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

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2025 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки для управління ідентифікацією та доступом до інформаційних ресурсів”

Виконав здобувач вищої освіти
IV курсу, групи КБ-22-МБ
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Каптінар М.І.
«____» _____ 2025 р.

Керівник проекту
кандидат технічних наук
_____ Смірнова Т.В.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

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

Спеціальність *125 “*Кібербезпека*”*

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

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

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

Каптінару Максиму Ігоровичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки для управління ідентифікацією та доступом до інформаційних ресурсів*

2. Керівник роботи *Смірнова Тетяна Віталіївна, канд. техн. наук*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 51-02 від 17.01.2025 року

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

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи кібербезпеки для управління ідентифікацією та доступом до інформаційних ресурсів*

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

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи кібербезпеки в промислову експлуатацію.

6. Висновки

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

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

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

КАЛЕНДАРНИЙ ПЛАН

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

Дата видачі завдання
« 17 » січня 2025 р.

Підпис керівника
_____ Смірнова Т.В.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Каптінар М.І.
(прізвище та ініціали)

АНОТАЦІЯ

Каптінар М.І. Програмне забезпечення системи кібербезпеки для управління ідентифікацією та доступом до інформаційних ресурсів. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки для управління ідентифікацією та доступом до інформаційних ресурсів.

Метою розробки є програмне забезпечення системи кібербезпеки для управління ідентифікацією та доступом до інформаційних ресурсів.

Результат роботи – програмна реалізація системи кібербезпеки для управління ідентифікацією та доступом до інформаційних ресурсів.

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

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

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

Програму розроблено в середовищі Python.

Ключові слова: кібербезпека, управління ідентифікацією та доступом

ABSTRACT

Kaptinar M.I. Cybersecurity system software for managing identification and access to information resources. 125 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed that is intended for a cybersecurity system for managing identification and access to information resources.

The purpose of the development is to develop a cybersecurity system software for managing identification and access to information resources.

The result of the work is a software implementation of a cybersecurity system for managing identification and access to information resources.

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: cybersecurity, identity and access management

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ЕЦП	–	електронний цифровий підпис
ІТ	–	інформаційні технології
КУМЗ	–	класи уніфікованих математичних завдань
ОМЗ	–	основні математичні завдання
ПЗ	–	програмне забезпечення
СКЗІ	–	система контролю та захисту інформації
СКУД	–	система контролю й управління доступом у приміщення
ОАОР	–	загальний річний ріст
OTP	–	OneTime Password
PKI	–	інфрастуктура відкритих ключів
USB	–	universal serial bus

ВСТУП

Актуальність теми. Питання забезпечення інформаційної безпеки повинні вирішуватися системно й комплексно. Важливу роль у цьому грають надійні механізми захищеного доступу, у тому числі автентифікація користувачів і захист переданих даних.

Інформаційна безпека неможлива без автентифікації, а проникнення в корпоративне середовище мобільних пристроїв і хмарних технологій не може не впливати на принципи забезпечення ІБ. Автентифікація – процедура підтвердження дійсності суб'єкта в інформаційній системі по якімсь ідентифікаторі. Надійна й адекватна система автентифікації користувачів – найважливіший компонент корпоративної системи інформаційної безпеки. Звичайно, у різних комунікаційних каналах можуть і повинні застосовуватися різні механізми автентифікації, кожний з яких має свої достоїнства й недоліки, відрізняється по надійності й вартості рішень, зручності застосування й адміністрування. Тому при їхньому виборі необхідно аналізувати ризики й оцінювати економічну доцільність впровадження.

Для автентифікації користувачів застосовуються різні технології – від паролів, смарт-карт, токенів до біометрії, заснованої на таких персональних властивостях людини, як відбиток пальця, структура сітківки ока й т.д. Системи строгої автентифікації перевіряють два й більше фактори.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки для управління ідентифікацією та доступом до інформаційних ресурсів.

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

– Огляд існуючих систем для управління ідентифікацією та доступом до інформаційних ресурсів.

– Дослідження системи кібербезпеки для управління ідентифікацією та доступом до інформаційних ресурсів.

– Програмна реалізація системи кібербезпеки для управління ідентифікацією та доступом до інформаційних ресурсів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі для управління ідентифікацією та доступом до інформаційних ресурсів.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки для управління ідентифікацією та доступом до інформаційних ресурсів, 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

Кращою практикою вважається двостороння сувора автентифікація, заснована на технології електронного цифрового підпису (ЕЦП). Коли цю технологію використовувати неможливо або недоцільно, рекомендується застосовувати одноразові паролі, а при мінімальному рівні ризиків – багаторазові.

Без автентифікації неможливо говорити про безпеку в інформаційній системі. Є різні її способи – наприклад, за допомогою багаторазових або одноразових паролів. Однак більше надійна багатофакторна автентифікація, коли безпека ґрунтується не тільки на знанні якогось секрету (пароля), але й на володінні спеціальним пристроєм, а як третій фактор виступає одна або кілька біометричних характеристик користувача. Пароль може бути украдений або підібраний, але без пристрою автентифікації – USB-токена, смарт-карти або SIM-карти – зловмисникові не вдасться одержати доступ до системи. Використання пристроїв, робота з якими підтримується не тільки на робочих станціях, але й на мобільних платформах, дозволяє застосовувати багатофакторну автентифікацію відносно власників мобільних телефонів або планшетів. Це стосується й моделі Bring Your Own Device (BYOD)».

Сьогодні ми спостерігаємо ріст інтересу до токенам – генераторів одноразових паролів (One Time Password, OTP) різних виробників. Впровадження подібних систем уже стало стандартом де-факто для інтернет-банкінгу й усе більше затребувано при організації віддаленого доступу з мобільних пристроїв. Вони зручні й прості у використанні, що дозволяє застосовувати їх повсюдно. Фахівці виявляють цікавість до технологій, які не вимагають установки й керування «зайвим» програмним забезпеченням на стороні клієнта. Відповідне рішення інтегрується з різними Web-сервісами, не має потреби в клієнтській частині й підтримує українські криптоалгоритми.

1.2 Область застосування

Програмний комплекс призначений для централізованого захищеного зберігання закритих ключів користувачів, а також для віддаленого виконання операцій по створенню електронного підпису. Він підтримує різні способи автентифікації: однофакторну – за логіном й паролем; двофакторну – з використанням цифрових сертифікатів і USB-токенів або смарт-карт; двофакторну – з додатковим уведенням одноразового пароля, що доставляється по SMS. Одна із ключових тенденцій у сфері ІБ пов'язана з ростом числа мобільних пристроїв, за допомогою яких офісні або віддалені співробітники працюють із конфіденційною корпоративною інформацією: електронною поштою, сховищами документів, різними бізнес-додатками й др. При цьому й в Україні, і за рубежем усе більше популярною стає модель BYOD, коли на роботі дозволяється використовувати особисті пристрої. Протидія виникаючим при цьому погрозам – одна з найбільш актуальних і складних проблем ІБ, що має бути вирішувати в найближчі п'ять – сім років.

Розуміння учасниками ринку необхідності впровадження надійних систем автентифікації й цифрового підпису (ЕЦП), зміна законодавства в області захисту персональних даних, прийняття вимог по сертифікації (СБУ України), реалізація таких проектів, як «Електронний уряд» і «Портал державних послуг», розвиток дистанційного банківського обслуговування й інтернет-банкінгу, пошук можливостей для визначення відповідальності в кіберпросторі – все це сприяє підвищеному інтересу до програмно-апаратних рішень, що сполучають у собі зручність використання й посилений захист.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки для управління ідентифікацією та доступом до інформаційних ресурсів, 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

Програмне забезпечення для шифрування пропонує функції симетричної та асиметричної криптографії для шифрування та захисту файлів, а також розшифровки зашифрованих файлів за допомогою відкритого чи закритого ключа. Він захищає ваші локальні файли та веб-матеріали від невиправданих шпигунів або шпигунів. Пройшовши фазу тестування та експериментування, я дізнався, як шаблони шифрування змінилися та перейшли від традиційних алгоритмів шифрування до сучасних функцій, таких як хешування, парольні фрази та блокування.

Захоплююче спостерігати, як ми пройшли довгий шлях від шифрування та захисту файлів за допомогою зашифрованого тексту до зберігання їх у хмарних сховищах або шафках. Незалежно від того, чи зберігаються мої дані локально чи в хмарі, ці найкращі інструменти шифрування розширюють свої послуги за допомогою простої та безпечної автентифікації до публічних і приватних хмар. Крім того, функція інтерфейсу відкритого ключа (PKI) допомогла мені обмінюватися захищеними даними, надавши загальний відкритий ключ одержувачу, зберігаючи його конфіденційність від інших користувачів.

Одне можна сказати напевно: це програмне забезпечення створює цифрову фортецю навколо ваших конфіденційних даних, недоступних для шпигунів чи федералів, і забезпечує наскрізний захист від зловмисного програмного забезпечення та атак грубою силою. Але хоча я ділюся своїм особистим досвідом щодо того, наскільки я задоволений послугами шифрування, я б сказав, що підписавшись на пробну версію або зареєструвавшись на спеціальну

демонстрацію, ви зможете на власні очі ознайомитися з інструментами короткого списку для процесів шифрування даних.

Я провів кілька тижнів, експериментуючи та тестуючи близько 40 найкращих програм для шифрування та оцінюючи плюси та мінуси, функції, ціни та обслуговування клієнтів, щоб скласти остаточний список із 6 найкращих систем шифрування, які забезпечують наскрізну безпеку даних і захист від шкідливого програмного забезпечення та запобігають несанкціонованому витоку даних.

На етапі тестування я також скористався допомогою штучного інтелекту, щоб звузити основні функції та варіанти використання кожного програмного модуля. Крім того, цей список також враховує реальні відгуки користувачів, потреби, больові точки та ступінь задоволення. Крім того, я також перевірив Grid Reports, щоб побачити порівняння цих програм для шифрування. Усі ці дослідження привели до створення найкращого програмного забезпечення для шифрування, створеного саме для вас.

Мої основні параметри програмного забезпечення для шифрування під час тестування

Вибір програмного забезпечення для шифрування був для мене подвійним процесом: я хотів інструменти, які пропонують наскрізне шифрування файлів і які легко налаштувати, застосувати та запустити. Я експериментував із різними локальними та хмарними пристроями з різним об’ємом пам’яті, щоб визначити, який інструмент шифрування забезпечує максимальну сумісність і можливості обміну даними для передачі даних із точки А в точку Б без втрати пакетів.

Окрім складання контрольного списку алгоритмів шифрування, присутніх у програмному забезпеченні для шифрування, я шукав додаткові інструменти plug-and-play та онлайн-сервіси резервного копіювання, щоб подвійно перевірити, що в системі не залишилося незашифрованих файлів. У зв’язку з цим я визначив відповідні параметри для кожного інструменту шифрування, які ви повинні попередньо перевірити, щоб отримати віддачу від інвестицій і створити міцне

середовище кібербезпеки для вашої компанії:

– Знищення файлів в Інтернеті: якщо інструмент шифрування залишає зашифрований файл після блокування зашифрованої версії, це може збільшити ризик шкідливого програмного забезпечення або атак грубої сили. Наявність опції подрібнення або перезапису файлів гарантує відсутність слідів оригінального файлу на диску або двійкових кодів, які можуть бути зламані або розшифровані зовнішніми хакерами. Інструменти шифрування, які пропонували додаткову інтеграцію з інструментами безпечного видалення або шредерами файлів, також потрапили до цього списку, оскільки після того, як я зашифрував файл, я міг використовувати ці інструменти для автоматичного видалення оригінального файлу.

– Алгоритми хешування: встановлення головного пароля для шафки чи сховища дуже заплутало мене, якщо в інструменті немає диспетчера паролів, і я зрештою встановив передбачувані паролі. Ось чому алгоритми хешування є обов'язковою добавкою до програмного забезпечення для шифрування, яке встановлює гарячі клавіші для спільних даних з одержувачем. Удосконалені алгоритми хешування, такі як SHA-3 або Argon2, додають додатковий рівень захисту зашифрованим файлам, щоб вони не були розшифровані, навіть якщо хтось знає пароль.

– Алгоритми шифрування: алгоритми шифрування, такі як AES, RSA або Blowfish, використовуються для зберігання конфіденційної інформації про співробітників або захисту даних за допомогою токенізації. Я використовував віртуальну приватну мережу, яка робила мої файли даних досить вразливими та схильними до вірусів, щоб я не використовував алгоритми шифрування для захисту своїх файлів від веб-трафіку. Програмне забезпечення для шифрування має запропонувати алгоритми шифрування, щоб запобігти зовнішнім перериванням або порушенням під час процесу передачі даних.

– Підтримка апаратного шифрування: ще однією типовою функцією, яку я оцінив, була можливість шифрувати та токенізувати файли в усіх апаратних

обчислювальних системах. Будь то Windows, віртуальна машина macOS чи USB-накопичувач, я відібрав інструменти, які інтегруються з такими апаратними модулями, як модулі надійної платформи (TPM) АБО модулі апаратної безпеки (HSM) для захисту цифрових файлів на різних пристроях і створення швидкого й ефективного робочого процесу шифрування для компаній.

– Інтеграція інфраструктури відкритих ключів (PKI): за допомогою функції інфраструктури відкритих ключів (PKI) я міг би зашифрувати файл за допомогою відкритого ключа та поділитися ним з одержувачем, який розшифровує його за допомогою закритого ключа. Це означає, що якщо файл відкритий, будь-хто може отримати до нього доступ за допомогою відкритого ключа, але якщо він заблокований, він абсолютно недоступний для всіх, і тільки я можу відкрити його за допомогою закритого ключа. Ця система ідеально підходить для безпечного зв'язку, де я можу вільно ділитися відкритими ключами, не турбуючись, що всі прочитають вміст файлу, призначеного для мене.

– Повне шифрування диска та розділів: повне шифрування диска робить усе: шифрує дискові файли, видаляє дублікати файлів і перезаписує дисковий простір новими папками для забезпечення більшої цілісності даних. Шифрування розділів також корисне, оскільки я можу вибирати, стискати та шифрувати лише важливі документи на своєму диску та зберігати їх у сховищі, і ця функція є найбільш вигідною, якщо інші також мають доступ до вашого диска. Повне шифрування диска є важливою функцією, оскільки воно блокує та захищає всі файли та папки, збережені на диску, і навіть дозволяє хмарне шифрування для хмарних дисків.

– Депонування ключа безпеки: Депонування ключа безпеки – це моя мережа безпеки. Таке відчуття, ніби ви надаєте надійну інформацію про свої дані відомим зацікавленим сторонам і ділитеся ключем шифрування через іншу систему обміну повідомленнями, щоб мати резервну копію на випадок, якщо я забуду ключ. Для тих, хто має проблеми з керуванням декількома паролями, як я, це позбавить мене від клопоту щодо налаштування відновлення пароля, оскільки

я можу надати ключ авторизованим учасникам для доступу до зашифрованих файлів і папок.

– Безпечне завантаження та інтеграція мікропрограми: Безпечне завантаження – це довіра. Коли я вмикаю свій пристрій, безпечне завантаження гарантує, що завантажуються лише надійні операційні системи та ядра. Це запобігає будь-якому зовнішньому втручанню або стеження під час завантаження операційної системи для захисту апаратних і програмних файлів. Програмне забезпечення для шифрування, яке забезпечує захист вбудованого програмного забезпечення, має важливе значення, якщо ви хочете додати кілька рівнів захисту до ваших локальних файлів і веб-файлів.

– Розбиття ключів: деякі інструменти шифрування повністю звели нанівець будь-які можливості крадіжки даних, оскільки вони не лише керували моїм паролем для сховищ, у яких зберігалися мої документи, але й розділяли ключ на частини та зберігали кожну частину ключа доступу в іншому місці. Лише я зміг зібрати свій ключ і захистити свої файли та папки від доступу під час моєї відсутності.

Ці параметри відрізняють інструмент шифрування від шуму різних систем кібербезпеки на ринку. Тепер, маючи мій особистий досвід випробування та дослідження найкращих 7 інструментів шифрування з 40+ інструментів, які я відібрав, я впевнений, що кожне з цих рішень розроблено для захисту ваших локальних баз даних і баз даних хмарного сервера та встановлює високі стандарти безпеки даних, одночасно зменшуючи витік даних або несанкціонований доступ.

Хоча наведені вище функції дали мені необхідні переваги, забезпечуючи плавний і безпечний обмін даними та створюючи сховища для моєї зручності, компанії мають інший процес вибору відповідного інструменту. Перш ніж потрапити до цього списку, окресліть свої вимоги, бюджет, робочі процеси мережевої архітектури та існуючі рішення для аналізу найбільш прийнятного програмного забезпечення для шифрування.

Список нижче містить справжні відгуки користувачів про програмне

забезпечення для шифрування. Важливо зазначити, що постачальники, які пропонують безкоштовну пробну версію, також вважаються безкоштовними в цьому контексті.

Щоб бути включеним до цієї категорії, рішення має:

– Захистіть дані та файли за допомогою зашифрованого тексту.

– Підготуйте до шифрування дані в стані спокою, дані в дорозі або дані, що використовуються.

– Дозволити користувачам вибирати та керувати своїми файлами в налаштуваннях шифрування..

Signal

З мого досвіду Signal був ідеальною системою обміну миттєвими повідомленнями для надсилання й отримання зашифрованих файлів, а також для редагування чи зміни розшифрованих файлів. Оскільки я помітив, що важко надсилати зашифровані файли одержувачам за допомогою інших інструментів, Signal надав службу обміну миттєвими повідомленнями для вкладення користувальницьких документів із зашифрованими вкладеннями та обміну ними на будь-якій хмарній платформі, як-от Dropbox, Google Drive тощо.

Найбільшою перевагою Signal є його наскрізне шифрування. Він блокує дані ваших повідомлень, записи дзвінків і збережені файли в безпечному сховищі. Мені не потрібно було турбуватися про те, що хтось підглядає за моїми розмовами, і це мене заспокоює, оскільки сьогодні особисті дані вразливі для всіх глобальних серверів.

Я також дізнався, що він використовує протокол Signal для захисту файлів під час завантаження чи завантаження. Однією з перших речей, які я помітив, був простий і чистий інтерфейс. Він не захаращений непотрібними функціями, тому ви можете шифрувати та надсилати свої документи чи файли без жодних проблем.

Незважаючи на те, що сигнал не переповнює вас можливостями, він все одно пропонує все, що мені потрібно від платформи обміну миттєвими

повідомленнями. Він може знищувати незашифровані повідомлення чи файли, знищувати дані магнітних файлів, входити з мобільного чи будь-якого іншого гаджета та створювати групові чати для надсилання та отримання зашифрованих файлів.

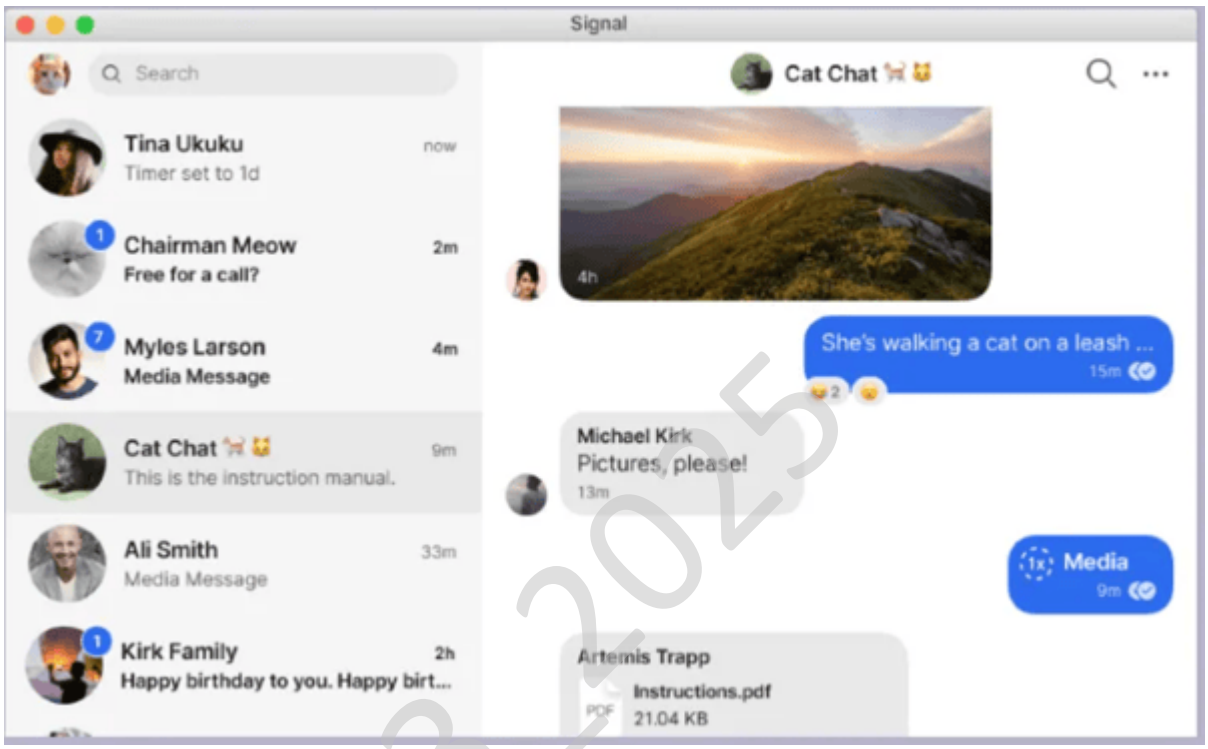

Рисунок 2.1 – Інтерфейс користувача Signal

Однак, хоча ви можете завантажувати документи на замовлення, Signal не міг надати мені жодної підтримки для зберігання на диску чи хмарного сховища. Він не забезпечує шифрування для локальних системних файлів, доки ви не налаштуєте його як форматований текстовий документ і не завантажите на сервер Signal.

Загалом програма була проста у використанні, але є крива навчання для таких функцій, як зникнення повідомлень або керування резервними копіями. Крім того, якщо особа, якій ви надсилаєте повідомлення, не користується Signal, чати не будуть зашифровані.

Сигнал не забезпечує покриття алгоритму шифрування чи менеджера

паролів. Загалом, хоча я вважаю, що це гарна платформа для ділового спілкування, йому дещо бракує з точки зору підтримки конфіденційності та безпеки даних.

Microsoft Bitlocker

Якщо є програма, яка виділяється тим, що шифрує файли та пропонує високотехнічний шифр для захисту даних компанії, для мене це не що інше, як Microsoft Bitlocker.

Microsoft Bitlocker – це вбудований інструмент шифрування Microsoft для Windows. Він може шифрувати файли або навіть просто блокувати незашифровані файли без їх шифрування.

Якщо шафка відкрита, я можу переглядати всі файли та папки. Але якщо шафка закрита, дані залишаються невловимими та непростежуваними. Він використовує алгоритм розширеного стандарту шифрування (AES) із 128-бітними або 256-бітними ключами для захисту ваших даних від сторонніх очей. Він також пропонує додаткові служби безпечного онлайн-видалення та резервного копіювання.

Я також розгорнув Microsoft Bitlocker у різних операційних системах Windows, але з'ясував, що він ідеально інтегрований із Windows 10 або Windows 11. Вам не потрібна додаткова інтеграція для налаштування та запуску програми.

Іншою ключовою функцією було те, що я міг зберігати всі свої паролі у вбудованому менеджері паролів, додавати рівні хешування до шафок і навіть входити в хмарну консоль, щоб отримати доступ до всіх збережених шафок. Microsoft Bitlocker забезпечив повну хмарну сумісність і тісні служби шифрування для моїх файлів.

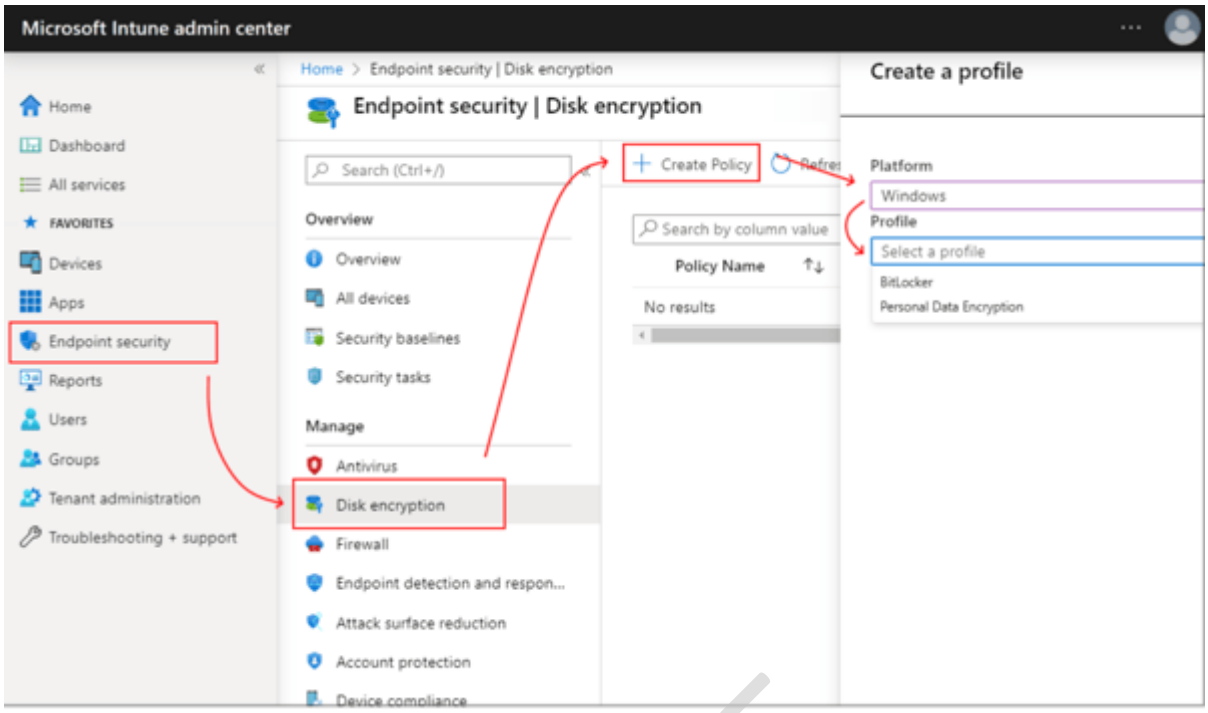

Рисунок 2.2 – Інтерфейс користувача Microsoft Bitlocker

Якщо говорити про паролі, то була різниця між тим, як встановити ключі від шафок. Якщо ви забудете ключ відновлення, ваші дані зникнуть назавжди. Хоча це робить Bitlocker безпечним, на комп'ютері не було вихідного сліду мого файлу, окрім зашифрованого файлу, до якого я не міг отримати доступ.

Якщо ви користуєтеся MacOS або Android, ви не зможете запустити Microsoft Bitlocker. Я також не міг поділитися будь-якими файлами чи папками з користувачами операційної системи, відмінної від Windows.

Я міг помітити, що деякі функції також не були повністю зручними для користувача. Коли мою систему потрібно було оновити для перевірки TPM, увесь процес нагадував програму інтерфейсу командного рядка, яку було важко зрозуміти. Мені не вдалося завершити процес відновлення пароля, щоб розблокувати свої файли без сторонньої допомоги.

Progress MOVEit

Як тільки я протестував Progress MOVEit, я припустив, що це буде ідеальний вибір для команд із кібербезпеки середнього та корпоративного рівня.

Він керує, контролює та автоматизує всі процеси обміну та передачі файлів на одній централізованій платформі.

Незалежно від того, чи йдеться про послуги шифрування, автоматизацію файлів, відстеження рівня активності чи підтримку угод про рівень обслуговування, Progress MOVEit не має браку функцій. Він використовує розширені протоколи передачі файлів і асиметричну криптографію для доставки файлів з однієї системи в іншу.

Крім того, це також дозволило мені підключити нових партнерів із обробки даних, налаштувати вхід у хмарну шафку та запустити резервну синхронізацію незашифрованих документів, щоб гарантувати, що дані не буде втрачено. Від чистого інтерфейсу до інтуїтивно зрозумілих робочих процесів вам не потрібно бути IT-генієм, щоб працювати з цією платформою. Я міг ділитися зашифрованими файлами зі швидким оборотом і навіть налаштувати автоматизацію робочого процесу, щоб зменшити ручний моніторинг даних і діагностику.

Щоразу, коли я стикався з проблемою, їхня служба підтримки клієнтів завжди була на місці. Крім того, база знань, яку він надає, є скарбницею корисної інформації, яка ідеально підходить для самостійного вирішення незначних проблем. Незалежно від того, чи виконуєте ви звичайні передачі чи міграцію великих даних, Progress MOVEit є надійним і надійним інструментом для роботи.

Але, як і будь-яке інше програмне забезпечення для шифрування, Progress MOVEit не ідеальне. Моя система часто відставала під час тривалих оновлень системи, що не давало мені можливості шифрувати файли в потрібний час.

Мені також було важко отримати короткий виклад кількості відсканованих дисків, деталі останнього резервного копіювання, дату й час у формі настроюваного аналітичного звіту. Якщо ви покладаєтеся на детальну аналітику, вам може не вистачити поточних пропозицій.

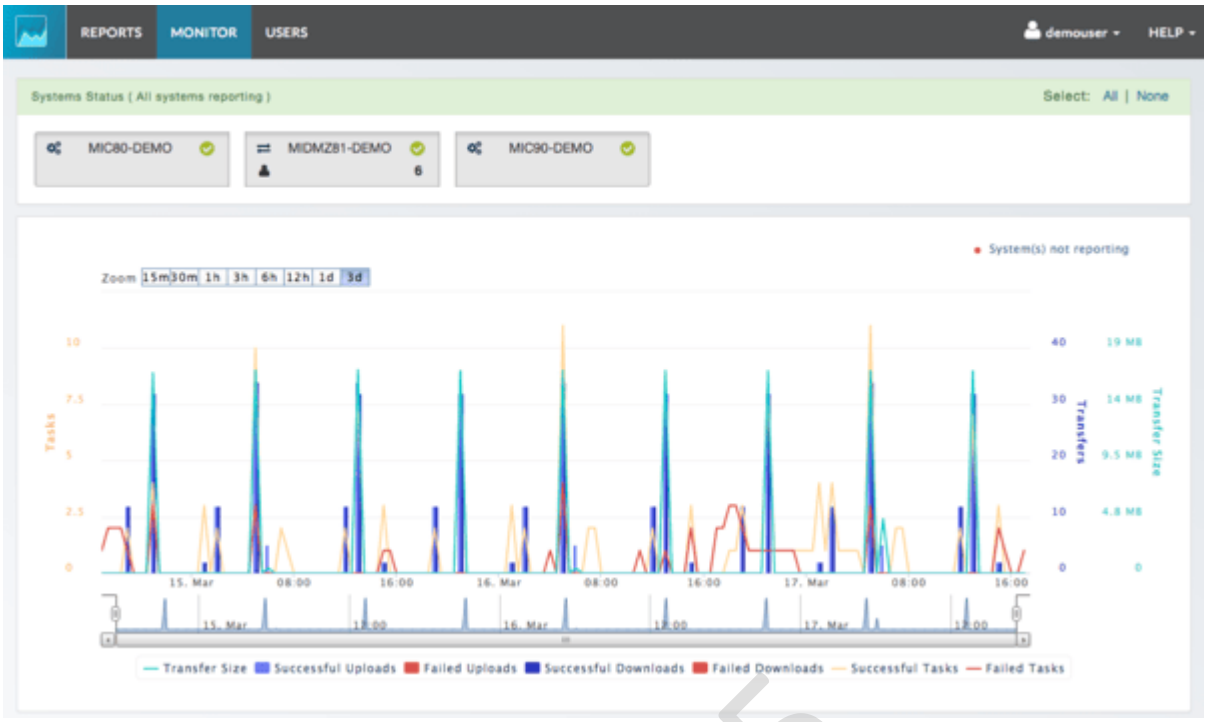

Рисунок 2.3 – Інтерфейс користувача Progress MOVEit

Я не заперечую незламні стандарти шифрування Progress MOVEit, але минулі інциденти безпеки змусили деяких користувачів бути обережними. Йому потрібен надійніший пароль і механізм хешування, щоб стиснути й захистити кілька файлів і створити більше довіри.

FileVault

FileVault не лише запропонував послуги шифрування для моїх окремих файлів, але й розширив службу шифрування на весь диск. FileVault пропонує такі функції шифрування, як контейнери для зберігання, сховища та шафки, які можуть зберігати будь-що, від документів, що засвідчують особу, до інформації про кредитну картку.

FileVault – це вбудована функція шифрування macOS. Він надає такі послуги, як повне шифрування диска, онлайн-подрібнення файлів, безпечний протокол файлів, резервне копіювання та відновлення онлайн. Це безкоштовно, входить до кожного Mac і не потребує додаткового налаштування.

Що мені найбільше подобається у FileVault, так це його герметичний

захист. Він використовує шифрування XTS-AES-128 із 256-бітним ключем, що звучить супертехнічно – і це так – але якщо говорити простою мовою, це означає, що мої дані заблоковано, як Форт-Нокс, головним паролем.

Будь то особисті документи, робочі файли чи щось інше, FileVault гарантує, що файли зашифровано та перетворено на шифр, щоб захистити їх від неетичних хижаків. Усі дані пов'язані з обліковими даними мого Mac, тобто лише я або хтось із моїм закритим ключем може отримати до них доступ.

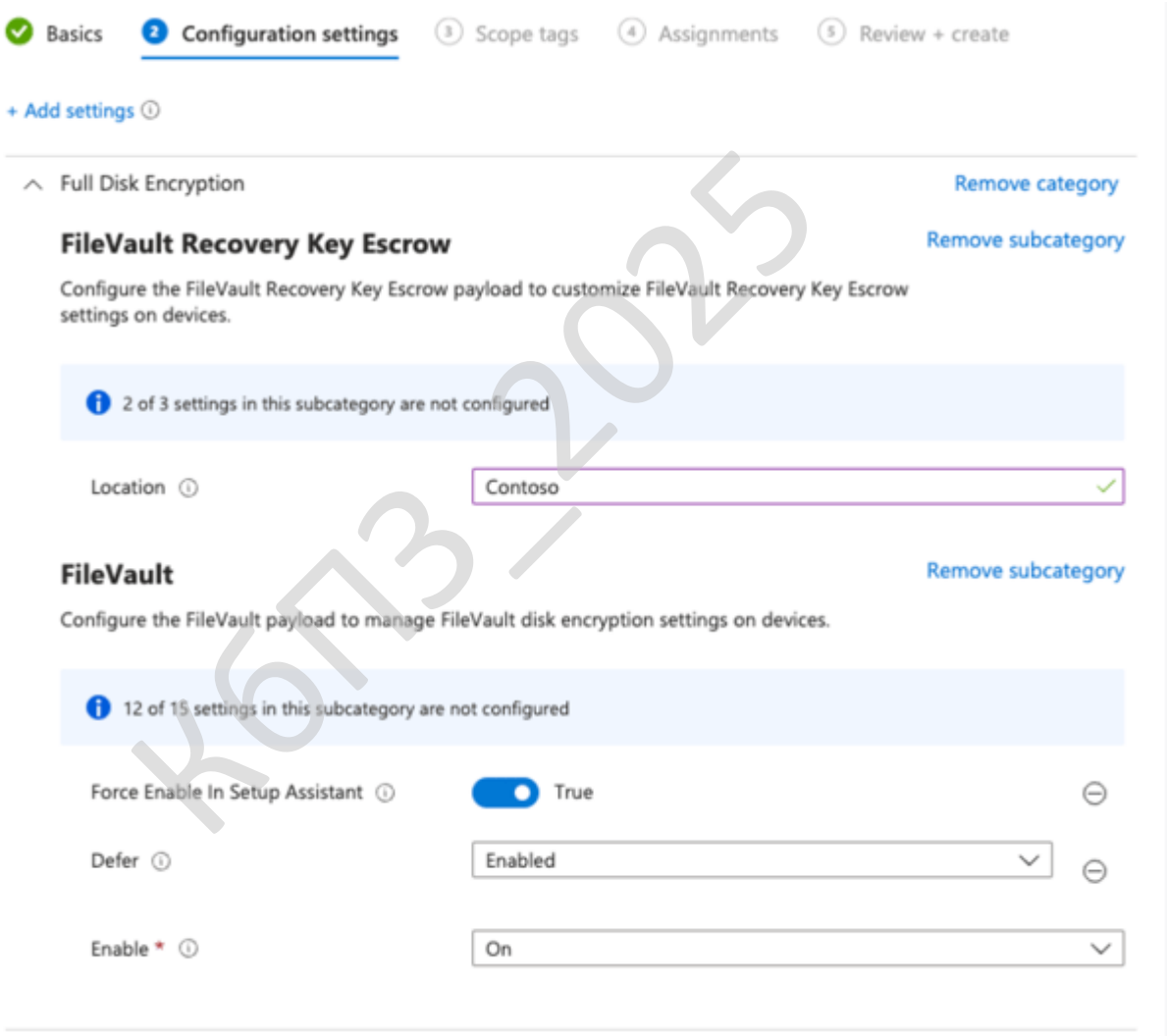

Рисунок 2.4 – Інтерфейс користувача FileVault

Тим не менш, у FileVault є недоліки. Одне, що я помітив, це незначне зниження продуктивності, коли ввімкнено FileVault. Для мене це не перешкода,

але якщо ви працюєте зі старішою macOS або програмами, які потребують великих ресурсів, система може відставати або бути вразливою до зовнішніх помилок.

Ще одна проблема – доступність. Хоча це чудово на моєму Mac, немає мобільної програми для запуску програми та доступу до зашифрованих файлів. Якщо одним Mac користується кілька людей, буде важко налаштувати маскування даних і захистити дані від інших користувачів.

Звичайно, FileVault не ідеальний, але він забезпечує найвищий ступінь захисту кінцевої точки та шифрування ваших локальних і глобальних файлів.

Virtru

З мого досвіду, Virtru має бути найуніверсальнішим інструментом шифрування електронної пошти для передачі ваших файлів електронною поштою з одного робочого столу на інший.

Virtru захищає вміст електронної пошти, який було надіслано, а також надає мені контрольний журнал вкладень, безпечно надісланих сервером електронної пошти. Мені особливо подобається, як він легко інтегрується з такими платформами, як Gmail або іншими клієнтами електронної пошти. Вам просто потрібно ввімкнути його, і він забезпечить оптимальну безпеку та підтримку алгоритму шифрування для ваших вкладень електронної пошти.

Virtru також використовує наскрізне шифрування файлів, щоб гарантувати, що ви та ваш одержувач матимете доступ до електронної пошти. Мені просто потрібно було встановити відкритий ключ для шифрування електронної пошти та поділитися ним через пристрій обміну миттєвими повідомленнями з користувачем, який може ввести пароль і відкрити вкладення. Це здавалося легким, зручним і безпечним водночас.

Мене також вразив набір інтеграцій Virtru в реальному часі, особливо ті, що використовують Microsoft Outlook і Google Workspace. Йдеться не лише про блокування вашої власної папки "Вхідні", але й про захист конфіденційних даних, як-от ділові дані, особисті відомості про транзакції тощо.

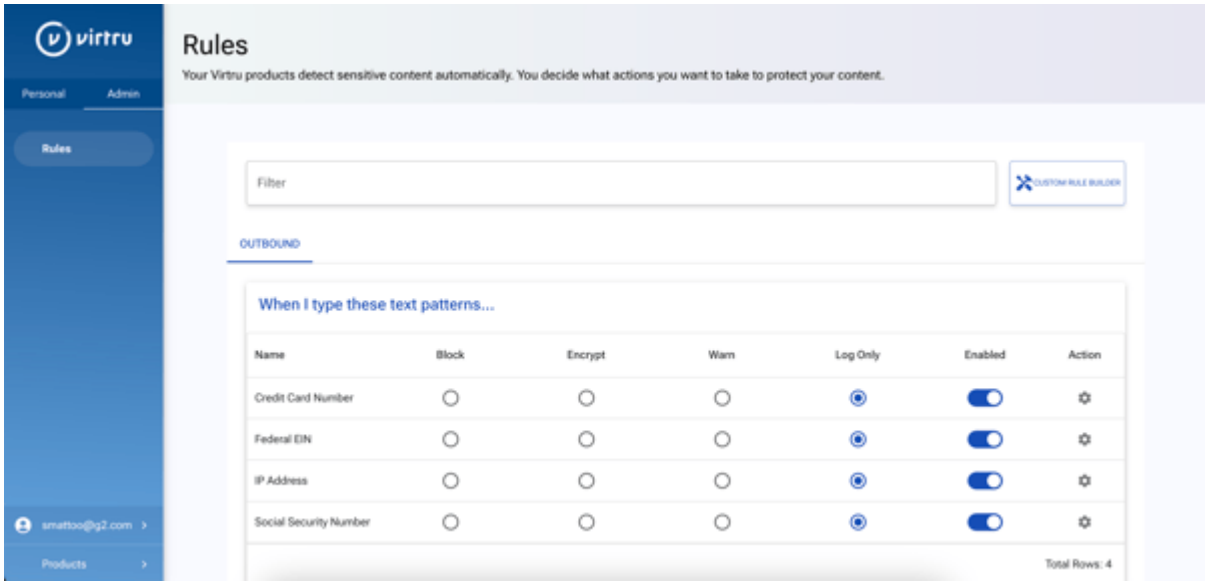

Рисунок 2.5 – Інтерфейс користувача Virtru

Але, коли я спробував отримати доступ до Virtru на мобільному пристрої, я зіткнувся з проблемами сумісності. Іноді мої електронні листи не відкривалися так гладко, як мали б, що трохи засмучувало на ходу.

Якщо ви ділитеся файлами, Virtru забезпечить вас шифруванням електронної пошти та детальним контролем доступу. Але що, якщо розмір файлу занадто великий? Я не міг приєднати документи форматованого тексту з кількома зашифрованими вкладеннями до електронного листа, який повідомляв, що інструмент не може надсилати чи отримувати файли, розмір яких перевищує певний.

Тим не менш, мінуси Virtru не затьмарюють його блискучі плюси. Для таких, як я, хто трохи параноїк щодо безпеки даних електронної пошти, це здавалося чистим золотом. Крім того, я вважаю, що він ідеально підходить для галузей, які часто обмінюються конфіденційними даними, як-от охорона здоров’я, банки, роздрібна торгівля та компанії електронної комерції.

Tresorit

Якщо ви маніяк безпеки, як я, Tresorit стане єдиним магазином для редагування, керування та дешифрування файлів із покращеним покриттям

безпеки.

Tresorit – це інструмент для спільної роботи над контентом, який дає змогу взаємодіяти, запускати розмови, ділитися вкладеннями та ресурсами, а також керувати журналами аудиту даних для процесів вашої компанії. Я міг заблокувати отримані дані в папках і приховати їх у головному меню, щоб вони залишалися недоступними для інших співавторів вмісту.

Для компаній і всіх, хто хоче відповідати вимогам GDPR або HIPAA, Tresorit пропонує параметри резидентності даних для токенізації файлів даних і доставки їх через захищені мережі.

Іншою функцією, яку я вважав дуже зручною, було встановлення безпечних посилань на депозит. Це дозволяє людям безпечно надсилати мені файли. І їм до смішного легко ним користуватися. Крім того, у моєму розпорядженні було кілька варіантів, як-от шифрування файлів і папок, подрібнення файлів, перезапис файлів і доступ до сховищ через командний рядок для керування та захисту цифрових файлів і відстеження активності користувачів у реальному часі в усіх офісах компанії.

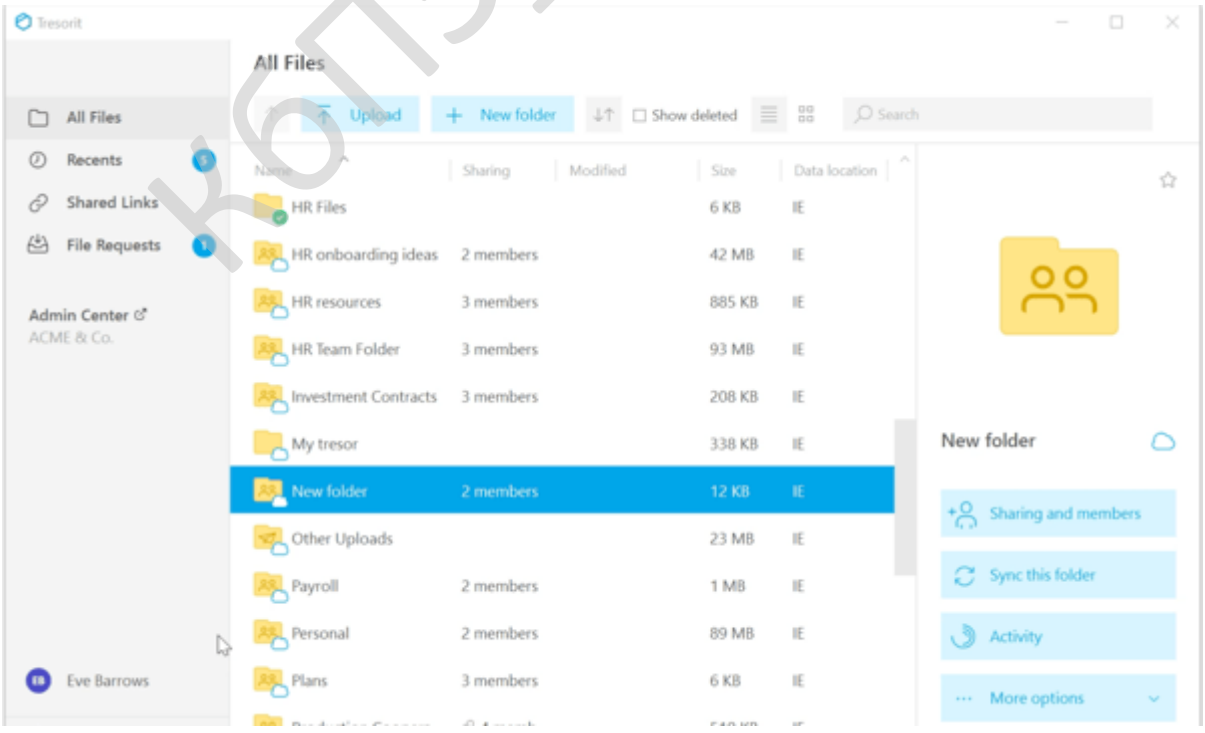

Рисунок 2.6 – Інтерфейс користувача Tresorit

Tresorit здавався чудовою системою кібербезпеки для керування контентом, але є деякі очевидні мінуси. Я припускаю, що підписка буде дорогою для фрілансерів або малого бізнесу, який працює з обмеженим бюджетом.

Завантажувати файли для використання в автономному режимі також було не так просто, як хотілося б, а керувати файлами без підключення до Інтернету буде незручно. Якщо ви інтегруєте Tresorit зі своєю віртуальною приватною мережею, їй потрібні ваші IP-адреси та веб-адреси «https», щоб захистити вас від трафіку сервера.

Я також помітив, що процес синхронізації або шифрування займає трохи часу та затримує ваші бізнес-операції. Це не порушує угоду, але якщо ви працюєте в стислі терміни для доступу та редагування нових баз даних, це може стати перешкодою.

2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки та мови програмування

Python – високорівнева мова програмування, яку називають другою за популярністю в світі. Її використовують для розробки вебзастосунків, програмного забезпечення, машинного навчання. Python застосовують для вирішення робочих завдань у компаніях Google, Instagram, Facebook, IBM, NASA, Dropbox, Netflix та інших. Розробники цінують цю мову програмування за простоту у вивченні, ефективність та мультиплатформність.

Python – скриптова мова програмування з досить простим синтаксисом. Для розуміння достатньо порівняти принципи написання найпростішої програми, яка виводить на екран текстове повідомлення. Саме тому мова програмування Python більш доступна для новачків, а професіонали встигли адаптувати її для вирішення великої кількості завдань. Це мультиплатформне рішення, тому знання Python дає можливість працювати у різних сферах: від розробки мобільних застосунків до ігрової індустрії та штучного інтелекту.

У мови програмування динамічна типізація: є можливість передавати до функцій будь-який тип даних без попереднього вказання. Інтерпретованість дозволяє знаходити помилки у коді ще до повної збірки у робочий застосунок. При цьому Python дуже чітко дає зрозуміти, де та через що виникла помилка.

Це мова об'єктноорієнтованого програмування (ООП). Програмне забезпечення на Python оформлене у вигляді моделей, які можуть бути зібраними у пакети. Тип та структуру кожного об’єкта можна запитати під час виконання програми. Для кожного з об’єктів можна отримати всю інформацію щодо його внутрішньої структури. Окрім того:

– у мови логічний синтаксис, завдяки чому вихідний код легко читати та розуміти;

– гнучкість та масштабованість Python дозволяє адаптувати високорівневу логіку та розширяти складні застосунки, як тільки виникне така необхідність;

– розробка на Python у більшості випадків проходить швидше, ніж на інших мовах програмування;

– Python – інтерпретована мова програмування. Це значить, що код можна написати у будь-якому текстовому файлі на будь-якій платформі, і потім успішно запустити;

– у Python – колосальна спільнота однодумців. Тож будь-які складнощі конкретних розробників вирішуються колективно.

Проте є декілька особливостей, які можна віднести до недоліків. Це повільність (ця мова програмування хоч і універсальна, проте повільніша за інші), велика кількість ресурсів, необхідних для роботи та «прив’язаність» до системних бібліотек.

Мова програмування Python використовується у наступних сферах:

1. Розробка програмних застосунків будь-якого напрямку.

2. Розробка серверної частини мобільних застосунків (найпопулярніший напрямок).

3. Ігри. Багато сучасних ігор для комп'ютерів (наприклад, World of Tanks) частково чи повністю написані на Python.

4. Вбудовані системи для різних пристроїв. Дуже часто Python використовують для написання внутрішніх платформ управління банкоматами.

5. Скрипти та плагіни до уже реалізованих програм для автоматизації процесів чи створення інших рішень.

6. Тестування (автоматизація цього процесу).

7. Машинне навчання. – основна мова для написання алгоритмів і аналітичних застосунків у сфері Machine Learning.

Бібліотеки Python

Різні бібліотеки Python використовують для виконання конкретних завдань. Наприклад, Matplotlib підходить для відображення даних у двовимірній та тривимірній графіці. Pandas підходить для зручної роботи з даними. NumPy дозволяє створювати масиви та керувати ними. Requests використовується для веброзробки. OpenCV-Python відкриває можливості для обробки зображень з метою оптимізації систем «машинного зору».

Найвідоміші фреймворки для мови програмування Python

Фреймворки Python допомагають створити зручне та функціональне середовище для розробки. У них міститься набір інструментів, модулів та бібліотек, корисних для виконання конкретних завдань. Це значно полегшує роботу: наприклад, дає змогу не витрачати час на розписування дій, які повторюються, а використати релевантний інструмент. Тож є можливість позбутися рутинних процесів та сконцентруватися на логіці проєкту.

Серед найпопулярніших фреймворків для Python:

– Django – найстаріший та найвідоміший. Створений для реалізації великих інтерактивних проєктів;

– Pyramid – зручний у налаштуваннях, і дає можливість реалізувати складні нестандартні ідеї;

– Web2py – підходить в першу чергу для вебзастосунків і може використовуватись на будь-яких архітектурах.

Популярні Python IDE

IDE або інтегровані середовища розробки – це програмне забезпечення, яке надає розробникам необхідні інструменти для написання, редагування, тестування та налаштування коду. Для розробки на Python найчастіше використовують IDE PyCharm, IDLE, Spyder та Atom.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи кібербезпеки для управління ідентифікацією та доступом до інформаційних ресурсів.

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

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

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

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

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

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Цифровий підпис

Застосовувані в смарт-картах і токенах технології встановлення дійсності відправника стають усе більше зрілими, практичними й зручними. Наприклад, їх можна використовувати як механізм автентифікації на Web-ресурсах, в електронних сервісах і платіжних додатках з ЕЦП.

Цифровий підпис співвідносить конкретного користувача із приватним ключем асиметричного шифрування. Приєднується до електронного повідомлення, вона дозволяє одержувачеві впевнитися, чи є його відправник тим, за кого себе видає. Форми шифрування при цьому можуть бути різними.

Приватний ключ, у якого тільки один власник, використовується для цифрового підпису й шифрування при передачі даних. Саме повідомлення може бути прочитане кожним, хто має відкритий ключ. Електронний цифровий підпис може генеруватися USB-токеном – апаратним пристроєм, що формує пари ключів. Іноді різні методи автентифікації комбінуються – наприклад, смарт-карту доповнюють токеном із криптографічним ключем, а для доступу до останнього передбачається уведення PIN-коду (двофакторна автентифікація). При використанні токенів і смарт-карт закритий ключ підпису не залишає меж токена. Таким чином, виключається можливість компрометації ключа.

Застосування ЕЦП забезпечується спеціальними засобами й технологіями, що становлять інфраструктуру відкритих ключів (Public Key Infrastructure, PKI). Основним компонентом PKI є центр, що засвідчує, що відповідає за видачу ключів і гарантує дійсність сертифікатів, що підтверджують відповідність між відкритим ключем і інформацією, що ідентифікує власника ключа.

Розроблювачі пропонують різні компоненти для вбудовування криптографічних функцій в Web-додатки – наприклад, SDK і модулі, які підключаються до браузерів із програмним інтерфейсом доступу до криптографічних функцій. З їхньою допомогою можна реалізовувати функції шифрування, автентифікації й ЕЦП із високим рівнем безпеки. Засоби цифрового підпису надаються також у вигляді онлайнових сервісів.

ЕЦП як сервіс

Для автоматизації підпису, обміну й зберігання електронних документів можна скористатися онлайновим сервісом. Всі реєструємі в ньому електронні документи захищаються цифровим підписом, причому формування й перевірка ЕЦП здійснюються на стороні клієнта за допомогою криптографічного модуля для браузера, установлюваного при першому звертанні до порталу. Робочі станції взаємодіють із Web-сервером порталу по протоколі TLS у режимі однобічної автентифікації сервера. Автентифікація користувачів виконується на основі персональних ідентифікаторів і паролів, переданих на сервер після встановлення захищеного з'єднання. Сервіс підтримується інфраструктурою відкритих ключів на базі центра, що засвідчує, але можуть прийматися й сертифікати, випущені іншим центром, що засвідчує.

Інфраструктура PKI є найкращою схемою для безпечної автентифікації користувачів. Нічого надійніше цифрових сертифікатів у даній області ще не придумали. Це можуть бути сертифікати державного зразка для фізичних осіб для застосування в хмарному сервісі або корпоративні цифрові сертифікати для реалізації моделі BYOD. Причому перші могли б використовуватися для доступу до внутрикорпоративних ресурсів, якби комерційні компанії мали можливість підключатися до державних центрів, що засвідчують. Коли цифрові сертифікати будуть застосовуватися скрізь, де необхідна автентифікація користувачів, життя звичайних громадян помітно спроститься. Надійне ж і безпечне зберігання цифрових сертифікатів на даний момент забезпечується тільки одним способом – за допомогою смарт-карт і токенів.

З огляду на підвищену увагу держави до таких напрямків, як цифровий підпис і смарт-карти, а також важливість наявності надійної й зручної багатофакторної автентифікації в різних інформаційних системах і розвитку електронних платіжних систем і юридично значимого електронного документообігу, вітчизняні розроблювачі продовжують удосконалювати свої рішення й розширювати лінійки продуктів.

USB-токени й смарт-карти

Смарт-карти й USB-токени можуть служити персональним засобом автентифікації й електронного підпису для організації захищеного і юридично значимого документообігу. Більше того, їхнє застосування дозволяє уніфікувати засобу автентифікації – починаючи з операційних систем і закінчуючи системами контролю доступу в приміщення.

Українські розроблювачі програмно-апаратних засобів інформаційної безпеки нагромадили солідний досвід у створенні електронних ключів (токенів) і ідентифікаторів. У продуктах із двофакторною автентифікацією (пристрій і PIN-код) використовуються 32-розрядні мікропроцесори ARM для генерації ключових пар, формування й перевірки електронного підпису (алгоритм ДСТУ 4145-2002), а також захищені мікроконтролери з енергонезалежною пам'яттю для зберігання користувальницьких даних.

При формуванні ЕЦП використовується криптографія на еліптичних кривих. Пропонований модуль, який підключається, для браузерів (для його установки не потрібні права адміністратора) уміє працювати з USB-токеном і має програмний інтерфейс доступу до криптографічних функцій. Плагін дозволяє інтегрувати із системами дистанційного банківського обслуговування й електронного документообігу.

Пристрій PINPad (ЕЦП із екраном) дозволяє візуалізувати підписується документ, що, перед застосуванням електронного підпису й тим самим захиститися від атак, чинених за допомогою засобів віддаленого керування з метою підміни вмісту документа при передачі на підпис.

Пристрою апаратно реалізують українські криптоалгоритми й сертифіковані СБУ як персональні засоби електронного підпису. Таким чином, вони можуть застосовуватися при автентифікації з використанням механізмів електронного підпису, для формування електронного підпису на документах або підтвердження різних операцій в інформаційних системах, а також при роботі із хмарними сервісами.

У пристроях криптоалгоритми реалізовані на рівні мікропроцесора, а крім того, задіяна схема роботи із закритим ключем, що не витягається, підпису. Такий підхід виключає можливість розкрадання закритого ключа підпису, а формування ЕЦП із його використанням виконується усередині пристрою. Ключі, що втримуються, і сертифікати можуть застосовуватися для строгої двофакторної автентифікації й формування посиленого кваліфікованого електронного підпису відразу в декількох інформаційних системах, що працюють у рамках однієї або більше інфраструктур PKI.

Пристрою автентифікації випускаються в різних форм-факторах, але мають однакову функціональність, що дозволяє застосовувати ті самі механізми автентифікації в багатьох інформаційних системах, на Web-порталах, у хмарних сервісах і мобільних додатках.

Підхід, при якому однакові пристрої використовуються для рішення ряду завдань, здобуває останнім часом все більшу популярність. Завдяки наявності в смарт-карті однієї або двох міток RFID і інтеграції зі СКУД, її можна застосовувати для контролю доступу в приміщення. А підтримка закордонних криптоалгоритмів (RSA) і інтеграція з більшістю продуктів світових вендоров (Microsoft, Citrix, VMware, Wyse і ін.) дають можливість використовувати смарт-карту як засіб строгої автентифікації в інфраструктурних корпоративних рішеннях, включаючи вхід користувача в Microsoft Windows, роботу з VDI і інші популярні сценарії. Програмне забезпечення допрацьовувати не потрібно – підтримка включається шляхом застосування певних настроювань і політик.

Біометричні методи автентифікації

Системи двофакторної автентифікації застосовуються в таких областях, як електронна комерція, включаючи інтернет-банкінг, і автентифікація при віддаленому доступі з недовіренного робітника місця. Більш строгу автентифікацію забезпечують біометричні методи. Такі системи реалізують доступ по відбитку пальця, геометрії особи, відбитку або рисунку вен долоні, структурі сітківки ока, рисунку райдужної оболонки ока й голосу та ін. Біометричні методи постійно вдосконалюються – вартість відповідних рішень знижується, а їхня надійність підвищується. Найбільш затребувані й надійні технології – біометрична автентифікація з використанням відбитка пальця й райдужної оболонки ока. Системи сканування відбитка пальця й розпізнавання геометрії особи застосовуються навіть у споживчих пристроях – смартфонах і ноутбуках.

Автентифікація з використанням біометрії дозволяє підвищити рівень безпеки при критично важливих операціях. Надання доступу людині, що не має на це права, практично виключено, однак помилкова відмова в доступі трапляється досить часто. Щоб уникнути таких непорозумінь, можна застосовувати системи багатофакторної автентифікації, коли особистість ідентифікується, наприклад, і по відбитку пальця, і по геометрії особи. Загальний ступінь надійності системи росте пропорційно числу використовуваних факторів.

Крім того, при використанні біометрії для доступу до ключів і сертифікатів на смарт-карті робота з останньої й процес автентифікації спрощуються: замість уведення складного пароля досить доторкнутися пальцем до сканера.

3.2 Розробка структурної схеми

Автоматизувати роботу адміністратора безпеки й швидко реагувати на зміни політик безпеки допомагають системи централізованого керування

життєвим циклом засобів автентифікації й ЕЦП. Система призначена для обліку й реєстрації всіх апаратних і програмних засобів автентифікації й зберігання ключової інформації, використовуваних співробітниками в масштабах підприємства.

Її завдання – керування життєвим циклом цих засобів, аудит їхнього використання, підготовка звітів, відновлення автентифікаційних даних, надання або відкликання прав доступу до додатків при зміні службових обов'язків або звільненні співробітника, заміна пристрою у випадку його втрати або ушкодження, вивід устаткування з експлуатації. З метою аудита засобів автентифікації фіксуються всі факти використання пристрою на комп'ютері підприємства й зміни даних, що зберігаються в його пам'яті.

За допомогою системи реалізуються також технічна підтримка й супровід користувачів: заміна забутого PIN-коду, синхронізація генератора одноразових паролів, обробка типових ситуацій втрати або поломки токена. А завдяки програмному віртуальному токену користувач, що перебуває поза офісом, навіть у випадку втрати токена може продовжити роботу з комп'ютером або одержати безпечний доступ до ресурсів без зниження рівня захищеності.

Система підвищує рівень корпоративної безпеки завдяки використанню сертифікатів відкритого ключа стандарту X.509 і зберіганню закритих ключів у захищеній пам'яті смарт-карт і USB-токенів. Вона спрощує впровадження й експлуатацію рішень PKI з використанням USB-токенів і смарт-карт за рахунок автоматизації типових процедур адміністрування й аудита.

Сьогодні усе більше важливу роль здобувають системи ідентифікації й керування доступом до інформаційних ресурсів підприємства (IDM). Розроблений продукт відповідає вітчизняній нормативній базі в області ІБ, підтримує вітчизняні центри, що засвідчують, смарт-карти й токени, забезпечує технологічну незалежність у такій важливій області, як комплексне керування доступом до конфіденційної інформації.

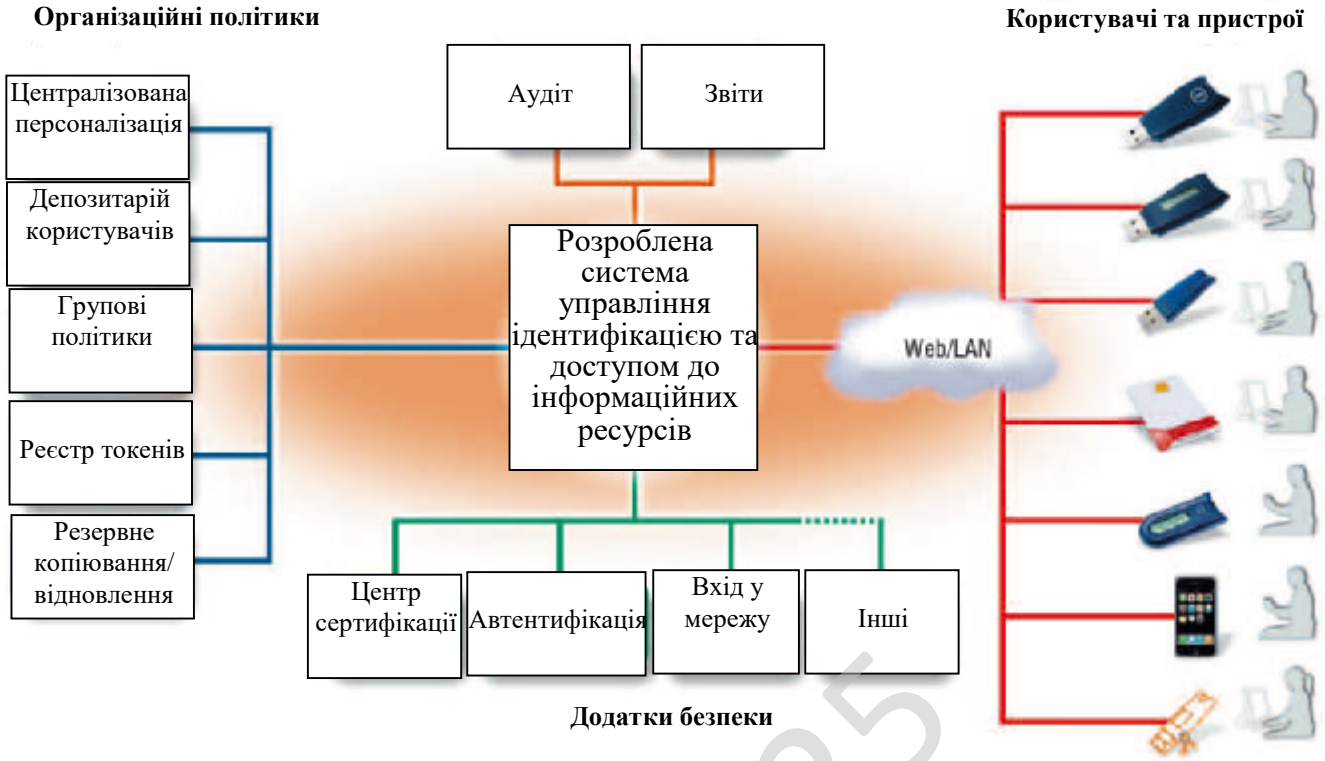

Рисунок 3.1 – Структурна схема системи

Програмне забезпечення вирішує завдання комплексного керування доступом: IDM використовується як центральну ланку корпоративної системи ІБ, з якої інтегруються інфраструктури PKI і єдиної авторизації (Single Sign On, SSO). ПЗ передбачає підтримку двох- і трифакторної автентифікації й біометричних технологій ідентифікації, реалізацію SSO для мобільних платформ Android і iOS, а також коннектори (модулі сполучення) з різними додатками.

На українському ринку популярність інтегрованих систем, що включають IDM, СКУД і апаратні засоби біометричної автентифікації, буде поступово рости, однак ще більш затребуваними стануть програмні технології й рішення, у яких задіяні мобільні пристрої: смартфони й планшети.

Можна створювати комплексні засоби керування доступом для систем, що поєднують традиційні додатки й приватні хмари, що працюють по моделі IaaS, PaaS і SaaS (в останньому випадку потрібно API хмарного додатка).

Рішення для приватних хмар і гібридних систем уже повністю пророблено, а його комерційне просування почнеться, як тільки на українському ринку з'явиться стійкий попит на подібні продукти.

У модуль SSO включений функціонал підтримуючі популярні мобільні платформи Android і iOS. Цей модуль надає безпечний доступ через браузер з мобільних пристроїв до внутрикорпоративних порталів і інтранет-додаткам (портал, корпоративна пошта Microsoft Outlook Web App і ін.). Версія для OS Android містить убудовану повнофункціональну систему SSO, що підтримує VoIP-телефонію, відео- і відео-конференц-зв'язок (Skype, SIP), а також будь-які Android-додатка (наприклад, клієнти корпоративних систем: бухгалтерських, CRM, ERP, HR і ін.) і хмарні Web-сервіси.

Завдяки цьому користувачам не потрібно запам'ятовувати безліч ідентифікаційних пар (логін-пароль), а організація може застосовувати правила безпеки, що вимагають застосування стійких, часто мінливих паролів, які розрізняються у всіх додатках.

Триваюче посилення державного регулювання сфери інформаційної безпеки в Україні сприяє росту вітчизняного ринку засобів ІБ. Так, по даним IDC, в 2015 році сукупний обсяг продажів програмних рішень безпеки склав більше 412 млн. доларів, перевищивши аналогічні показники попереднього періоду більш ніж на 9%. А зараз, після очікуваного спаду, прогнози оптимістичні: у наступні три роки середньорічні темпи росту можуть перевищувати 6%. Найбільші обсяги продажів доводяться на програмні рішення для захисту кінцевих пристроїв (більше 50% ринку засобів ІБ), моніторингу уразливостей і контролю безпеки в корпоративних мережах, а також на засоби ідентифікації й контролю доступу.

Вітчизняні криптоалгоритми не уступають західним стандартам і навіть, на думку багатьох, їх перевершують. Що стосується інтеграції українських розробок в області автентифікації й ЕЦП із закордонними продуктами, то було б корисно створити бібліотеки з відкритим кодом і криптопровайдери для різних

рішень із контролем їхньої якості з боку СБУ України й сертифікацією окремих стабільних версій. У такому випадку виробники могли б вбудовувати їх у пропоновані продукти, які, у свою чергу, проходили б сертифікацію як СКЗІ з урахуванням коректності використання вже перевіреної бібліотеки. Варто відзначити, що деяких успіхів уже вдалося досягти: гарним прикладом є розробка латок для підтримки ДСТ в openssl. Однак варто задуматися про організацію цього процесу на державному рівні.

Українська галузь IT іде по шляху вбудовування вітчизняних сертифікованих компонентів у закордонні інформаційні системи. Цей шлях на даний момент оптимальний, тому що розробка повністю українських інформаційних і операційних систем з нуля буде невиправдано складною й дорогою. Поряд з організаційними й фінансовими проблемами, а також недостатньо проробленими законами, широкому поширенню засобів автентифікації й цифрового підпису заважає низький рівень освіченості населення в сфері ІБ і відсутність державної інформаційної підтримки вітчизняних постачальників рішень безпеки. Драйверами ринку, безумовно, є торговельні площадки, системи податкової звітності, корпоративні й державні системи документообігу. За останні п'ять років прогрес у розвитку галузі величезний.

3.3 Розробка функціональної схеми

Функціональна схема розробленої системи зображена на рисунку 3.2. З рисунку видно, що розроблена система складається з наступних частин, які реалізують типи прав доступу до USB-ключа:

1. Адміністраторський – надає наступні можливості:

– право міняти PIN-код користувача, не знаючи його;

– право зміни пароля адміністратора;

– право налаштовувати параметри кешування змісту закритої області пам'яті й додаткового захисту закритих ключів паролем, а також можливість робити ці настроювання доступними в користувальницькому режимі.

2. Користувальницький – надає наступні можливості:

– право переглядати, змінювати й видаляти об'єкти в закритих, відкритих і вільній областях пам'яті;

– можливість одержання загальної інформації відносно USB-ключа;

– право міняти PIN-код і перейменовувати USB-ключ;

– право налаштовувати параметри кешування змісту закритої області пам'яті й додаткового захисту закритих ключів паролем (при відсутності пароля адміністратора або з дозволу адміністратора)

– право перегляду й видалення сертифікатів у сховищі USB-ключа і ключових контейнерах RSA.

3. Гостьовий – надає наступні можливості:

– можливість переглядати об'єкти у відкритій області пам'яті;

– можливість одержання із системної області пам'яті загальної інформації відносно USB-ключа, що включає ім'я USB-ключа, ідентифікатори й деякі інші параметри.

При гостьовому доступі знання PIN-коду не обов'язково.

4. Ініціалізаційний – право форматувати USB-ключ.

Для одержання доступу до даних, що зберігається в пам'яті USB-ключа, необхідно ввести PIN-код (Personal Identification Number). В PIN-коді не рекомендується використовувати пробіли й кирилицю. При цьому PIN-код повинен задовольняти критеріям якості, заданим у файлі %systemroot%\system32\etcpass.ini.

Редагування цього файлу, що містить критерії якості PIN-коду, здійснюється за допомогою утиліти USB-ключа.

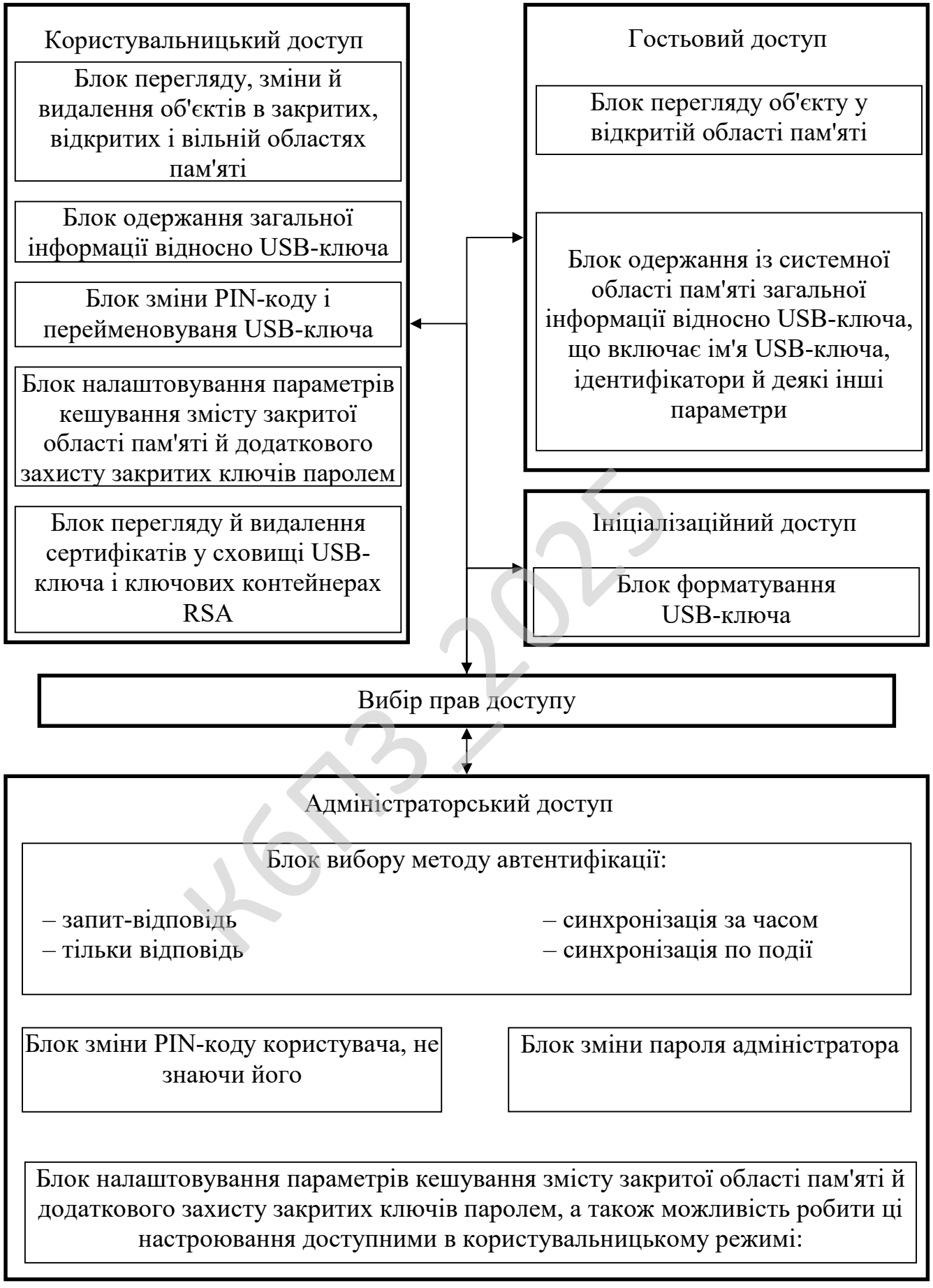

Рисунок 3.2 – Функціональна схема системи

Адміністраторський доступ до USB-ключа може бути зроблений тільки після правильного введення пароля адміністратора. Якщо ж у процесі форматування пароль адміністратора не заданий, то звернутися із правами адміністратора не можна.

За допомогою розробленого програмного забезпечення можна:

– налаштовувати параметри USB-ключа і його драйверів;

– переглядати загальну інформацію відносно USB-ключа;

– імпортувати, переглядати й видаляти сертифікати (за винятком сертифікатів зі сховища USB-ключів) і ключові контейнери RSA;

– форматувати USB-ключ;

– налаштовувати критерії якості PIN-кодів.

Для установки даного програмного забезпечення необхідні права локального адміністратора. Варто пам'ятати, що до установки розробленого програмного забезпечення не можна підключати USB-ключ.

Якщо на комп'ютері встановлене програмне забезпечення, підключите USB-ключ до порту USB або до подовжувального кабелю. Після цього почнеться процес обробки нового обладнання, що може зайняти якийсь час. По завершенні процесу обробки нового обладнання на USB-ключ засвітиться світловий індикатор.

Утиліта "Властивості USB-ключа" дозволяє виконувати основні операції по керуванню токенами, такі як зміна паролів, перегляд інформації й сертифікатів, розташованих у пам'яті USB-ключа. Крім того, за допомогою утиліти "Властивості USB-ключа" можна швидко й легко переносити сертифікати між комп'ютером і USB-ключем, а також імпортувати ключі до пам’яті USB-ключа.

Кнопка "Розблокувати" необхідна, якщо користувач забув свій PIN-код, і не може прийти до адміністратора USB-ключа (наприклад, користувач перебуває у відрядженні). Звернувшись до адміністратора по e-mail, користувач зможе одержати для цього USB-ключа від адміністратора шестнадцятирічний запит,

сформований на підставі даних, що зберігаються в базі TMS, занесши який у поле "відповідь" користувач одержить доступ на зміну PIN-коду.

При зміні PIN-коду необхідно щоб новий PIN-код відповідав вимогам якості введеного пароля. Якість пароля перевіряється відповідно до введених критеріїв. Для того щоб перевірити відповідність пароля обраним критеріям, введіть пароль у рядок. Під цим рядком виводиться інформація про причини невідповідності введеного пароля обраним критеріям у відсотках, а також графічно й у відсотках умовно відображається якість введеного пароля відповідно до обраних критеріїв.

Для того щоб задати список неприпустимих або небажаних паролів, створіть текстовий файл. Або можливо скористатися так званими частотними словниками, які використовуються для підбора паролів. Файли таких словників можна взяти на сайті www.passwords.com.

Приклад такого словника:

– anna

– annette

– bill

– password

– william

Призначте критерію "Dictionary" шлях до створеного файлу. При цьому шлях до файлу словника на кожному комп'ютері повинен збігатися зі значенням критерію "Dictionary".

Вхід у систему за допомогою USB-ключа

При автентифікації в Windows використовуються ім'я користувача й пароль, що зберігаються в пам'яті USB-ключ. Це дає можливість застосовувати строгу автентифікацію на основі токенів.

Разом з тим хотілося б додати, що у великих компаніях, що використовують доменну структуру, необхідно подумати про впровадження PKI і централізованому застосуванні SmartCardLogon.

При використанні USB-ключа можуть застосовуватися нікому не відомі випадкові складні паролі. Крім того, передбачена можливість використання сертифікатів, що зберігаються в пам'яті USB-ключа, для реєстрації на основі смарт-карт, що підвищує безпека входу в Windows.

Це стало можливим завдяки тому, що система Windows XP/7/8/10 дозволяє використовувати різні механізми доступу, що заміняють метод автентифікації за замовчуванням. Механізми ідентифікації й автентифікації служби входу в Windows (winlogon), що забезпечує інтерактивну реєстрацію в системі, убудовані в замінну бібліотеку, що приєднується динамічно (DLL), іменовану GINA (Graphical Identification and Authentication, робочий стіл автентифікації). Коли система має потребу в іншому методі автентифікації, який би замінив механізм "ім'я користувача/пароль" (використовуваний за замовчуванням) стандартну msgina.dll заміняють новою бібліотекою.

При установці USB-ключа заміняється бібліотека робочого стола автентифікації й створюються нові параметри реєстру. GINA відповідає за політику інтерактивного підключення й здійснює ідентифікацію й діалог з користувачем. Заміна бібліотеки робочого стола автентифікації робить USB-ключ основним механізмом перевірки дійсності, що розширює можливості стандартної автентифікації Windows XP/7/8/10, заснованої на застосуванні ім'я користувача й пароля.

Користувачі можуть самостійно записувати до пам’яті USB-ключ інформацію, необхідну для входу в Windows (профілі), якщо це дозволено політикою безпеки підприємства. Профілі можна створювати за допомогою майстра створення профілів USB-ключа Windows Logon.

USB-ключ SecurLogon автентифікує користувача Windows XP/7/8/10 с допомогою USB-ключ, використовуючи або сертифікат користувача зі смарт-картою, або ім'я користувача й пароль, які зберігаються в пам'яті USB-ключа. USB-ключ містить у собі всі необхідні файли й драйвери, що забезпечують підтримку USB-ключа в Windows Logon.

Всі нові USB-ключ мають той самий PIN-код, установлений за замовчуванням при виробництві. Цей PIN-код 1234567890. Для забезпечення строгої, двофакторної автентифікації й повної функціональності користувач обов'язково повинен замінити PIN-код за замовчуванням власним PIN-кодом відразу після одержання нового USB-ключ.

Важливо: PIN-код не слід плутати з паролем користувача Windows.

Блокування робочої станції

Можливо забезпечувати безпеку вашого комп'ютера, не виходячи із системи, шляхом блокування комп'ютера. При від'єднанні USB-ключа від порту USB або кабелю (після вдалої реєстрації) операційна система автоматично заблокує ваш комп'ютер.

Розблокування ваш комп'ютера

Коли ваш комп'ютер заблокований, з'являється вікно "Блокування комп'ютера Computer Locked". Підключите USB-ключ до порту USB або кабелю. У вікні, що з'явилося, введіть PIN-код у поле " USB-ключ Password" і натисніть кнопку "OK" – комп'ютер розблокований. У випадку натискання "CTRL+ALT+DEL" і введення пароля комп'ютер буде розблокований без використання USB-ключа.

Установка

Для того щоб установити USB-ключ Windows Logon:

– увійдіть у систему як користувач із правами адміністратора;

– двічі клацніть SecurLogon.msi;

– з'явиться вікно майстра установки USB-ключ SecurLogon;

– натисніть кнопку "Next", з'явиться ліцензійна угода USB-ключ Enterprise;

– прочитайте угоду, натисніть кнопку "I accept" (Приймаю), а потім кнопку "Next";

– наприкінці установки виробляється перезавантаження.

Використання USB-ключ SecurLogon

USB-ключ SecurLogon дозволяє користувачам реєструватися в Windows XP/7/8/10 за допомогою USB-ключа із записаним у пам'яті паролем.

Зміна пароля

Можливо перемінити пароль Windows після входу в систему за допомогою USB-ключ. Для того щоб перемінити пароль після входу в систему за допомогою USB-ключ:

– увійдіть у систему, використовуючи USB-ключ;

– натисніть "CTRL+ALT+DEL", з'явиться вікно "Безпека Windows / Windows Security";

– Клацніть кнопку "Зміна пароля / Change Password", якщо поточний пароль був створений вручну, те з'явиться вікно "Зміна пароля / Change Password", але якщо поточний пароль був створений випадковим образом, то переходимо до пункту 5;

– Уведіть новий пароль у полях "Новий пароль / New Password" і "Підтвердження / Confirm New Password" і натисніть кнопку "OK";

– Якщо поточний пароль був створений випадковим образом, то й новому паролі буде створений автоматично;

– у діалоговому вікні, що з'явилося, введіть PIN-код USB-ключ і натисніть кнопку "OK"

– з'явиться вікно з підтверджувальним повідомленням.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування). Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи. Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Діаграми потоків даних містять чотири типи елементів:

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

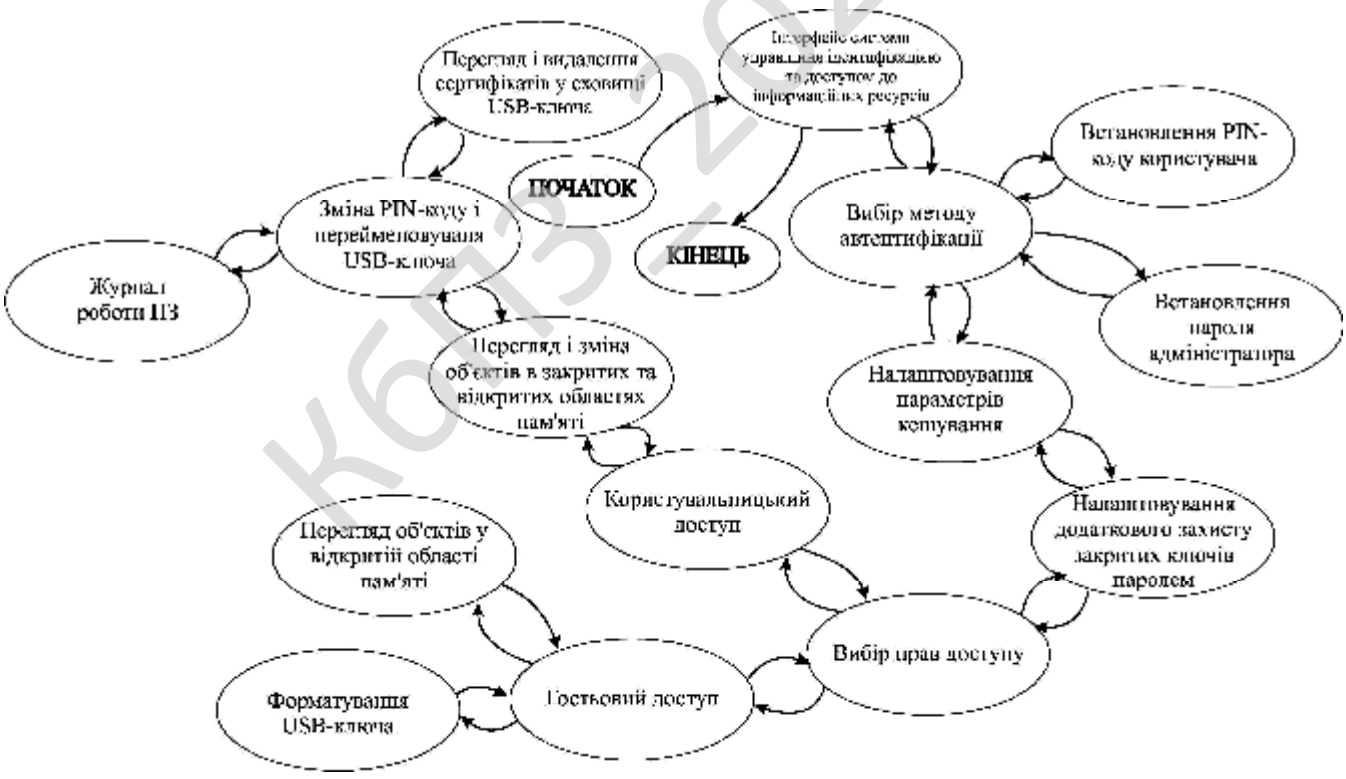

Рисунок 3.3 – Діаграма взаємодії процесів.

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

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 було враховано, 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 системи управління ідентифікацією та доступом до інформаційних ресурсів.

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.

Під час роботи над бакалаврською дипломною роботою було створено блок-схеми. Перед їх розглядом необхідно провести роз’яснення який саме тип блок-схем використовується.

Блок-схема це представлення задачі для її аналізу або розв'язування за допомогою спеціальних символів (геометричних образів), які позначають такі елементи, як операції, потік, дані тощо. Блок вхідних та вихідних даних прийнято позначати паралелограмом, блок обчислень (обробки) даних – прямокутником, блок прийняття рішень – ромбом, еліпсом – початок та кінець алгоритму.

У інформаційних технологіях функціональна схема складається з функціональних блоків, які являють собою конструктивно відособлені частини (елементи або пристрої) автоматичних систем, які виконують певні функції.

Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів.

Функціональні схеми можуть виконуватися в укрупненому і розгорненому вигляді. У першому випадку на схемі зображають найважливіші блоки системи і зв'язки між ними.

У другому варіанті схема відображається більш детально, що полегшує її читання та ілюструє принцип роботи.

Основні елементи схем алгоритму це термінатор, процес, рішення, зумовлений процес (підпрограма), дані та з'єднувач.

Термінатор це елемент відображає вхід із зовнішнього середовища або вихід з неї (найчастіше застосування – початок і кінець програми). Всередині фігури записується відповідна дія.

Процес це виконання однієї або кількох операцій, обробка даних будь-якого виду (зміна значення даних, форми подання, розташування). Всередині фігури записують безпосередньо самі операції.

Рішення це показує рішення або функцію перемикального типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов, визначених всередині цього елементу. Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елементу. Якщо виходів два чи три то зазвичай кожен вихід позначається лінією, що виходить з решти вершин (бічних і нижній). Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижній) елемента, яка потім розгалужується. Відповідні результати обчислень можуть записуватися поруч з лініями, що відображають ці шляхи.

Зумовлений процес (підпрограма) це символ відображає виконання процесу, що складається з однієї або кількох операцій, що визначені в іншому

місці програми (у підпрограмі, модулі). Всередині символу записується назва процесу і передані в нього дані.

Дані це перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення). Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи).

З'єднувач це символ відображає вихід в частину схеми і вхід з іншої частини цієї схеми. Використовується для обриву лінії та продовження її в іншому місці (приклад: поділ блок-схеми, що не поміщається на листі). Відповідні сполучні символи повинні мати одне (при тому унікальне) позначення.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

З яких 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.

Також при розробці бакалаврської дипломної роботи було використано наступні підходи UML: діаграма діяльності (діаграми поведінки типу); діаграма прецедентів (діаграми поведінки типу); Діаграма класів; Діаграма компонент.

Діаграма діяльності. Це візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій. Дія є фундаментальною одиницею 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.

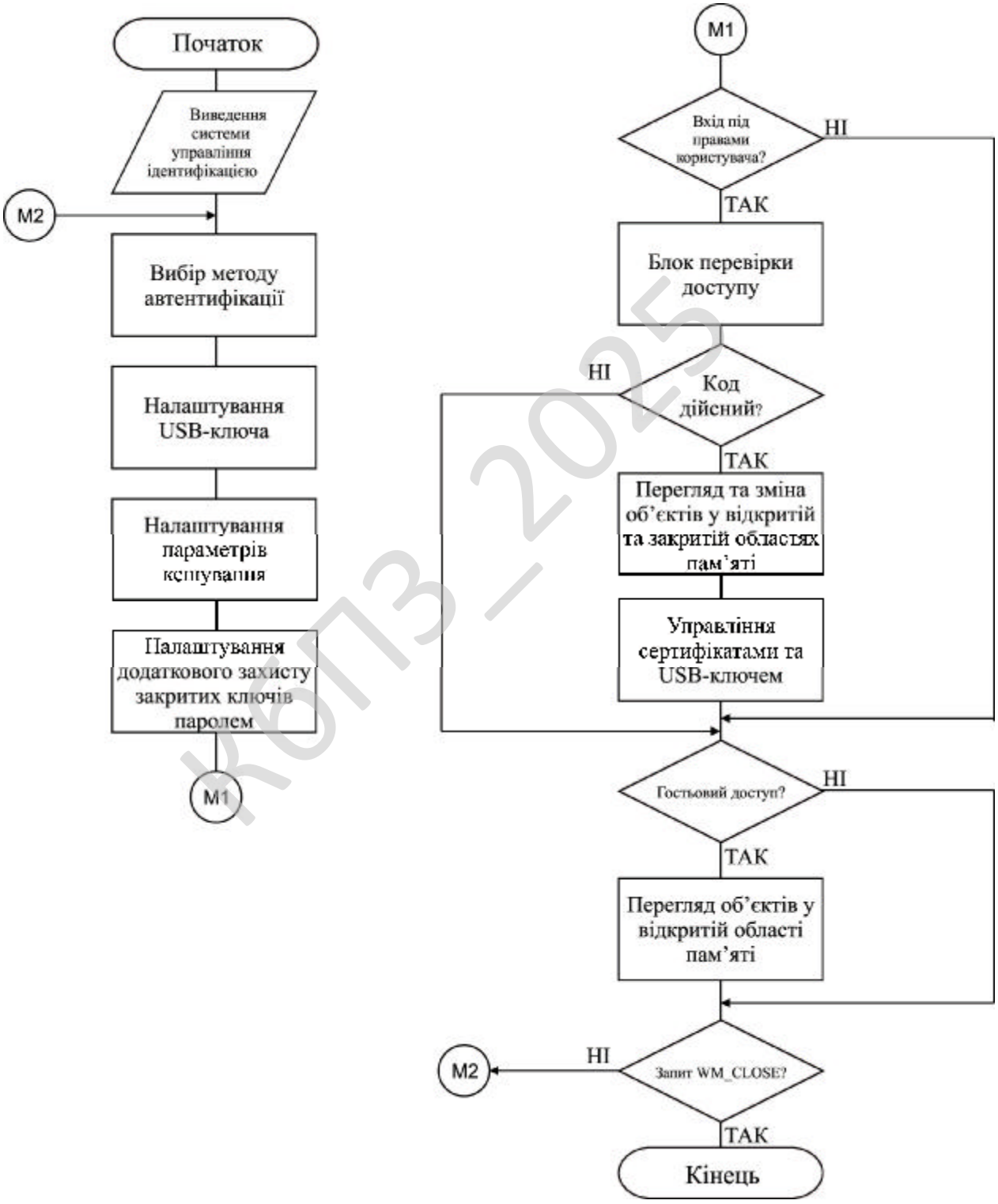

Рисунок 4.1 – Блок-схема основної програми

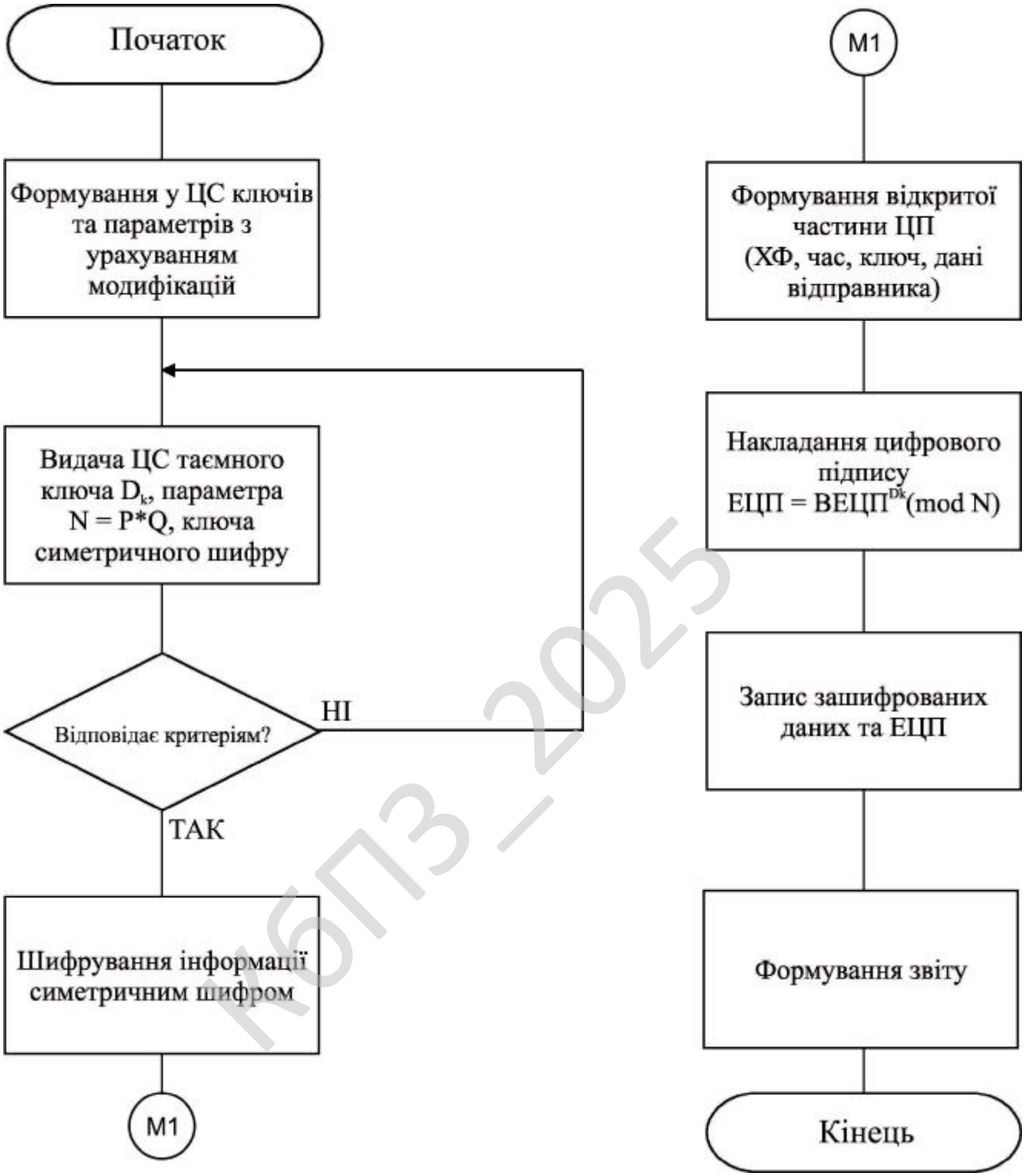

Рисунок 4.2 – Блок-схема роботи підпрограми

Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

Діаграма прецедентів це діаграма, на якій зображено відношення між акторами та прецедентами в системі. Також, перекладається як діаграма варіантів використання.

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених границею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами. Діаграми прецедентів відображають елементи моделі варіантів використання.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Варіант використання (use case) використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система при діалозі з актором.

При цьому нічого не говориться про те, яким чином буде реалізована взаємодія акторів із системою.

У мові UML є кілька стандартних видів відношень між акторами і варіантами використання:

– асоціації (association relationship);

– включення (include relationship);

– розширення (extend relationship);

– узагальнення (generalization relationship).

При цьому загальні властивості варіантів використання можуть бути представлені трьома різними способами, а саме – за допомогою відношень включення, розширення і узагальнення.

Відношення асоціації – одне з фундаментальних понять у мові UML і в тій чи іншій мірі використовується при побудові всіх графічних моделей систем у формі канонічних діаграм.

Включення (include) у мові UML – це різновид відношення залежності між базовим варіантом використання і його спеціальним випадком. При цьому відношенням залежності (dependency) є таке відношення між двома елементами моделі, при якому зміна одного елемента (незалежного) приводить до зміни іншого елемента (залежного).

Відношення розширення (extend) визначає взаємозв'язок базового варіанта використання з іншим варіантом використання, функціональна поведінка якого задіюється базовим не завжди, а тільки при виконанні додаткових умов.

Діаграма класів це статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення.

Діаграма класів, також, може містити позначення для пакетів та може містити позначення для вкладених пакетів. Також, діаграма класів може містити позначення деяких елементів поведінки, однак їх динаміка розкривається в інших типах діаграм.

Діаграма класів (class diagram) служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування. На цій діаграмі показують класи, інтерфейси, об'єкти й кооперації, а також їхні відносини.

В UML існують наступні типи зв'язків які використовуються у діаграмі класів: Асоціації; Агрегація; Композиція.

Асоціації це якщо між двома класами визначена асоціація, то можна переміщатися від об'єктів одного класу до об'єктів іншого. Цілком припустимі випадки, коли обидва кінці асоціації відносяться до одного і того ж класу. Це означає, що з об'єктом деякого класу дозволено зв'язати інші об'єкти з того ж класу. Асоціація, що зв'язує два класи, називається бінарної. Можна, хоча це рідко

буває необхідним, створювати асоціації, що зв'язують відразу кілька класів. Графічно асоціація зображується у вигляді лінії, що з'єднує клас сам з собою або з іншими класами.

Асоціації може бути присвоєно ім'я, яке описує природу відносини. Зазвичай ім'я асоціації не вказується, якщо тільки ви не хочете явно задати для неї рольові імена або у вашій моделі настільки багато асоціацій, що виникає необхідність посилатися на них і відрізняти один від одного. Ім'я буде особливо корисним, якщо між одними і тими ж класами існує кілька різних асоціацій.

Клас, що бере участь в асоціації, грає в ній деяку роль. По суті, це "обличчя", яким клас, що знаходиться на одній стороні асоціації, звернений до класу з іншого її боку. Можна явно позначити роль, яку клас грає в асоціації.

Часто при моделюванні буває важливо вказати, скільки об'єктів може бути пов'язано допомогою одного примірника асоціації. Це число називається кратністю (Multiplicity) ролі асоціації та записується або як вираз, значенням якого є діапазон значень, або в явному вигляді.

Вказуючи кратність на одному кінці асоціації, ви тим самим говорите, що на цьому кінці саме стільки об'єктів повинно відповідати кожному об'єкту на протилежному кінці. Кратність можна задати рівною одиниці (1), можна вказати діапазон: "нуль або одиниця" (0..1), "багато" (0 .. *), "одиниця або більше" (1 .. *). Дозволяється також вказувати певне число (наприклад, 3). За допомогою списку можна задати і більш складні кратності, наприклад 0. . 1, 3..4, 6 .. *, що означає "будь-яке число об'єктів, крім 2 і 5".

Агрегація це проста асоціація між двома класами відображає структурний відношення між рівноправними сутностями, коли обидва класу знаходяться на одному концептуальному рівні і ні один не є більш важливим, ніж інший. Але іноді доводиться моделювати відношення типу «частина/ціле», в якому один з класів має більш високий ранг (ціле) і складається з декількох менших за рангом (частин).

Ставлення такого типу називають агрегацією; воно зараховане до відносин типу «має» (з урахуванням того, що об'єкт-ціле має кілька об'єктів-частин). Агрегація є окремим випадком асоціації і зображується у вигляді простої асоціації з незафарбованим ромбом з боку «цілого». Графічно агрегація представляється порожнім ромбом на блоці класу, і лінією, яка від цього ромба до міститься класу.

Композиція це більш суворий варіант агрегації. Відома також як агрегація за значенням.

Композиція має жорстку залежність часу існування екземплярів класу контейнера та примірників містяться класів. Якщо контейнер буде знищений, то весь його вміст буде також знищено. Графічно представляється як і агрегація, але з зафарбовані ромбиком.

Діаграма компонент в UML це діаграма, на якій відображаються компоненти, залежності та зв'язки між ними.

Діаграма компонент відображає залежності між компонентами програмного забезпечення, включаючи компоненти вихідних кодів, бінарні компоненти, та компоненти, що можуть виконуватись.

Модуль програмного забезпечення може бути представлено в якості компоненти. Деякі компоненти існують під час компіляції, деякі – під час компонування, а деякі під час роботи програми.

Діаграма компонент відображає лише структурні характеристики, для відображення окремих екземплярів компонент слід використовувати діаграму розгортування.

Компоненти об'єднуються разом використовуючи структурні зв'язки (assembly connector) щоб об'єднати інтерфейси двох компонент. Це ілюструє зв'язок типу «клієнт-сервер». Структурна взаємодія – «зв'язок двох компонент, який передбачає, що один з них надає послуги, потрібні іншому компоненту».

При використанні діаграми компонент щоб показати внутрішню структуру компонента, клієнтські та серверні інтерфейси можуть утворювати пряме з'єднання з внутрішніми. Таке з'єднання називається з'єднанням делегації.

Розгортання служб сертифікації

Для впровадження системи сертифікації й керування відкритими ключами в системі автентифікації користувачів пропонується використовувати дворівневу ієрархію центрів сертифікації: ізольований ЦС (Stand-alone Root CA) як центр сертифікації й ізольований підлеглий ЦС (Stand-alone subordinate CA) як центр реєстрації.

Розгортання системи сертифікації й керування відкритими ключами виконується в наступній послідовності:

– установка центра сертифікації;

– конфігурування центра сертифікації;

– установка й настроювання центра реєстрації;

– установка й настроювання сховища сертифікатів;

– установка й настроювання допоміжних систем і додатків.

Центр сертифікації поєднує людей, процеси, програмні й апаратні засоби, залучені в безпечне зв'язування імен користувачів і їхніх відкритих ключів. Центр сертифікації відомий суб'єктам інфраструктура відкритих ключів алгоритму RSA по двох атрибутах: назві й відкритому ключу. Центр сертифікації включає своє ім'я в кожний випущений їм сертифікат і в **список анульованих сертифікатів** (CAC) і підписує їх за допомогою власного секретного ключа. Користувачі можуть легко ідентифікувати сертифікати по ім'ю центр сертифікації і переконатися в їхній дійсності, використовуючи його відкритий ключ. Центр сертифікації – головний керуючий компонент інфраструктури відкритих ключів алгоритму RSA – виконує наступні основні функції:

– формує власний секретний ключ; якщо є головним центр сертифікації, то видає й підписує свій сертифікат, який називається **самовиданим** або **самопідписаним**;

– випускає (тобто створює й підписує) сертифікати відкритих ключів підлеглих центрів, що засвідчують, і кінцевих суб'єктів інфраструктури відкритих ключів алгоритму RSA; може випускати крос-сертифікати, якщо зв'язано

відносинами довіри з іншими інфраструктурами відкритих ключів алгоритму RSA;

– підтримує реєстр сертифікатів (базу всіх виданих сертифікатів) і формує списки CAC із регулярністю, певної регламентом сертифікаційного центра;

– публікує інформацію про статус сертифікатів і списків CAC.

Реєстраційний центр (РЦ) є необов'язковим компонентом інфраструктури відкритих ключів алгоритму RSA. Звичайно РЦ одержує від центра, що засвідчує, повноваження реєструвати користувачів, забезпечувати їхня взаємодія із центр сертифікації і перевіряти інформацію, що заноситься в сертифікат. Сертифікат може містити інформацію, що надана суб'єктом, що подає заявку на сертифікат і пред'являє документ (паспорт, права водія, чекову книжку й т.п.) або третьою стороною (наприклад, кредитним агентством – про кредитний ліміт пластикової карти). Іноді в сертифікат включається інформація з відділу кадрів або дані, що характеризують повноваження суб'єкта в компанії (наприклад, право підпису документів певної категорії). РЦ агрегує цю інформацію й надає її центр сертифікації.

Репозиторій – спеціальний об'єкт інфраструктури відкритих ключів, база даних, у якій зберігається реєстр сертифікатів. Репозиторій значно спрощує керування системою й доступ до ресурсів. Він надає інформацію про статус сертифікатів, забезпечує зберігання й поширення сертифікатів і CAC, управляє внесеннями змін у сертифікати. До репозиторію пред'являються наступні вимоги:

– простота й стандартність доступу;

– регулярність відновлення інформації;

– вбудована захищеність;

– простота керування;

– сумісність із іншими сховищами (необов'язкова вимога).

Архів сертифікатів виконує функцію довгострокового зберігання (від імені сертифікаційного центра) і захисту інформації про всі видані сертифікати. Архів підтримує базу даних, використовувану при виникненні споровши із приводу

надійності електронних цифрових підписів, якими в минулому засвідчувалися документи. Архів підтверджує якість інформації в момент її одержання й забезпечує цілісність даних під час зберігання. Інформація, надавана центр сертифікації архіву, повинна бути достатньої для визначення статусу сертифікатів і їхнього видавця. Архів повинен бути захищений відповідними технічними засобами й процедурами.

Кінцеві суб'єкти, або користувачі, інфраструктури відкритих ключів алгоритму RSA діляться на дві категорії: власники сертифікатів і сторони, що довіряють. Вони використовують деякі сервіси й функції, щоб одержати сертифікати або перевірити сертифікати інших суб'єктів. Власником сертифіката може бути фізична або юридична особа, додаток, сервер і т.д. сторони, Що Довіряють, запитують і покладаються на інформацію про статус сертифікатів і відкритих ключів підпису своїх партнерів по діловому спілкуванню.

Операційні протоколи – це протоколи для доставки сертифікатів (або інформації про їхній статус) і списків анульованих сертифікатів до клієнтських систем, що використовують сертифікати.

Протоколи керування необхідні для підтримки взаємодій між користувачем інфраструктури відкритих ключів і суб'єктами керування. Протоколи керування підтримують:

– реєстрацію суб'єкта для одержання сертифіката;

– ініціалізацію (наприклад, генерації пари ключів);

– випуск сертифіката;

– відновлення пари ключів;

– відновлення пари ключів після закінчення терміну дії сертифіката;

– обіг із запитом про анулювання сертифіката;

– крос-сертифікацію, коли два сертифікаційних центри обмінюються інформацією для генерації крос-сертифіката.

Політика застосування сертифікатів і регламент центра сертифікації є в документах, де приведенні зобов'язання сторін і правила використання сертифікатів.

Загальна схема функціонування інфраструктури відкритих ключів алгоритму RSA працює наступним чином. Користувач відправляє запит на сертифікат у РЦ (транзакція керування). Якщо запит фактично схвалений, то направляється безпосередньо в центр сертифікації для завірення цифровим підписом. Центр сертифікації перевіряє запит на сертифікат, і якщо той проходить верифікацію, то підписується й випускається сертифікат. Сертифікат публікується в репозиторії; залежно від конкретної конфігурації інфраструктури відкритих ключів алгоритму RSA, ця функція може бути покладена на реєстраційний або центр, що засвідчує. Процес анулювання сертифіката аналогічний процесу його генерації. Кінцевий суб'єкт запитує центр сертифікації про анулювання свого сертифіката, РЦ приймає рішення й направляє запит про анулювання в центр сертифікації. Центр сертифікації вносить зміни в список анульованих сертифікатів і публікує його в репозиторії. Кінцеві суб'єкти можуть перевірити статус конкретного сертифіката через операційний протокол

Поточні завдання системи сертифікації й керування відкритими ключами

До завдань, постійно виконуваних системою сертифікації й керування відкритими ключами, відносяться:

- архівування й відновлення центрів сертифікації;
- відмова або схвалення запитів на сертифікати;
- відкликання сертифікатів;
- публікація списків відкликаних сертифікатів (CRL);
- відновлення сертифікатів центрів сертифікації;
- відновлення сертифікатів користувачів.

Відновлення після збою

Серйозні збої, такі, як відмова жорсткого диска або компрометація

сертифіката ЦС, здатні повністю порушити роботу служб сертифікації. Існує кілька шляхів мінімізації негативних наслідків таких збоїв і забезпечення своєчасного відновлення. До операцій по зниженню ризику відмови або компрометації ЦС відносяться:

– захист закритих ключів центрів сертифікації;

– розробка планів відновлення.

Наступні операції дозволяють попередити ризик відмови ЦС і звести до мінімуму час простою служб ЦС:

– створення дублюючих ЦС;

– часте архівування ЦС, що дозволяє швидко й з мінімальними втратами даних відновити центр сертифікації;

– установка служб сертифікації на дискових масивах і масивах RAID-5;

– розробка планів відновлення й навчання адміністраторів виконанню цих планів;

– збереження дані конфігурації ЦС для безпроблемного й точного відновлення конфігурації у випадку збою.

Забезпечення безпеки центрів сертифікації

Комп'ютери, на яких працюють ЦС, – це найбільш імовірні мети для атаки зловмисників, що намагаються порушити роботу служб або скомпрометувати захист і інформаційних систем. Одержавши доступ до ЦС або скориставшись недоліками захисту, ті що атакують можуть одержати доступ до ресурсів і скомпрометувати безпеку цілого ланцюжка довіри. Тому ЦС мають потребу в більше надійному захисті, чим звичайні.

Ризик атаки на ЦС залежить від багатьох факторів, у тому числі від захищеності , цілей атаки й витрат на неї. До можливих негативних наслідків компрометації ЦС відносяться:

– втрата інформації, що становить інтелектуальну власність організації;

– відмова або простій служб;

– ушкодження або руйнування ресурсів;

– витрати на усунення наслідків компрометації ЦС і повторне розгортання центрів сертифікації й сертифікатів.

Компрометація кореневого ЦС обійдеться набагато дорожче, ніж проміжного або випускаючого ЦС. Щоб знизити потенційні збитки, варто створити в організації ієрархію з декількох ЦС.

Визначаючи адекватність мер безпеки, необхідно зважити й оцінити витрати на ці заходи щодо забезпечення безпеки й можливі збитки у випадку компрометації ЦС. До заходів щодо забезпечення безпеки ЦС відносяться:

– розміщення у захищених центрах даних і надання фізичного доступу до них тільки довіреним адміністраторам;

– використання апаратних центрів сертифікації або апаратних постачальників CSP для забезпечення максимального захисту закритих ключів ЦС;

– конфігурування параметрів безпеки для забезпечення високого рівня захисту, такого, як рівень High Security (Високий) захисту шаблонів;

– використання службової програми Windows 2000 System Key (SysKey) для забезпечення шифрування захищених сховищ ЦС;

– аудит безпеки для контролю й спостереження за можливими атаками на ЦС;

– обмеження надання прав користувачам шляхом надання прав тільки відповідній групі адміністраторів (іншим користувачам або групам треба заборонити перегляд або виконання будь-яких завдань на локальному комп'ютері зі ЦС);

– відключення непотрібних служб на х зі ЦС; працюючі служби являють собою додаткові «лазівки» для зловмисників;

– розгортання політики й виконання процедур безпеки при розгортанні ЦС на підприємстві.

При виборі мір безпеки ЦС варто зважити витрати на їхню реалізацію й підтримку, з одного боку, і ризик нападу на ЦС і можливі збитки від

компрометації ЦС, з іншої сторони. У загальному випадку чим вище ризик нападу й збитки від компрометації, тим більше витрати на заходи щодо забезпечення безпеки ЦС. Максимальний захист варто забезпечити кореневим ЦС, а проміжні ЦС треба убезпечити надійніше, ніж випускаючі ЦС.

Допустимо, що в організації ухвалено рішення захистити великий обсяг надзвичайно коштовної й конфіденційної інформації, використавши рішення на основі відкритих ключів. Також вирішено придбати дорогий апаратний ЦС для виконання функцій кореневого ЦС і розмістити його для безпеки в захищеному сховищі, розташованому в головному офісі організації. Доступ до кореневого ЦС, що сертифікує всі проміжні ЦС у підрозділах, надається тільки довіреним адміністраторам. Проміжні ЦС – це ізольовані ЦС на комп'ютерах під керуванням Windows XP/8/10, від'єднаних від й розміщених у захищених центрах даних під наглядом адміністратора підрозділу. Проміжний ЦС використовується в міру необхідності для сертифікації випускаючих ЦС під керуванням Windows XP/8/10 відповідно до потреб кожного підрозділу. Випускаючі ЦС – це ЦС Windows XP/8/10 підприємства або ізольовані ЦС, розміщені в захищених центрах даних кожного підрозділу. Політика безпеки організації повинна мати на увазі самі строгі міри безпеки виконання запиту, авторизації й впровадження кореневого, проміжних і випускаючих ЦС на підприємстві й контроль за ними.

З іншого боку, якщо в організації рішення безпеки на базі відкритого ключа застосовуються для захисту не дуже коштовної інформації, цілком достатньо ізольованого кореневого ЦС Windows XP/8/10, розміщеного в центрі даних, а не згаданого в попередньому прикладі дорогого апаратного ЦС, розміщеного в захищеному сховищі. При цьому припустимо розмістити проміжні й випускаючі ЦС у підрозділах за межами центра даних. Також не потрібно таких строгих обмежень на впровадження, запити й авторизацію ЦС.

Служби сертифікації Windows XP/8/10/11 на базі постачальника Microsoft Base CSP зможуть задовольнити більшість потреб у захисті ЦС. Для забезпечення найвищої безпеки ЦС варто використовувати апаратні ЦС.

Захист закритих ключів алгоритму RSA центрів сертифікації

Якщо в зловмисника є доступ до комп'ютера ЦС – безпосередньо або по , він у стані розшифрувати особистий ключ і потім виступати від імені ЦС і одержувати доступ до коштовних ресурсів. Він зможе красти інформацію, порушити роботу служб і знищити ресурси. Скомпрометований ключ алгоритму RSA ЦС підриває й зводить «на ні» всю систему безпеки, забезпечену цим ЦС, і всю його ієрархію ЦС. Необхідно регулярно проводити заходу щодо зниження атаки на ключі ЦС.

Необхідно забезпечити захист із центрами сертифікації, як описано раніше в цьому розділі. Забезпечення фізичної безпеки мінімізує ризик одержання атакуючого доступу до ЦС або захищеному сховищу (будь воно апаратним або програмним), де зберігається ключ алгоритму RSA ЦС. Забезпечення безпеки й (програмного забезпечення) знижує ризик того, що порушники одержать доступ до ЦС або скористаються недоліками додатків або служб цього для компрометації ключа ЦС. Необхідно забезпечити посилений захист ключів центра сертифікації. Якщо потрібна максимальна безпека закритих ключів, необхідно скористатися апаратними постачальниками CSP, тому що в цьому випадку ключі зберігаються на стійкі до злому апаратних пристроях і ніколи не надаються операційній системі. Необхідно використовувати службову програму SysKey для забезпечення додаткового захисту закритих ключів ЦС, збережених постачальниками Microsoft CSP.

Використання в центрах сертифікації ключів великої довжини знижує ризик атаки на ключ алгоритму RSA, однак довгі ключі вимагають більше дискового простору й обчислювальних потужностей для підписання сертифікатів. Варто вибрати максимальну можливу довжину ключа й урахувати обмеження на пам'ять і продуктивність ЦС.

Наприклад, 4096-бітний ключ алгоритму RSA ЦС забезпечує чудову безпеку, але підписання сертифікатів таким довгим ключем займає занадто багато часу навіть при наявності плати криптоакселератора. Такий ключ алгоритму RSA

цілком придатний для коренев або проміжного ЦС, які використовуються нечасто й тільки для сертифікації підлеглого ЦС. Для більшості випускаючих ЦС 4096-бітних ключів неприйнятний через неприпустиме зниження продуктивності роботи ЦС. На випускаючих ЦС варто використовувати ключі, які забезпечують задовільну безпеку, не сповільнюють роботу ЦС і відповідають довгостроковим цілям конкретної служби сертифікації. Для підвищення продуктивності випускаючого ЦС і використання більше довгих ключів рекомендується встановити плату криптоакселератора. Перш ніж розгортати ЦС на підприємстві, необхідно протестувати його продуктивність для різних довжин ключів у лабораторних умовах і на пілотних системах.

Необхідно визначити адекватні терміни дії ключів ЦС. Чим більше термін дії ключа, тим вище ризик його компрометації, тому що в атакуючих більше часу на злом. Не існує простого правила визначення максимальних термінів дії ключів. Однак у загальному випадку термін дії ключів у значній мірі залежать від якості їхнього захисту й довжини. У загальному випадку, терміни дії більше довгих ключів більше. Аналогічно більше захищені ключі служать довше. Наприклад, ключі, що зберігаються в стійкі до злому апаратних криптоустройствах надійніше, ніж ключі, розміщені на жорсткому диску локального комп'ютера. Тому для двох ключів однієї довжини термін дії ключа, збереженого на апаратному криптографічному пристрої, звичайно більше, ніж ключа, розміщеного в програмному постачальнику CSP на жорсткому диску.

Розробка планів відновлення

Розробка детального плану відновлення дозволяє швидко повернути ЦС у робочий стан у випадку збоїв служб сертифікації або компрометації. Рекомендується протестувати складений план, щоб переконатися в його коректності. Крім того варто провести навчання співробітників, щоб бути впевненим у тім, що вони знають, як діяти відповідно до цього плану.

План відновлення повинен передбачати:

– процедури відновлення й контрольні списки для адміністраторів;

– набори засобів і службових програм відновлення;

– план дій у непередбачених обставинах.

Існує безліч причин, по яких ЦС може потерпіти збій, у тому числі поломка жорсткого диска , відмова мережної карти або вихід з ладу системної (материнської) плати . Деякі збої усуваються швидко – шляхом локалізації й виправлення джерела неполадки ЦС. Наприклад, після заміни несправної мережної карти або системної плати для відновлення служби сертифікації досить запустити знову комп'ютер.

Жорсткий диск, що відмовив, заміняється, і ЦС відновлюється з останнього архіву. При ушкодженні або руйнуванні ЦС його відновлюють також з останнього архіву. Після цього ЦС установлюють у вихідній конфігурації й з вихідним особистим ключем і сертифікатом ЦС.

При виявленні факту компрометації ЦС треба негайно:

– відкликати сертифікат скомпрометованого ЦС;

– опублікувати новий список CRL з відкликаним сертифікатом ЦС;

– видалити скомпрометовані сертифікати ЦС зі сховища TRCA (Довірені кореневі центри сертифікації) і із всіх списків довіри (CTL);

– сповістити всіх зацікавлених користувачів і адміністраторів про компрометацію й відізвати сертифікати, які випущені скомпрометованим ЦС;

– усунути всі «лазівки», що стали причиною компрометації.

Для відновлення ієрархії центрів сертифікації варто повторно виконати розгортання нового ЦС, далі – повторно випустити всі сертифікати для користувачів, комп'ютерів і служб.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм ДСТУ 8845:2019 – алгоритм симетричного потокового перетворення. В основі ДСТУ 8845:2019 лежить класична схема підсумовуючого

генератора подібна генератору SNOW 2.0, SNOW 3.0 та SNOW V. В основі ДСТУ 8845:2019 збережені всі базові операції шифрів сімейства SNOW, а раунд шифрування AES замінений на функцію нелінійної підстановки T, що реалізовує перестановку елементів скінченного поля $GF(2^{64})$ за допомогою компонентів національного стандарту симетричного криптоперетворення ДСТУ 7624:2014.

ДСТУ 8845:2019 використовує 256-бітний вектор ініціалізації IV та 256-бітний або 512-бітний секретний ключ K і забезпечує високий та надвисокий рівень стійкості із врахуванням можливого застосування квантового криптографічного аналізу. При розробці алгоритму ДСТУ 8845:2019 орієнтувалися на сучасні 64-бітні обчислювальні системи, тому розмір слова обрано рівним 8 байт. запису байтів застосовують подання від старшого до молодшого. Генератор ключових потоків ДСТУ 8845:2019 у режимі генерації гами шифру схематично приведено у стандарті.

Як випливає з генератора ключових потоків ДСТУ 8845:2019 у режимі генерації гами шифру основними компонентами генератору є регістр зсуву з лінійним зворотнім зв'язком та скінчений автомат на базі якого виконується нелінійне перетворення T. Вхідні дані (ключ шифрування K та вектор ініціалізації IV) використовуються для ініціалізації змінної стану $S_i (i \geq 0)$, яка складається із двох компонент до складу яких входять[12]:

– 16 змінних $s^{(i)}$ – комірок регістра зсуву з лінійним зворотнім зв'язком: $s^{(i)} = (s_{15}^{(i)}, s_{14}^{(i)}, s_{13}^{(i)}, s_{12}^{(i)}, s_{11}^{(i)}, s_{10}^{(i)}, s_9^{(i)}, s_8^{(i)}, s_7^{(i)}, s_6^{(i)}, s_5^{(i)}, s_4^{(i)}, s_3^{(i)}, s_2^{(i)}, s_1^{(i)}, s_0^{(i)})$;

– Два регістри скінченного автомату $r^{(i)}$: $r^{(i)} = (r_2^{(i)}, r_1^{(i)})$. На виході отримуємо ключовий потік (гамма), який формується з 8-байтних слів Z_i.

З рисунку слідує, що відводи регістра зсуву з лінійним оберненим зв'язком побудовані за примітивним над полем $GF(2^{64})$ поліномом: $f(x) = x^{16} + x^{13} + \alpha^{-1} x^{11} + \alpha$, де α є коренем примітивного над полем $GF(2^8)$ поліному $gz(z) = z^8 + \beta^{170} z^7 + \beta^{166} z^6 + \beta^2 z^5 + \beta^{224} z^4 + \beta^{70} z^3 + \beta^2$. Поле $GF(2^8)$ як і в ДСТУ 7624:2014 побудовано за примітивним на полем GF(2) поліномом $p(y) = x^8 + x^4 + x^3 + x^2 + 1$, а коефіцієнти $g(z)$ подаються через ступінь

примітивного елементу β поля GF(2^8), тобто β корінь поліному $p(y)$. Тобто, у нас є вежа полів: $GF(2) \subset GF(2^8) \subset GF(2^{64}) \subset GF(2^{1024})$, де:

– поле $GF(2^{1024})$ задається відводами зворотного зв'язку як фактор кільце $GF(2^{64})[x]/(f(x))$;

– поле $GF(2^{16424})$ задається як фактор кільце $GF(2^8)[z]/(g(z))$;

– поле $GF(2^{1024})$ задається як фактор кільце $GF(2)[y]/(p(y))$.

З вищезазначеного, слідує що період вихідної послідовності становить 2^{1024}.

Структурно в алгоритмі симетричного потокового перетворення ДСТУ 8845:2019 виділяють три основні функції:

– функція ініціалізації, яка приймає в якості вхідних даних 256-бітний вектор ініціалізації IV та 256-бітний або 512-бітний секретний ключ K, і виробляє початкове значення змінної стану $S_0 = (s^{(0)}, r^{(0)})$;

– функція наступного стану Next, яка приймає на вхід зміну стану $S_i = (s^{(i)}, r^{(i)})$ та виробляє наступне значення змінної стану $S_{i+1} = (s^{(i+1)}, r^{(i+1)})$;

– функція ключового потоку Strm, що приймає на вході змінну стану $S_i = (s^{(i)}, r^{(i)})$ та виробляє на виході 64-бітний ключовий потік Z^i.

Також функція Next може виконуватися в двох режимах, в залежності від способу виконання ітерації, як частина реалізації ініціалізації алгоритму ДСТУ 8845:2019 або як частина функції ключового потоку Strm.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено інтерфейс програмного забезпечення, розробленого у результаті виконання бакалаврської дипломної роботи. Розроблене програмне забезпечення системи управління ідентифікацією та доступом до інформаційних ресурсів складається з наступних функціональних блоків:

– Навігаційне меню: Система автентифікації; Сертифікати; Параметри; Журнал подій; Довідка.

– Вікна обрання профілю.

– Вікно виведення результату роботи системи.

– Функціональних кнопок ПЗ: Параметри; Про програму ; Змінити ПІН; Журнал; Додати; Видалити; Редагувати; Додати сертифікат.

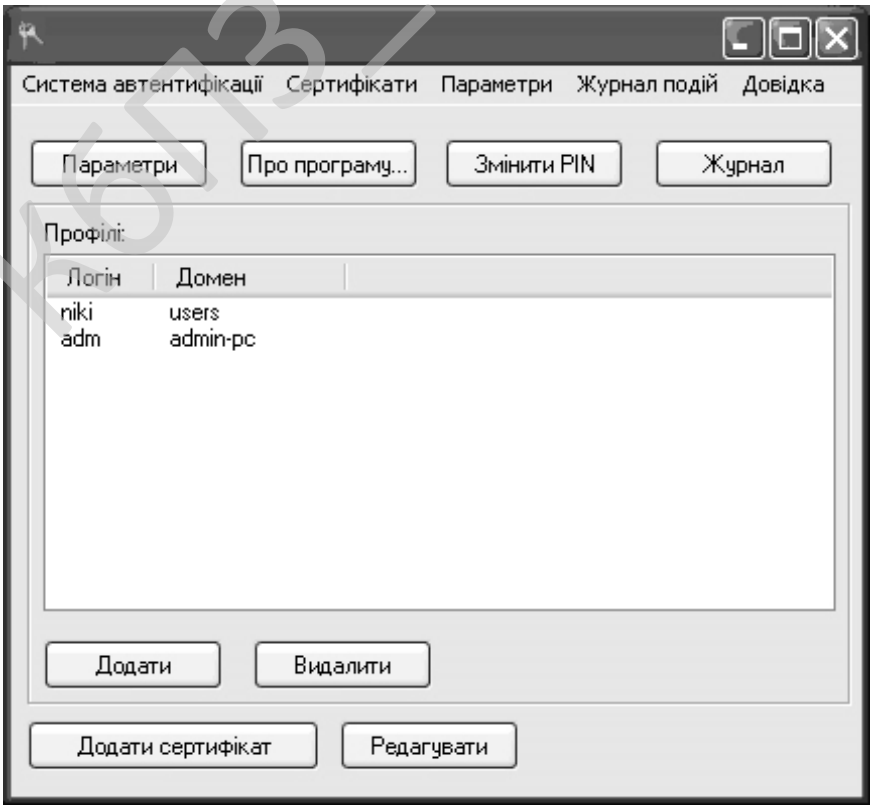

Рисунок 5.1 – Головне вікно розробленого ПЗ

Для перегляду короткої довідки про програму слід натиснути на основному вікні кнопку авторського права, після чого на екрані з’явиться вікно показане на рисунку 5.2.

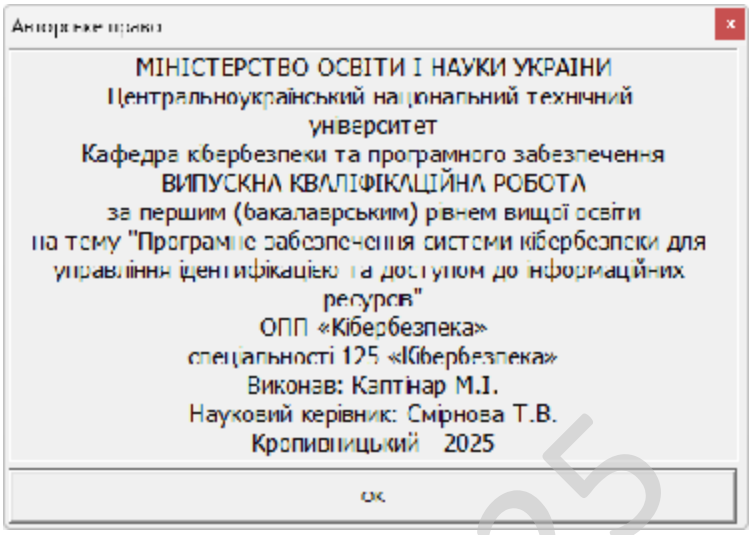

Рисунок 5.2 – Вікно розробника ПЗ

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів.

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

Проводилось тестування форматом білої скриньки засноване на аналізі керуючої структури програми. Програма вважається повністю перевіреною, якщо проведено вичерпне тестування маршрутів (шляхів) її графа управління.

У цьому випадку формуються тестові варіанти, в яких:

– Гарантується перевірка всіх незалежних маршрутів програми.

– Знаходяться гілки True, False для всіх логічних рішень.

– Виконуються всі цикли (у межах їхніх кордонів та діапазонів).

– Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

– Кількість незалежних маршрутів може бути дуже велика.

– Повне тестування маршрутів не гарантує відповідності програми вихідним вимогам до неї.

– У програмі можуть бути пропущені деякі маршрути.

– Не можна виявити помилки, поява яких залежить від даних.

Переваги тестування "білої скриньки" пов'язані з тим, що принцип «білої скриньки» дозволяє врахувати особливості програмних помилок:

– Кількість помилок мінімально в «центрі» і максимально на «периферії» програми.

– Попередні припущення про ймовірність потоку керування або даних у програмі часто бувають некоректними. У результаті типовим може стати маршрут, модель обчислень за яким опрацьована слабо.

– При записі алгоритму програмного забезпечення у вигляді тексту на мові програмування можливе внесення типових помилок трансляції (синтаксичних та семантичних).

– Деякі результати в програмі залежать не від вихідних даних, а від внутрішніх станів програми.

Обрано умови розповсюдження – commercial software.

Програмне забезпечення, створене комерційною організацією з метою отримання прибутку від його використання іншими, наприклад, шляхом продажу копій.

Найважливішою особливістю комерційний програмних продуктів є підтримка великих компаній, прямо зацікавлених у поширенні програм. Багато організацій надають виключно платну підтримку своїх продуктів, такий підхід, як правило, використовують організації надають відкриті вихідні коди. Для продуктів, що розповсюджуються на комерційній основі діють зазвичай безкоштовні служби підтримки, покликані збільшити рівень довіри у клієнтів і потенційних покупців.

Далеко не завжди, але як правило терміни критично важливих змін в комерційних продуктах значно менше, ніж у некомерційних проектів. Це пов'язано з тим, що над комерційним продуктом працюють цілі групи розробників і ця робота є їх основним заняттям. Розробникам-початківцям як правило доводиться шукати додаткові способи заробітку, і це збільшує час, що витрачається на доповнення і зміни програм. Так як основним рушійним фактором створення комерційного ПЗ є одержання прибутку, то комерційні програмні продукти першими заповнюють вільні ніші та пропонують варіанти вирішення завдань відразу по мірі виявлення вакууму в будь-якому секторі ринку.

Окремий вид комерційних програм, коли їх розробка оплачується безпосередньо замовником. Такі програми найчастіше позбавлені всіх переваг комерційних продуктів, оскільки мають обмежений бюджет, але більш адаптовані до вимог замовника, ніж аналоги.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки для управління ідентифікацією та доступом до інформаційних ресурсів.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем для управління ідентифікацією та доступом до інформаційних ресурсів.

– Досліджена система для управління ідентифікацією та доступом до інформаційних ресурсів.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки для управління ідентифікацією та доступом до інформаційних ресурсів.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання для управління ідентифікацією та доступом до інформаційних ресурсів.

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

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

Програма реалізована на мові високого рівня Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи кібербезпеки для управління ідентифікацією та доступом до

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

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм ДСТУ 8845:2019.

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. Awais Rashid, Howard Chivers, George Danezis, Emil Lupu, Andrew Martin. CyBOK The Cyber Security Body of Knowledge. The National Cyber Security Centre. 2019. 854 p.

2. Loren Kohnfelder. Designing Secure Software. No Starch Press. 2022. 332 p.

3. Samir Kumar Rakshit. Ethical Hacker's Penetration Testing Guide. BPB Online. 2022. 509 p.

4. Corey J. Ball. Hacking APIs. No Starch Press. 2022. 353 p.

5. Kevin Beaver. Hacking for Dummies. John Wiley & Sons. 2022. 419 p.

6. Mark S. Merkow. Practical Security for Agile and DevOps. CRC Press. 2022. 236 p

7. Derek Fisher. Application Security Program Handbook. Manning Publications. 2021. 155 p.

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

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

10. 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.

11. 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.

12. 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.

13. 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.

14. 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.

15. 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.

16. 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.

17. 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.

18. 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.

19. 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,

20. 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.

21. 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.

22. 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

23. 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.

24. 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.

25. 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.*

26. 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.

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

28. 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.

29. 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.

30. 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.

31. 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.

32. 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.

33. 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.

34. 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.

35. 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.

36. 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.

37. 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.

38. 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.

39. 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.

40. 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.

41. 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.

42. 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.

43. 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.

44. 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.

45. 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.

46. 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.

47. 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.

48. 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.

49. 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.

50. 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.

51. 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.

52. 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.

53. 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.

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки для управління ідентифікацією та доступом до інформаційних ресурсів.

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. № 51-02 від 17.01.2025 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи кібербезпеки для управління ідентифікацією та доступом до інформаційних ресурсів.

4 Джерела розробки

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

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

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

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи кібербезпеки для управління ідентифікацією та доступом до інформаційних ресурсів;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Python.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 77 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 4.06.2025 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

_____ Смірнова Т.В.

Програмне забезпечення системи кібербезпеки для управління ідентифікацією та доступом до інформаційних ресурсів

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 20

Літера: РП

Кропивницький – 2025 року

Основна програма

```
#!/usr/bin/env python3
#Імпорт необхідних модулів для роботи системи кібербезпеки
import sqlite3
#Імпорт модуля для роботи з хешуванням паролів
import hashlib
#Імпорт модуля для генерації криптографічно стійких випадкових чисел
import secrets
#Імпорт модуля для роботи з датою та часом
import datetime
#Імпорт модуля для ведення логів
import logging
#Імпорт модуля для роботи з часом
import time
#Імпорт модуля для роботи з системними параметрами
import sys

#Налаштування системного логування
logging.basicConfig(filename="cybersecurity.log", level=logging.INFO,
format='%(asctime)s %(message)s')
#Налаштування додаткових параметрів логування
logging.info("Система кібербезпеки ініціалізована")
#Додатковий запис в лог для відслідковування запуску системи
logging.info("Запуск програми кібербезпеки")

#Клас для роботи з базою даних системи кібербезпеки
class DatabaseManager:
    def __init__(self, db_file="cybersecurity.db"):
        self.db_file = db_file
        self.conn = sqlite3.connect(self.db_file)
        self.create_tables()
        #Запис в лог про створення об’єкту керування базою даних
        logging.info("Ініціалізовано менеджер бази даних")

    def create_tables(self):
        cursor = self.conn.cursor()
#Створення таблиці користувачів з полями id, username, password_hash, salt,
role, created_at
        cursor.execute("""
        CREATE TABLE IF NOT EXISTS users (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            username TEXT UNIQUE NOT NULL,
            password_hash TEXT NOT NULL,
            salt TEXT NOT NULL,
            role TEXT NOT NULL,
            created_at TEXT NOT NULL
        )
        """)
#Створення таблиці для зберігання логів доступу з полями id, username, action,
timestamp
        cursor.execute("""
        CREATE TABLE IF NOT EXISTS access_logs (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            username TEXT NOT NULL,
            action TEXT NOT NULL,
            timestamp TEXT NOT NULL
        )
        """)
        self.conn.commit()
        #Логування успішного створення таблиць
        logging.info("Таблиці бази даних створені або вже існують")
```

```
    def query_user(self, username):
        cursor = self.conn.cursor()
#Виконання SQL-запиту для отримання даних користувача за username
        cursor.execute("SELECT id, username, password_hash, salt, role,
created_at FROM users WHERE username = ?", (username,))
        result = cursor.fetchone()
        #Логування виконання запиту до бази даних
        logging.info("Запит даних користувача: " + str(username))
        return result

    def insert_user(self, username, password_hash, salt, role):
        cursor = self.conn.cursor()
#Отримання поточної дати та часу у форматі ISO
        created_at = datetime.datetime.utcnow().isoformat()
#Виконання SQL-запиту для вставки нового користувача
        cursor.execute("INSERT INTO users (username, password_hash, salt, role,
created_at) VALUES (?, ?, ?, ?, ?)",
                       (username, password_hash, salt, role, created_at))
        self.conn.commit()
        #Логування успішної реєстрації нового користувача
        logging.info("Користувача додано до бази даних: " + username)

    def log_access(self, username, action):
        cursor = self.conn.cursor()
#Отримання поточного часу для запису події
        timestamp = datetime.datetime.utcnow().isoformat()
#Виконання SQL-запиту для логування дій користувача
        cursor.execute("INSERT INTO access_logs (username, action, timestamp)
VALUES (?, ?, ?)",
                       (username, action, timestamp))
        self.conn.commit()
        #Логування факту запису події
        logging.info("Дія '" + action + "' для користувача " + username + "
залогована")

    def close(self):
#Закриття з'єднання з базою даних
        self.conn.close()
        #Логування завершення роботи з базою даних
        logging.info("З'єднання з базою даних закрито")

#Функція для хешування пароля за допомогою алгоритму SHA256 із додаванням солі
def hash_password(password, salt):
    return hashlib.sha256((password + salt).encode()).hexdigest()

#Функція для генерації унікальної солі для кожного користувача
def generate_salt():
    return secrets.token_hex(16)

#Функція для генерації токена сесії користувача
def generate_token():
    return secrets.token_hex(32)

#Клас, що реалізує основну логіку системи кібербезпеки
class CyberSecuritySystem:
    def __init__(self):
        self.db_manager = DatabaseManager()
        self.active_sessions = {}  #Зберігання активних сесій: token -> username
        #Логування створення об'єкту системи кібербезпеки
        logging.info("Система кібербезпеки ініціалізована")

    def register_user(self, username, password, role="user"):
#Перевірка чи існує користувач з таким ім'ям
```

```
        existing = self.db_manager.query_user(username)
        if existing:
            logging.info("Спроба реєстрації вже існуючого користувача: " +
username)
            return False, "Користувач вже існує"
#Генерація солі для нового користувача
        salt = generate_salt()
#Хешування пароля користувача із використанням згенерованої солі
        password_hash = hash_password(password, salt)
#Вставка даних користувача до бази даних
        self.db_manager.insert_user(username, password_hash, salt, role)
        logging.info("Користувача зареєстровано успішно: " + username)
        return True, "Користувача зареєстровано"

    def authenticate_user(self, username, password):
#Отримання даних користувача з бази даних
        user_record = self.db_manager.query_user(username)
        if not user_record:
            logging.info("Спроба аутентифікації невдалась, користувача не
знайдено: " + username)
            return False, "Користувача не знайдено"
        user_id, db_username, db_password_hash, salt, role, created_at =
user_record
#Перевірка правильності введеного пароля шляхом порівняння хешів
        if hash_password(password, salt) == db_password_hash:
#Генерація сесійного токена для користувача
            token = generate_token()
            self.active_sessions[token] = username
#Логування події входу в систему
            self.db_manager.log_access(username, "login")
            logging.info("Користувача аутентифіковано: " + username)
            return True, token
        else:
            logging.info("Невдала спроба аутентифікації для користувача: " +
username)
            return False, "Невірний пароль"

    def logout_user(self, token):
#Перевірка наявності сесійного токена
        if token in self.active_sessions:
            username = self.active_sessions[token]
#Логування події виходу користувача з системи
            self.db_manager.log_access(username, "logout")
            del self.active_sessions[token]
            logging.info("Користувача вийшов з системи: " + username)
            return True, "Вихід виконано"
        logging.info("Спроба виходу з системи з невірним токеном")
        return False, "Невірний токен сесії"

    def change_password(self, username, old_password, new_password):
#Отримання даних користувача для зміни пароля
        user_record = self.db_manager.query_user(username)
        if not user_record:
            logging.info("Зміна пароля невдала, користувача не знайдено: " +
username)
            return False, "Користувача не знайдено"
        user_id, db_username, db_password_hash, salt, role, created_at =
user_record
#Перевірка правильності старого пароля
        if hash_password(old_password, salt) == db_password_hash:
#Генерація нової солі для нового пароля
            new_salt = generate_salt()
#Хешування нового пароля
```

```
            new_password_hash = hash_password(new_password, new_salt)
            cursor = self.db_manager.conn.cursor()
#Виконання SQL-запиту для оновлення пароля користувача
            cursor.execute("UPDATE users SET password_hash = ?, salt = ? WHERE
username = ?", (new_password_hash, new_salt, username))
            self.db_manager.conn.commit()
#Логування події зміни пароля
            self.db_manager.log_access(username, "password_change")
            logging.info("Пароль змінено для користувача: " + username)
            return True, "Пароль змінено"
        else:
            logging.info("Спроба зміни пароля невдала через невірний старий
пароль для користувача: " + username)
            return False, "Невірний старий пароль"

    def assign_role(self, username, new_role):
#Отримання даних користувача для призначення нової ролі
        user_record = self.db_manager.query_user(username)
        if not user_record:
            logging.info("Призначення ролі невдале, користувача не знайдено: " +
username)
            return False, "Користувача не знайдено"
        cursor = self.db_manager.conn.cursor()
#Виконання SQL-запиту для оновлення ролі користувача
        cursor.execute("UPDATE users SET role = ? WHERE username = ?",
(new_role, username))
        self.db_manager.conn.commit()
#Логування зміни ролі користувача
        self.db_manager.log_access(username, "role_change")
        logging.info("Роль користувача " + username + " змінено на: " +
new_role)
        return True, "Роль оновлено"

    def check_permission(self, token, required_role):
#Перевірка дійсності сесійного токена
        if token not in self.active_sessions:
            logging.info("Перевірка дозволів невдала через невірний токен")
            return False, "Невірна сесія"
        username = self.active_sessions[token]
#Отримання даних користувача для перевірки дозволів
        user_record = self.db_manager.query_user(username)
        if not user_record:
            logging.info("Перевірка дозволів невдала, користувача не знайдено: "
+ username)
            return False, "Користувача не знайдено"
        user_role = user_record[4]
#Порівняння ролі користувача з необхідною роллю для доступу
        if user_role == required_role or user_role == "admin":
            logging.info("Дозвіл надано користувачу: " + username)
            return True, "Дозвіл надано"
        logging.info("Доступ заборонено для користувача: " + username)
        return False, "Доступ заборонено"

    def get_access_logs(self, token):
#Перевірка дозволу на доступ до логів для адміністратора
        permission, msg = self.check_permission(token, "admin")
        if not permission:
            return False, "Немає прав доступу"
        cursor = self.db_manager.conn.cursor()
#Виконання SQL-запиту для отримання всіх записів логів доступу
        cursor.execute("SELECT username, action, timestamp FROM access_logs")
        logs = cursor.fetchall()
        logging.info("Адміністратор отримав логи доступу")
```

```
        return True, logs

    def simulate_attack(self):
#Імітація процесу виявлення кібератаки
        logging.warning("Виявлено потенційну атаку")
        time.sleep(1)
        logging.warning("Імітація атаки завершена")
        return "Імітація атаки завершена"

    def cleanup_sessions(self):
#Очищення списку активних сесій (імплементація за замовчуванням)
        logging.info("Очищення активних сесій")
        self.active_sessions.clear()
        return "Активні сесії очищено"

    def run_security_audit(self):
#Запуск повного аудиту безпеки системи
        logging.info("Аудит безпеки розпочато")
        audit_report = []
        audit_report.append("Звіт аудиту безпеки")
        audit_report.append("Кількість активних сесій: " +
str(len(self.active_sessions)))
        audit_report.append("Дані логів доступу отримано")
        logging.info("Аудит безпеки завершено")
        return audit_report

    def __del__(self):
#Закриття з'єднання з базою даних під час видалення об'єкту системи
        self.db_manager.close()
        #Логування завершення роботи об'єкту системи кібербезпеки
        logging.info("Об'єкт системи кібербезпеки знищено")

#Функція для виконання резервного копіювання бази даних
def perform_database_backup(db_file="cybersecurity.db",
backup_file="cybersecurity_backup.db"):
    conn = sqlite3.connect(db_file)
    bck = sqlite3.connect(backup_file)
    with bck:
        conn.backup(bck)
    bck.close()
    conn.close()
    logging.info("Резервне копіювання бази даних виконано")
    return "Резервне копіювання успішне"

#Функція для відновлення бази даних з резервної копії
def restore_database(backup_file="cybersecurity_backup.db",
db_file="cybersecurity.db"):
    conn = sqlite3.connect(backup_file)
    bck = sqlite3.connect(db_file)
    with bck:
        conn.backup(bck)
    bck.close()
    conn.close()
    logging.info("Відновлення бази даних виконано")
    return "Відновлення успішне"

#Функція для генерації детального звіту роботи системи
def generate_report(system):
    report = []
    report.append("=== Звіт системи кібербезпеки ===")
    report.append("Активних сесій: " + str(len(system.active_sessions)))
    report.append("Статус резервного копіювання: Успішно")
    report.append("Деталі аудиту: " + ", ".join(system.run_security_audit()))
```

```
    logging.info("Звіт системи згенеровано")
    return "\n".join(report)

#Додаткова допоміжна функція для розширення коду (функція-заглушка)
def dummy_function_one():
#Ініціалізація змінної результату
    result = 0
#Цикл для обчислення суми чисел
    for i in range(10):
        result += i
#Повернення результату обчислень
    return result

#Додаткова допоміжна функція для розширення коду (функція-заглушка)
def dummy_function_two():
#Ініціалізація списку для зберігання значень
    values = []
#Цикл для обчислення квадратів чисел
    for j in range(10, 20):
        values.append(j * j)
#Повернення списку обчислених значень
    return values

#Додаткова допоміжна функція для розширення коду (функція-заглушка)
def dummy_function_three():
#Ініціалізація текстового рядка для обробки
    text = "Cybersecurity"
#Ініціалізація змінної результату
    result = ""
#Цикл для обробки кожного символу тексту
    for char in text:
        result += char + "-"
#Повернення обробленого тексту
    return result

#Додаткова допоміжна функція для розширення коду (функція-заглушка)
def dummy_function_four():
#Створення списку чисел від 0 до 49
    numbers = [i for i in range(50)]
#Обчислення суми чисел у списку
    sum_numbers = sum(numbers)
#Повернення суми обчислених чисел
    return sum_numbers

#Додаткова допоміжна функція для розширення коду (функція-заглушка)
def dummy_function_five():
#Ініціалізація словника для зберігання пар чисел
    data = {}
#Цикл для заповнення словника парами ключ-значення
    for k in range(5):
        data[k] = k * k
#Повернення заповненого словника
    return data

#Головна функція для взаємодії з користувачем через командний рядок
def main():
    system = CyberSecuritySystem()
    while True:
#Виведення меню для користувача
        print("=== Меню системи кібербезпеки ===")
        print("1. Реєстрація користувача")
        print("2. Аутентифікація користувача")
        print("3. Вихід користувача")
```

```
        print("4. Зміна пароля")
        print("5. Призначення ролі")
        print("6. Перегляд логів доступу (тільки для адміністратора)")
        print("7. Імітація атаки")
        print("8. Очищення активних сесій")
        print("9. Запуск аудиту безпеки")
        print("10. Резервне копіювання бази даних")
        print("11. Відновлення бази даних")
        print("12. Генерація звіту")
        print("13. Dummy Function One")
        print("14. Dummy Function Two")
        print("15. Dummy Function Three")
        print("16. Dummy Function Four")
        print("17. Dummy Function Five")
        print("0. Вихід")
        choice = input("Введіть свій вибір: ")
        if choice == "1":
            username = input("Введіть ім’я користувача: ")
            password = input("Введіть пароль: ")
            role = input("Введіть роль (user/admin): ")
            success, message = system.register_user(username, password, role)
            print(message)
        elif choice == "2":
            username = input("Введіть ім’я користувача: ")
            password = input("Введіть пароль: ")
            success, result = system.authenticate_user(username, password)
            if success:
                print("Аутентифікацію виконано. Ваш сесійний токен:")
                print(result)
            else:
                print(result)
        elif choice == "3":
            token = input("Введіть сесійний токен: ")
            success, message = system.logout_user(token)
            print(message)
        elif choice == "4":
            username = input("Введіть ім’я користувача: ")
            old_password = input("Введіть старий пароль: ")
            new_password = input("Введіть новий пароль: ")
            success, message = system.change_password(username, old_password,
new_password)
            print(message)
        elif choice == "5":
            username = input("Введіть ім’я користувача: ")
            new_role = input("Введіть нову роль: ")
            success, message = system.assign_role(username, new_role)
            print(message)
        elif choice == "6":
            token = input("Введіть сесійний токен адміністратора: ")
            success, logs = system.get_access_logs(token)
            if success:
                for log in logs:
                    print("Користувач:", log[0], "| Дія:", log[1], "| Час:",
log[2])
            else:
                print(logs)
        elif choice == "7":
            result = system.simulate_attack()
            print(result)
        elif choice == "8":
            result = system.cleanup_sessions()
            print(result)
        elif choice == "9":
```

```
            report = system.run_security_audit()
            for line in report:
                print(line)
        elif choice == "10":
            result = perform_database_backup()
            print(result)
        elif choice == "11":
            result = restore_database()
            print(result)
        elif choice == "12":
            report = generate_report(system)
            print(report)
        elif choice == "13":
            result = dummy_function_one()
            print("Результат Dummy Function One:", result)
        elif choice == "14":
            result = dummy_function_two()
            print("Результат Dummy Function Two:", result)
        elif choice == "15":
            result = dummy_function_three()
            print("Результат Dummy Function Three:", result)
        elif choice == "16":
            result = dummy_function_four()
            print("Результат Dummy Function Four:", result)
        elif choice == "17":
            result = dummy_function_five()
            print("Результат Dummy Function Five:", result)
        elif choice == "0":
            print("Вихід із системи...")
            break
        else:
            print("Невірний вибір, спробуйте ще раз.")
        time.sleep(1)
        print("\n")
#Виклик головної функції, якщо модуль запущено напряму
if __name__ == '__main__':
    main()
```

Файл TwoFactorAuth.py

```
import threading
import time
import datetime
import secrets
import hashlib
import smtplib

class TwoFactorAuth:
    def __init__(self):
        self.otp_storage = {}
    def generate_otp(self, username):
        otp = str(secrets.randbelow(1000000)).zfill(6)
        self.otp_storage[username] = otp
        return otp
    def verify_otp(self, username, otp):
        if username in self.otp_storage and self.otp_storage[username] == otp:
            del self.otp_storage[username]
            return True
        return False

class AccountLockoutManager:
    def __init__(self, threshold=3, lockout_duration=300):
        self.failed_attempts = {}
        self.threshold = threshold
        self.lockout_duration = lockout_duration
    def record_failed_attempt(self, username):
        now = time.time()
        if username in self.failed_attempts:
            count, first_attempt = self.failed_attempts[username]
            self.failed_attempts[username] = (count + 1, first_attempt)
        else:
            self.failed_attempts[username] = (1, now)
    def is_locked(self, username):
        if username in self.failed_attempts:
            count, first_attempt = self.failed_attempts[username]
            if count >= self.threshold:
                if time.time() - first_attempt < self.lockout_duration:
                    return True
                else:
                    self.failed_attempts[username] = (0, time.time())
                    return False
        return False
    def reset_attempts(self, username):
        if username in self.failed_attempts:
            self.failed_attempts[username] = (0, time.time())

class SessionExpirationManager:
    def __init__(self, expiration_time=600):
        self.sessions = {}
        self.expiration_time = expiration_time
        self.lock = threading.Lock()
        self.thread = threading.Thread(target=self._cleanup_sessions)
        self.thread.daemon = True
        self.thread.start()
    def add_session(self, token, username):
        with self.lock:
            self.sessions[token] = (username, time.time())
    def update_session(self, token):
        with self.lock:
            if token in self.sessions:
                username, _ = self.sessions[token]
```

```
                self.sessions[token] = (username, time.time())
    def remove_session(self, token):
        with self.lock:
            if token in self.sessions:
                del self.sessions[token]
    def get_session(self, token):
        with self.lock:
            return self.sessions.get(token, None)
    def _cleanup_sessions(self):
        while True:
            time.sleep(5)
            with self.lock:
                tokens_to_remove = []
                for token, (username, last_activity) in self.sessions.items():
                    if time.time() - last_activity > self.expiration_time:
                        tokens_to_remove.append(token)
                for token in tokens_to_remove:
                    del self.sessions[token]

class NotificationSystem:
    def __init__(self, smtp_server='localhost', smtp_port=25):
        self.smtp_server = smtp_server
        self.smtp_port = smtp_port
    def send_notification(self, recipient, subject, message):
        try:
            server = smtplib.SMTP(self.smtp_server, self.smtp_port)
            email_message = "Subject: {}\n\n{}".format(subject, message)
            server.sendmail("noreply@example.com", recipient, email_message)
            server.quit()
        except Exception as e:
            pass

class AnomalyDetection:
    def __init__(self):
        self.login_history = {}
    def record_login(self, username, login_time):
        if username not in self.login_history:
            self.login_history[username] = []
        self.login_history[username].append(login_time)
    def analyze_login(self, username, current_time):
        if username not in self.login_history or
len(self.login_history[username]) < 5:
            return False
        times = self.login_history[username][-5:]
        avg = sum(times) / len(times)
        if abs(current_time - avg) > 3600:
            return True
        return False

class EnhancedCyberSecuritySystem:
    def __init__(self):
        self.users = {}
        self.two_factor = TwoFactorAuth()
        self.lockout_manager = AccountLockoutManager()
        self.session_manager = SessionExpirationManager()
        self.notification = NotificationSystem()
        self.anomaly_detector = AnomalyDetection()
    def register_user(self, username, password, email):
        salt = secrets.token_hex(16)
        pwd_hash = hashlib.sha256((password + salt).encode()).hexdigest()
        self.users[username] = {'password_hash': pwd_hash, 'salt': salt,
'email': email}
    def authenticate_user(self, username, password):
```

```
        if username not in self.users:
            return False, "User not found"
        if self.lockout_manager.is_locked(username):
            return False, "Account locked"
        salt = self.users[username]['salt']
        pwd_hash = hashlib.sha256((password + salt).encode()).hexdigest()
        if pwd_hash != self.users[username]['password_hash']:
            self.lockout_manager.record_failed_attempt(username)
            return False, "Invalid credentials"
        self.lockout_manager.reset_attempts(username)
        otp = self.two_factor.generate_otp(username)
        email = self.users[username]['email']
        self.notification.send_notification(email, "Your OTP Code", "Your OTP
is: " + otp)
        return True, "OTP sent"
    def verify_otp(self, username, otp):
        if self.two_factor.verify_otp(username, otp):
            token = secrets.token_hex(32)
            self.session_manager.add_session(token, username)
            now = time.time()
            self.anomaly_detector.record_login(username, now)
            anomaly = self.anomaly_detector.analyze_login(username, now)
            if anomaly:
                email = self.users[username]['email']
                self.notification.send_notification(email, "Anomaly Detected",
"Unusual login activity detected.")
            return True, token
        return False, "Invalid OTP"
    def logout_user(self, token):
        self.session_manager.remove_session(token)
        return True, "Logged out"
    def check_session(self, token):
        session = self.session_manager.get_session(token)
        if session:
            self.session_manager.update_session(token)
            return True, session[0]
        return False, "Session expired"
    def change_password(self, username, old_password, new_password):
        if username not in self.users:
            return False, "User not found"
        salt = self.users[username]['salt']
        pwd_hash = hashlib.sha256((old_password + salt).encode()).hexdigest()
        if pwd_hash != self.users[username]['password_hash']:
            return False, "Invalid current password"
        new_salt = secrets.token_hex(16)
        new_pwd_hash = hashlib.sha256((new_password +
new_salt).encode()).hexdigest()
        self.users[username]['salt'] = new_salt
        self.users[username]['password_hash'] = new_pwd_hash
        email = self.users[username]['email']
        self.notification.send_notification(email, "Password Changed", "Your
password has been successfully changed.")
        return True, "Password changed"

def main():
    system = EnhancedCyberSecuritySystem()
    system.register_user("alice", "password123", "alice@example.com")
    system.register_user("bob", "securepass", "bob@example.com")
    print("Login for alice:")
    success, message = system.authenticate_user("alice", "password123")
    print(message)
    if success:
        otp = input("Enter OTP for alice: ")
```

```
        success, token = system.verify_otp("alice", otp)
        if success:
            print("Alice logged in, token:", token)
            valid, user = system.check_session(token)
            print("Session valid for:", user)
            res, msg = system.change_password("alice", "password123",
"newpass456")
            print(msg)
            system.logout_user(token)
        else:
            print("OTP verification failed")
    print("Login for bob with wrong password:")
    success, message = system.authenticate_user("bob", "wrongpass")
    print(message)
    success, message = system.authenticate_user("bob", "wrongpass")
    print(message)
    success, message = system.authenticate_user("bob", "wrongpass")
    print(message)
    success, message = system.authenticate_user("bob", "securepass")
    print(message)
    if success:
        otp = input("Enter OTP for bob: ")
        success, token = system.verify_otp("bob", otp)
        if success:
            print("Bob logged in, token:", token)
            system.logout_user(token)
    print("End of simulation.")

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

Файл AccessControlPanel.py

```
from flask import Flask, request, jsonify
import threading
import time
import datetime
import sqlite3
import logging
import random
import secrets

class ExtendedLogger:
    def __init__(self, filename="extended.log"):
        self.filename = filename
        self.logger = logging.getLogger("ExtendedLogger")
        self.logger.setLevel(logging.INFO)
        handler = logging.FileHandler(self.filename)
        formatter = logging.Formatter('%(asctime)s - %(ip)s - %(user_agent)s -
%(message)s')
        handler.setFormatter(formatter)
        self.logger.addHandler(handler)
    def log(self, ip, user_agent, message):
        extra = {'ip': ip, 'user_agent': user_agent}
        self.logger.info(message, extra=extra)

class AccessControlPanel:
    def __init__(self):
        self.policies = {}
    def add_policy(self, policy_name, policy_detail):
        self.policies[policy_name] = policy_detail
    def remove_policy(self, policy_name):
        if policy_name in self.policies:
            del self.policies[policy_name]
    def update_policy(self, policy_name, policy_detail):
        self.policies[policy_name] = policy_detail
    def list_policies(self):
        return self.policies
    def run_panel(self):
        while True:
            print("Access Control Panel")
            print("1. Add Policy")
            print("2. Remove Policy")
            print("3. Update Policy")
            print("4. List Policies")
            print("0. Exit Panel")
            choice = input("Enter choice: ")
            if choice=="1":
                name = input("Policy name: ")
                detail = input("Policy detail: ")
                self.add_policy(name, detail)
                print("Policy added")
            elif choice=="2":
                name = input("Policy name to remove: ")
                self.remove_policy(name)
                print("Policy removed")
            elif choice=="3":
                name = input("Policy name to update: ")
                detail = input("New policy detail: ")
                self.update_policy(name, detail)
                print("Policy updated")
            elif choice=="4":
                policies = self.list_policies()
                for k,v in policies.items():
```

```
                    print(k, ":", v)
            elif choice=="0":
                break
            else:
                print("Invalid choice")
            time.sleep(1)

class IDSIntegration:
    def __init__(self, db_file="ids_logs.db"):
        self.db_file = db_file
        self.conn = sqlite3.connect(self.db_file, check_same_thread=False)
        self.create_table()
        self.running = True
        self.thread = threading.Thread(target=self.monitor_logs)
        self.thread.daemon = True
        self.thread.start()
    def create_table(self):
        cursor = self.conn.cursor()
        cursor.execute("CREATE TABLE IF NOT EXISTS ids_events (id INTEGER
PRIMARY KEY AUTOINCREMENT, event TEXT, timestamp TEXT)")
        self.conn.commit()
    def record_event(self, event):
        timestamp = datetime.datetime.utcnow().isoformat()
        cursor = self.conn.cursor()
        cursor.execute("INSERT INTO ids_events (event, timestamp) VALUES (?,
?)", (event, timestamp))
        self.conn.commit()
    def monitor_logs(self):
        while self.running:
            if random.random() < 0.1:
                self.record_event("Suspicious activity detected")
            time.sleep(2)
    def stop(self):
        self.running = False
        self.thread.join()
        self.conn.close()

app = Flask(__name__)
extended_logger = ExtendedLogger()
access_control_panel = AccessControlPanel()
ids_integration = IDSIntegration()

@app.route("/api/login", methods=["POST"])
def api_login():
    data = request.get_json()
    username = data.get("username", "")
    password = data.get("password", "")
    token = secrets.token_hex(16)
    ip = request.remote_addr
    user_agent = request.headers.get("User-Agent", "")
    extended_logger.log(ip, user_agent, "User {} attempted
login".format(username))
    return jsonify({"status": "success", "token": token})

@app.route("/api/data", methods=["GET"])
def api_data():
    ip = request.remote_addr
    user_agent = request.headers.get("User-Agent", "")
    extended_logger.log(ip, user_agent, "Data accessed")
    return jsonify({"data": "Secure data payload"})

class CustomReporting:
    def __init__(self, db_file="ids_logs.db"):
```

```
        self.db_file = db_file
    def generate_report(self):
        conn = sqlite3.connect(self.db_file)
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM ids_events")
        events = cursor.fetchall()
        conn.close()
        report = "Custom Security Report\n"
        report += "Report generated at: " +
datetime.datetime.utcnow().isoformat() + "\n"
        report += "Number of IDS events: " + str(len(events)) + "\n"
        for event in events:
            report += "Event: " + str(event[1]) + " Timestamp: " + str(event[2])
+ "\n"
        return report

custom_reporting = CustomReporting()

def run_flask_app():
    app.run(port=5001)

def main():
    flask_thread = threading.Thread(target=run_flask_app)
    flask_thread.daemon = True
    flask_thread.start()
    panel_thread = threading.Thread(target=access_control_panel.run_panel)
    panel_thread.daemon = True
    panel_thread.start()
    try:
        while True:
            print("Custom Reporting")
            print(custom_reporting.generate_report())
            time.sleep(10)
    except KeyboardInterrupt:
        ids_integration.stop()
        print("Exiting")

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

Файл LDAPIntegration.py

```
import ldap3
import time
import random
import threading
import sqlite3
import os
import hashlib
import requests
import json
import schedule

class LDAPIntegration:
    def __init__(self, server_uri, base_dn):
        self.server_uri = server_uri
        self.base_dn = base_dn
        self.server = ldap3.Server(self.server_uri)
        self.connection = None
    def connect(self, user_dn, password):
        self.connection = ldap3.Connection(self.server, user=user_dn,
password=password, auto_bind=True)
        return self.connection.bound
    def search_user(self, username):
        search_filter = f"(uid={username})"
        self.connection.search(self.base_dn, search_filter,
attributes=ldap3.ALL_ATTRIBUTES)
        if self.connection.entries:
            return self.connection.entries[0]
        return None
    def disconnect(self):
        if self.connection:
            self.connection.unbind()
            self.connection = None

class PatchManagement:
    def __init__(self, db_file="patches.db"):
        self.db_file = db_file
        self.conn = sqlite3.connect(self.db_file)
        self.create_table()
    def create_table(self):
        cursor = self.conn.cursor()
        cursor.execute("CREATE TABLE IF NOT EXISTS patches (id INTEGER PRIMARY
KEY AUTOINCREMENT, name TEXT, version TEXT, applied INTEGER)")
        self.conn.commit()
    def check_for_updates(self):
        updates = []
        for i in range(1, 4):
            updates.append({"name": f"Patch_{i}", "version":
f"{random.randint(1,5)}.{random.randint(0,9)}"})
        return updates
    def download_patch(self, patch):
        time.sleep(1)
        patch_data = f"Data for {patch['name']} version {patch['version']}"
        return patch_data
    def apply_patch(self, patch, patch_data):
        cursor = self.conn.cursor()
        cursor.execute("INSERT INTO patches (name, version, applied) VALUES (?,
?, ?)", (patch['name'], patch['version'], 1))
        self.conn.commit()
        return True
    def schedule_patch_updates(self):
```

```
        def job():
            updates = self.check_for_updates()
            for patch in updates:
                patch_data = self.download_patch(patch)
                self.apply_patch(patch, patch_data)
        schedule.every(10).seconds.do(job)
        while True:
            schedule.run_pending()
            time.sleep(1)
    def close(self):
        self.conn.close()

class ComplianceModule:
    def __init__(self, config_file="system_config.json"):
        self.config_file = config_file
        if not os.path.exists(self.config_file):
            default_config = {"firewall_enabled": True, "password_policy":
{"min_length": 8, "require_numbers": True}, "logging_enabled": True}
            with open(self.config_file, "w") as f:
                json.dump(default_config, f)
    def load_config(self):
        with open(self.config_file, "r") as f:
            config = json.load(f)
        return config
    def check_compliance(self):
        config = self.load_config()
        issues = []
        if not config.get("firewall_enabled", False):
            issues.append("Firewall is disabled")
        policy = config.get("password_policy", {})
        if policy.get("min_length", 0) < 8:
            issues.append("Password minimum length less than 8")
        if not config.get("logging_enabled", False):
            issues.append("Logging is disabled")
        return issues
    def remediate(self):
        config = self.load_config()
        if not config.get("firewall_enabled", False):
            config["firewall_enabled"] = True
        policy = config.get("password_policy", {})
        if policy.get("min_length", 0) < 8:
            policy["min_length"] = 8
        if not config.get("logging_enabled", False):
            config["logging_enabled"] = True
        with open(self.config_file, "w") as f:
            json.dump(config, f)
        return config

class SelfServicePortal:
    def __init__(self, db_file="users_selfservice.db"):
        self.db_file = db_file
        self.conn = sqlite3.connect(self.db_file)
        self.create_table()
    def create_table(self):
        cursor = self.conn.cursor()
        cursor.execute("CREATE TABLE IF NOT EXISTS users (id INTEGER PRIMARY KEY
AUTOINCREMENT, username TEXT, email TEXT, password_hash TEXT)")
        self.conn.commit()
    def register_user(self, username, email, password):
        salt = os.urandom(16).hex()
        password_hash = hashlib.sha256((password + salt).encode()).hexdigest()
        cursor = self.conn.cursor()
```

```
        cursor.execute("INSERT INTO users (username, email, password_hash)
VALUES (?, ?, ?)", (username, email, password_hash))
        self.conn.commit()
    def update_profile(self, username, new_email):
        cursor = self.conn.cursor()
        cursor.execute("UPDATE users SET email = ? WHERE username = ?",
(new_email, username))
        self.conn.commit()
    def reset_password(self, username, new_password):
        salt = os.urandom(16).hex()
        new_hash = hashlib.sha256((new_password + salt).encode()).hexdigest()
        cursor = self.conn.cursor()
        cursor.execute("UPDATE users SET password_hash = ? WHERE username = ?",
(new_hash, username))
        self.conn.commit()
    def get_user(self, username):
        cursor = self.conn.cursor()
        cursor.execute("SELECT id, username, email FROM users WHERE username =
?", (username,))
        return cursor.fetchone()
    def interactive_menu(self):
        while True:
            print("Self-Service Portal")
            print("1. Register")
            print("2. Update Profile")
            print("3. Reset Password")
            print("4. Get User Info")
            print("0. Exit")
            choice = input("Enter choice: ")
            if choice == "1":
                username = input("Username: ")
                email = input("Email: ")
                password = input("Password: ")
                self.register_user(username, email, password)
                print("User registered")
            elif choice == "2":
                username = input("Username: ")
                new_email = input("New Email: ")
                self.update_profile(username, new_email)
                print("Profile updated")
            elif choice == "3":
                username = input("Username: ")
                new_password = input("New Password: ")
                self.reset_password(username, new_password)
                print("Password reset")
            elif choice == "4":
                username = input("Username: ")
                user = self.get_user(username)
                print("User Info:", user)
            elif choice == "0":
                break
            else:
                print("Invalid choice")
            time.sleep(1)

def main():
    ldap_integration = LDAPIntegration("ldap://localhost", "dc=example,dc=com")
    try:
        ldap_integration.connect("cn=admin,dc=example,dc=com", "adminpassword")
        entry = ldap_integration.search_user("testuser")
        if entry:
            print("LDAP Entry:", entry)
        ldap_integration.disconnect()
```

```
    except Exception as e:
        print("LDAP integration error")
    patch_manager = PatchManagement()
    patch_thread = threading.Thread(target=patch_manager.schedule_patch_updates)
    patch_thread.daemon = True
    patch_thread.start()
    compliance_module = ComplianceModule()
    issues = compliance_module.check_compliance()
    if issues:
        print("Compliance issues found:", issues)
        compliance_module.remediate()
        print("Compliance remediated")
    else:
        print("System is compliant")
    self_service = SelfServicePortal()
    menu_thread = threading.Thread(target=self_service.interactive_menu)
    menu_thread.daemon = True
    menu_thread.start()
    try:
        while True:
            time.sleep(5)
    except KeyboardInterrupt:
        patch_manager.close()
        print("Exiting")
if __name__ == "__main__":
    main()
```