:
            octet_result = _lib.X509_EXTENSION_get_data(self._extension)
            string_result = _ffi.cast("ASN1_STRING*", octet_result)
            char_result = _lib.ASN1_STRING_get0_data(string_result)
            result_length = _lib.ASN1_STRING_length(string_result)
            return _ffi.buffer(char_result, result_length)[:]


    @deprecated(
        "CSR support in pyOpenSSL is deprecated. You should use the APIs "
        "in cryptography."
    )
    class X509Req:

        def __init__(self) -> None:
            req = _lib.X509_REQ_new()
            self._req = _ffi.gc(req, _lib.X509_REQ_free)
            self.set_version(0)

        def to_cryptography(self) -> x509.CertificateSigningRequest:
            from cryptography.x509 import load_der_x509_csr

            der = _dump_certificate_request_internal(FILETYPE_ASN1, self)

            return load_der_x509_csr(der)

        @classmethod
        def from_cryptography(
```

```
    cls, crypto_req: x509.CertificateSigningRequest
) -> X509Req:
    if not isinstance(crypto_req, x509.CertificateSigningRequest):
        raise TypeError("Must be a certificate signing request")

    from cryptography.hazmat.primitives.serialization import Encoding

    der = crypto_req.public_bytes(Encoding.DER)
    return _load_certificate_request_internal(FILETYPE_ASN1, der)

def set_pubkey(self, pkey: PKey) -> None:
    set_result = _lib.X509_REQ_set_pubkey(self._req, pkey._pkey)
    _openssl_assert(set_result == 1)

def get_pubkey(self) -> PKey:
    pkey = PKey.__new__(PKey)
    pkey._pkey = _lib.X509_REQ_get_pubkey(self._req)
    _openssl_assert(pkey._pkey != _ffi.NULL)
    pkey._pkey = _ffi.gc(pkey._pkey, _lib.EVP_PKEY_free)
    pkey._only_public = True
    return pkey

def set_version(self, version: int) -> None:
    if not isinstance(version, int):
        raise TypeError("version must be an int")
    if version != 0:
        raise ValueError(
            "Invalid version. The only valid version for X509Req is 0."
        )
    set_result = _lib.X509_REQ_set_version(self._req, version)
    _openssl_assert(set_result == 1)

def get_version(self) -> int:
    return _lib.X509_REQ_get_version(self._req)

def get_subject(self) -> X509Name:
    name = X509Name.__new__(X509Name)
    name._name = _lib.X509_REQ_get_subject_name(self._req)
    _openssl_assert(name._name != _ffi.NULL)

    name._owner = self

    return name

def add_extensions(self, extensions: Iterable[X509Extension]) -> None:
    warnings.warn(
        (
            "This API is deprecated and will be removed in a future "
            "version of pyOpenSSL. You should use pyca/cryptography's "
            "X.509 APIs instead."
        ),
        DeprecationWarning,
        stacklevel=2,
    )

    stack = _lib.sk_X509_EXTENSION_new_null()
    _openssl_assert(stack != _ffi.NULL)

    stack = _ffi.gc(stack, _lib.sk_X509_EXTENSION_free)

    for ext in extensions:
        if not isinstance(ext, X509Extension):
            raise ValueError("One of the elements is not an X509Extension")
```

```
            _lib.sk_X509_EXTENSION_push(stack, ext._extension)

        add_result = _lib.X509_REQ_add_extensions(self._req, stack)
        _openssl_assert(add_result == 1)

    def get_extensions(self) -> list[X509Extension]:
        warnings.warn(
            (
                "This API is deprecated and will be removed in a future "
                "version of pyOpenSSL. You should use pyca/cryptography's "
                "X.509 APIs instead."
            ),
            DeprecationWarning,
            stacklevel=2,
        )

        exts = []
        native_exts_obj = _lib.X509_REQ_get_extensions(self._req)
        native_exts_obj = _ffi.gc(
            native_exts_obj,
            lambda x: _lib.sk_X509_EXTENSION_pop_free(
                x,
                _ffi.addressof(_lib._original_lib, "X509_EXTENSION_free"),
            ),
        )

        for i in range(_lib.sk_X509_EXTENSION_num(native_exts_obj)):
            ext = X509Extension.__new__(X509Extension)
            extension = _lib.X509_EXTENSION_dup(
                _lib.sk_X509_EXTENSION_value(native_exts_obj, i)
            )
            ext._extension = _ffi.gc(extension, _lib.X509_EXTENSION_free)
            exts.append(ext)
        return exts

    def sign(self, pkey: PKey, digest: str) -> None:
        if pkey._only_public:
            raise ValueError("Key has only public part")

        if not pkey._initialized:
            raise ValueError("Key is uninitialized")

        digest_obj = _lib.EVP_get_digestbyname(_byte_string(digest))
        if digest_obj == _ffi.NULL:
            raise ValueError("No such digest method")

        sign_result = _lib.X509_REQ_sign(self._req, pkey._pkey, digest_obj)
        _openssl_assert(sign_result > 0)

    def verify(self, pkey: PKey) -> bool:
        if not isinstance(pkey, PKey):
            raise TypeError("pkey must be a PKey instance")

        result = _lib.X509_REQ_verify(self._req, pkey._pkey)
        if result <= 0:
            _raise_current_error()

        return result


class X509:
```

```
def __init__(self) -> None:
    x509 = _lib.X509_new()
    _openssl_assert(x509 != _ffi.NULL)
    self._x509 = _ffi.gc(x509, _lib.X509_free)

    self._issuer_invalidator = _X509NameInvalidator()
    self._subject_invalidator = _X509NameInvalidator()

@classmethod
def _from_raw_x509_ptr(cls, x509: Any) -> X509:
    cert = cls.__new__(cls)
    cert._x509 = _ffi.gc(x509, _lib.X509_free)
    cert._issuer_invalidator = _X509NameInvalidator()
    cert._subject_invalidator = _X509NameInvalidator()
    return cert

def to_cryptography(self) -> x509.Certificate:
    from cryptography.x509 import load_der_x509_certificate

    der = dump_certificate(FILETYPE_ASN1, self)
    return load_der_x509_certificate(der)

@classmethod
def from_cryptography(cls, crypto_cert: x509.Certificate) -> X509:
    if not isinstance(crypto_cert, x509.Certificate):
        raise TypeError("Must be a certificate")

    from cryptography.hazmat.primitives.serialization import Encoding

    der = crypto_cert.public_bytes(Encoding.DER)
    return load_certificate(FILETYPE_ASN1, der)

def set_version(self, version: int) -> None:
    if not isinstance(version, int):
        raise TypeError("version must be an integer")

    _openssl_assert(_lib.X509_set_version(self._x509, version) == 1)

def get_version(self) -> int:
    return _lib.X509_get_version(self._x509)

def get_pubkey(self) -> PKey:
    pkey = PKey.__new__(PKey)
    pkey._pkey = _lib.X509_get_pubkey(self._x509)
    if pkey._pkey == _ffi.NULL:
        _raise_current_error()
    pkey._pkey = _ffi.gc(pkey._pkey, _lib.EVP_PKEY_free)
    pkey._only_public = True
    return pkey

def set_pubkey(self, pkey: PKey) -> None:
    if not isinstance(pkey, PKey):
        raise TypeError("pkey must be a PKey instance")

    set_result = _lib.X509_set_pubkey(self._x509, pkey._pkey)
    _openssl_assert(set_result == 1)

def sign(self, pkey: PKey, digest: str) -> None:
    if not isinstance(pkey, PKey):
        raise TypeError("pkey must be a PKey instance")

    if pkey._only_public:
        raise ValueError("Key only has public part")
```

```
    if not pkey._initialized:
        raise ValueError("Key is uninitialized")

    evp_md = _lib.EVP_get_digestbyname(_byte_string(digest))
    if evp_md == _ffi.NULL:
        raise ValueError("No such digest method")

    sign_result = _lib.X509_sign(self._x509, pkey._pkey, evp_md)
    _openssl_assert(sign_result > 0)

def get_signature_algorithm(self) -> bytes:
    sig_alg = _lib.X509_get0_tbs_sigalg(self._x509)
    alg = _ffi.new("ASN1_OBJECT **")
    _lib.X509_ALGOR_get0(alg, _ffi.NULL, _ffi.NULL, sig_alg)
    nid = _lib.OBJ_obj2nid(alg[0])
    if nid == _lib.NID_undef:
        raise ValueError("Undefined signature algorithm")
    return _ffi.string(_lib.OBJ_nid2ln(nid))

def digest(self, digest_name: str) -> bytes:
    digest = _lib.EVP_get_digestbyname(_byte_string(digest_name))
    if digest == _ffi.NULL:
        raise ValueError("No such digest method")

    result_buffer = _ffi.new("unsigned char[]", _lib.EVP_MAX_MD_SIZE)
    result_length = _ffi.new("unsigned int[]", 1)
    result_length[0] = len(result_buffer)

    digest_result = _lib.X509_digest(
        self._x509, digest, result_buffer, result_length
    )
    _openssl_assert(digest_result == 1)

    return b":".join(
        [
            b16encode(ch).upper()
            for ch in _ffi.buffer(result_buffer, result_length[0])
        ]
    )

def subject_name_hash(self) -> int:
    return _lib.X509_subject_name_hash(self._x509)

def set_serial_number(self, serial: int) -> None:
    if not isinstance(serial, int):
        raise TypeError("serial must be an integer")

    hex_serial = hex(serial)[2:]
    hex_serial_bytes = hex_serial.encode("ascii")

    bignum_serial = _ffi.new("BIGNUM**")

    result = _lib.BN_hex2bn(bignum_serial, hex_serial_bytes)
    _openssl_assert(result != _ffi.NULL)

    asn1_serial = _lib.BN_to_ASN1_INTEGER(bignum_serial[0], _ffi.NULL)
    _lib.BN_free(bignum_serial[0])
    _openssl_assert(asn1_serial != _ffi.NULL)
    asn1_serial = _ffi.gc(asn1_serial, _lib.ASN1_INTEGER_free)
    set_result = _lib.X509_set_serialNumber(self._x509, asn1_serial)
    _openssl_assert(set_result == 1)
```

```
def get_serial_number(self) -> int:
    asn1_serial = _lib.X509_get_serialNumber(self._x509)
    bignum_serial = _lib.ASN1_INTEGER_to_BN(asn1_serial, _ffi.NULL)
    try:
        hex_serial = _lib.BN_bn2hex(bignum_serial)
        try:
            hexstring_serial = _ffi.string(hex_serial)
            serial = int(hexstring_serial, 16)
            return serial
        finally:
            _lib.OPENSSL_free(hex_serial)
    finally:
        _lib.BN_free(bignum_serial)

def gmtime_adj_notAfter(self, amount: int) -> None:
    if not isinstance(amount, int):
        raise TypeError("amount must be an integer")

    notAfter = _lib.X509_getm_notAfter(self._x509)
    _lib.X509_gmtime_adj(notAfter, amount)

def gmtime_adj_notBefore(self, amount: int) -> None:
    if not isinstance(amount, int):
        raise TypeError("amount must be an integer")

    notBefore = _lib.X509_getm_notBefore(self._x509)
    _lib.X509_gmtime_adj(notBefore, amount)

def has_expired(self) -> bool:
    time_bytes = self.get_notAfter()
    if time_bytes is None:
        raise ValueError("Unable to determine notAfter")
    time_string = time_bytes.decode("utf-8")
    not_after = datetime.datetime.strptime(time_string, "%Y%m%d%H%M%SZ")

    UTC = datetime.timezone.utc
    utcnow = datetime.datetime.now(UTC).replace(tzinfo=None)
    return not_after < utcnow

def _get_boundary_time(self, which: Any) -> bytes | None:
    return _get_asn1_time(which(self._x509))

def get_notBefore(self) -> bytes | None:
    return self._get_boundary_time(_lib.X509_getm_notBefore)

def _set_boundary_time(
    self, which: Callable[..., Any], when: bytes
) -> None:
    return _set_asn1_time(which(self._x509), when)

def set_notBefore(self, when: bytes) -> None:
    return self._set_boundary_time(_lib.X509_getm_notBefore, when)

def get_notAfter(self) -> bytes | None:
    return self._get_boundary_time(_lib.X509_getm_notAfter)

def set_notAfter(self, when: bytes) -> None:
    return self._set_boundary_time(_lib.X509_getm_notAfter, when)

def _get_name(self, which: Any) -> X509Name:
    name = X509Name.__new__(X509Name)
    name._name = which(self._x509)
    _openssl_assert(name._name != _ffi.NULL)
```

```
    name._owner = self

    return name

def _set_name(self, which: Any, name: X509Name) -> None:
    if not isinstance(name, X509Name):
        raise TypeError("name must be an X509Name")
    set_result = which(self._x509, name._name)
    _openssl_assert(set_result == 1)

def get_issuer(self) -> X509Name:
    name = self._get_name(_lib.X509_get_issuer_name)
    self._issuer_invalidator.add(name)
    return name

def set_issuer(self, issuer: X509Name) -> None:
    self._set_name(_lib.X509_set_issuer_name, issuer)
    self._issuer_invalidator.clear()

def get_subject(self) -> X509Name:
    name = self._get_name(_lib.X509_get_subject_name)
    self._subject_invalidator.add(name)
    return name

def set_subject(self, subject: X509Name) -> None:
    self._set_name(_lib.X509_set_subject_name, subject)
    self._subject_invalidator.clear()

def get_extension_count(self) -> int:
    return _lib.X509_get_ext_count(self._x509)

def add_extensions(self, extensions: Iterable[X509Extension]) -> None:
    warnings.warn(
        (
            "This API is deprecated and will be removed in a future "
            "version of pyOpenSSL. You should use pyca/cryptography's "
            "X.509 APIs instead."
        ),
        DeprecationWarning,
        stacklevel=2,
    )

    for ext in extensions:
        if not isinstance(ext, X509Extension):
            raise ValueError("One of the elements is not an X509Extension")

        add_result = _lib.X509_add_ext(self._x509, ext._extension, -1)
        _openssl_assert(add_result == 1)

def get_extension(self, index: int) -> X509Extension:
    warnings.warn(
        (
            "This API is deprecated and will be removed in a future "
            "version of pyOpenSSL. You should use pyca/cryptography's "
            "X.509 APIs instead."
        ),
        DeprecationWarning,
        stacklevel=2,
    )

    ext = X509Extension.__new__(X509Extension)
    ext._extension = _lib.X509_get_ext(self._x509, index)
```

```
        if ext._extension == _ffi.NULL:
            raise IndexError("extension index out of bounds")

        extension = _lib.X509_EXTENSION_dup(ext._extension)
        ext._extension = _ffi.gc(extension, _lib.X509_EXTENSION_free)
        return ext


class X509StoreFlags:

    CRL_CHECK: int = _lib.X509_V_FLAG_CRL_CHECK
    CRL_CHECK_ALL: int = _lib.X509_V_FLAG_CRL_CHECK_ALL
    IGNORE_CRITICAL: int = _lib.X509_V_FLAG_IGNORE_CRITICAL
    X509_STRICT: int = _lib.X509_V_FLAG_X509_STRICT
    ALLOW_PROXY_CERTS: int = _lib.X509_V_FLAG_ALLOW_PROXY_CERTS
    POLICY_CHECK: int = _lib.X509_V_FLAG_POLICY_CHECK
    EXPLICIT_POLICY: int = _lib.X509_V_FLAG_EXPLICIT_POLICY
    INHIBIT_MAP: int = _lib.X509_V_FLAG_INHIBIT_MAP
    CHECK_SS_SIGNATURE: int = _lib.X509_V_FLAG_CHECK_SS_SIGNATURE
    PARTIAL_CHAIN: int = _lib.X509_V_FLAG_PARTIAL_CHAIN


class X509Store:

    def __init__(self) -> None:
        store = _lib.X509_STORE_new()
        self._store = _ffi.gc(store, _lib.X509_STORE_free)

    def add_cert(self, cert: X509) -> None:
        if not isinstance(cert, X509):
            raise TypeError()

        res = _lib.X509_STORE_add_cert(self._store, cert._x509)
        _openssl_assert(res == 1)

    def add_crl(self, crl: x509.CertificateRevocationList) -> None:
        if isinstance(crl, x509.CertificateRevocationList):
            from cryptography.hazmat.primitives.serialization import Encoding

            bio = _new_mem_buf(crl.public_bytes(Encoding.DER))
            openssl_crl = _lib.d2i_X509_CRL_bio(bio, _ffi.NULL)
            _openssl_assert(openssl_crl != _ffi.NULL)
            crl = _ffi.gc(openssl_crl, _lib.X509_CRL_free)
        else:
            raise TypeError(
                "CRL must be of type "
                "cryptography.x509.CertificateRevocationList"
            )

        _openssl_assert(_lib.X509_STORE_add_crl(self._store, crl) != 0)

    def set_flags(self, flags: int) -> None:
        _openssl_assert(_lib.X509_STORE_set_flags(self._store, flags) != 0)

    def set_time(self, vfy_time: datetime.datetime) -> None:
        param = _lib.X509_VERIFY_PARAM_new()
        param = _ffi.gc(param, _lib.X509_VERIFY_PARAM_free)

        _lib.X509_VERIFY_PARAM_set_time(
            param, calendar.timegm(vfy_time.timetuple())
        )
        _openssl_assert(_lib.X509_STORE_set1_param(self._store, param) != 0)
```

```
    def load_locations(
        self,
        cafile: StrOrBytesPath | None,
        capath: StrOrBytesPath | None = None,
    ) -> None:
        if cafile is None:
            cafile = _ffi.NULL
        else:
            cafile = _path_bytes(cafile)

        if capath is None:
            capath = _ffi.NULL
        else:
            capath = _path_bytes(capath)

        load_result = _lib.X509_STORE_load_locations(
            self._store, cafile, capath
        )
        if not load_result:
            _raise_current_error()


class X509StoreContextError(Exception):

    def __init__(
        self, message: str, errors: list[Any], certificate: X509
    ) -> None:
        super().__init__(message)
        self.errors = errors
        self.certificate = certificate


class X509StoreContext:

    def __init__(
        self,
        store: X509Store,
        certificate: X509,
        chain: Sequence[X509] | None = None,
    ) -> None:
        self._store = store
        self._cert = certificate
        self._chain = self._build_certificate_stack(chain)

    @staticmethod
    def _build_certificate_stack(
        certificates: Sequence[X509] | None,
    ) -> None:
        def cleanup(s: Any) -> None:
            for i in range(_lib.sk_X509_num(s)):
                x = _lib.sk_X509_value(s, i)
                _lib.X509_free(x)
            _lib.sk_X509_free(s)

        if certificates is None or len(certificates) == 0:
            return _ffi.NULL

        stack = _lib.sk_X509_new_null()
        _openssl_assert(stack != _ffi.NULL)
        stack = _ffi.gc(stack, cleanup)

        for cert in certificates:
            if not isinstance(cert, X509):
```

```
                raise TypeError("One of the elements is not an X509 instance")

            _openssl_assert(_lib.X509_up_ref(cert._x509) > 0)
            if _lib.sk_X509_push(stack, cert._x509) <= 0:
                _lib.X509_free(cert._x509)
                _raise_current_error()

        return stack

    @staticmethod
    def _exception_from_context(store_ctx: Any) -> X509StoreContextError:
        message = _ffi.string(
            _lib.X509_verify_cert_error_string(
                _lib.X509_STORE_CTX_get_error(store_ctx)
            )
        ).decode("utf-8")
        errors = [
            _lib.X509_STORE_CTX_get_error(store_ctx),
            _lib.X509_STORE_CTX_get_error_depth(store_ctx),
            message,
        ]
        _x509 = _lib.X509_STORE_CTX_get_current_cert(store_ctx)
        _cert = _lib.X509_dup(_x509)
        pycert = X509._from_raw_x509_ptr(_cert)
        return X509StoreContextError(message, errors, pycert)

    def _verify_certificate(self) -> Any:
        store_ctx = _lib.X509_STORE_CTX_new()
        _openssl_assert(store_ctx != _ffi.NULL)
        store_ctx = _ffi.gc(store_ctx, _lib.X509_STORE_CTX_free)

        ret = _lib.X509_STORE_CTX_init(
            store_ctx, self._store._store, self._cert._x509, self._chain
        )
        _openssl_assert(ret == 1)

        ret = _lib.X509_verify_cert(store_ctx)
        if ret <= 0:
            raise self._exception_from_context(store_ctx)

        return store_ctx

    def set_store(self, store: X509Store) -> None:
        self._store = store

    def verify_certificate(self) -> None:
        self._verify_certificate()

    def get_verified_chain(self) -> list[X509]:
        store_ctx = self._verify_certificate()

        cert_stack = _lib.X509_STORE_CTX_get1_chain(store_ctx)
        _openssl_assert(cert_stack != _ffi.NULL)

        result = []
        for i in range(_lib.sk_X509_num(cert_stack)):
            cert = _lib.sk_X509_value(cert_stack, i)
            _openssl_assert(cert != _ffi.NULL)
            pycert = X509._from_raw_x509_ptr(cert)
            result.append(pycert)

        _lib.sk_X509_free(cert_stack)
        return result
```

```
def load_certificate(type: int, buffer: bytes) -> X509:
    if isinstance(buffer, str):
        buffer = buffer.encode("ascii")

    bio = _new_mem_buf(buffer)

    if type == FILETYPE_PEM:
        x509 = _lib.PEM_read_bio_X509(bio, _ffi.NULL, _ffi.NULL, _ffi.NULL)
    elif type == FILETYPE_ASN1:
        x509 = _lib.d2i_X509_bio(bio, _ffi.NULL)
    else:
        raise ValueError("type argument must be FILETYPE_PEM or FILETYPE_ASN1")

    if x509 == _ffi.NULL:
        _raise_current_error()

    return X509._from_raw_x509_ptr(x509)


def dump_certificate(type: int, cert: X509) -> bytes:
    bio = _new_mem_buf()

    if type == FILETYPE_PEM:
        result_code = _lib.PEM_write_bio_X509(bio, cert._x509)
    elif type == FILETYPE_ASN1:
        result_code = _lib.i2d_X509_bio(bio, cert._x509)
    elif type == FILETYPE_TEXT:
        result_code = _lib.X509_print_ex(bio, cert._x509, 0, 0)
    else:
        raise ValueError(
            "type argument must be FILETYPE_PEM, FILETYPE_ASN1, or "
            "FILETYPE_TEXT"
        )

    _openssl_assert(result_code == 1)
    return _bio_to_string(bio)


def dump_publickey(type: int, pkey: PKey) -> bytes:
    bio = _new_mem_buf()
    if type == FILETYPE_PEM:
        write_bio = _lib.PEM_write_bio_PUBKEY
    elif type == FILETYPE_ASN1:
        write_bio = _lib.i2d_PUBKEY_bio
    else:
        raise ValueError("type argument must be FILETYPE_PEM or FILETYPE_ASN1")

    result_code = write_bio(bio, pkey._pkey)
    if result_code != 1:  # pragma: no cover
        _raise_current_error()

    return _bio_to_string(bio)


def dump_privatekey(
    type: int,
    pkey: PKey,
    cipher: str | None = None,
    passphrase: PassphraseCallableT | None = None,
) -> bytes:
    bio = _new_mem_buf()
```

```
    if not isinstance(pkey, PKey):
        raise TypeError("pkey must be a PKey")

    if cipher is not None:
        if passphrase is None:
            raise TypeError(
                "if a value is given for cipher "
                "one must also be given for passphrase"
            )
        cipher_obj = _lib.EVP_get_cipherbyname(_byte_string(cipher))
        if cipher_obj == _ffi.NULL:
            raise ValueError("Invalid cipher name")
    else:
        cipher_obj = _ffi.NULL

    helper = _PassphraseHelper(type, passphrase)
    if type == FILETYPE_PEM:
        result_code = _lib.PEM_write_bio_PrivateKey(
            bio,
            pkey._pkey,
            cipher_obj,
            _ffi.NULL,
            0,
            helper.callback,
            helper.callback_args,
        )
        helper.raise_if_problem()
    elif type == FILETYPE_ASN1:
        result_code = _lib.i2d_PrivateKey_bio(bio, pkey._pkey)
    elif type == FILETYPE_TEXT:
        if _lib.EVP_PKEY_id(pkey._pkey) != _lib.EVP_PKEY_RSA:
            raise TypeError("Only RSA keys are supported for FILETYPE_TEXT")

        rsa = _ffi.gc(_lib.EVP_PKEY_get1_RSA(pkey._pkey), _lib.RSA_free)
        result_code = _lib.RSA_print(bio, rsa, 0)
    else:
        raise ValueError(
            "type argument must be FILETYPE_PEM, FILETYPE_ASN1, or "
            "FILETYPE_TEXT"
        )

    _openssl_assert(result_code != 0)

    return _bio_to_string(bio)


class _PassphraseHelper:
    def __init__(
        self,
        type: int,
        passphrase: PassphraseCallableT | None,
        more_args: bool = False,
        truncate: bool = False,
    ) -> None:
        if type != FILETYPE_PEM and passphrase is not None:
            raise ValueError(
                "only FILETYPE_PEM key format supports encryption"
            )
        self._passphrase = passphrase
        self._more_args = more_args
        self._truncate = truncate
        self._problems: list[Exception] = []
```

```
    @property
    def callback(self) -> Any:
        if self._passphrase is None:
            return _ffi.NULL
        elif isinstance(self._passphrase, bytes) or callable(self._passphrase):
            return _ffi.callback("pem_password_cb", self._read_passphrase)
        else:
            raise TypeError(
                "Last argument must be a byte string or a callable."
            )

    @property
    def callback_args(self) -> Any:
        if self._passphrase is None:
            return _ffi.NULL
        elif isinstance(self._passphrase, bytes) or callable(self._passphrase):
            return _ffi.NULL
        else:
            raise TypeError(
                "Last argument must be a byte string or a callable."
            )

    def raise_if_problem(self, exceptionType: type[Exception] = Error) -> None:
        if self._problems:
            try:
                _exception_from_error_queue(exceptionType)
            except exceptionType:
                pass

            raise self._problems.pop(0)

    def _read_passphrase(
        self, buf: Any, size: int, rwflag: Any, userdata: Any
    ) -> int:
        try:
            if callable(self._passphrase):
                if self._more_args:
                    result = self._passphrase(size, rwflag, userdata)
                else:
                    result = self._passphrase(rwflag)
            else:
                assert self._passphrase is not None
                result = self._passphrase
            if not isinstance(result, bytes):
                raise ValueError("Bytes expected")
            if len(result) > size:
                if self._truncate:
                    result = result[:size]
                else:
                    raise ValueError(
                        "passphrase returned by callback is too long"
                    )
            for i in range(len(result)):
                buf[i] = result[i : i + 1]
            return len(result)
        except Exception as e:
            self._problems.append(e)
            return 0


def load_publickey(type: int, buffer: str | bytes) -> PKey:
    if isinstance(buffer, str):
```

```
        buffer = buffer.encode("ascii")

    bio = _new_mem_buf(buffer)

    if type == FILETYPE_PEM:
        evp_pkey = _lib.PEM_read_bio_PUBKEY(
            bio, _ffi.NULL, _ffi.NULL, _ffi.NULL
        )
    elif type == FILETYPE_ASN1:
        evp_pkey = _lib.d2i_PUBKEY_bio(bio, _ffi.NULL)
    else:
        raise ValueError("type argument must be FILETYPE_PEM or FILETYPE_ASN1")

    if evp_pkey == _ffi.NULL:
        _raise_current_error()

    pkey = PKey.__new__(PKey)
    pkey._pkey = _ffi.gc(evp_pkey, _lib.EVP_PKEY_free)
    pkey._only_public = True
    return pkey


def load_privatekey(
    type: int,
    buffer: str | bytes,
    passphrase: PassphraseCallableT | None = None,
) -> PKey:
    if isinstance(buffer, str):
        buffer = buffer.encode("ascii")

    bio = _new_mem_buf(buffer)

    helper = _PassphraseHelper(type, passphrase)
    if type == FILETYPE_PEM:
        evp_pkey = _lib.PEM_read_bio_PrivateKey(
            bio, _ffi.NULL, helper.callback, helper.callback_args
        )
        helper.raise_if_problem()
    elif type == FILETYPE_ASN1:
        evp_pkey = _lib.d2i_PrivateKey_bio(bio, _ffi.NULL)
    else:
        raise ValueError("type argument must be FILETYPE_PEM or FILETYPE_ASN1")

    if evp_pkey == _ffi.NULL:
        _raise_current_error()

    pkey = PKey.__new__(PKey)
    pkey._pkey = _ffi.gc(evp_pkey, _lib.EVP_PKEY_free)
    return pkey


def dump_certificate_request(type: int, req: X509Req) -> bytes:
    bio = _new_mem_buf()

    if type == FILETYPE_PEM:
        result_code = _lib.PEM_write_bio_X509_REQ(bio, req._req)
    elif type == FILETYPE_ASN1:
        result_code = _lib.i2d_X509_REQ_bio(bio, req._req)
    elif type == FILETYPE_TEXT:
        result_code = _lib.X509_REQ_print_ex(bio, req._req, 0, 0)
    else:
        raise ValueError(
            "type argument must be FILETYPE_PEM, FILETYPE_ASN1, or "
```

```
            "FILETYPE_TEXT"
        )

    _openssl_assert(result_code != 0)

    return _bio_to_string(bio)


_dump_certificate_request_internal = dump_certificate_request

utils.deprecated(
    dump_certificate_request,
    __name__,
    (
        "CSR support in pyOpenSSL is deprecated. You should use the APIs "
        "in cryptography."
    ),
    DeprecationWarning,
    name="dump_certificate_request",
)


def load_certificate_request(type: int, buffer: bytes) -> X509Req:
    if isinstance(buffer, str):
        buffer = buffer.encode("ascii")

    bio = _new_mem_buf(buffer)

    if type == FILETYPE_PEM:
        req = _lib.PEM_read_bio_X509_REQ(bio, _ffi.NULL, _ffi.NULL, _ffi.NULL)
    elif type == FILETYPE_ASN1:
        req = _lib.d2i_X509_REQ_bio(bio, _ffi.NULL)
    else:
        raise ValueError("type argument must be FILETYPE_PEM or FILETYPE_ASN1")

    _openssl_assert(req != _ffi.NULL)

    x509req = X509Req.__new__(X509Req)
    x509req._req = _ffi.gc(req, _lib.X509_REQ_free)
    return x509req
```