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 другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи захисту персональних даних у мережевих Cloud-системах”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-24М
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Василенко К.О.
«____» _____ 2025 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Смірнов С.А.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

АНОТАЦІЯ

Василенко К.О. Дослідження та програмна реалізація системи захисту персональних даних у мережевих Cloud-системах. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи захисту персональних даних у мережевих Cloud-системах.

Метою розробки є дослідження та програмна реалізація системи захисту персональних даних у мережевих Cloud-системах.

Об'єктом дослідження є процес захисту персональних даних у мережевих Cloud-системах.

Предметом дослідження є методи захисту персональних даних у мережевих Cloud-системах.

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

Результат роботи – програмна реалізація системи захисту персональних даних у мережевих Cloud-системах.

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

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

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

Програму розроблено в середовищі Python.

Ключові слова: комп'ютерна інженерія, захисту даних, Cloud-системи

ABSTRACT

Vasylenko K.O. Research and software implementation of a personal data protection system in network Cloud systems. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the second (master's) level of higher education, software has been developed, which is intended for a personal data protection system in network Cloud systems.

The purpose of the development is the research and software implementation of a personal data protection system in network Cloud systems.

The object of the research is the process of personal data protection in network Cloud systems.

The subject of the research is methods of personal data protection in network Cloud systems.

The research methods are based on methods of information protection in the network, methods of mathematical statistics, methods of software development.

The result of the work is a software implementation of a personal data protection system in network Cloud systems.

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 the software are provided.

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

The program was developed in the Python environment.

Keywords: computer engineering, data protection, Cloud systems

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 3
ВСТУП..... 4
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 7
1.1 Призначення системи..... 7
1.2 Область застосування..... 8
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 9
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти..... 9
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування..... 12
2.3 Розгорнута постановка завдання 15
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 16
3.1 Опис функціонування системи 16
3.2 Розробка структурної схеми..... 25
3.3 Розробка функціональної схеми 40
3.4 Розробка діаграми процесів..... 51
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ..... 53
4.1 Розробка блок-схем та опис алгоритмів функціонування системи..... 53
4.2 Захист розробленого програмного забезпечення..... 76
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ..... 78
6 НАУКОВА НОВИЗНА 83

7 МАРКЕТИНГОВЕ ТА ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ ІТ-ПРОЄКТУ 84
7.1 Визначення цільової аудиторії кінцевого готового продукту 84
7.2 Оцінка привабливості шляхом застосування методів експертних оцінок ... 85
7.3 Вибір методу оцінки вартості ПЗ 86
7.4 Розрахунок економічної ефективності від впровадження реалізованого ПЗ як фактору його привабливості 86
7.5 Пропозиція алгоритму просування проєкту розробки ПЗ 88
7.6 Оптимізація каналів збуту та шляхів реалізації ПЗ 89
7.7 Визначення ключових факторів успіху конкретного проєкту 90
8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ 91
8.1 Вступ 91
8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером 92
8.3 Аналіз умов праці 93
8.4 Техніка безпеки та протипожежна профілактика 97
8.5 Розрахункова частина 99
9 ОСНОВНІ ВИСНОВКИ 102
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 104

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ГПВЧ – генератор псевдовипадкових чисел

ДСТ – державний стандарт

ЕОМ – електронна обчислювальна машина

ІБ – інформаційна безпека

ІС – інформаційна система

КСЗІ – комплекс системи захисту інформації

НЗД – найбільший загальний дільник

НСД – несанкціонований доступ

ПЗ – програмне забезпечення

ПЗП – Постійно Запам'ятовувальний Пристрій

EEPROM – Electronically EPROM

EPROM – Erasable Programmable ROM або як Electrically Programmable ROM

Flash – Flash Erase EEPROM

NVRWM – nonvolatile read-write memory

PROM – Programmable ROM

RAM – Random Access Memory

ROM – Read Only Memory

ВСТУП

Актуальність теми. Стандартизація необхідна для ефективного функціонування будь-якої галузі, і ринок хмарних послуг не є виключенням. Його повноцінний розвиток неможливо без стандартів захисту інформації, переносимості даних і додатків, оцінки рівня надаваного сервісу й т.д. Причому їхній вплив може бути настільки ж великий, як і законодавчих вимог і норм. Проте забезпечення належної відповідності їм, а також сертифікація залишаються комерційною справою кожного провайдеру, якщо немає відповідної вимоги регулятора. Ключову роль у прийнятті необхідних стандартів і розвитку ринку може грати уряд – потенційно найбільш великий споживач хмарних сервісів. Підтримка хмарних обчислень припускає прийняття безлічі різних стандартів, що – з урахуванням наявності десятків органів по стандартизації – чревате їхньою фрагментарністю, непогодженістю й взаємним дублюванням. У свій час Європейська комісія навіть виразила заклопотаність із цього приводу, назвавши «мішанину стандартів» головною перешкодою на шляху переходу до хмар, що гальмує розвиток ринку. Найбільше побоювання викликало те, що галузь не зможе прийти до згоди відносно інтероперабельності сервісів і переносимості даних. Кожний великий гравець прагне до домінування на ринку, а тому, на думку комісії, не зацікавлений у стандартизації. У результаті замовники можуть виявитися прив'язаними до конкретного провайдеру, не маючи можливості його перемінити. Крім того, комісія виразила тривогу із приводу відсутності стандартів для забезпечення безпеки даних, відповідно до яких можна було б сертифікувати провайдерів хмарних послуг, їхню інфраструктуру й сервіси, що дозволило б гарантувати схоронність користувальницьких даних і допомогло галузі розвиватися.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи захисту персональних даних у мережевих Cloud-системах.

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

– Огляд існуючих систем захисту персональних даних у мережевих Cloud-системах.

– Дослідження системи захисту персональних даних у мережевих Cloud-системах.

– Програмна реалізація системи захисту персональних даних у мережевих Cloud-системах.

Об'єктом дослідження є процес захисту персональних даних у мережевих Cloud-системах.

Предметом дослідження є методи захисту персональних даних у мережевих Cloud-системах.

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

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

– Удосконалено метод захисту персональних даних у мережевих Cloud-системах.

– Розроблено вітчизняний продукт захисту персональних даних у мережевих Cloud-системах, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі захисту персональних даних у мережевих Cloud-системах.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Робота апробована на LVII Науково-технічній конференції здобувачів вищої освіти LV науково-технічної конференції «Наука в ЦНТУ: основні досягнення та перспективи розвитку» (2025 р.), основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №15.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи захисту персональних даних у мережевих Cloud-системах, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

У звіті Групи по координації хмарних стандартів (Cloud Standard Coordination, CSC) Європейського інституту телекомунікаційних стандартів (European Telecommunications Standards Institute, ETSI) ці побоювання Європейської комісії були визнані надмірними: ситуація в області хмарних стандартів характеризується як складна, але «динамічна». На думку ETSI, проблема полягає не в плутанині, а в необхідності інтенсифікації роботи над технічними стандартами, де дотепер залишаються пробіли. Крім того, відсутність стандартів інтероперабельності визнається менш істотним у порівнянні з потребою в стандартах безпеки й конфіденційності даних.

Такий достаток норм, стандартів і специфікацій зовсім не свідчить про проблеми на ринку хмарних обчислень. Скоріше, воно відбиває розмаїтість і комплексний характер використовуваних технологій. До того ж ситуація з технічними стандартами набагато краще, ніж з іншими – зокрема, із сертифікаційними, для прийняття яких потрібно більше часу, оскільки вони повинні бути убудовані в існуюче правове поле.

Ключову роль у прийнятті необхідних стандартів здатен зіграти уряд – потенційно найбільш великий споживач хмарних сервісів. Через своїх представників воно може або прямо брати участь у розробці стандарту, або встановлювати регулюючі або законодавчі норми, куди включаються вже наявні стандарти. У деяких державах уводяться обов'язкові стандарти, необхідні для функціонування хмарного ринку. Так, у Сінгапурі діє специфікація на багаторівневий захист хмарних сервісів (Specification for multi-tiered cloud computing security, MTCC SS). У більшості ж країн уряди не займаються розробкою стандартів, підтримуючи розвиток хмар за допомогою відповідних стратегічних ініціатив.

1.2 Область застосування

Вимоги до інформаційної безпеки відбиті в безлічі стандартів – міжнародних і національних, загальних і галузевих. До їхнього числа відносяться ISO 22301, ITIL, TIER, SSAE, ISAE, NIST, 152-ФЗ, 382-П, SOX-4, HIPAAS і ще десятки іншими, прийнятими різними законодавчими й нормативними органами. На щастя, як відзначає експерт KMPG, локальні стандарти й акти звичайно базуються на міжнародних нормативах, так що застосування передових практик дозволяє задовольнити якщо не всі, те більшість вимог. До того ж багато хто з перерахованих стандартів призначені для вузького сегмента інформаційної безпеки.

На Заході «практично обов'язковою» для провайдерів є сертифікація по ISO 27001:2013 на системи менеджменту інформаційної безпеки. Тільки в країнах ЄС щорічно видається більше 8000 сертифікатів на ISO 27001. В Україні, незважаючи на наявність ДСТ, цей стандарт мало розповсюджений – щороку його одержують усього кілька десятків компаній. Таке положення справ можна пояснити «зфокусованістю стандарту на загальних підходах до керування ІБ, а не на ефективності окремих елементів керування, недовірою до інших сертифікатів ISO і дорожнечею одержання сертифіката». Проте очікується, що затребуваність сертифіката буде рости.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи захисту персональних даних у мережевих Cloud-системах, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Якщо глянути на хмарне середовище з технологічної точки зору, то умови роботи додатків не сильно відрізняються від традиційних. Бізнес-системи також запускаються на окремих обчислювальних потужностях, тільки в хмарі вони стають віртуальними. Дані як і раніше зберігаються на серверах, але тепер вони можуть бути розподілені на кілька обчислювальних вузлів або навпаки, упаковані більшою кількістю на один потужний сервер. Саме тому багато експертів уважають, що захист інформації в хмарі повинна бути побудована по тим же принципам, що й захист традиційних систем, а різниця складається лише в організаційних моментах, пов'язаних з механізмами надання сервісів.

Таким чином, захист даних – завдання, рішення якого лягає на плечі не тільки оператора, але й самого клієнта. При цьому в кожному окремому випадку можуть використовуватися свої методи захисту даних, які будуть відрізнятися залежно від виду використовуваних хмарних сервісів.

Приватна хмара

Найпростіше гарантувати безпека в приватному хмарному середовищі. Адже при роботі із власною хмарою ми одержуємо лише переваги від сервісної моделі, а також віртуалізації обчислювальних ресурсів і сховищ даних. При цьому вся коштовна інформація залишається усередині компанії. Вона може залишати мережа тільки в строго певних випадках і, як правило, зовсім не зберігається на кінцевих пристроях, наприклад, при роботі з віртуальним робочим столом. У приватній хмарі можна реалізувати не тільки повний функціонал платформи й додатків, але й забезпечити максимальний набір засобів

захисту. У приватній хмарі можна використовувати й кодування даних, і захист даних від адміністратора, і кластерне рішення, і резервування для забезпечення катастрофостійкості, і аудит операцій, і маскування даних, і, звичайно, весь арсенал автентифікації й розмежування прав користувачів.

У дійсності сучасні програмні рішення дозволяють робити багато чого. Навіть самі системи керування базами даних відкривають можливості для віртуалізації й підвищення операційної гнучкості. Зокрема, такі функції як Run-Time Privilege Analysis і Data Redaction, дозволяють організаціям визначати реально використовувані привілеї й ролі доступу до даних, у тому числі, що зберігається в хмарному середовищі. Установлюючи мінімум привілеїв і крім непотрібних прав доступу, компанії можуть зберегти повний спектр і цілісність своїх бізнес-процесів, одночасно не допускаючи до даних зайвих осіб. Як відзначають фахівці Oracle, при цьому немає необхідності або модифікувати код додатків.

Однак не варто забувати, що приватна хмарна структура вимагає наявності кваліфікованих кадрів, які зможуть обслуговувати рівень серверів, забезпечувати безвідмовну й ефективну роботу віртуалізаційного ПЗ, а також відповідати за роботу самих бізнес-додатків у хмарному середовищі й підтримувати потрібний рівень сервісу. Як наслідок, вони ж повинні бути великими й досвідченими фахівцями в сфері хмарної безпеки. На жаль, далеко не у всіх організаціях таке можливо, і тому сьогодні усе більш популярними стають публічні хмари.

Публічна хмара

Головна особливість публічного хмарного середовища полягає в тому, що за ваші дані відповідає інша організація, що забезпечує їхнє зберігання й передачу на вимогу. У зв'язку з тим, що коштовна інформація регулярно залишає фактичний периметр корпоративної мережі, для неї потрібна додатковий захист. На жаль, жодне публічна або гібридна хмара в принципі не може забезпечити той же рівень безпеки, що й частка, у якому захист може досягати такого ж рівня, що й у традиційних, розгорнутих на підприємстві, системах. Тому для підвищення

рівня безпеки в публічній хмарі сервіс провайдери часто змушені прибігати до обмеження функціонала надаваних сервісів.

Проте, для багатьох організацій можливість забезпечення хмарної безпеки з боку провайдеру є важливою перевагою. Останнім часом значно підсилилися побоювання щодо того, що дані, розміщені в хмарних сховищах, можуть бути уразливі й контролюватися з боку користувачів інших держав. Але ті організації, які роблять крок назад, як правило, доходять висновку, що більшість провайдерів хмарних технологій є експертами в забезпеченні безпеки, і деякі компанії мають навички, порівнянними з компетенцією провайдерів хмарних технологій.

Технологія захисту

Втім, при продуманій стратегії й наявності достатнього потенціалу в сфері IT сучасні технології дозволяють забезпечити в хмарному середовищі практично будь-який рівень безпеки, аж до найвищих вимог до захисту персональних даних. Справа в тому, що в хмарних обчисленнях завжди можна чітко вказати зону відповідальності учасників і визначити вимоги на кожному структурному рівні надання хмарних послуг. Засоби реалізації таких вимог є вже сьогодні. При цьому акцент повинен бути на безпечному розміщенні й використанні прикладного ПЗ. Саме рівень прикладного ПЗ надає доступ до даних, коштовним для потенційного порушника й саме прикладне ПЗ перебуває на передній лінії доступу й піддано максимальної небезпеки.

Найпоширеніші ризики, що мають відношення до хмарних середовищ, представлені такими інцидентами як крадіжки віртуальних машин з використанням знімних носіїв, зміни мережної топології IT-інфраструктури з використанням тільки програмних налаштувань, атаки на IT-сервіси в обхід мережних механізмів захисту. Дані ризики знижуються завдяки захисту на всіх рівнях побудови віртуального середовища: апаратного забезпечення, системного ПЗ віртуалізації (гіпервізора), усередині самої віртуальної інфраструктури, у системі керування й системі зберігання.

Сучасні рішення дозволяють створювати міжмережні екрани для віртуальних машин, також робити постійний моніторинг і оперативно реагувати

на інциденти на віртуальних машинах. Як відзначають експерти, такі рішення називаються System Information Event Management SIEM. Вони представлені продутими IBM QRadar, Security Vision, ArcSight і іншими. Подібні системи збирають повідомлення із всіх засобів захисту організації, у тому числі й з віртуальної інфраструктури, корелюючи й ранжируя інциденти для швидкого й зваженого прийняття рішень залежно від ситуації.

Відповідність вимогам

Розвиток компетенцій українських ЦОД і ріст хмарного ринку більш ніж на 30% у рік при основному побоюванні клієнтів у сфері безпеки доводить, що технічно захист організувати нескладно. «Потрібно використовувати резервне копіювання, шифрування даних на етапах зберігання й передачі; регулярно сканувати своє програмне забезпечення на наявність уразливостей, використовувати міжмережні екрани й інші засоби. Для більшості організацій коштує питання дотримання вимоги українських регуляторів і одержання відповідних ліцензій. Багато в чому труднощі даного процесу перешкоджає переходу в хмари, однак усе більше центрів обробки даних починають надавати гарантії захисту даних аж до К2 і використовують це як конкурентна перевага. Таким чином, незважаючи на існуючі сьогодні складності, ринок неминуче рухається до переносу в хмари не тільки інформації різних організацій, але й відповідальності за її захист.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Як мова програмування обрана Python. Python – високорівнева мова програмування загального призначення з акцентом на продуктивність розроблювача й читаність коду. Синтаксис ядра Python мінімалістичний. У той же час стандартна бібліотека включає великий обсяг корисних функцій.

Python підтримує кілька парадигм програмування, у тому числі структурне, об'єктно-орієнтоване, функціональне, імперативне й аспектно-

орієнтоване. Основні архітектурні риси – динамічна типізація, автоматичне керування пам'яттю, повна інтроспекція, механізм обробки виключень, підтримка багатопоточні обчислень і зручні високорівневі структури даних. Код у Python організовується у функції й класи, які можуть поєднуватися в модулі (які у свою чергу можуть бути об'єднані в пакети).

Еталонною реалізацією Python є інтерпретатор CPython, що підтримує більшість активно використовуваних платформ. Він поширюється вільно під дуже ліберальною ліцензією, що дозволяє використовувати його без обмежень у будь-яких застосунках, включаючи пропрієтарні. Є реалізації інтерпретаторів для JVM (з можливістю компіляції), MSIL (з можливістю компіляції), LLVM і інших. Проект PyPy пропонує реалізацію Python на самому Python, що зменшує витрати на зміни мови й постановку експериментів над новими можливостями.

Python – мова програмування, що активно розвивається, нові версії (з додаванням/зміною язикових властивостей) виходять приблизно раз у два з половиною року. Внаслідок цього й деяких інших причин на Python відсутні ANSI, ISO або інші офіційні стандарти, їхня роль виконує CPython.

Python портований і працює майже на всіх відомих платформах – від КПК до мейнфреймів. Існують порти під Microsoft Windows, практично всі варіанти UNIX (включаючи FreeBSD і Linux), Plan 9, Mac OS і Mac OS X, iPhone OS 2.0 і вище, Palm OS, OS/2, Amiga, AS/400 і навіть OS/390, Symbian і Android.

При цьому, на відміну від багатьох портуємих систем, для всіх основних платформ Python має підтримку характерних для даної платформи технологій (наприклад, Microsoft COM/DCOM). Більше того, існує спеціальна версія Python для віртуальної машини Java – Jython, що дозволяє інтерпретаторові виконуватися на будь-якій системі, що підтримує Java, при цьому класи Java можуть безпосередньо використовуватися з Python й навіть бути написаними на Python. Також кілька проектів забезпечують інтеграцію із платформою Microsoft .NET, основні з яких – IronPython і Python.Net.

Python підтримує динамічну типізацію, тобто тип змінної визначається тільки під час виконання. Тому замість «присвоювання значення змінної» краще говорити про «зв'язування значення з деяким ім'ям». У Python є убудовані типи: бульові, рядки, Unicode-рядки, цілі числа довільної точності, числа із плаваючої коми, комплексні числа й деякі інші. З колекцій Python підтримує кортежі (*tuples*), списки, словники (асоціативні масиви) і, починаючи з версії 2.4, безлічі. Всі значення в Python є об'єктами, у тому числі функції, методи, модулі, класи.

Додати новий тип можна або написавши клас (class), або визначивши новий тип у модулі розширення (наприклад, написаному мовою C). Система класів підтримує спадкування (одиночне й множинне) і метапрограмування. Можливе спадкування від більшості убудованих типів і типів розширень.

Всі об'єкти діляться на посилальні й атомарні. До атомарного ставляться int, long, complex і деякі інші. При присвоюванні атомарних об'єктів копіюється їхнє значення, у той час як для посилальних копіюється тільки покажчик на об'єкт, таким чином, обидві змінні після присвоювання використовують те саме значення. Посилальні об'єкти бувають змінювані й незмінні. Наприклад, рядки й кортежі є незмінними, а списки, словники й багато інших об'єктів – змінюваними. Кортеж у Python є, по суті, незмінним списком. У багатьох випадках кортежі працюють швидше списків, тому якщо ви не плануєте змінювати послідовність, то краще використовувати саме їх.

Мова має чіткий і послідовний синтаксис, продуману модульність й масштабованість, завдяки чому вихідний код написаних на Python програм легко читаємий.

Python – стабільна й розповсюджена мова. Він використовується в багатьох проектах і в різних якостях: як основна мова програмування або для створення розширень і інтеграції застосунків. На Python реалізоване велика кількість проектів, також він активно використовується для створення прототипів майбутніх програм. Python використовується в багатьох великих компаніях.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи захисту персональних даних у мережевих Cloud-системах.

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

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

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

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

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

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Релевантними для провайдерів стандартами ISO із групи 27000 є ISO 27002 на методи й засоби інформаційної безпеки й ISO 27005 на керування ризиками в області інформаційної безпеки. У першому більш докладно розглядаються засоби керування, що перераховуються в додатку до ISO 27001. Обидва стандарти (27002 і 27005) є рекомендаційними, сертифікація на відповідність їм не передбачається. Проте провайдер може замовити незалежну оцінку, щоб перевірити, якою мірою дотримуються запропоновані рекомендації. Деякі замовники запитують таку оцінку замість сертифікації по 27001. Всі три стандарти не є специфічними для провайдерів. Тим часом улітку 2014 року ISO опублікувала стандарт ISO 27018:2015 про захист персональних даних у хмарі, а наприкінці 2015 року – ISO 27017:2015 про засоби контролю інформаційної безпеки для хмарних рішень.

В ISO 27017 передбачаються додаткові елементи безпеки для хмари, відсутні в ISO 27002. Повна офіційна назва цього стандарту: «Звід правил для засобів керування інформаційною безпекою на базі ISO/IEC 27002 для хмарних сервісів» («Code of practice for information security controls based on ISO/IEC 27002 for cloud services»). Незважаючи на те що його фінальна редакція була опублікована лише 30 листопада 2015 року, Amazon Web Services уже має відповідний сертифікат, отриманий ще в жовтні 2015-го. Тим часом, будучи доповненням до ISO 27002, новий стандарт не припускає сертифікації, однак через його популярність багато органів сертифікації планують це робити: імовірно, будуть видавати «свідчення про відповідність». У всякому разі, на дане питання ще немає ясної відповіді.

Конкретні рекомендації стосовно до хмарних сервісів даються до 37 з більш ніж сотні засобів керування безпекою, які визначені в ISO 27002. Вони адресовані не тільки провайдерам хмарних послуг, але й замовникам, чим підкреслюється їхня взаємна відповідальність за безпеку сервісів: замовник повинен розробити політикові використання хмарних сервісів, а провайдер – надати йому необхідну інформацію. Найбільші зміни стосуються розділу, присвяченого контролю доступу: розглядаються реєстрація й вихід користувача, надання доступу, керування привілейованим доступом, обмеження доступу до інформації й використання привілейованих службових програм. Крім того, уводяться сім нових елементів керування:

– загальні ролі й відповідальність у хмарному середовищі;
– видалення й повернення активів клієнта хмарних сервісів;
– сегрегація у віртуальних обчислювальних середовищах;
– посилення віртуальних машин;
– операційна безпека адміністратора;
– моніторинг хмарних сервісів;
– узгодження керування безпекою для віртуальних і фізичних середовищ.

Звичайно, це далеко не всі стандарти, навіть якщо обмежитися тільки інформаційною безпекою. Чим їх більше, тим сутужніше в них розібратися. Як би те не було, ISO 27001 представляється найкращим базовим стандартом для всіх компаній, що бажають захистити свою інформацію (і єдиним із серії стандартів інформаційної безпеки ISO, на який видається реальний сертифікат). Залишаючи осторонь питання про захист персональних даних (ISO 27018 і/або 152-ФЗ), можна сказати, що реалізований постачальником хмарних послуг комплекс рекомендацій ISO 20001 і 20017 є необхідним «джентльменським набором» в області інформаційної безпеки, якщо провайдер хоче розсіяти сумніву замовника в безпеці хмарних сервісів.

Однак, як випливає з ISO 20017, навіть проходження провайдером його рекомендаціям не знімає відповідальності із клієнта, що повинен виконати свою

частину домашнього завдання й також ретельно додержуватися рекомендацій стандарту. Наприклад, відповідно до вимог контролю доступу А.9.4.1, він повинен обмежити доступ до інформації в хмарі у відповідності зі своєю корпоративною політикою.

Стандарти можуть робити настільки ж значний вплив на ринкову ситуацію, як і введення законодавчих вимог і норм. Проте при відсутності офіційних вимог забезпечення відповідності тим або іншим стандартам, а також сертифікація – це комерційна справа кожного провайдеру. У більшості країн діяльність хмарних провайдерів державою ніяк спеціальним образом не регулюється. Донедавна в жодній країні миру не було окремих законів, присвячених хмарам. Торік першопрохідником на цьому шляху стала Південна Корея.

До кінця 2017 року були прийняті необхідні правові акти по врегулюванню використання хмарних технологій при здійсненні державного керування. Однак на даний момент спеціальні нормативно-правові акти, де б установлювалися правила надання хмарних послуг, відсутні. Проте в Україні дуже багато законів, у яких і до замовників, і до постачальників послуг пред'являються досить специфічні й детальні вимоги. Головні з них – Цивільний кодекс (ГК), а також закони про інформацію й персональні дані.

Основні обмеження в частині обробки інформації стосуються персональних даних (ПДн). З юридичної точки зору у відносинах, що формуються у зв'язку з використанням хмарних послуг, беруть участь три сторони: суб'єкти даних, замовники й провайдери, – причому замовник є також оператором даних для тих суб'єктів даних, інформацію про які він збирає й потім передає в хмару. При буквальному читанні закону про персональні дані провайдер може класифікуватися як оператор ПДн залежно від обсягу надаваної хмарної послуги. У такому випадку виникає необхідність відповідати величезній кількості критеріїв, у числі яких – реєстрація в спеціальному реєстрі регулятора й

відповідальність перед суб'єктом персональних даних, що перебувають у хмарі, що збільшує навантаження на провайдеру і його витрати.

Щоб уникнути цієї ситуації радимо провайдерам скористатися особливим режимом обробки даних – обробкою з доручення. Він рятує від необхідності діставати згоду на обробку ПДн у суб'єктів і дозволяє уникнути відповідальності безпосередньо перед ними: відносини провайдеру із приводу дотримання законодавства про ПДн будуть обмежені його контрактом із замовником – розбиратися з тими або іншими претензіями конкретних суб'єктів даних буде сам замовник, він же оператор даних. Для цього в договорі із провайдером повинні бути прописані наступні умови:

– перелік дій по обробці Пдн, які будуть відбуватися провайдером, а також мети обробки;

– зобов'язання провайдеру дотримувати конфіденційності Пдн і забезпечити їхня безпеку в ході обробки;

– перелік вимог до провайдеру по захисту оброблюваних Пдн.

При відсутності в договорі цих умов провайдер ризикує порушити безліч статей закону про Пдн, а при їхній наявності він хоча й звільняється від безпосередньої відповідальності перед суб'єктом даних, але відповідає за недотримання передбачених договором заходів щодо захисту Пдн.

Багато питань викликало вимогу закону про локалізацію, що вказує на те, що персональні дані українських громадян повинні оброблятися з використанням баз даних, що перебувають в Україні, і – згідно «твердому трактуванню» – винятково на вітчизняних серверах. Однак у підсумку вибрали більше м'який варіант: обробка даних українських громадян можлива й на закордонних серверах, якщо ці дані втримуються також у базах даних, розташованих на території Україні, при цьому не допускається наявності за межами країни Пдн, які відсутні в українській базі даних. Як відзначає Микола Феоктистов, дане роз'яснення не відповідає на цілий ряд питань, наприклад: якщо нові дані створюються за рубежем і вони доступні користувачеві в Україні, те чи належні

вони відразу реплікуватися в українську базу даних і яка відповідальність провайдеру в той період часу, поки ця реплікація відбувається?

Послуги хмарних обчислень – послуги з надання обчислювальних потужностей, включаючи технічні засоби й права використання програм для електронних обчислювальних машин з метою обробки й зберігання інформації органів державної влади, органів місцевого самоврядування, органів керування державними позабюджетними фондами з використанням технічних засобів, взаємодіючих через інформаційно-телекомунікаційні мережі.

Фактично в цьому визначенні хмарні сервіси розглядалися як послуги винятково для державних структур, тому після обговорення експертів згадування про органи державної влади й іже з ними було виключено (до слова, було подано всього одна пропозиція по внесенню зміни, текст якого, на жаль, недоступний; наскільки можна судити по приводяться адресам, що, електронної пошти, текст виправлень розсилався на експертизу в основному представникам бізнес-співтовариства): «Послуги хмарних обчислень – послуги з надання обчислювальних потужностей, включаючи технічні засоби й права використання програм для електронних обчислювальних машин з метою обробки й зберігання інформації з використанням технічних засобів, взаємодіючих через інформаційно-телекомунікаційні мережі».

Однак поточне формулювання як і раніше носить занадто загальний характер (наприклад, під дане визначення підходять послуги аутсорсингу обчислювальної інфраструктури, коли оператор не тільки обслуговує відомчу мережу, але й надає в оренду встаткування. Крім того, вона не відбиває специфіки хмарних сервісів: оперативне одержання послуг при мінімальній участі провайдеру, оплату за фактичне використання й інші особливості. Далі, не тільки не проводиться явного розходження між різними моделями надання хмарних сервісів («інфраструктура як сервіс» і «ПЗ як сервіс»), але вони фактично змішуються – «надання обчислювальних потужностей (IaaS), включаючи…права використання програм (SaaS)». (І нарешті, винятково

редакторське зауваження – після «обчислювальних машин» необхідна кома, інакше виходить нелегкотравна фраза про «технічні засоби … з використанням технічних засобів».)

У виправленнях даються визначення не тільки хмарних сервісів, але також хмарної інфраструктури й постачальника послуг хмарних обчислень, які по суті носять тавтологічний характер: хмарна інфраструктура – це інфраструктура для надання хмарних сервісів. Втім, якщо підходити формально, з визначення не цілком ясно, чи входять у неї мережі доступу або тільки мережі усередині ЦОДа, оскільки передача інформації (надання її клієнтові) не згадується в числі операцій з даними: «хмарна інфраструктура – сукупність програмно-технічних засобів і інформаційно-телекомунікаційних мереж, що забезпечують обробку й зберігання інформації з метою надання послуг хмарних обчислень».

Як провайдери можуть виступати як юридичні особи, так і індивідуальні підприємці. До них пред'являються три основних вимоги: постачальник повинен забезпечити доступність інформації й програмного забезпечення, можливість обробки даних (включаючи повне видалення) і, нарешті, захист інформації (відповідно до вимог ст. 16 з 5-й частини закону про інформацію). Окремо обмовляється, що постачальник не має ніяких прав на інформацію (не є її власником). Здійснювати поставку хмарних послуг державним і муніципальним органам можуть тільки українські компанії (і навіть індивідуальні підприємці), а їхня хмарна інфраструктура повинна перебувати на території України. Провайдерам необхідно буде одержати акредитацію. Крім того, міністерство зобов'язане розробити вимоги до контрактів і договорів на надання хмарних послуг. Порядок надання послуг визначить уряд України.

Перший закон про хмари був прийнятий у Південній Кореї. Його повна назва – «Закон про розвиток хмарних обчислень і захисту користувачів» («Act on the Development of Cloud Computing and Protection of Users», скорочено – Korean Cloud Act). Держава зважилася на дерегулювання галузі, оскільки Південна Корея була однієї з деяких країн, де суспільним інститутам заборонялася орендувати

хмарні сервери в приватних провайдерів. Головною причиною заборони було побоювання щодо можливих погроз інформаційної безпеки, який, як виявляється з назви, у законі приділяється особлива увага.

На розробку й прийняття цього документа пішло біля півтора років (постанова уряду бути схвалено у вересні 2013 року, а закон прийнятий у березні 2015-го й набув чинності у вересні того ж року). Причому спочатку комітет з науці, ИКТ, майбутньому плануванню й комунікаціям не квапився з його розглядом, уважаючи більше важливим рішення інших завдань, зокрема прийняття закону про віщання.

Закон поширюється на всі суспільні інститути – центральний уряд, громадські організації, установи охорони здоров'я й утворення – і покликаний стимулювати першочергове використання хмарних сервісів для підвищення продуктивності й конкурентоспроможності.

Місцевий ринок хмарних обчислень відносно невеликий і нерозвинений у порівнянні з локальним IT-ринком, щодо цього він схожий з українським. По оцінках інституту економічних досліджень Digieco, в 2014 році його обсяг склав 863 млн доларів, що порівнянно з розміром українського хмарного ринку (до знецінення гривни). Як очікується, його щорічний ріст повинен скласти 30% за рахунок наступних мір:

– нарощування інвестицій у розвиток хмарного ринку й розширення підтримки, насамперед з боку уряду;

– дозвіл і заохочення повсюдного використання хмарних сервісів, включаючи публічні сервіси;

– введення мер по забезпеченню безпеки з боку провайдерів хмарних послуг.

Корейське Міністерство науки, IT і перспективного планування надає й ряд стимулюючих преференцій, адресованих насамперед невеликим розроблювачам хмарного програмного забезпечення: податкові відрахування й технічна допомога. Крім того, такі компанії одержать пріоритет при проведенні

державних тендерів на реалізацію науково-дослідних проектів у сфері хмарних технологій.

Вся відповідальність за захист даних переноситься із клієнта на провайдеру хмарних сервісів. Як затверджується, це зроблено для того, щоб компаніям без досвіду роботи в IT не доводилося додатково витрачатися на безпеку, і тоді хмарні сервіси стануть для них більше привабливими. Власники даних повинні підписати угоду із провайдером і вказати, яка збережена в хмарі інформація може бути розкрита, а яка немає.

Спочатку передбачалося, що контроль за відповідністю вимогам безпеки стане здійснювати Національне агентство розвідки (НАР). Зокрема, провайдери повинні були б повідомляти НАР про всі інциденти безпеки із хмарними сервісами. Однак побоювання у відношенні того, що НАР одержить необмежений контроль за хмарними сервісами й персональними даними, привело до суспільних протестів, і ця стаття була вилучена з підсумкового тексту. Відповідно до закону про інциденти провайдер повинен повідомляти про підозрілі випадки в галузеве міністерство, і саме воно буде ініціювати перевірку.

Вимоги до хмарних провайдерів

При виборі провайдеру клієнти повинні з'ясувати його відповідність вимогам закону про хмари. Відповідно до нового південнокорейського законодавства, на провайдерів хмарних послуг покладає ряд зобов'язань:

– провайдер зобов'язаний повідомити про факт витоку інформації, якщо такий мав місце, своїм клієнтам і в профільне міністерство; останнє може провести розслідування інциденту;

– провайдер не повинен надавати приналежним клієнтам інформацію третій стороні або використовувати неї для інших, відмінних від застережених, цілей без згоди клієнта;

– провайдер повинен повернути або видалити дані клієнта після закінчення строку контракту про надання хмарних послуг;

– при розміщенні інформації за межами Південної Кореї клієнт може зажадати від провайдеру розкрити її місцезнаходження;

– якщо клієнт поніс втрати внаслідок навмисних дій провайдеру або через його недбалість і порушення закону про хмари, то клієнт може зажадати відшкодування від провайдеру, що повинен сам доводити свою невинність;

– підзаконними актами міністерства будуть деталізовані вимоги до якості/функціональності хмарних сервісів і належних рівнів обслуговування, а також стандарти для захисту інформації;

– крім цього, планується введення системи сертифікації хмарних сервісів і стандартизованих контрактів на використання хмарних сервісів.

У законі про хмари поки немає якої-небудь прив'язки до міжнародних стандартів. У ряді країн для оцінки провайдеру хмарних послуг регулювальні органи орієнтуються, наприклад, на рекомендації ISO/IEC 27001 і ISO/IEC 27018. Передбачені останнім інструменти контролю відповідають багатьом положенням південнокорейського закону (і навіть пред'являють більше тверді вимоги), тому в тих провайдерів, які вже забезпечили відповідність ISO/IEC 27018, не повинне виникати яких-небудь проблем з його виконанням.

Як сподіваються в Південній Кореї, прийняття цього закону, з одного боку, активізує розвиток суміжних галузей, а з іншої, буде мотивувати недержавні компанії до використання хмарних сервісів. Розвиток необхідної інфраструктури для надання хмарних сервісів повинне сприяти росту таких зв'язаних галузей, як телемедицина, фінансові послуги й Інтернет речей. А як показує досвід Китаю, що у цьому випадку є прикладом, впровадження хмарних послуг у державному секторі сприяє їхній популяризації в приватному секторі.

3.2 Розробка структурної схеми

Розуміння основ комп’ютерної безпеки ще ніколи не було таким важливим. З огляду на те, що наше життя дедалі більше перебуває в Інтернеті, захист вашої особистої інформації є ключем до збереження вашої конфіденційності та запобігання крадіжці особистих даних.

Комп'ютерна безпека охоплює широкий спектр практик, протоколів і технологій, спрямованих на захист IT-систем від несанкціонованого доступу, витоків даних та кіберзагроз. Вона передбачає впровадження різних заходів безпеки, включаючи брандмауери, антивірусний захист, персональний брандмауер та інструменти шифрування, всі з яких розроблені для захисту конфіденційної інформації від зловмисників.

Перш ніж ми поговоримо про послуги комп’ютерної безпеки, давайте розглянемо визначення комп’ютерної безпеки. Комп’ютерна безпека включає захисні заходи та практики, які ви впроваджуєте для запобігання несанкціонованому доступу до IT-систем. Водночас вона забезпечує цілісність, конфіденційність та доступність ваших даних та цифрової інформації.

Якщо ви хочете досягти надійної комп'ютерної безпеки, потрібен багатогранний підхід, який включає різні компоненти. Зазвичай це включає шифрування даних для захисту конфіденційної інформації, гарантуючи, що навіть у разі перехоплення дані залишатимуться нечитабельними без відповідних ключів розшифрування. Стандарти шифрування також відіграють значну роль у забезпеченні безпеки ваших даних.

Контроль доступу є життєво важливим для обмеження дозволів користувачів та доступу до системи лише уповноваженими особами, тим самим мінімізуючи потенційні вразливості. Методи автентифікації, такі як захист паролем, біометричне сканування та багатофакторна автентифікація, забезпечують додаткові рівні безпеки, перевіряючи особу користувачів.

Разом ці елементи створюють комплексну систему, необхідну для захисту ваших цифрових активів від потенційних загроз та порушень.

Важливість комп'ютерної безпеки для захисту даних

Ефективні заходи захищають конфіденційну інформацію від таких загроз, як крадіжка особистих даних та витоки даних, зокрема тих, що виникають внаслідок підозрілих електронних листів.

У сучасному цифровому середовищі, де величезні обсяги особистої та фінансової інформації поширюються онлайн, наслідки недостатньої безпеки можуть бути руйнівними. Це може призвести не лише до фінансових втрат, але й до значного порушення довіри між користувачами. Ви повинні розуміти ризики, пов’язані із незахищеними системами, оскільки один недогляд може призвести до несанкціонованого доступу до особистих даних. Захистіть свої IT-системи від несанкціонованого доступу, застосовуючи надійні заходи безпеки.

Складний взаємозв'язок між комп'ютерною безпекою та конфіденційністю в Інтернеті стає дедалі важливішим, що підкреслює необхідність надійних захисних заходів.

Надаючи пріоритет безпеці, ви можете краще гарантувати конфіденційність своїх даних, тим самим зберігаючи свою конфіденційність в епоху кіберзагроз. Розгляньте можливість використання VPN для безпечного з’єднання Wi-Fi та уникнення публічних Wi-Fi, щоб покращити свою конфіденційність в Інтернеті.

Види загроз комп'ютерній безпеці та онлайн-загроз

Вам слід знати про різні типи загроз комп’ютерній безпеці, які становлять значні ризики для захисту даних. До них належать:

– кібератаки;

– фішингові атаки;

– шкідливе програмне забезпечення, що походить з підозрілих електронних листів.

Всі вони прагнуть використати вразливості в IT-системах, включаючи вразливості програмного забезпечення та програмне забезпечення сторонніх розробників.

Поширені загрози безпеці даних

Хочете знати, які поширені загрози безпеці даних? Ось огляд для вас: шкідливе програмне забезпечення, фішингові атаки та несанкціонований доступ – це вважаються поширеними загрозами безпеці даних.

Вони можуть призвести до серйозних наслідків, таких як крадіжка особистих даних та витік даних, які впливають як на організації, так і на окремих користувачів. Впровадження заходів безпеки програм та захисту електронної пошти може зменшити ці ризики.

Ці загрози діють за допомогою різних механізмів, спрямованих на вразливості в системах, мережах або поведінці людини. Наприклад, шкідливе програмне забезпечення може проникнути на пристрій користувача через заражені вкладення електронної пошти або завантаження шкідливого програмного забезпечення, зрештою компрометуючи конфіденційну інформацію. Фішингові атаки часто маскуються під законні повідомлення, обманом змушуючи людей розкривати особисті дані або облікові дані для входу. Несанкціонований доступ зазвичай передбачає використання слабких паролів або застарілих заходів безпеки, що дозволяє кіберзлочинцям порушувати бази даних та витягувати конфіденційні дані.

Реальні приклади, такі як витік даних Equifax та злом мережі Sony PlayStation Network, підкреслюють руйнівний вплив цих загроз безпеці – не лише на постраждалі організації, але й на мільйони людей, чиї дані можуть бути використані неналежним чином.

Розуміння кібератак

Кібератаки – це зловмисні спроби порушення безпеки IT-систем та мереж з метою отримання доступу, викрадення або маніпулювання конфіденційними

даними. Такі порушення можуть призвести до серйозних наслідків як для окремих осіб, так і для організацій, включаючи проблеми з безпекою обладнання.

Ці загрози можуть виникати з різних мотивів, включаючи фінансову вигоду, політичні цілі або особисту помсту. Кіберзлочинці використовують низку методів, таких як фішингові електронні листи, програми-вимагачі та розподілені атаки типу «відмова в обслуговуванні» (DDoS), для використання вразливостей у системах та IT-обладнанні.

Гучні інциденти, такі як витік даних Equifax у 2017 році та атака програм-вимагача WannaCry, служать яскравим нагадуванням про руйнівний вплив, який ці порушення можуть мати на конфіденційність, фінансову стабільність, цілісність організації та IT-системи.

Щоб зменшити ці ризики, як підприємства, так і приватні особи повинні впроваджувати надійні заходи безпеки. Регулярні оновлення програмного забезпечення, навчання співробітників та передові системи виявлення загроз можуть значно посилити захист від постійно мінливого ландшафту кіберзагроз. Розгляньте можливість звернутися за порадою до споживачів з авторитетних джерел, таких як PCMag, Wired та The Guardian, щоб дізнатися про найновіші методи безпеки.

Найкращі практики для комп'ютерної безпеки

Щоб захистити вашу конфіденційну інформацію та зменшити ризики, пов’язані з кіберзагрозами, впровадження найкращих практик комп’ютерної безпеки може допомогти вашій організації.

Надаючи пріоритет наступним практикам, ви можете покращити загальний рівень безпеки та захистити критично важливі дані від потенційних вразливостей.

1. Впровадження надійних паролів та автентифікації

Номер один: впровадження надійних паролів та багатофакторної автентифікації – це фундаментальний крок у захисті даних, який значно знижує ризик несанкціонованого доступу до конфіденційної інформації та IT-систем.

Уникайте використання особистої інформації у своїх паролях та будьте обережні з підозрілими електронними листами із запитом на введення облікових даних.

Вживаючи цих заходів безпеки, ви можете створити надійну лінію захисту від кіберзагроз. Надійний пароль повинен складатися з комбінації великих і малих літер, цифр і спеціальних символів, в ідеалі довжина пароля повинна перевищувати 12 символів. Ця складність ускладнює для зловмисників злом паролів методами грубої сили. Не поширюйте свої паролі через незахищені канали, такі як демонстрація екрана або публічні Wi-Fi.

Багатофакторна автентифікація забезпечує додатковий рівень безпеки, вимагаючи від вас підтвердження вашої особи за допомогою другого засобу, такого як код текстового повідомлення або додаток для автентифікації. Такий підхід суттєво мінімізує ймовірність несанкціонованого доступу, навіть якщо паролі скомпрометовані, особливо у віддаленому місці.

Найкращі практики також включають регулярне оновлення паролів та використання менеджера паролів для безпечного зберігання, що ще більше покращує загальну гігієну ваших паролів. Переконайтеся, що ваш менеджер паролів захищений надійними стандартами шифрування.

2. Регулярні оновлення програмного забезпечення та управління патчами

Звичайно, всі ми знаємо, що нам слід регулярно оновлювати наші пристрої, але це особливо важливо для безпеки даних. Регулярні оновлення програмного забезпечення та ефективне управління виправленнями допомагають вашій організації усунути вразливості безпеки, які кіберзлочинці часто використовують для отримання несанкціонованого доступу до IT-систем та конфіденційних даних. Це включає забезпечення актуальності програмного забезпечення сторонніх розробників для запобігання зловживанням.

Слідкування за цими оновленнями не лише зміцнює ваш захист від потенційних порушень, але й підвищує загальну продуктивність і надійність

системи. В епоху, коли витоки даних стають дедалі поширенішими, важливість управління виправленнями важко переоцінити.

Правильно впроваджені оновлення знижують ризик експлуатації, усуваючи лазівки, якими можуть скористатися кіберзлочинці. Найкращі практики включають встановлення послідовного графіка оновлень, визначення пріоритетів критичних виправлень на основі оцінки ризиків та використання інструментів автоматизації, коли це можливо, для оптимізації процесу. Організації також повинні впроваджувати налаштування конфіденційності для контролю доступу до конфіденційної інформації.

Крім того, організації отримують користь від навчання персоналу важливості обслуговування програмного забезпечення, сприяючи колективному зобов'язанню щодо захисту конфіденційної інформації. Такі ресурси, як програми навчання з безпеки від Microsoft, можуть бути безцінними в цьому відношенні.

3. Стратегії резервного копіювання даних

Впровадження ефективних стратегій резервного копіювання даних допомагає запобігти втраті даних через кібератак, збої обладнання або випадкове видалення. Це гарантує, що ваша цінна інформація залишатиметься доступною та безпечною протягом усього її життєвого циклу. Розгляньте можливість використання хмарного резервного копіювання та регулярного тестування резервних даних, щоб забезпечити їх цілісність.

Існує кілька методів резервного копіювання, які варто розглянути, зокрема хмарні резервні копії, які забезпечують можливості зберігання даних поза офісом та автоматичну синхронізацію. Крім того, ви можете вибрати знімні варіанти сховища, такі як зовнішні жорсткі диски або USB-флеш-накопичувачі, для локального зберігання даних. Також доцільно регулярно створювати резервні копії даних для захисту від потенційних випадків витоку даних.

Регулярне резервне копіювання не лише захищає вас від непередбачених катастроф, але й відіграє життєво важливу роль у підтримці цілісності даних та забезпеченні їхнього захисту з часом.

Щоб створити надійну стратегію резервного копіювання, важливо встановити графік резервного копіювання, який відповідає частоті оновлення ваших даних, використовувати шифрування для будь-якої конфіденційної інформації та регулярно тестувати процеси відновлення, щоб переконатися, що все працює безперебійно, коли це необхідно. Використовуйте інструменти шифрування для ефективного захисту ваших даних.

Застосовуючи ці методи, ви можете ефективно захистити свої дані від різноманітних загроз та досягти спокою. Розгляньте можливість інтеграції багатофакторної автентифікації для подальшого підвищення безпеки ваших IT-систем.

Захист даних у різних середовищах

Мобільні пристрої, віддалені місця та загальнодоступні мережі Wi-Fi – дані потребують захисту в різних середовищах. Тому вам потрібно впровадити індивідуальні заходи безпеки.

Такий підхід є важливим для збереження цілісності та конфіденційності конфіденційної інформації. Переконайтеся, що IT-обладнання захищене, а параметри конфіденційності в Інтернеті налаштовані належним чином.

Захист даних на мобільних пристроях

Мобільні пристрої часто стають мішенями для кіберзагроз, включаючи шкідливе програмне забезпечення та несанкціонований доступ, через їхню портативність та зручність підключення. Використання рішень мобільної безпеки може значно підвищити безпеку пристроїв.

Впровадження надійних методів, таких як шифрування, допомагає захистити конфіденційну інформацію, навіть якщо пристрій скомпрометовано. Контроль доступу діє як перша лінія захисту, обмежуючи тих, хто може переглядати дані або взаємодіяти з ними. Використання програм безпеки, які забезпечують виявлення загроз у режимі реального часу, може ще більше посилити цей захисний рівень, а впровадження антивірусного захисту може запобігти кібератакам.

Однак, самих лише технологій недостатньо; підвищення обізнаності користувачів щодо розпізнавання потенційних загроз, таких як фішингові атаки, підозрілі завантаження або підозрілі електронні листи, є надзвичайно важливим. Застосовуючи проактивний підхід та навчаючи користувачів, організації можуть значно посилити свою мобільну безпеку, гарантуючи безпеку даних у швидкозмінному цифровому середовищі.

Захист даних під час віддаленої роботи

Віддалена робота означає доступ до конфіденційної інформації через загальнодоступні мережі Wi-Fi або незахищені мережі, оскільки ці середовища створюють унікальні проблеми безпеки. Захист даних у такому середовищі має бути головним пріоритетом.

Для ефективного захисту конфіденційних даних важливо використовувати віртуальну приватну мережу (VPN). VPN шифрує ваше інтернет-з'єднання, значно знижуючи ризик перехоплення кіберзлочинцями. Розгляньте можливість використання особистої точки доступу як альтернативи публічному Wi-Fi для безпечнішого з'єднання Wi-Fi.

Крім того, переконайтеся, що ваші з'єднання захищені, використовуючи надійні паролі та двофакторну автентифікацію, коли це можливо. Також важливо оновлювати антивірусне програмне забезпечення та брандмауери, оскільки ці заходи додатково захищають ваші пристрої від потенційних загроз.

Регулярний перегляд та зміна налаштувань безпеки на ваших особистих пристроях допоможе створити безпечніше середовище для віддаленої роботи, захищаючи від несанкціонованого доступу та витоків даних.

Забезпечення безпеки хмарного сховища

Забезпечення безпеки хмарного сховища вимагає впровадження надійних заходів, таких як шифрування та контроль доступу, для захисту конфіденційних даних від несанкціонованого доступу та порушень. Розгляньте можливість використання стандартів шифрування та регулярного оновлення налаштувань конфіденційності для підтримки безпеки даних.

У сучасному цифровому середовищі, де витоки даних стають все більш поширеними, вкрай важливо застосовувати проактивний підхід до захисту хмарного сховища. Організації повинні усвідомлювати, що хоча хмарні рішення пропонують значні переваги, вони також несуть низку потенційних ризиків, включаючи втрату даних та викрадення облікових записів.

Використовуючи найкращі практики, такі як надійні методи шифрування, для захисту даних як під час зберігання, так і під час передачі, компанії можуть значно зменшити ймовірність компрометації своєї інформації. Регулярні аудити безпеки мають вирішальне значення для виявлення вразливостей та забезпечення актуальності протоколів безпеки.

Розуміння цих стратегій може значно покращити загальний рівень безпеки та вселити користувачам впевненість у безпеці їхніх цінних даних.

Технології безпеки даних

Технології безпеки даних допомагають вам у боротьбі з кіберзагрозами, надаючи низку методів захисту конфіденційної інформації та забезпечення цілісності ваших IT-систем.

1. Методи шифрування

Методи шифрування пропонують критично важливий рівень безпеки, який захищає цифрову інформацію від несанкціонованого доступу та витоків даних.

Ці методи шифрують конфіденційні дані як під час передачі, так і в стані спокою, гарантуючи, що лише уповноважені особи можуть розшифрувати інформацію та отримати до неї доступ. З огляду на зростаючу залежність від цифрових платформ для особистих та ділових транзакцій, важливість надійних методів шифрування неможливо переоцінити.

Такі інструменти, як AES (Advanced Encryption Standard) та RSA (Rivest-Shamir-Adleman), широко використовуються завдяки своїй ефективності у підтримці цілісності даних.

Застосовуючи передові практики, такі як регулярне оновлення ключів шифрування та впровадження багатофакторної автентифікації, ви можете ще більше покращити свої заходи безпеки. У швидкозмінному цифровому середовищі інформування про нові технології шифрування є критично важливим для будь-якої організації, яка прагне захистити цінні дані від потенційних загроз.

2. Брандмауери та антивірусне програмне забезпечення

Брандмауери та антивірусне програмне забезпечення є критично важливими компонентами захисту даних, що слугують першою лінією захисту від шкідливого програмного забезпечення та несанкціонованого доступу у ваших IT-системах. Впровадження захисту брандмауером може допомогти запобігти несанкціонованому доступу до вашої мережі.

Ці інструменти безпеки працюють шляхом моніторингу та контролю вхідного та вихідного мережевого трафіку, ефективно створюючи бар'єр між вашими довіреними внутрішніми мережами та ненадійними зовнішніми. Брандмауери ретельно перевіряють пакети даних і можуть блокувати потенційні загрози, перш ніж вони досягнуть ваших пристроїв, тоді як антивірусне програмне забезпечення виявляє, поміщає в карантин та видаляє шкідливі програми. Разом вони створюють надійну систему, яка захищає конфіденційну інформацію та забезпечує цілісність системи. Регулярний перегляд та оновлення налаштувань безпеки має вирішальне значення для підтримки ефективної безпеки мережі.

У швидкозмінному кіберсередовищі важливо регулярно оновлювати як брандмауери, так і антивірусні рішення. Таке обслуговування не лише забезпечує їх найновішими визначеннями загроз, але й підвищує їхню здатність боротися з новими вразливостями, що підкреслює важливість проактивних стратегій кіберзахисту.

3. Інструменти запобігання втраті даних

Інструменти запобігання втраті даних (DLP) є важливими для захисту конфіденційної інформації шляхом моніторингу та контролю передачі даних, що

ефективно мінімізує ризик несанкціонованого доступу та витоків даних. Використання методів управління даними разом з інструментами DLP може ще більше покращити захист цифрової інформації.

У сучасному цифровому середовищі, де кіберзагрози стають дедалі складнішими та поширенішими, ці інструменти стають ще більш важливими. Рішення DLP працюють, використовуючи комбінацію методів, включаючи перевірку контенту, контекстний аналіз та поведінкову аналітику, для виявлення та захисту конфіденційних даних на різних платформах. Регулярне навчання заходам безпеки та навчання співробітників з безпеки можуть допомогти розпізнати та пом'якшити загрози.

Організації зазвичай впроваджують стратегії, що включають шифрування конфіденційної інформації, забезпечення суворого контролю доступу та навчання співробітників найкращим практикам захисту даних. Вживаючи цих заходів, ви не лише покращуєте загальний рівень безпеки, але й забезпечуєте дотримання правил, зрештою будуючи довіру з клієнтами та зацікавленими сторонами. Регулярні оновлення програмного забезпечення сторонніх розробників можуть допомогти мінімізувати вразливості програмного забезпечення, які можуть бути використані.

Відповідність нормативним вимогам та безпека даних

Дотримання нормативних вимог є вирішальним аспектом безпеки даних для вашої організації. Важливо дотримуватися різних правил захисту даних, таких як GDPR та CCPA, щоб захистити конфіденційну інформацію та зберегти довіру ваших клієнтів. Регулярне оновлення налаштувань конфіденційності даних допоможе забезпечити дотримання вимог.

Огляд правил захисту даних (GDPR, CCPA тощо)

Нормативні акти щодо захисту даних, такі як GDPR та CCPA, забезпечують важливі рамки, яких ви повинні дотримуватися, щоб забезпечити відповідність вашим практикам безпеки даних та захистити особисту інформацію людей.

Ці правила спрямовані на створення балансу між правами осіб контролювати свою особисту інформацію та вашими обов’язками як організації щодо відповідального управління цими даними. Наприклад, GDPR наголошує на суворих вимогах щодо згоди та вимагає розробки чітких протоколів обробки даних, що включає забезпечення прозорості щодо використання даних та забезпечення прав осіб на доступ до своєї особистої інформації або її видалення. Аналогічно, CCPA надає жителям Каліфорнії право знати, які персональні дані збираються та передаються, тим самим розширюючи їхню автономію щодо особистої інформації.

Наслідки цих правил виходять за рамки простого дотримання; недотримання може призвести до значних фінансових штрафів та шкоди репутації. Це підкреслює критичну важливість впровадження надійних стратегій захисту даних у вашій організації.

Організаційні обов'язки щодо безпеки даних

Організації несуть значну відповідальність за безпеку даних, що включає впровадження ефективних стратегій кібербезпеки, забезпечення дотримання нормативних актів та захист конфіденційної інформації. Поради організацій, що надають консультації споживачам, таких як Національний центр кібербезпеки (NCSC), можуть бути безцінними в цьому відношенні.

Щоб ефективно захистити дані, ви повинні застосовувати проактивний підхід, усвідомлюючи, що цифровий ландшафт постійно розвивається та створює нові загрози. Це вимагає регулярної оцінки та оновлення ваших протоколів кібербезпеки для захисту від потенційних порушень та атак шкідливого програмного забезпечення. Дотримання законів про захист даних, таких як GDPR або CCPA, має вирішальне значення для підтримки довіри клієнтів та уникнення значних штрафів.

Також важливо брати участь у програмах навчання та підвищення обізнаності співробітників, оскільки співробітники часто виступають першою лінією захисту від кібератак. Інтегруючи ці заходи, організації можуть створити

надійну систему безпеки, яка не лише захищає дані, але й сприяє культурі безпеки та відповідальності.

Реагування на інциденти та відновлення після них

Реагування на інциденти та відновлення є важливими процесами в управлінні порушеннями даних та забезпеченні стійкості організації до кіберзагроз. Ці процеси спрямовані на відновлення нормальної роботи, одночасно захищаючи конфіденційну інформацію.

Дії, які слід вжити після витоку даних

Після витоку даних організаціям вкрай важливо вжити негайних заходів для зменшення збитків, оцінки наслідків та ініціювання процесів відновлення для захисту конфіденційної інформації та відновлення довіри.

Це передбачає швидке стримування порушення для запобігання будь-якому подальшому несанкціонованому доступу, а потім ретельне розслідування для розуміння масштабів порушення та виявлення вразливостей, які були використані. Після завершення розслідування важливо повідомити про інцидент постраждалі сторони, регуляторні органи та зацікавлені сторони, оскільки прозорість має вирішальне значення для підтримки довіри.

Зусилля з відновлення повинні бути зосереджені на відновленні систем та впровадженні заходів для запобігання повторенню. Проведення оцінки після інциденту є життєво важливим для вдосконалення протоколів безпеки, забезпечення того, щоб отриманий досвід спонукав до оновлення існуючих захисних механізмів, та підвищення загальної стійкості організації до майбутніх загроз.

Відновлення даних та систем

Відновлення даних і систем після порушення є критично важливим компонентом процесу реагування на інциденти, що гарантує, що ваша організація може відновити втрачені дані та підтримувати безпеку своєї IT-інфраструктури.

Цей процес не лише спрямований на відновлення скомпрометованої інформації, але й наголошує на впровадженні надійних заходів безпеки для

запобігання майбутнім інцидентам. Організації часто покладаються на резервні дані як фундаментальний елемент своєї стратегії відновлення, що дозволяє швидко відновити роботу, мінімізуючи час простою.

Ефективне відновлення вимагає поєднання регулярного резервного копіювання даних, впровадження посилених протоколів безпеки та постійного навчання персоналу розпізнаванню потенційних загроз. Постійне вдосконалення є важливим; воно спонукає організації вдосконалювати свої системи, оцінювати вразливості та інтегрувати найновіші технологічні оновлення та виправлення.

Застосовуючи проактивний підхід до кібербезпеки, ваша організація може краще захистити критично важливі дані та підвищити загальну стійкість системи.

На рисунку 3.1 зображена структурна схема роботи системи. Схема розділена на три основних компоненти:

– Flash накопичувач;
– розроблена програма;
– користувач.

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

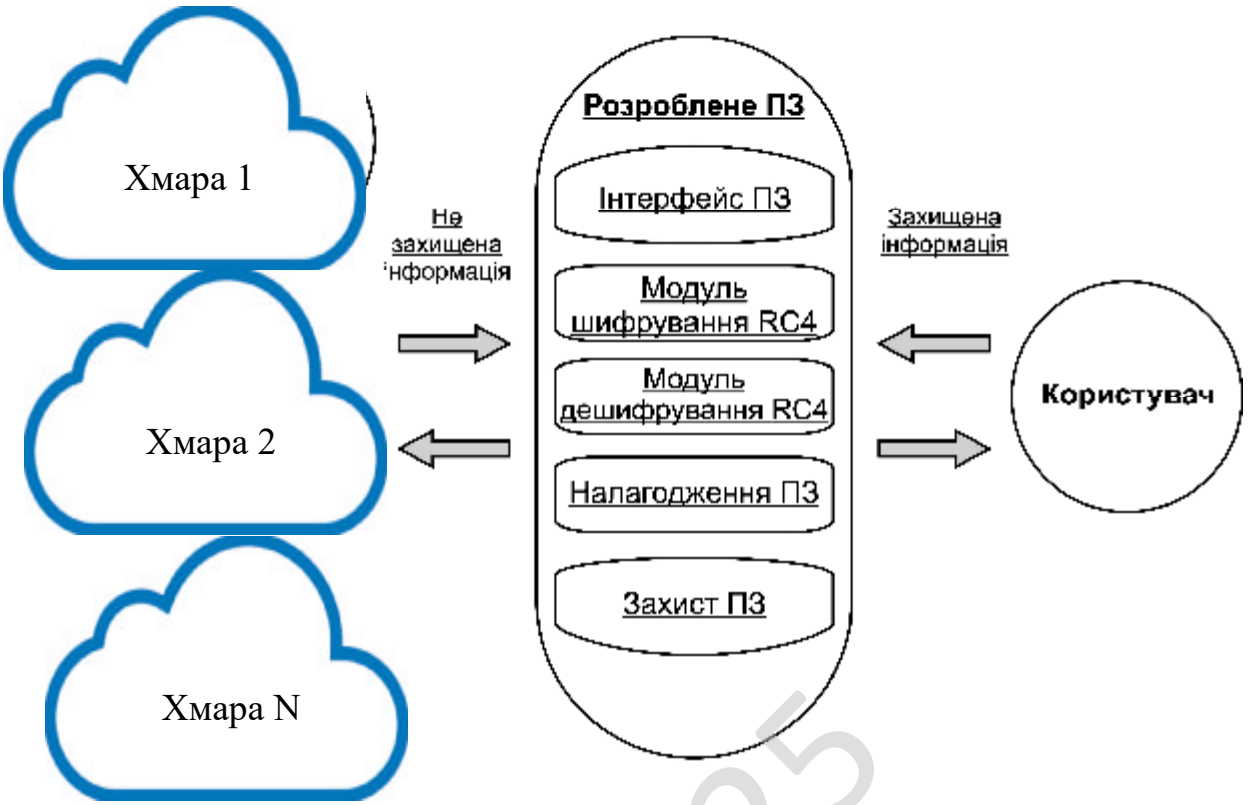

Рисунок 3.1 – Структурна схема роботи системи

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

Розроблена програма складається з декількох частин:

– інтерфейсу програми;

– модуля потокового шифрування RC4;

– модуля потокового дешифрування RC4;

– налаштування програми й захисту програми.

Розроблена програма управляє процесом обміну інформацією між Flash накопичувачем і персональним комп'ютером, використовуючи потоковий алгоритм шифрування інформації RC4. Завдяки такому підходу, можливо використовувати всі існуючі на даний момент Flash накопичувачі не зупиняючись на окремих реалізаціях з підвищеними вимогами захищеності Flash накопичувача (рисунок 3.1).

3.3 Розробка функціональної схеми роботи системи

Опис алгоритму шифрування

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

$$c_i = p_i \oplus k_i \quad (3.1)$$

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

$$p_i = c_i \oplus k_i \quad (3.2)$$

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

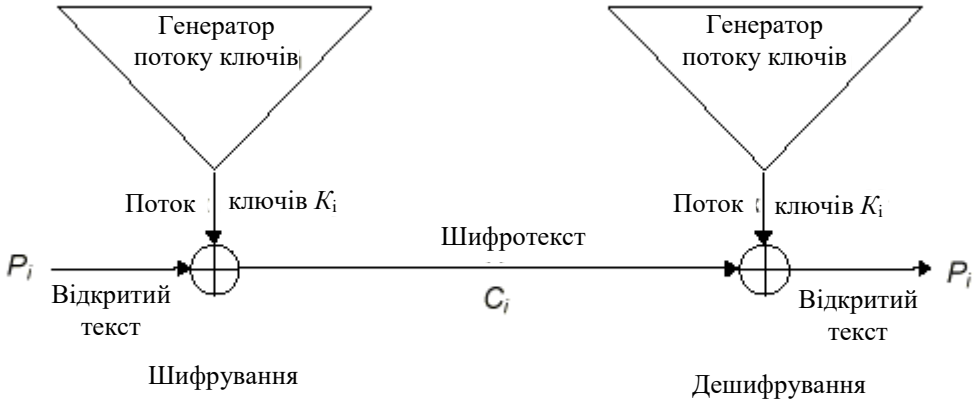

Рисунок 3.2 – Потоковий шифр

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

Генератор потоку ключів складається із трьох основних частин:

– Внутрішній стан описує поточний стан генератора потоку ключів.

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

– Функція виходу по внутрішньому стану генерує біт потоку ключів.

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

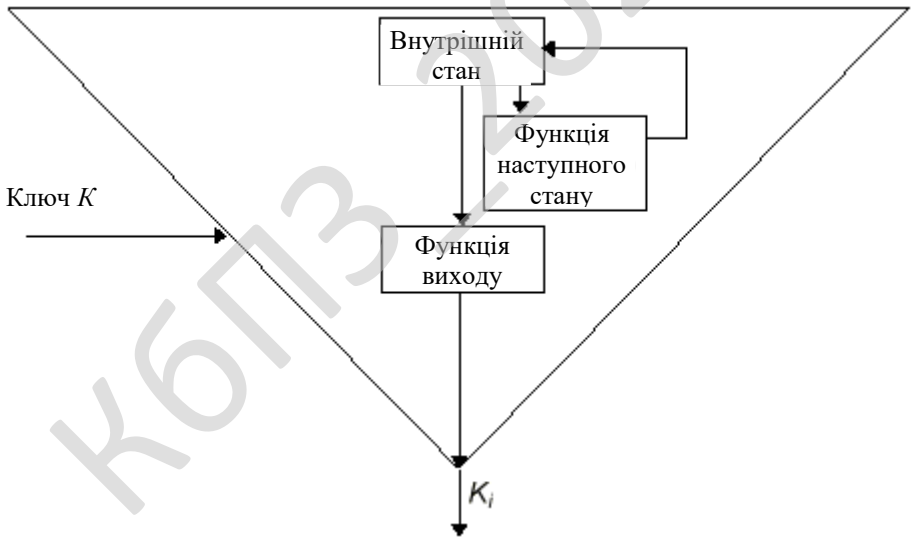

Рисунок 3.3 – Пристрій генератора потоку ключів

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

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

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

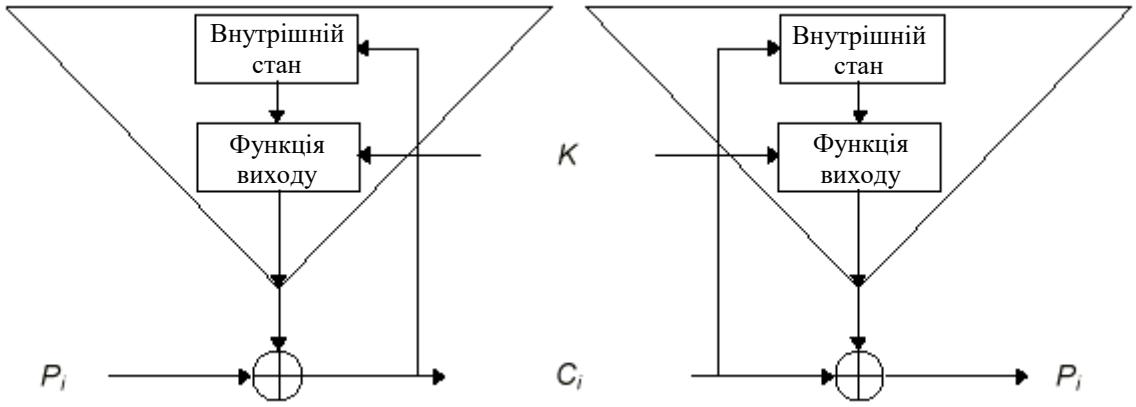

Рисунок 3.4 – Генератор потоку ключів, що самосинхронізується

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

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

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

Алгоритм RC4 і його криптоаналіз

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

криптоалгоритмів як можливій альтернативі апаратним схемам на регістрах зрушення.

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

Він володіє наступними властивостями:

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

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

– адаптивністю на процесори різних довжин слова;

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

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

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

– простотою й легкістю виконання.

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

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

Для генерації випадкового байта гами виконуються наступні операції:

$$i = (i+1) \bmod 256 \quad (3.3)$$

$$j = (j+S_i) \bmod 256 \quad (3.4)$$

$$\text{swap}\ (S_i, S_j) \quad (3.5)$$

$$t = (S_i+S_j) \bmod 256 \quad (3.6)$$

$$K = S_t \quad (3.7)$$

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

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

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

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

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

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

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

$$i_t = i_{t-1} + 1 \quad (3.8)$$

$$j_t = j_{t-1} + S_{t-1}(i_t) \quad (3.9)$$

$$S_t(i_t) = S_{t-1}(j_t) \quad (3.10)$$

$$S_t(j_t) = S_{t-1}(i_t) \quad (3.11)$$

$$Z_t = S_t(S_t(i_t) + S_t(j_t)), \quad (3.12)$$

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

$$K = (K_L)_{t=0}^{2n-1} \quad (3.13)$$

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

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

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

Відзначається, що для послідовностей, генеруємих RC4, не підходять методи статистичного аналізу. Але, з іншого боку, для блоків, розмір яких перевищує M (розмір внутрішньої пам'яті генератора), завжди існує лінійна статистична слабість або так звана "лінійна модель". Таку модель можна ефективно визначати за допомогою методу апроксимації лінійною послідовною

схемою. Лінійна статистична слабість – це лінійне співвідношення між бітами гами, що виконується з імовірністю, що відрізняється від 1/2.

За допомогою методу АЛПС були виведені лінійні моделі для RC4. Метод АЛПС полягає в знаходженні й рішенні послідовної лінійної схеми, що апроксимує генератор гами й приводить до лінійних моделей з відносно великим кореляційним коефіцієнтом c, де ймовірність відповідного лінійного співвідношення між бітами гами становить $(1 + c)/2$. При аналізі використовувалася техніка двійкових похідних. Нехай $Z = (Z_t)_{t=1}^{\infty}$ позначає послідовність самих молодших біт слів виходу RC4, і нехай $Z^{/}=(Z^{/}_t = Z_t + Z_{t+1})_{t=1}^{\infty}$ і $Z^{//}=(Z^{//}_t = Z_t + Z_{t+2})_{t=1}^{\infty}$ позначають її перші й другу двійкові похідні, відповідно. Показано, що $Z^{/}$ не корелює ні з 1, ні з 0, але $Z^{//}$ корелює з 1 з кореляційним коефіцієнтом, близьким до $15^* 2\text{-}3^n$ при великих $2n$, де n – довжина ключа. Оскільки довжина вихідної послідовності, необхідна для виявлення статистичної слабості з кореляційним коефіцієнтом c, становить O(c-2), те ця довжина дорівнює приблизно $64^n /225$. Наприклад, якщо $n = 8$, як рекомендується в більшості додатків, те необхідна довжина близька до 2^{40}.

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

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

завданні й рішенні лінійної послідовної схеми (ЛПС), що апроксимує вузол, що комбінує, з пам'яттю. Ця ЛПС має додаткові незбалансовані входи й заснована на лінійних апроксимаціях функції виходу й всіх компонентів функції наступного стану. Лінійна апроксимація булевої функції – це будь-яка лінійна функція, з якою задана булева функція скорельована. Описаний метод застосуємо до довільних вузлів, що комбінують, з пам'яттю без обмежень на функції виходу й наступний стан.

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

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

$$S_{t+1} = A \cdot S_t + B \cdot X_t + \Delta(X_t, S_t),\ t \geq 0, \tag{3.14}$$

$$y_t = C \cdot S_t + D \cdot X_t + \varepsilon(X_t, S_t),\ t \geq 0, \tag{3.15}$$

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

можна вирішувати цю ЛПС із використанням техніки виробляючих функцій (*D*-перетворень). Зокрема, нехай *S*, *X*, *Δ*, *ε*, у позначають виробляючі функції від змінної *z* для послідовностей $\{S_t\}$, $\{X_t\}$, $\Delta(X_t, S_t)\}$, $\varepsilon(X_t, S_t)\}$, $y_t\}$, відповідно. Тоді рівняння зводяться до виду:

$$y = \left(D - \frac{C \cdot adj(zA - I)B}{\det(zA - I)} \right) X - \frac{C \cdot adj(zA - I)}{\det(zA - I)} (z\Delta + S_0) + \varepsilon \qquad (3.16)$$

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

$$y = \frac{1}{\varphi(z)} \sum_{i=1}^{N} g_i(z) x_i + \frac{1}{\varphi(z)} \sum_{j=1}^{M} h_j(z)(z\delta_j + s_{j0}) + \varepsilon \qquad (3.17)$$

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

$$\varphi(z) = \sum_{k=0}^{M} \varphi_k z^k, \quad g_i(z) = \sum_{k=0}^{M} g_{ik} z^k, \quad h_j(z) = \sum_{k=0}^{M-1} h_{jk} z^k \qquad (3.18)$$

рішення можна перетворити до виду:

$$\sum_{k=0}^{M} \varphi_k y_{l-k} = \sum_{i=1}^{N} \sum_{k=0}^{M} g_{ik} x_{i,l-k} + e\left(X_i^{M+1}, S_{l-M}\right), \qquad t \geq M, \qquad (3.19)$$

$$e\left(X_i^{M+1}, S_{l-M}\right) = \sum_{j=1}^{N} \sum_{k=0}^{M-1} h_{jk} \delta_j \left(X_{t-1-k}, S_{t-1-k}\right) + \sum_{k=0}^{M-1} \varphi_k \varepsilon\left(X_{t-k}, S_{t-k}\right), \qquad t \geq M, \qquad (3.20)$$

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

шумових функцій не потрібно бути незалежними, у принципі не можна виключати можливість, що коефіцієнт кореляції *e* з константною нульовою функцією дорівнює нулю або дуже близький до цього значення.

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

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

статистичному аналізу потокового генератора RC4, у якій були використані результати роботи для знаходження значення компонент *S*-боксу. Приблизний час роботи цього методу становить 2^{6n}, де n – порція біт у вихідному потоці, довжина вихідної послідовності, необхідна для виявлення статистичної слабості, близька до 2^{30}. Отриманий результат указує на істотну слабість генератора й можливість відновити параметри i і n. *S*-бокс може приймати 2^{nk}, де n_k – число біт ключа.

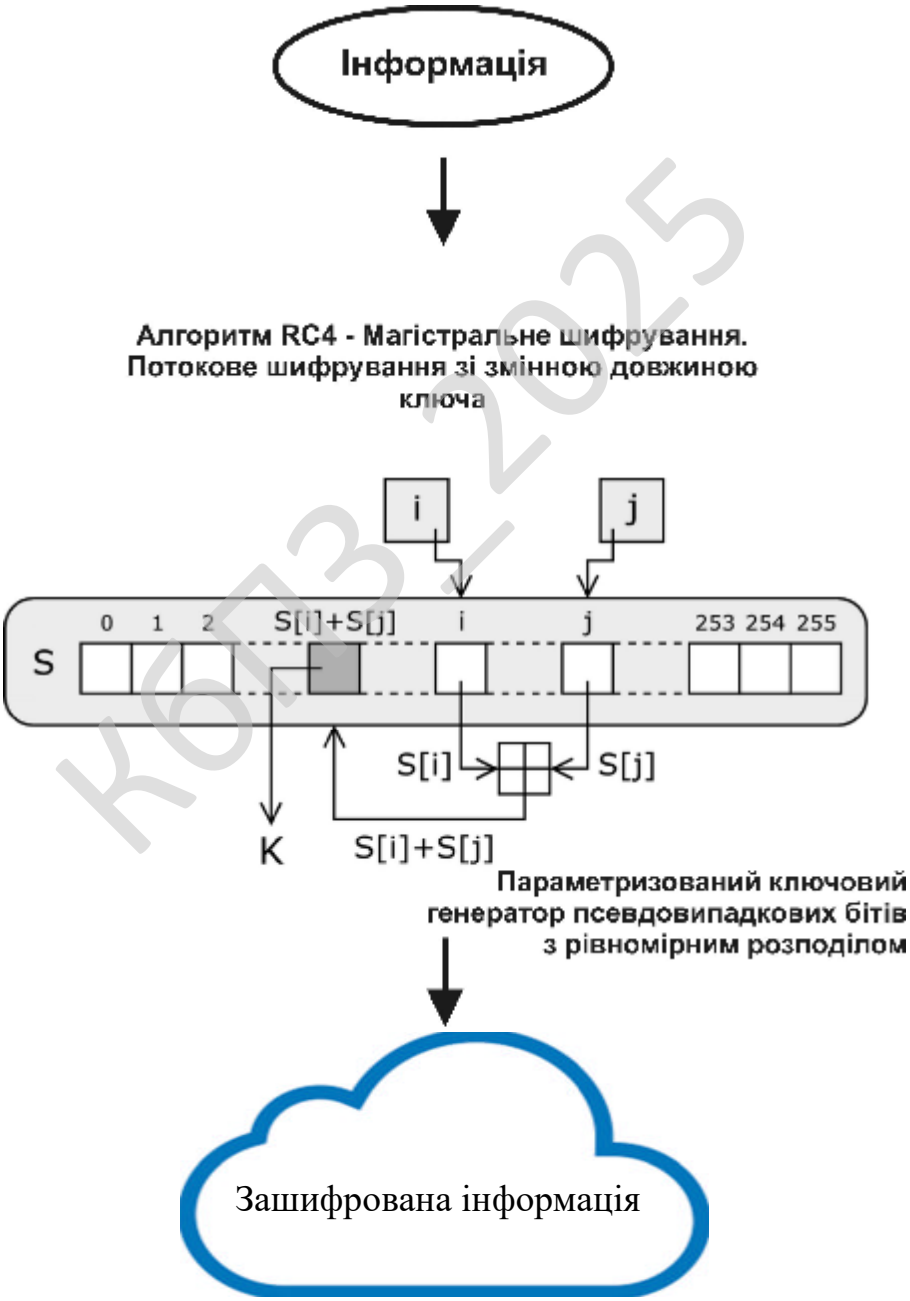

Рисунок 3.5 – Функціональна схема роботи системи

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

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

Процес дешифрації складається з регенерації цього ключового потоку й підсумовування його із шифрограммою по модулю два, відновлюючи вихідний текст. Також важливий елемент алгоритму функція ініціалізації, використовує ключ змінної довжини для створення початкового стану генератора ключового потоку. Параметр n є розміром слова для алгоритму. За замовчуванням програма встановлює значення, $n = 8$. для підвищення рівня безпеки необхідно збільшити це значення.

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

Ключем як і іншими налаштуваннями алгоритму управляє користувач через меню налаштування програми.

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

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.6. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

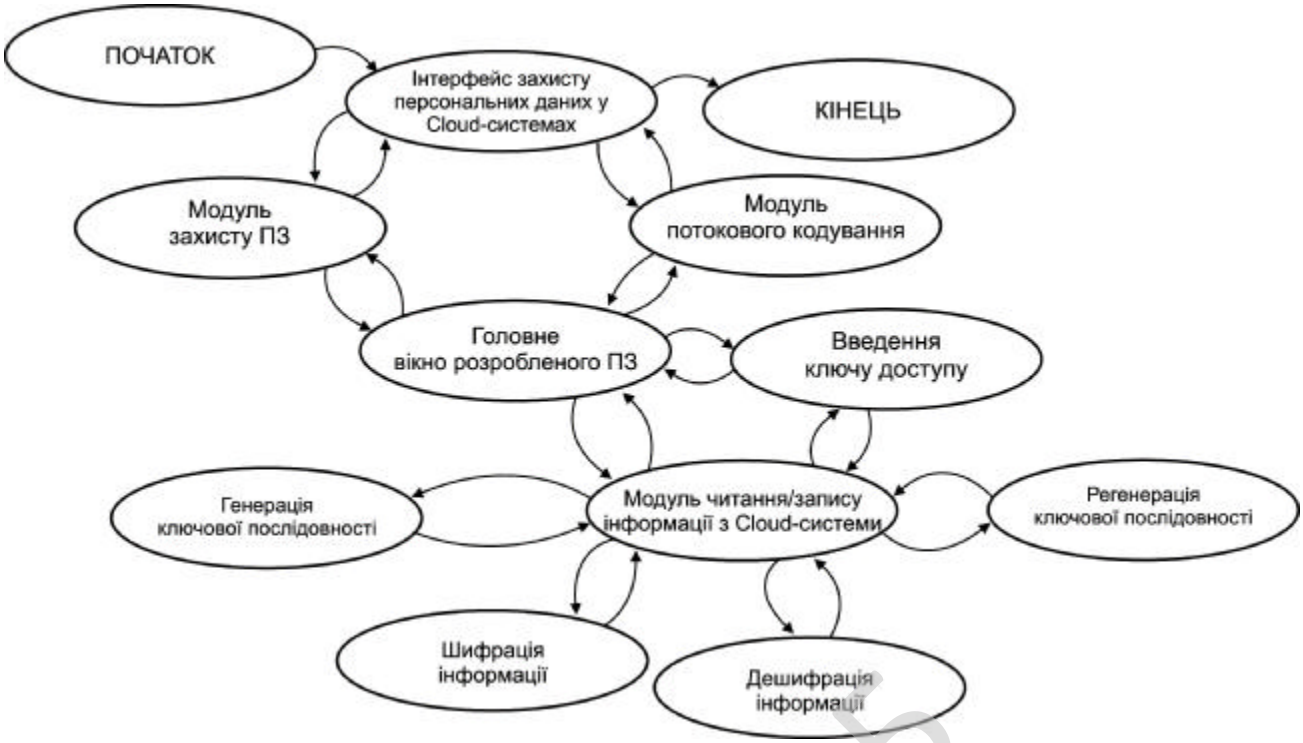

Рисунок 3.6 – Діаграма взаємодії процесів

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.

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Блок-схеми є основою ПЗ. Тому від точності і детальності проробки 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, також те, що при розробці програми слід надати особливу увагу модулю захисту персональних даних у Cloud-системах.

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

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 – перевірки поточного стану та поверненням на початок схеми чи з завершенням роботи розробленого ПЗ.

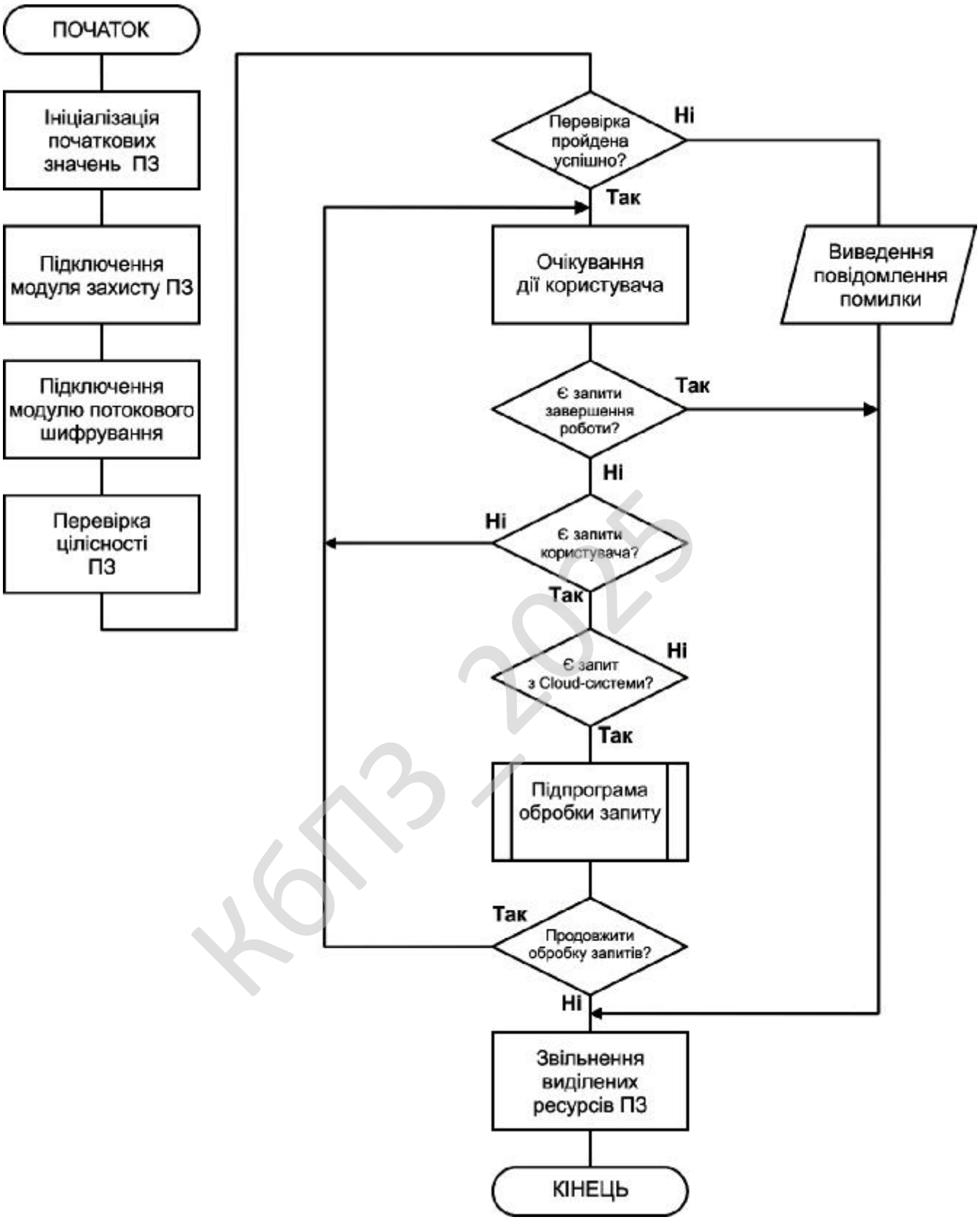

Рисунок 4.1 – Блок-схема основної програми

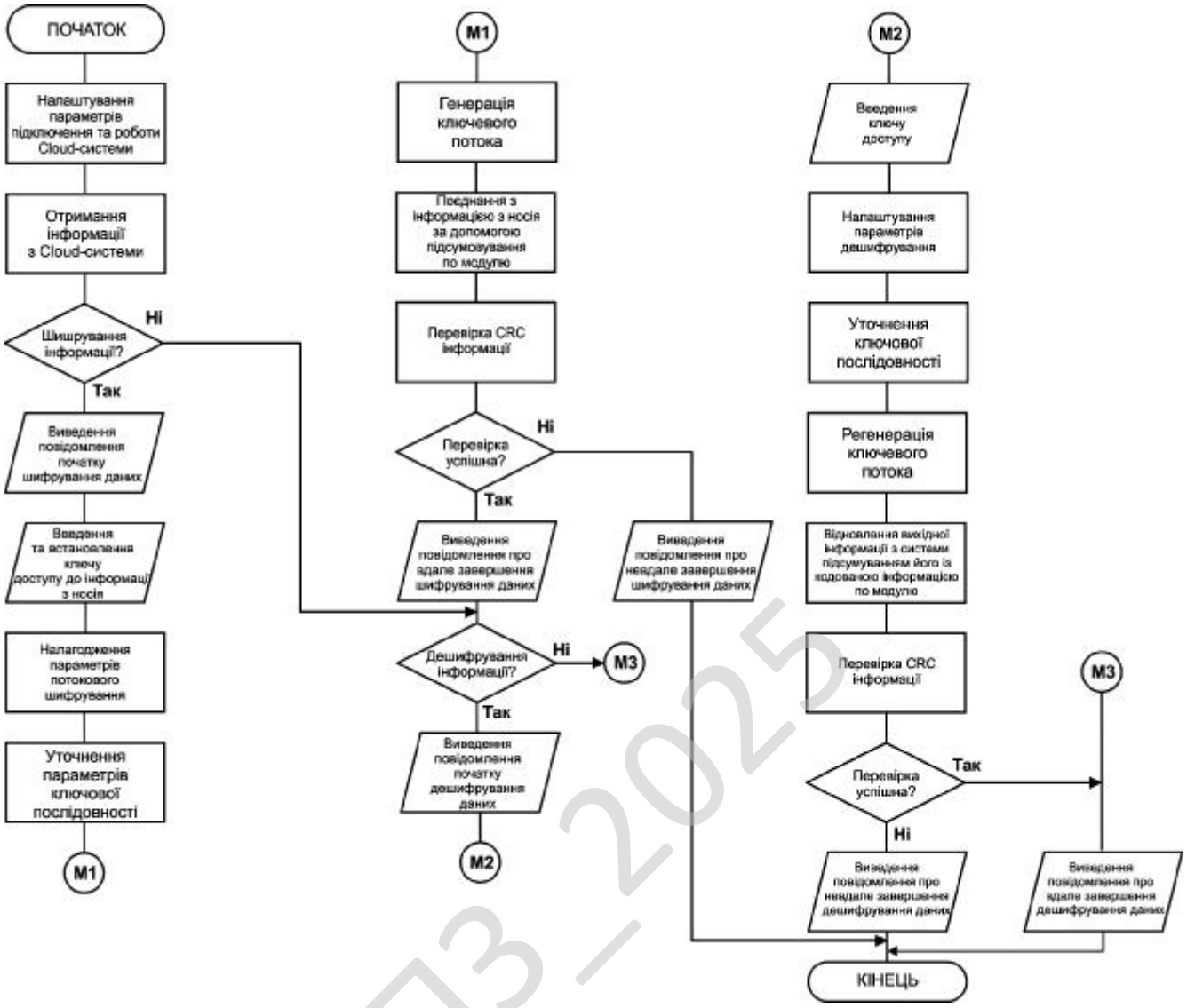

Рисунок 4.2 – Блок-схема роботи підпрограми

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

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю.

UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

Розглянемо використані технології та їх основні компоненти що підтверджують правильність використаних проектних рішень.

Текстовий опис програмної системи захисту персональних даних у мережевих Cloud системах

Програмна система захисту персональних даних у мережевих Cloud середовищах реалізується у вигляді багаторівневої Python програми. Система моделює типову архітектуру захисту даних у хмарній інфраструктурі і включає модулі моделювання користувачів та записів персональних даних, конфігурацію політик безпеки, шифрування, керування доступом, аудит подій, аналіз ризиків, а також сервіс, що обробляє запити до персональних даних.

Уся логіка системи представлена у вигляді класів та функцій мовою Python. Опис кожного значущого елемента вихідного коду розміщується у вигляді коментарів, що починаються з нового рядка. Коментарі пояснюють призначення класів, полів та методів, а також демонструють, як відбувається застосування політик безпеки до персональних даних у Cloud середовищі.

Моделювання ролей, класифікації та користувачів

На першому кроці система вводить узгоджені позначення для ролей користувачів і рівнів класифікації даних. Це спрощує опис політик доступу та дозволяє уніфіковано опрацьовувати персональні та службові записи. Далі система описує користувача як сутність з унікальним ідентифікатором, логіном, роллю, належністю до підрозділу та станом блокування.

Конфігурація безпеки та криптографічний модуль

Система зберігає параметри безпеки у датакласі конфігурації. Сюди входить ключ шифрування, шлях до файлу журналу безпеки, вимога до використання двофакторної автентифікації для чутливих даних, а також граничні

значення для обмеження кількості невдалих входів.

Оскільки система орієнтується на Cloud середовище, всі персональні дані перед зберіганням у сховищі шифруються. Для цього застосовується менеджер шифрування, що використовує симетричний алгоритм з ключем, заданим у конфігурації. У демонстраційному варіанті система використовує бібліотеку cryptography з механізмом Fernet. Це дає можливість безпечно кодувати та декодувати рядки.

Модель запису персональних даних і сховище

Запис персональних даних у системі відображається окремим датакласом. Він містить ідентифікатор запису, ідентифікатор власника, рівень класифікації, шифрований вміст, час створення та номер версії. Шифрований вміст зберігається як рядок, що отримується через менеджер шифрування.

Усі записи розміщуються у репозиторії. У демонстраційній реалізації цей репозиторій представлений як пам’яткова структура на основі словника Python. Такий підхід спрощує пояснення алгоритмів захисту, не прив’язуючи систему до конкретного Cloud провайдера.

Аудит та журнал безпеки

У Cloud середовищі ключову роль відіграє простежуваність дій з персональними даними. Для цього система реалізує модуль аудиту. Кожна значуща операція з даними та зміна стану безпеки фіксується у журналі. У прикладі журнал записується у текстовий файл формату JSON Lines, де кожен рядок містить час, користувача, тип дії та деталі.

Аналіз ризиків і керування доступом

Для обґрунтування безпечності проєктних рішень система додає простий модуль оцінки ризику доступу. Ризик оцінюється за рівнем класифікації запису, роллю користувача, відмінністю власника та особливостями контексту, наприклад репутацією IP адреси та часом доступу. Цей модуль надає числову оцінку ризику, яку використовує політика доступу.

Анонімізація та псевдонімізація

Для зниження ризиків витоку система реалізує модуль анонімізації. Якщо користувач не має повного доступу до запису, сервіс може повернути анонімізоване представлення, де ПІБ, електронна пошта та номер телефону частково замасковані. Такий режим корисний для аналітиків або аудиторів, які не повинні бачити повні персональні дані.

Основний сервіс захисту персональних даних

Серцем системи є сервіс, що приймає запити щодо зберігання, читання та оновлення персональних даних. Він використовує модулі шифрування, сховища, керування доступом, аудиту, анонімізації та аналізу ризиків.

Під час збереження даних сервіс шифрує JSON представлення запису і створює об'єкт PersonalDataRecord. Під час читання сервіс спочатку оцінює політику доступу. Якщо доступ дозволяється, дані розшифровуються і повертаються у первинному вигляді. Якщо доступ забороняється, але потрібен обмежений перегляд, сервіс формує анонімізоване представлення.

Побудова системи і демонстраційний сценарій

Щоб продемонструвати роботу системи у пояснювальній записці, вихідний код містить функцію побудови усіх компонентів і демонстраційний сценарій. Сценарій створює декілька користувачів з різними ролями, формує запис з чутливими персональними даними та показує, як власник і сторонній користувач отримують різні результати. Такий підхід дозволяє підтвердити правильність обраних проєктних рішень. У журнал безпеки записуються події збереження даних, успішного та заблокованого доступу, а також видачі анонімізованих представлень. Студент може проаналізувати записаний журнал, оцінити ризики і показати, що система реагує на ризиковий контекст блокуванням доступу до персональних даних.

```
from dataclasses import dataclass, field
from datetime import datetime
from typing import Dict, Any, List, Optional
import json
import uuid
```

```
from cryptography.fernet import Fernet

# Перелік ролей користувачів інформаційної системи
class Role:
    ADMIN = "admin"
    SECURITY_OFFICER = "security_officer"
    USER = "user"
    AUDITOR = "auditor"

# Рівні класифікації інформації
class Classification:
    PUBLIC = "public"
    INTERNAL = "internal"
    PERSONAL = "personal"
    SENSITIVE = "sensitive"

# Конфігурація параметрів безпеки системи
@dataclass
class SecurityConfig:
    encryption_key: bytes = field(default_factory=lambda: Fernet.generate_key())
    log_file: str = "security_audit.log"
    require_two_factor_for_sensitive: bool = True
    max_failed_logins: int = 5

# Модель користувача системи захисту
@dataclass
class User:
    user_id: str
    login: str
    role: str
    department: str
    failed_logins: int = 0
    is_blocked: bool = False
    two_factor_enabled: bool = True

# Модель запису персональних даних у сховищі
@dataclass
class PersonalDataRecord:
    record_id: str
    owner_id: str
    classification: str
    encrypted_payload: str
```

```
    created_at: datetime = field(default_factory=datetime.utcnow)
    version: int = 1

# Менеджер симетричного шифрування персональних даних
class CryptoManager:
    def __init__(self, config: SecurityConfig):
        self._config = config
        self._fernet = Fernet(self._config.encryption_key)
    def encrypt_text(self, text: str) -> str:
        # Шифрування текстового представлення даних
        data = text.encode("utf-8")
        token = self._fernet.encrypt(data)
        return token.decode("ascii")
    def decrypt_text(self, token: str) -> str:
        # Розшифрування текстового представлення даних
        data = self._fernet.decrypt(token.encode("ascii"))
        return data.decode("utf-8")

# Пам'яткове репозиторійне сховище персональних даних
class PersonalDataRepository:
    def __init__(self):
        self._records: Dict[str, PersonalDataRecord] = {}
    def save(self, record: PersonalDataRecord) -> None:
        # Збереження або оновлення запису у сховищі
        self._records[record.record_id] = record
    def get(self, record_id: str) -> Optional[PersonalDataRecord]:
        # Пошук запису за ідентифікатором
       return self._records.get(record_id)
    def find_by_owner(self, owner_id: str) -> List[PersonalDataRecord]:
        # Отримання всіх записів для конкретного власника
        return [r for r in self._records.values() if r.owner_id == owner_id]

# Модуль ведення журналу безпеки
class AuditLogger:
    def __init__(self, config: SecurityConfig):
        self._config = config
    def log(self, user: Optional[User], action: str, details: Dict[str, Any]) ->
None:
        # Запис події безпеки у журнал у форматі JSON
        entry = {
            "time": datetime.utcnow().isoformat(),
            "user_id": user.user_id if user else None,
```

```
            "login": user.login if user else None,
            "action": action,
            "details": details,
        }
        line = json.dumps(entry, ensure_ascii=False)
        with open(self._config.log_file, "a", encoding="utf-8") as f:
           f.write(line + "\n")

# Модуль анонімізації полів персональних даних
class DataAnonymizer:
    def anonymize(self, data: Dict[str, Any]) -> Dict[str, Any]:
        # Формування анонімізованої копії словника з даними
        masked: Dict[str, Any] = {}
        for key, value in data.items():
            if key.lower() in ("name", "full_name"):
                masked[key] = self._mask_name(str(value))
            elif key.lower() in ("email", "e_mail"):
                masked[key] = self._mask_email(str(value))
            elif key.lower() in ("phone", "phone_number", "mobile"):
                masked[key] = self._mask_phone(str(value))
            else:
                masked[key] = value
        return masked
    def _mask_name(self, name: str) -> str:
        # Маскування повного імені користувача
        parts = name.split()
        if not parts:
            return name
        first = parts[0]
        masked_rest = " ".join(p[0] + "." for p in parts[1:])
        return first + " " + masked_rest if masked_rest else first
    def _mask_email(self, email: str) -> str:
        # Маскування електронної адреси
        if "@" not in email:
            return email
        local, domain = email.split("@", 1)
        if len(local) <= 2:
            return "*" * len(local) + "@" + domain
        return local[0] + "*" * (len(local) - 2) + local[-1] + "@" + domain
    def _mask_phone(self, phone: str) -> str:
        # Маскування номеру телефону
        digits = [c for c in phone if c.isdigit()]
```

```
        if len(digits) < 4:
            return "*" * len(phone)
        visible = digits[-4:]
        return "***" + "".join(visible)

# Модуль обчислення інтегрального ризику доступу
class RiskAnalyzer:
    def __init__(self, high_risk_threshold: int = 60):
        self.high_risk_threshold = high_risk_threshold
    def evaluate_access_risk(
        self,
        user: User,
        record: PersonalDataRecord,
        context: Dict[str, Any],
    ) -> int:
        # Обчислення сумарного ризику запиту доступу
        risk = 0
        if record.classification == Classification.SENSITIVE:
            risk += 40
        if context.get("ip_reputation") == "low":
            risk += 30
        if context.get("access_time") == "night":
            risk += 10
        if user.role == Role.USER and record.owner_id != user.user_id:
            risk += 20
        return risk

# Модуль перевірки прав доступу до персональних даних
class AccessControl:
    def __init__(self, config: SecurityConfig, audit: AuditLogger, risk_analyzer:
RiskAnalyzer):
        self._config = config
        self._audit = audit
        self._risk_analyzer = risk_analyzer
    def can_read(
        self,
        user: User,
        record: PersonalDataRecord,
        context: Dict[str, Any],
    ) -> bool:
        # Перевірка права читання запису з урахуванням ролі і ризику
        if user.is_blocked:
```

```
            self._audit.log(user, "blocked_user_read_attempt", {"record_id":
record.record_id})
            return False
        base_allowed = False
        if user.role in (Role.ADMIN, Role.SECURITY_OFFICER, Role.AUDITOR):
            base_allowed = True
        elif record.classification in (Classification.PERSONAL,
Classification.SENSITIVE):
            base_allowed = record.owner_id == user.user_id
        else:
            base_allowed = True
        risk_value = self._risk_analyzer.evaluate_access_risk(user, record,
context)
        if risk_value >= self._risk_analyzer.high_risk_threshold:
            self._audit.log(
                user,
                "high_risk_access_blocked",
                {"record_id": record.record_id, "risk": risk_value},
            )
            return False
        if base_allowed:
            self._audit.log(
                user,
                "access_granted",
                {"record_id": record.record_id, "risk": risk_value},
            )
            return True
        self._audit.log(
            user,
            "access_denied",
            {"record_id": record.record_id, "risk": risk_value},
        )
        return False

# Сервіс високого рівня для роботи з персональними даними
class PersonalDataService:
    def __init__(
        self,
        crypto: CryptoManager,
        repo: PersonalDataRepository,
        access: AccessControl,
        audit: AuditLogger,
```

```
        anonymizer: DataAnonymizer,
    ):
        self._crypto = crypto
        self._repo = repo
        self._access = access
        self._audit = audit
        self._anonymizer = anonymizer

    def store_personal_data(
        self,
        user: User,
        owner_id: str,
        classification: str,
        payload: Dict[str, Any],
        context: Dict[str, Any],
    ) -> str:
        # Збереження нового запису персональних даних з попереднім шифруванням
        serialized = json.dumps(payload, ensure_ascii=False)
        encrypted_payload = self._crypto.encrypt_text(serialized)
        record_id = str(uuid.uuid4())
        record = PersonalDataRecord(
            record_id=record_id,
            owner_id=owner_id,
            classification=classification,
            encrypted_payload=encrypted_payload,
        )
        self._repo.save(record)
        self._audit.log(
            user,
            "store_personal_data",
            {"record_id": record_id, "classification": classification},
        )
        return record_id

    def read_personal_data(
        self,
        user: User,
        record_id: str,
        context: Dict[str, Any],
        allow_anonymized: bool = True,
    ) -> Optional[Dict[str, Any]]:
        # Отримання запису персональних даних з урахуванням політики доступу
```

```
        record = self._repo.get(record_id)
        if record is None:
            self._audit.log(user, "record_not_found", {"record_id": record_id})
            return None
        if self._access.can_read(user, record, context):
            decrypted = self._crypto.decrypt_text(record.encrypted_payload)
            data = json.loads(decrypted)
            return data
        if allow_anonymized:
            decrypted = self._crypto.decrypt_text(record.encrypted_payload)
            data = json.loads(decrypted)
            anonymized = self._anonymizer.anonymize(data)
            self._audit.log(
                user,
                "anonymized_view_returned",
                {"record_id": record_id},
            )
            return anonymized
        self._audit.log(
            user,
            "read_denied_no_anonymized",
            {"record_id": record_id},
        )
        return None

# Комплексна система захисту персональних даних
@dataclass
class PrivacySystem:
    config: SecurityConfig
    crypto: CryptoManager
    repo: PersonalDataRepository
    audit: AuditLogger
    risk_analyzer: RiskAnalyzer
    access: AccessControl
    anonymizer: DataAnonymizer
    service: PersonalDataService

# Побудова повної конфігурації системи захисту персональних даних
def build_privacy_system() -> PrivacySystem:
    config = SecurityConfig()
    audit = AuditLogger(config)
    crypto = CryptoManager(config)
```

```
    repo = PersonalDataRepository()
    risk_analyzer = RiskAnalyzer()
    access = AccessControl(config, audit, risk_analyzer)
    anonymizer = DataAnonymizer()
    service = PersonalDataService(crypto, repo, access, audit, anonymizer)
    return PrivacySystem(
        config=config,
        crypto=crypto,
        repo=repo,
        audit=audit,
        risk_analyzer=risk_analyzer,
        access=access,
        anonymizer=anonymizer,
        service=service,
    )

# Демонстраційний сценарій використання системи
def demo_scenario() -> None:
    system = build_privacy_system()
    owner = User(
        user_id="u1",
        login="owner",
        role=Role.USER,
        department="it",
    )
    outsider = User(
        user_id="u2",
        login="guest",
        role=Role.USER,
        department="external",
    )
    admin = User(
        user_id="admin",
        login="admin",
        role=Role.ADMIN,
        department="security",
    )
    payload = {
        "name": "Іван Іваненко",
        "email": "ivan@example.com",
        "phone": "+380671112233",
        "diagnosis": "Приклад чутливих медичних даних",
```

```
    }
    context_normal = {
        "ip_reputation": "normal",
        "access_time": "day",
    }
    context_suspicious = {
        "ip_reputation": "low",
        "access_time": "night",
    }
    record_id = system.service.store_personal_data(
        user=admin,
        owner_id=owner.user_id,
        classification=Classification.SENSITIVE,
        payload=payload,
        context=context_normal,
    )
    data_for_owner = system.service.read_personal_data(
        user=owner,
        record_id=record_id,
        context=context_normal,
        allow_anonymized=False,
    )
    print("Дані для власника")
    print(data_for_owner)
    data_for_outsider = system.service.read_personal_data(
        user=outsider,
        record_id=record_id,
        context=context_suspicious,
        allow_anonymized=True,
    )
    print("Дані для стороннього користувача")
    print(data_for_outsider)

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

Redmine – вільне серверне ПЗ для управління проектами та відстежування помилок. До системи входить календар-планувальник та діаграми Ганта для візуального представлення ходу робіт за проектом та строків виконання. Redmine написано на мові Ruby і є ПЗ розробленим з використанням відомого веб-

фреймворку Ruby on Rails, що означає легкість в розгортанні системи та її адаптації під конкретні вимоги. Для кожного проекту можна вести свої вікі та форуми.

Функціональні можливості:

– Ведення декількох проектів.

– Гнучка система доступу з використанням ролей.

– Система відстеження помилок.

– Діаграми Ганта та календар.

– Ведення новин проекту, документів та управління файлами.

– Сповіщення про зміни за допомогою RSS-потоків та електронної пошти.

– Власна Wiki для кожного проекту.

– Форуми для кожного проекту.

– Облік часових витрат.

– Налаштування власних (custom) полів для задач, затрат часу, проектів та користувачів.

– Легка інтеграція із системами керування версіями (SVN, CVS, Git, Mercurial, Bazaar и Darcs).

– Створення записів про помилки на основі отриманих листів

– Підтримка LDAP автентифікації.

– Можливість самореєстрації нових користувачів.

– Багатомовний інтерфейс (у тому числі українська мова).

– Підтримка СКБД: MySQL, PostgreSQL, SQLite.

Діаграма Ганта (Gantt chart, також стрічкова діаграма, графік Ганта) – це популярний тип діаграм, який використовується для ілюстрації плану, графіка робіт за будь-яким проектом. Є одним з методів планування та управління проектами.

Діаграма Ганта являє собою відрізки (графічні плашки), розміщені на горизонтальній шкалі часу. Кожен відрізок відповідає окремому завданню або підзадачі. Завдання і підзадачі, складові плану, розміщуються по вертикалі.

Початок, кінець і довжина відрізка на шкалі часу відповідають початку, кінцю і тривалості завдання. На деяких діаграмах Ганта також показується залежність між завданнями.

Діаграма може використовуватися для представлення поточного стану виконання робіт: частина прямокутника, що відповідає завданню, заштриховується, відзначаючи відсоток виконання завдання; показується вертикальна лінія, що відповідає моменту «сьогодні».

Часто діаграма Ганта використовується спільно з таблицею зі списком робіт, рядки якої відповідають окремо взятій задачі, зображеній на діаграмі, а стовпці містять додаткову інформацію про задачу.

Система відстеження помилок Багтрекер – прикладна програма для допомоги розробникам програмного забезпечення (програмістам, тестувальникам тощо) враховувати і контролювати помилки, знайдені у програмах, питання щодо функціональності, рішення та оновлення, побажання користувачів, а також стежити за процесом їх виконання.

Кожному, хто розробляв програмні продукти, добре знайоме співвідношення «20/80» – останні 20 % роботи тривають 80 % часу.

Як це не парадоксально, але нічого дивного в цій пропорції немає, адже саме на завершальній стадії починається тестування проекту, коли виявляються помилки, і що більший проект, то більше буде знайдено помилок.

Водночас досить часто виявляється, що більшість цих помилок були відомі та могли бути виправлені з меншими витратами на попередніх стадіях роботи, але не були вчасно описані, а потім загубилися серед інших важливих завдань.

Отже, система відстеження помилок у найпростішому варіанті – це процес, що включає в себе виявлення помилки, її опис, виправлення і перевірку цього виправлення, тобто процес «стеження» за багом протягом всього як його життєвого циклу, так і життєвого циклу розробки в цілому.

Сукупність інформації про дефект. Головний компонент такої системи – база даних, що містить відомості про виявлені дефекти. Ці відомості можуть включати в себе:

– номер (ідентифікатор) дефекту;

– хто повідомив про дефект;

– дата і час виявлення дефекту;

– версія продукту, в якій виявлено дефект;

– серйозність (критичність) дефекту та пріоритет рішення;

– опис кроків для відтворення дефекту (неправильної поведінки програми);

– відповідальний за усунення дефекту;

– обговорення можливих рішень та їх наслідків;

– поточний стан виправлення дефекту;

– версії продукту, в якій дефект виправлений.

Крім того, розвинені системи надають можливість прикріплювати файли, які допомагають описати проблему, наприклад, дамп пам'яті або скріншот.

Використання. Основна перевага систем відстеження помилок полягає в забезпеченні чітких централізованих оглядів, запитів на розробку (включаючи помилки і виправлення) та їх стан. У корпоративному середовищі, системи відстеження помилок можуть бути використані для генерації звітів по продуктивності програмістів виправлення помилок. Однак, це може іноді приводити до неточних результатів, тому що різні помилки можуть мати різні ступені пріоритету та серйозності, що пов'язано з складністю їх фіксації.

Життєвий цикл дефекту. Як правило, система відстеження помилок використовує той чи інший варіант «життєвого циклу» помилки, стадія якого визначається поточним станом помилки.

Типовий життєвий цикл дефекту:

1. Новий – дефект зареєстрований тестувальником.

2. Призначений – призначений відповідальний за виправлення дефекту.

3. Дозволений – дефект переходить назад у сферу відповідальності тестувальника. Як правило, супроводжується резолюцією, наприклад:

– Виправлено (виправлення включені у версію таку-то).

– Дубль (повторює дефект, що вже знаходиться в роботі).

– Не виправлено (працює відповідно до специфікації, має занадто низький пріоритет, виправлення відкладено до наступної версії тощо).

– «В мене все працює» (запит додаткової інформації про умови, в яких дефект проявляється).

4. Далі тестувальник проводить перевірку виправлення, залежно від чого дефект або знову переходить у стан «Призначений» (якщо він описаний як виправлений, але не виправлений), або у стан «Закрито».

5. Відкрито повторно – дефект знайдено знову в іншій версії.

Система може надавати адміністраторові можливість налаштування користувачі, які можуть переглядати і редагувати помилки залежно від їх стану, переводити їх в інший стан або видаляти.

У корпоративному середовищі, система відстеження помилок може використовуватися для отримання звітів, що показують продуктивність програмістів при виправленні помилок. Однак, часто такий підхід не дає достатньо точних результатів через те, що різні помилки мають різну ступінь серйозності та складності. При цьому серйозність проблеми прямо не стосується складності її усунення.

Хоча я реалізовував програму сам, було використано підходи Scrum для саморозвитку та пришвидшенню розробки, розглянемо цей метод. Scrum – підхід управління проектами для гнучкої розробки програмного забезпечення. Скрам чітко робить акцент на якісному контролі процесу розробки.

Підхід вперше описали Гіротака Такеучі та Ікуджіро Нонака в статті The New New Product Development Game (Гарвардський Діловий Огляд, січ–лют 1986). Вони відзначили, що проекти, над якими працюють невеликі, крос–функціональні команди, зазвичай систематично продукують кращі результати, і

пояснили це, як «підхід регбі». У 1991 році ДеҐрейс та Шталь у книжці Злі проблеми, справедливі рішення послалися на цей підхід, як на Scrum (штовханина; сутичка навколо м'яча (у регбі)), спортивний термін, згаданий в статті Такеучі і Нонака. Кен Швабер на початку 1990–х використовував підхід який привів Scrum в його компанію.

Вперше метод Scrum було представлено на загальний огляд задокументованим, чітко сформульованим та описаним спільно Сазерлендом та Швабером на OOPSLA'96 в Остіні. Швабер та Сазерленд протягом наступних років працювали разом щоб обробити та описати весь їхній досвід та найкращі практичні зразки для індустрії в одне ціле, в ту методологію, що відома сьогодні як Scrum. Швабер об'єднав зусилля з Майком Бідлом в 2001, щоб детально описати метод в книжці Agile Software Development with SCRUM. Не зважаючи на те, що для Scrum нарікли долю управління проектами з розробки ПЗ, він може також використовуватися в роботі команд обслуговувань програмного забезпечення (software maintenance teams), або як підхід управління розробкою і супроводом програм: Scrum of Scrums.

Scrum – це кістяк процесу, який включає набір методів і попередньо визначених ролей. Головні дійові особи – ScrumMaster, той хто опікується процесами, веде їх і працює як керівник проекту, Власник Продукту, людина, що представляє інтереси кінцевих користувачів та інших зацікавлених в продукті сторін, та Команду, яка включає розробників.

Протягом кожного спринту, 15–30 денного періоду (тривалість визначається командою), працівники створюють функціональний ріст програмного забезпечення.

Набір можливостей, які імплементуються кожного спринту, приходять з етапу, що має назву product backlog (документація запитів на виконання робіт), який має найвищу пріоритетність за рівнем вимог до роботи, що повинна бути виконана.

Запити на виконання робіт (backlog items), що визначені протягом наради з планування спринту (sprint planning meeting), переміщуються в етап спринту. Протягом цієї наради Власник Продукту інформує про завдання, які він хоче, аби були виконані. Тоді Команда визначає, скільки з бажаного вони можуть зробити, щоб завершити необхідні частини протягом наступного спринту. Протягом спринту команда виконує визначений фіксований список завдань (т.з. backlog items). Впродовж цього періоду ніхто не має права змінювати перелік запитів на виконання робіт, що слід розуміти, як заморожування вимог (requirements) протягом спринту.

Product backlog – це документ, який має список вимог до функціональності, які упорядковані згідно зі ступенем важливості. Product backlog представляє список того, що повинно бути реалізовано. Елементи цього списку називається «історіями» (user story) або елементами backlog–у (backlog items). Product backlog відкритий для редагування усім учасникам Scrum–процесу.

Обов'язкові поля:

1. ID – унікальний ідентифікатор, порядковий номер, який використовується для ідентифікації історій у разі їх перейменування.

2. Назва (Name) – стислий опис історії. Він повинен бути однозначним, щоб і розробники і product owner могли зрозуміти, про що йдеться і відрізнити одну історію від іншої.

3. Важливість (Importance) – ступінь важливості даної історії на погляд product owner ’а. Зазвичай являє собою натуральне число, іноді для цієї цілі використовуються числа Фібоначчі. Чим більше значення, тим більше пріоритет.

4. Попередня оцінка (initial estimate) – початкова оцінка об'єму робіт, необхідного для реалізації історії порівняно з іншими історіями. Вимірюється у story point'ах. Приблизно відповідає числу «ідеальних людино–днів».

5. Як продемонструвати (how to demo) – стисле пояснення того, як завершена задача буде продемонстрована у кінці спринта. Дане поле може являти собою код автоматизованого приймального тесту.

Додаткові поля. Іноді, також, використовуються додаткові поля у product backlog, в основному для того, щоб допомогти product owner'у визначитися з його пріоритетами.

Категорія (track). Наприклад, «панель управління» чи «оптимізація». За допомогою цього поля product owner може легко вибрати усі пункти категорії «оптимізація» і задати їм низький пріоритет.

Компоненти (components) – указує, які компоненти (наприклад, база даних, сервер, клієнт) будуть зачеплені при реалізації історії. Дане поле складається з групи checkbox'ів, які відмічаються, якщо відповідні компоненти потребують змін.

Ініціатор запиту (requestor). Product owner може захотіти зберігати інформацію про усіх замовників, зацікавлених у даній задачі. Це потрібно для того, щоб тримати їх у курсі діла про хід виконання робіт.

ID у системі обліку помилок (bug tracking ID) – якщо ви використовуєте окрему систему обліку помилок, тоді у описі історії корисно зберігати посилання на всі дефекти, які до неї відносяться.

Sprint backlog – містить функціональність, обрану Product Owner із Product Backlog. Всі функції розбиті по задачах, кожна з яких оцінюється командою. Кожен день команда оцінює об'єм роботи, який необхідно провести для завершення задачі.

Burndown chart – показує, скільки вже виконано і скільки ще залишається зробити.

Планування спринта (Sprint Planning Meeting)

Проходить на початку нової ітерації Спринта:

– Із Product Backlog обираються задачі, зобов'язання по виконанню яких за спринт приймає на себе команда;

– На основі обраних задач створюється Sprint Backlog. Кожна задача оцінюється у ідеальних людино–годинах;

– Рішення задачі не повинно займати більше 12 годин або одного дня. При необхідності задача розбивається на підзадачі;

– Обговорюється та визначається, яким чином буде реалізовано цей об'єм робіт;

– Тривалість наради обмежена зверху 4–8 годинами в залежності від тривалості ітерації, досвіду команди тощо;

– (перша частина наради) Беруть участь Product Owner + Команда: обирають задачі із Product Backlog;

– (друга частина наради) Бере участь лише команда: обговорюють технічні деталі реалізації, наповнюють Sprint Backlog.

Щоденна нарада (Daily Scrum Meeting)

Відбувається кожен день протягом спринта. Є «пульсом» ходу спринта. Нараді властиві наступні обмеження:

– починається точно вчасно;

– всі можуть спостерігати, але говорять тільки обрані;

– триває не більш ніж 15 хвилин;

– проводиться в одному і тому ж місці протягом одного спринта.

Протягом наради кожен член команди відповідає на 3 запитання:

– Що зроблено з моменту попередньої щоденної наради?;

– Що буде зроблено з моменту поточної наради до наступної?;

– Які проблеми заважають досягненню цілей спринта? (Над рішенням цих проблем працює ScrumMaster. Зазвичай це рішення проходить за рамками щоденної наради і у складі осіб, що безпосередньо займаються даною перешкодою.)

Демонстрація (Sprint Review Meeting):

– Проходить у кінці ітерації (спринта).

– Команда демонструє внесок функціональності до продукту всім зацікавленим особам.

– Залучається максимальна кількість глядачів.

– Усі члени команди беруть участь у демонстрації (одна людина на демонстрацію або кожен показує, що зробив за спринт).

– Обмежена 4–ма годинами в залежності від тривалості ітерації і змін у продукті.

Ретроспектива (Sprint Retrospective):

– Члени команди висловлюють свою думку про минулий спринт.

– Відповідають на два основних запитання: Що було зроблено добре у минулому спринті?; Що потрібно покращити в наступному?.

– Виконують покращення процесу розробки (вирішують питання та фіксують вдалі рішення).

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою алгоритму DES.

DES є класичною мережею Фейштеля із двома гілками. Дані шифруються 64-бітними блоками, використовуючи 56-бітний ключ. Алгоритм перетворить за кілька раундів 64-бітний вхід в 64-бітний вихід. Довжина ключа дорівнює 56 бітам. Процес шифрування складається із чотирьох етапів. На першому з них виконується початкова перестановка (IP) 64-бітного вихідного тексту (забілювання), під час якої біти переставляються у відповідності зі стандартною таблицею.

Наступний етап складається з 16 раундів однієї й тої ж функції, що використовує операції зрушення й підстановки. На третьому етапі ліва й права половини виходу останньої (16-й) ітерації міняються місцями.

Нарешті, на четвертому етапі виконується перестановка IP^{-1} результату, отриманого на третьому етапі. Перестановка IP^{-1} інверсна початковій перестановці.

Рисунок 4.3 – Загальна схема DES

Праворуч на рисунку показаний спосіб, яким використовується 56-бітний ключ. Спочатку ключ подається на вхід функції перестановки.

Потім для кожного з 16 раундів підключ K_i є комбінацією лівого циклічного зрушення й перестановки. Функція перестановки та сама для кожного раунду, але підключи K_i для кожного раунду виходять різні внаслідок повторюваного зрушення біт ключа.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ захисту персональних даних у Cloud-системах яке зображено на рисунку 5.1. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи:

– Навігаційне меню: Налаштування; Встановлення значень кодів доступу; Довідка.

– Розділу обрання Cloud системи.

– Вікно введення даних ідентифікації.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Функціональних кнопок ПЗ.

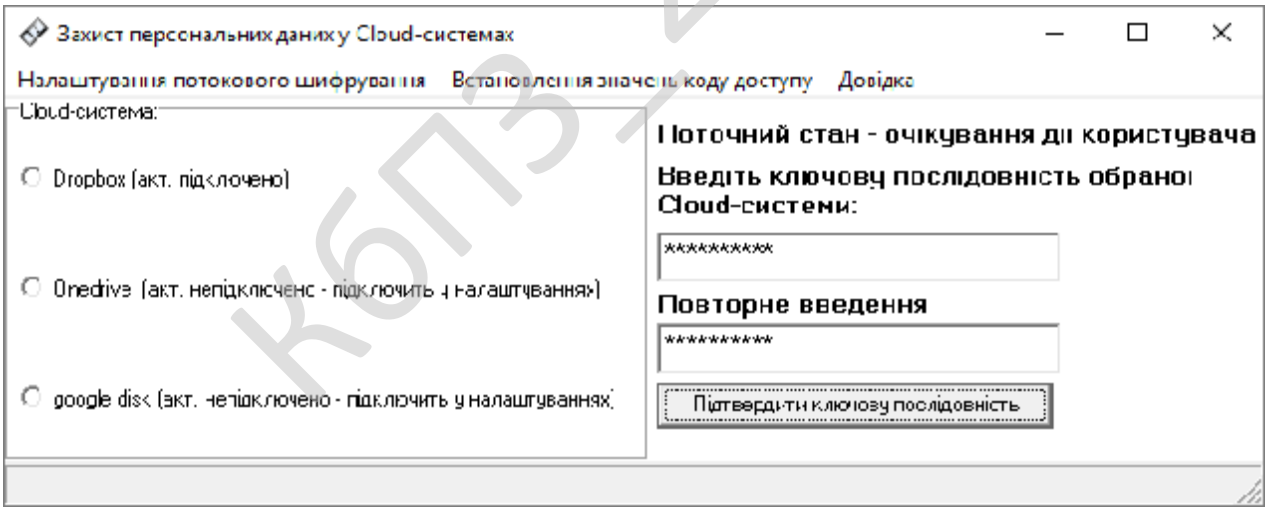

Рисунок 5.1 – Головне вікно ПЗ

Розроблена програма має дуже простий і інтуїтивно зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним

середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий.

Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

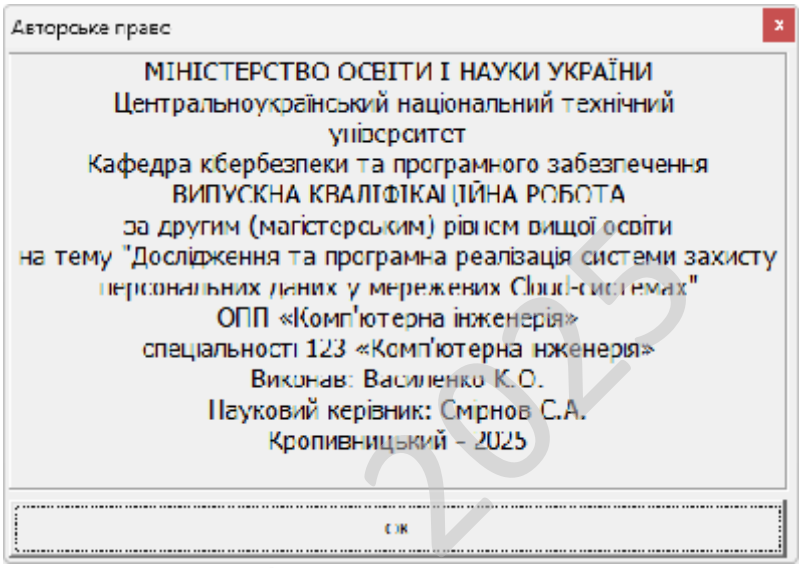

Рисунок 5.2 – Авторське право

Розглянемо процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом. Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення.

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

Тому, "розгортання" можна трактувати як загальний процес відповідно до певних вимог та характеристик. Розгортання може здійснюватись програмістом і в процесі розробки програмного забезпечення.

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

– Випуск.

– Встановлення та активація.

– Деактивація.

– Адаптація.

– Обновлення.

– Вмонтування.

– Відстежування версій.

– Видалення.

– Вилучення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

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

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

– Виконання робіт з загальної стандартизації існуючої діяльності організації, коли виділяються кращі практики виконання процедур і включаються в ІТ рішення за принципом найбільшої корисності для більшості учасників.

Відсоток таких процедур щодо загального обсягу автоматизації може бути невеликий, але це надає процесу побудови рішення вагу в організації за рахунок збільшення його необхідності.

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.

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 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи захисту персональних даних у мережевих Cloud-системах.

Метою розробки є дослідження та програмна реалізація системи захисту персональних даних у мережевих Cloud-системах.

Об'єктом дослідження є процес захисту персональних даних у мережевих Cloud-системах.

Предметом дослідження є методи захисту персональних даних у мережевих Cloud-системах.

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

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

– Удосконалено метод захисту персональних даних у мережевих Cloud-системах.

– Розроблено вітчизняний продукт захисту персональних даних у мережевих Cloud-системах, який має більш широкі можливості, на відміну від існуючих аналогів.

7 МАРКЕТИНГОВЕ ТА ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ ІТ-ПРОЄКТУ

7.1 Визначення цільової аудиторії кінцевого готового продукту

Результати цього дослідження насамперед можуть бути цікавими для компаній, які зберігають або обробляють персональні дані своїх клієнтів у хмарному середовищі. Сюди належать банки, медичні установи, інтернет-магазини, навчальні платформи – усі ті, хто має справу з конфіденційною інформацією. Для них надійна система захисту – це не лише вимога законодавства, а й питання репутації та довіри користувачів.

Також така розробка може зацікавити ІТ-компанії, що займаються хмарними сервісами або кібербезпекою. Для них результати дослідження можуть стати основою для створення нових продуктів чи вдосконалення існуючих рішень. Особливо актуально це для компаній, які працюють на міжнародному ринку, де вимоги до безпеки даних дедалі жорсткіші.

Викладачі та студенти ІТ-напрямів теж можуть знайти цю тему корисною, адже вона дає можливість глибше зрозуміти механізми захисту в Cloud-інфраструктурах і розібратися, як теоретичні алгоритми шифрування застосовуються на практиці. Такі знання сьогодні є надзвичайно затребуваними на ринку праці.

Крім того, дослідження може зацікавити державні структури, які регулюють політику кібербезпеки або займаються перевірками відповідності компаній стандартам захисту даних. Їм важливо бачити приклади ефективної реалізації таких систем у реальних умовах.

Загалом, тема захисту персональних даних у хмарних системах є універсально важливою, адже вона зачіпає кожного користувача інтернету. Тому

результати цієї роботи мають як наукову, так і прикладну цінність для багатьох сфер діяльності.

7.2 Оцінка привабливості шляхом застосування методів експертних оцінок

Оцінку привабливості системи захисту персональних даних у Cloud-середовищах можна провести за допомогою експертного аналізу. Для цього формується група спеціалістів – фахівці з інформаційної безпеки, представники IT-бізнесу, викладачі університетів і навіть юристи, які працюють з питаннями захисту даних. Їм пропонується оцінити систему за низкою критеріїв: технічна надійність, відповідність міжнародним стандартам, простота інтеграції у вже існуючу інфраструктуру та економічна доцільність впровадження.

Кожен експерт виставляє свої оцінки за шкалою, наприклад від 1 до 10, після чого розраховується середній бал для кожного критерію. На основі цих даних визначається загальний показник привабливості проєкту. Якщо він перевищує умовний поріг, наприклад 8 балів, можна вважати, що система є конкурентоспроможною і має реальні шанси на комерційне використання.

Важливо, що експертна оцінка включає не лише цифри, а й якісні коментарі. Фахівці можуть вказати, які елементи потребують доопрацювання, наприклад оптимізація алгоритмів шифрування, покращення зручності панелі адміністратора чи додавання системи сповіщень про підозрілі активності.

Зібрані дані потім аналізуються, і на основі висновків формується план удосконалення продукту. Таким чином, метод експертних оцінок дозволяє не лише визначити ринкову привабливість розробки, а й знайти напрямки для її покращення ще до виходу на ринок.

Такий підхід є особливо корисним для проєктів у сфері кібербезпеки, де навіть дрібна помилка може мати серйозні наслідки, а експертна думка допомагає запобігти ризикам на ранньому етапі.

7.3 Вибір методу оцінки вартості ПЗ

Для оцінки вартості реалізації системи захисту персональних даних у Cloud-середовищах найкраще підходить витратний метод. Це пояснюється тим, що створення подібних систем передбачає чітко визначені статті витрат: розробку програмного забезпечення, придбання ліцензій, налаштування серверів, оплату праці спеціалістів із кібербезпеки та постійне технічне обслуговування.

Витратний підхід дозволяє точно визначити, скільки коштів потрібно для створення мінімально життєздатного продукту (MVP), а також оцінити витрати на подальшу масштабизацію системи. Це дає змогу побудувати реалістичну фінансову модель і спланувати етапи розвитку проєкту.

Додатково можна врахувати елементи прибуткового підходу – тобто оцінити потенційний прибуток від впровадження системи. Наприклад, зменшення втрат від витоків даних або штрафів за порушення правил GDPR також можна трактувати як економічний ефект від інвестицій у захист.

Перевагою витратного методу є прозорість – кожну складову можна перевірити, обґрунтувати й при необхідності скоригувати. Це особливо важливо для проєктів, що можуть фінансуватися за рахунок грантів або інвесторів, яким потрібне чітке розуміння, куди саме підуть їхні кошти.

Таким чином, витратний підхід дає змогу об’єктивно оцінити ціну розробки системи, а також сформувати аргументовану базу для подальших переговорів з потенційними партнерами чи замовниками.

7.4 Розрахунок економічної ефективності від впровадження реалізованого ПЗ як фактору його привабливості

Підприємство використовує хмарну інфраструктуру для зберігання персональних даних клієнтів і співробітників. До впровадження системи захисту спостерігалися випадки несанкціонованого доступу, збої у збереженні даних, а

також витрати на відновлення інформації після кібератак. Нова система передбачає впровадження засобів шифрування, автентифікації користувачів, моніторингу активності та резервного копіювання. Вхідні дані та розрахунки зведемо до таблиці 7.1.

Таблиця 7.1 – Вхідні дані для розрахунків

Показник	**До впровадження**	**Після впровадження**	**Економічний ефект**
Кількість інцидентів безпеки на рік	5	1	–4
Середні втрати від одного інциденту (відновлення, репутаційні збитки, простої)	60 000 грн	10 000 грн	–50 000 грн
Річні витрати на аудит безпеки та підтримку	120 000 грн	80 000 грн	–40 000 грн
Витрати на навчання персоналу щодо роботи із системою	—	15 000 грн	–15 000 грн
Вартість розробки та впровадження системи	—	—	250 000 грн

Розрахунок економічного ефекту дає наступні дані: зменшення збитків від інцидентів безпеки – 240 000 грн/рік, економія на аудиті та технічному обслуговуванні – 40 000 грн/рік, витрати на навчання персоналу (одноразово) – 15 000 грн, разом економічний ефект (щорічно) – 280 000 грн/рік, термін окупності (Payback Period) – 0,89 року (~11 місяців), коефіцієнт економічної ефективності – 112%

Додаткові (немонетарні) вигоди: підвищення рівня довіри клієнтів і партнерів завдяки гарантії безпеки даних; зменшення репутаційних ризиків, пов'язаних із витоками інформації; відповідність міжнародним стандартам (gdpr, iso 27001); зростання продуктивності персоналу завдяки стабільній роботі cloud-систем; можливість масштабування рішення для інших відділів або філій.

Впровадження системи захисту персональних даних у мережевих Cloud-системах є економічно ефективним, оскільки дозволяє повністю окупити витрати протягом менше ніж одного року. Річна економія становить 280 000 грн, а коефіцієнт ефективності перевищує 100%, що свідчить про високу доцільність інвестицій. Крім фінансового ефекту, система забезпечує довгострокову вигоду у вигляді підвищення інформаційної безпеки, стабільності бізнес-процесів і зростання довіри користувачів до компанії.

7.5 Пропозиція алгоритму просування проєкту розробки ПЗ

Просування проєкту системи захисту персональних даних у хмарних середовищах варто почати з визначення цільової аудиторії. Основними споживачами будуть компанії, які працюють із великими обсягами персональної інформації, тому важливо донести до них ключову цінність – надійність і простоту інтеграції.

Першим кроком може бути створення демонстраційної версії системи, де потенційні користувачі зможуть перевірити її можливості на практиці. Це допоможе показати переваги рішення без технічних складнощів.

Наступним етапом є інформаційна кампанія – публікації в ІТ-медіа, участь у конференціях з кібербезпеки, створення кейсів і відеоогляді. Важливо, щоб просування базувалося не на загальних фразах, а на реальних прикладах, які демонструють ефективність системи у запобіганні витокам чи атакам.

Ефективним інструментом може стати співпраця з Cloud-провайдерами або консалтинговими компаніями, які можуть інтегрувати систему у свої сервіси. Таке партнерство розширить ринок і збільшить довіру до продукту.

Завершальним етапом має бути постійна підтримка користувачів і збір відгуків. Це дозволить не лише вдосконалювати продукт, а й формувати спільноту довкола бренду, що особливо важливо для рішень у сфері безпеки.

7.6 Оптимізація каналів збуту та шляхів реалізації ПЗ

Оптимізація каналів збуту системи захисту персональних даних має ґрунтуватися на поєднанні прямого продажу, партнерських відносин та онлайн-просування. Для великих підприємств доцільно створити індивідуальні пропозиції із демонстрацією безпосередніх вигод, наприклад – зниження ризику витоку даних чи спрощення відповідності міжнародним стандартам.

Для середнього бізнесу ефективним буде модель підписки (SaaS), яка дозволяє користувачам сплачувати лише за обсяг використання. Це знижує поріг входу і робить систему доступною для ширшого кола клієнтів.

Важливо забезпечити присутність продукту на платформах Cloud-сервісів і у маркетплейсах ІТ-рішень, щоб користувачі могли встановлювати його безпосередньо через знайоме середовище. Це значно спрощує процес придбання.

Додатково варто організовувати вебінари, навчальні відео та консультації для клієнтів, щоб допомогти їм краще зрозуміти цінність продукту. Підтримка після продажу також є критичною, адже у сфері кібербезпеки довіра формується через стабільність і професіоналізм.

Таким чином, оптимізовані канали збуту мають бути гнучкими, орієнтованими на різні типи клієнтів і підкріпленими високим рівнем сервісу, що створює довгострокову лояльність до продукту.

7.7 Визначення ключових факторів успіху конкретного проєкту

Ключовим фактором успіху системи захисту персональних даних у Cloud-середовищах є її надійність. Якщо продукт дійсно здатен гарантувати безпечне зберігання даних, навіть за умов кібератак, користувачі швидко почнуть йому довіряти.

Другим важливим чинником є простота інтеграції. Багато компаній уникають складних систем безпеки саме через їхню громіздкість, тому легке налаштування і зручний інтерфейс можуть стати вирішальною перевагою.

Не менш значущим є дотримання міжнародних стандартів – таких як GDPR або ISO 27001. Сертифікація і відповідність цим вимогам підвищує довіру і відкриває можливості для виходу на глобальний ринок.

Велике значення має і якість технічної підтримки. Постійна взаємодія з користувачами, швидке реагування на проблеми та оновлення системи допомагають зберігати позитивну репутацію.

І нарешті, успіх залежить від команди – її компетентності, здатності до інновацій і розуміння потреб ринку. Тільки поєднання технічної досконалості з людським підходом дозволить створити продукт, який стане не просто програмою, а справжнім гарантом цифрової безпеки.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Зі становленням та розвитком інформаційного суспільства спостерігається масове впровадження комп’ютерних технологій в усіх сферах життя і діяльності людини. Застосування персональних комп’ютерів і ЕОМ дозволило значно підвищити продуктивність праці, змінити характер і зміст праці.

Впровадження комп'ютерних технологій принципово змінило характер праці різних категорій фахівців. Програмісти у процесі роботи отримують негативний вплив на органи зору, а також мають значну розумову напругу і нервово-емоційне навантаження. Руки (м’язи рук та суглоби пальців) при роботі з клавіатурою мають теж істотне навантаження. До шкідливих факторів, які впливають на робітників галузі інформаційних технологій фахівці відносять високочастотні електромагнітні коливання під час роботи апаратної частини ЕОМ та виділення шкідливих газів [1, 2].

Ці шкідливі фактори можуть привести до професійних захворювань.

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

– недостатню площу і обсяг виробничого приміщення;

– недотримання вимог мікроклімату на робочих місцях;

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

– підвищений рівень низькочастотних магнітних полів від моніторів;

– порушення вимог організації робочих місць;

– недотримання вимог до режимів праці та відпочинку;

– надмірне виробниче навантаження працівників;

– відсутність навичок зниження впливу психоемоційної напруги.

Відповідно до ст. 14 Закону «Про охорону праці» [3] на роботодавця покладено обов'язок забезпечити: безпеку працівників при експлуатації устаткування; застосування засобів індивідуального захисту працівників; відповідні вимоги охорони праці, умов праці на кожному робочому місці; дотримання режиму праці та відпочинку працівників; навчання безпечним методам і прийомам виконання робіт; інструктаж з охорони праці; організацію контролю над станом умов праці на робочих місцях; проведення атестації робочих місць за умовами праці.

Максимально зменшити кількість шкідливих впливів на людину при високій продуктивності праці, створити комфортні умови для роботи людей – головна задача охорони праці [5].

8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

Електронно-обчислювальні машини (ЕОМ) та інше офісне обладнання є джерелами небезпеки ураження електричним струмом. Оскільки робота програміста характеризується істотним зоровим навантаженням, то вимагає належного освітлення. У приміщенні, в якому працюють ІТ-працівники, необхідно створити належний мікроклімат, параметри якого регламентуються, Державними санітарними правилами і нормами, зокрема ДСанПіН 3.3.2.007-98 [2].

Шкідливими факторами при роботі з персональним комп'ютером є іонізуюче випромінювання промислової частоти, підвищене нервово-емоційне навантаження на оператора, підвищене навантаження на органи зору та дрібні стереостатичні рухи кінцівок.

Ці фактори можуть викликати у працівника певні розлади здоров'я, зокрема підвищення артеріального тиску, кон'юктивіти, тендовагініти ті інші захворювання.

Комп’ютер, як і будь-який електричний прилад, особливо при його неправильному підключенні, може бути джерелом ураження оператора електричним струмом. Саме тому всі працівники, які працюють з персональним комп’ютером, повинні мати першу (або другу) групу допуску з електробезпеки.

Через наявність зазначених факторів працівники, які працюють з персональними комп’ютерами, підлягають попередньому та періодичному медичному огляду згідно з пунктом 6.2.3 додатку 4 до наказу Міністерства охорони здоров’я України “Про затвердження Порядку проведення медичних оглядів працівників певних категорій” від 21 травня 2007 року №246 [8].

8.3 Аналіз умов праці

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

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

Таблиця 8.1 – Перелік шкідливих та небезпечних виробничих чинників

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

За категорією вибухо– і пожежонебезпеки дане приміщення відноситься до категорії В – пожежонебезпечне, тому що присутні тверді матеріали, що горять, такі як дерев'яні столи, папір та інше. Виходячи з категорії

пожежонебезпеки і поверховості будинку, ступінь вогнестійкості будівлі II. Згідно з ДБН В 1.1-7-2016 «Пожежна безпека об'єктів будівництва» [5] ЕОМ повинні розташовуватись в будівлі не менше ніж II ступню вогнестійкості.

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

Для забезпечення оптимальних мікрокліматичних умов у приміщенні передбачена система опалення (загальне парове) в холодний період, та вентиляція і кондиціонування в теплий період року згідно ДБН2.5–67–2013 «Опалення, вентиляція та кондиціонування» [6]. При виконанні замірів параметрів мікроклімату, значення їх відповідали оптимальним та допустимим параметрам відповідно до ДСанПіН3.3.2.007 – 98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин».

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

Нормування освітлення здійснюється відповідно до ДБН В.2.5 – 28 – 2006 «Природне та штучне освітлення». [7]

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

Одним з найбільш поширеніших чинників зовнішнього середовища, який несприятливо впливає на людину, є шум. Вплив шуму на організм людини залежить від рівня звукового тиску, частотних характеристик, тривалості дії, а також індивідуальних особливостей людини.

При тривалій дії шуму у оператора ЕОМ виявляються симптоми утомленості, нервового збудження, що сприяють погіршенню працездатності і допущенні помилок при роботі. Для уникнення шкідливої дії шуму на організм

працюючого, необхідне дотримання нормованих параметрів, які не повинні перевищувати допустимих величин. При роботі на комп'ютері рівень шуму не повинен перевищувати 50 дБА. Приміщення розташоване вікнами у двір і знаходиться далеко від проїжджої частини вулиці. Основними джерелами шуму в приміщенні є устаткування і люди. Розглянута кімната не призначена для прийому відвідувачів і тому в ній не спостерігається великого скупчення людей. Тому основним джерелом шуму є комп'ютерна техніка.

Джерелами шуму при роботі ЕОМ є механічні частини принтера, що рухаються, і вентилятори(Lпк = 35 дБА, , Lprn = 48 дБА.) При роботі вентиляційної системи, що забезпечує оптимальний температурний режим електронних блоків ЕОМ і вмонтована в задню панель, створюється аеродинамічний шум. Шум, створюваний працюючим комп'ютером, може бути охарактеризований як широко смуговий постійний з аперіодичним посиленням при роботі принтера. Час роботи ПЕОМ – 6 – 8 год. за добу; принтери працюють не більш 1,5-2 год. за добу.

При наявності великої кількості джерел шуму еквівалентне значення шуму LЭКВ, дБА розраховують по наступній формулі:

$$L_{ЭКВ} = 10 \cdot \lg\left(\frac{1}{T}\sum_{i=1}^{n}\left(t_i \cdot 10^{0.1 L_i}\right)\right) \quad (8.1)$$

де

L_i – рівень шуму і-го джерела (пристрою),

ti – час роботи і-го джерела (пристрою),

T – загальний час роботи,

n – кількість джерел шуму даного типу;

Для даного приміщення необхідні змінні складають:

Загальний час роботи – робочий день, тобто T=8 годин.

Для фонового шуму (вентиляторів):

L1 = 35 дБА, T1 = 8 годин, n1=15 (5x3);

Для лазерного принтера Lexmark Jet:

L2 = 48дБА, T2 = 2 години, n2=1, для сканера L3 = 46дБА, T3 = 2 години.

Підставляємо отримані величини у формулу (8.1):

$$L_{экв} = 10 \cdot \lg\left(\frac{1}{8} \cdot \left(15 \cdot 8 \cdot 10^{0.1 \cdot 35} + 1 \cdot 2 \cdot 10^{0.1 \cdot 48} + 1 \cdot 2 \cdot 10^{0.1 \cdot 46}\right)\right) = 46{,}3 \text{ дБА}$$

Таким чином, еквівалентний рівень шуму в приміщенні за робочий день $L_{экв}$ = 46,3 дБА, що не перевищує норму 50 дБА.

8.4 Техніка безпеки та протипожежна профілактика

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

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

Робота обслуговуючого персоналу полягає в інсталяції необхідного програмного забезпечення й наступному його використанні в діалоговому режимі роботи з ЕОМ. Іноді може виникати необхідність написання допоміжних програм для поліпшення роботи вузла або для зниження витрат. З погляду забезпечення умов праці й вимог техніки безпеки для роботи програміста необхідно наступне: достатнє освітлення екрану дисплея й робочого місця; повна технічна справність устаткування, його електробезпечність; достатня пожежобезпечність приміщення; оптимальний мікроклімат, що сприяє продуктивній роботі; відповідність робочого місця вимогам ергономіки. До небезпечних і шкідливих факторів, дії яких піддається програміст, можна віднести: можливість поразки

електричним струмом, при несправності устаткування, порушенні заземлення або техніки безпеки; робота в мікрокліматі з неприпустимими параметрами; робота при недостатній освітленості екрану дисплея й робочого місця.

Відповідно НПАОП 40.1-1.21-98 “Правил безпечної експлуатації електроустановок споживачів” [8], приміщення можна віднести до приміщень без підвищеної небезпеки, оскільки це приміщення, сухе, з нормальною температурою й ізолюючими підлогами, що не має заземлених металоконструкцій.

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

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

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

8.5 Розрахункова частина

Запорукою безпечної роботи є виконання вимог електричної безпеки, оскільки все офісне обладнання заживлюється від електричної мережі. Одним з необхідних засобів електричної безпеки є встановлення захисного заземлення.

Початкові дані, необхідні для розрахунку захисного заземлення:

– допустимий опір розповсюдженню струму в землі від заземлю-вального пристрою $R_{зн}$ = 10 Ом;

– питомий опір грунту в місці встановлення заземлювача $\rho_з$ = 100 Ом/м;

– тип ґрунту – суглинок;

– тип заземлювача – труба, діаметром d=0.06 м і довжиною l =1.7 м;

– конструкція заземлювача – розташування заземлювачів по контуру. Розрахунок проводимо за стандартною методикою [7].

Визначимо розрахунковий опір землі:

$$\rho_{рз} = \phi \cdot \rho_3$$

де ϕ – коефіцієнт сезонності, що враховує коливання питомого опору при зміні вологості ґрунту протягом року; при використанні заземлювача довжиною l = 1.7 м при глибині закладання від вершини h = 0.65 м ϕ = 1.1 для четвертої кліматичної зони.

Схема розташування заземлювачів показана на рисунку 8.1.

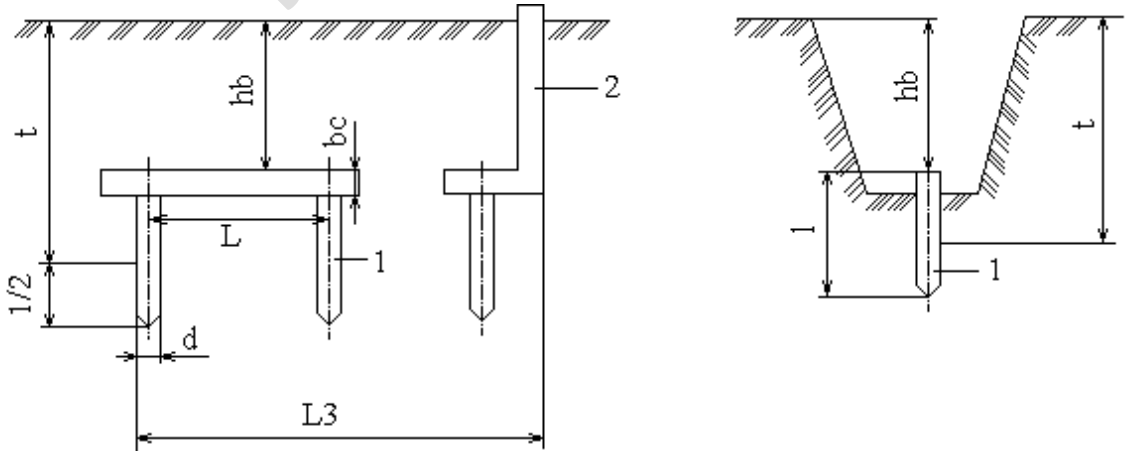

Рисунок 8.1 – Схема розташування заземлювачів

Опір землі:

$$\rho_{рз} = 1{,}1 \cdot 100 = 110 \text{ Ом·м}$$

Опір Rв, розповсюдженню струму в землі від одного вертикального заземлювача:

$$R_B = \frac{\rho_{рз}}{2\pi \cdot l}\left(\ln\frac{2 \cdot l}{d} + 0.5\ln\frac{4t + l}{4t - l}\right)$$

де

l – довжина заземлювача (l = 1.7 м);

d = 0.06 м – діаметр заземлювача при U<1 кВ та при S<100 кВА;

t – відстань від поверхні до середини заземлювача:

$$t = h + l/2 = 0.65 + 1.7/2 = 1.5 \text{ м.}$$

$$R_в = \frac{110}{2 \cdot 3.14 \cdot 1.7}\left(\ln\left(\frac{2 \cdot 1.7}{0.06}\right) + 0.5 \cdot \ln\left(\frac{4 \cdot 1.5 + 1.7}{4 \cdot 1.5 - 1.7}\right)\right) = 45.17 \text{ Ом}$$

Визначаємо потрібну кількість заземлювачів:

$$n' = \frac{R_в}{R_{зн}} = \frac{45.17}{10} = 4.5 \approx 5 \text{ шт.}$$

Коефіцієнт використання вертикальних заземлювачів враховує ефект екранування. При вибраному значенні k = a/l, де a – відстань між вертикальними заземлювачами, м; k = 1 при a = 2.4 м.

Таким чином, коефіцієнт використання вертикального заземлювача за довідковими даними дорівнює $\eta_В = 0{,}6$.

Кількість вертикальних заземлювачів з урахуванням коефіцієнту використання $\eta_В$ приблизно складає

$$n = \frac{R_в}{R_{зн} \cdot \eta_в} = \frac{45.17}{10 \cdot 0.6} = 7.53 \approx 8 \text{ шт.}$$

Довжина горизонтального заземлювача, необхідна для розміщення вертикальних заземлювачів по контуру

$$L = a \cdot n = 2.4 \cdot 8 = 19.2 \text{ м}$$

Опір горизонтального заземлювача $R_Г$, Ом, прокладеного на глибині h =

0.65 м від поверхні землі буде

$$R_г = \frac{R_{рз}}{2 \cdot 3.14 \cdot L} \cdot \ln \frac{2 \cdot L^2}{b \cdot h} = \frac{110}{2 \cdot 3.14 \cdot 19.2} \cdot \ln \frac{2 \cdot 19.2^2}{0.06 \cdot 0.65} = 10.61 \text{ Ом}$$

де b = 0.04 м – ширина сталевої смуги, з якої виготовлений заземлювач.

Обчислюємо загальний опір:

$$R_з = \frac{R_в \cdot R_г}{n \cdot R_г \cdot \eta_в + R_в \cdot \eta_г} = \frac{45.17 \cdot 10.61}{6 \cdot 10.61 \cdot 0.6 + 45.17 \cdot 0.34} = 8.33 \text{ Ом}$$

де $\eta_Г$ – коефіцієнт використання горизонтального заземлювача ($\eta_Г = 0.34$).

Маємо 8.33 < 10 Ом (за потужності генераторів та трансформаторів 100 кВт і менше), отже нормативне обмеження $R_з < R_{з.норм}$ виконується.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи захисту персональних даних у мережевих Cloud-системах.

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

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів захисту персональних даних у мережевих Cloud-системах.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем захисту персональних даних у мережевих Cloud-системах.

– Досліджена система захисту персональних даних у мережевих Cloud-системах.

– На основі отриманих результатів досліджень створена програмна реалізація системи захисту персональних даних у мережевих Cloud-системах.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання захисту персональних даних у мережевих Cloud-системах.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

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

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

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

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм DES.

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.

Проведено маркетингове та економічне обґрунтування ІТ-проєкту, що дозволило визначити ключові фактори успіху даного проекту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Василенко К.О. Дослідження та програмна реалізація системи захисту персональних даних у мережевих Cloud-системах // Збірник праць молодих науковців ЦНТУ. – Вип. 15. – Кропивницький: ЦНТУ, 2025.

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

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

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

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

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

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

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

9. Петрик В.М., Присяжнюк М.М., Аль-Файюмі Халед та ін. «Системи інформаційної зброї та технології інформаційної війни»: підручник / Петрик В.М., Присяжнюк М.М., Аль-Файюмі Халед, Жарков Я.М., Смірнов О.А, Буравченко К.О., Давидюк А.В., Кононович В.Г., Корчинский В.В., Кудирко В.М., Фесенко А.О.; за заг. ред. В.М. Петрика, М.М. Присяжнюка.– К.: Видавничий центр “Кафедра”, 2025.– 320 с.

10. Усік, П.С., Смірнова, Т.В., Буравченко, К.О., Смірнов, О.А., Улічев, О.С., Смірнов, С.А. «Дослідження технологій забезпечення кібербезпеки банківських систем з використанням штучного інтелекту». *Кібербезпека: освіта, наука, техніка.* 2025. Том 1 № 29. С.704–716, 2025

11. Kuznetsov, O., Frontoni, E., Kuznetsova, K., Arnesano, M., Smirnov, O.

«A secure biometric authentication architecture for blockchain-driven cyber-physical systems». *Security and Privacy of Cyber Physical Systems Emerging Trends Technologies and Applications*, 2025, pp. 193–224.

12. Kuznetsov, O., Atzeni, G., Arnesano, M., Randieri, C., Smirnov, O. «Secure IoT-based smart wheelchair system: From implementation to security enhancement strategy». *Security and Privacy of Cyber Physical Systems Emerging Trends Technologies and Applications*, 2025, pp. 225–257.

13. Kuznetsov, O., Smirnov, O., Kuznetsova, T., Shaikhanova, A., Svatowsky, I. «Privacy-utility trade-offs in IoT networks: A comparative analysis of differential privacy mechanisms for sensor data aggregation». *Security and Privacy of Cyber Physical Systems Emerging Trends Technologies and Applications*, 2025, pp. 589–622.

14. 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.

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

16. 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.

17. 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.

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

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

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

21. Батрак О., Смірнова Т., Гнатюк В., Одарченко Р., Смірнов О. «Дослідження показників ефективності функціонування та перспектив розвитку систем IP-телефонії». *Підводні технології,* 2024, № 13, с. 28-35.

22. Kuznetsov, O., Kryvinska, N., Ilchenko, O., Smirnova, T., Ulianovska, Y. «Comparative Analysis of Cryptocurrency Trading Platforms Using the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, 2023, 3628, pp. 106-115.

23. Akhalaia, G., Iavich, M., Iashvili, G., Prysiazhnyy, D., Smirnova, T. «Secure Encrypted Connection on Georgian Website». *CEUR Workshop Proceedings,* 2023, 3550, pp. 313-320.

24. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56

25. 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.

26. 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.

27. 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.

28. Смірнова Т.В., Гнатюк С.О., Бердибаєв Р.Ш., Сидоренко В.М., Жигаревич О.К., «Система корелювання подій та управління інцидентами кібербезпеки на об'єктах критичної інфраструктури». *Кібербезпека: освіта, наука, техніка*, №*3*(19), 2023, С. 176-196.

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

30. Козлов Я.О., Смірнова Т.В., Смірнов О.А. «Дослідження SIEM-систем для забезпечення кібербезпеки». *VII міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології" до 30-ти річчя кафедри кібербезпеки та програмного забезпечення,* м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 26.

31. Козлов Я.О., Козірова Н.Л., Смірнов О.А. «Дослідження структури та принципу роботи SIEM-системи». *VII міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології" до 30-ти річчя кафедри кібербезпеки та програмного забезпечення,* м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 59.

32. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А., Коваленко А.С. «Дослідження нормативних документів та галузевих стандартів

розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». *Системи управління, навігації та зв'язку*, 2023, вип. 2(72), С. 170-178.

33. 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,

34. 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.

35. Смірнова Т.В., Гнатюк С.О., Сидоренко В.М., Юдін О.Ю., Сидоренко С.Ю., «Модель визначення критичності галузевих інформаційно-телекомунікаційних систем». *Проблеми інформатизації та управління,* № 2(70). 2022. С. 28-37.

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

37. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»,* № 2 (307). С. 46-52. 2022.

38. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 1(67). С. 84-89.

39. 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

40. 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.

41. 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.

42. 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.*

43. 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.

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

45. 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.

46. 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.

47. 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.

48. 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.

49. 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.

50. 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.

51. 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.

52. 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.