Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Підвищення захищеності DRM-системи управління доступом динамічно змінюваними USB-ключами безпеки на основі вдосконаленого алгоритму HOTP (Hashed-Based One-Time Password) та системи блокування доступу»

Виконав: ст. 2-го курсу, групи КІТС-23м
спеціальності 125– Кібербезпека та захист інформації
Освітня програма – Кібербезпека інформаційних технологій та систем
(шифр і назва напряму підготовки, спеціальності)

Пуздрановський І.В.
(прізвище та ініціали)

Керівник: д. ф., доц. каф. МБІС
Салієва О.В.
(прізвище та ініціали)
«____» _____ 2024 р.

Опонент: к.т.н., доцент, каф. ОТ
Богомолов С. В.
(прізвище та ініціали)
«____» _____ 2024 р.

Допущено до захисту
Голова секції УБ кафедри МБІС

_____Юрій ЯРЕМЧУК
"09" грудня 2024 р.

Вінниця ВНТУ – 2024 рік

Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

Рівень вищої освіти II-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 125 – Кібербезпека та захист інформації
Освітньо-професійна програма – Кібербезпека інформаційних технологій та систем

ЗАТВЕРДЖУЮ
Голова секції УБ/ кафедра МБІС

_____ **Юрій ЯРЕМЧУК**
"20" вересня 2024 р.

З А В Д А Н Н Я
на магістерську кваліфікаційну роботу студенту
Пуздрановському І.В.
(прізвище, ім`я, по-батькові)

1. Тема роботи:

«Підвищення захищеності DRM-системи управління доступом динамічно змінюваними USB-ключами безпеки на основі вдосконаленого алгоритму HOTP (Hashed-Based One-Time Password) та системи блокування доступу»
Керівник роботи: д. ф., доц. каф. МБІС Салієва О.В.
(прізвище, ім`я, по-батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "17" вересня 2024 року № 310
2. Строк подання студентом роботи за тиждень до захисту.
3. Вихідні дані до роботи:
Стандарти, електронні джерела, підручники та наукові статті по темі, які стосуються теми магістерської кваліфікаційної роботи.
4. Зміст текстової частини:
У вступі роботи обґрунтовано актуальність дослідження, визначено мету, задачі, об'єкт, предмет, новизну та практичну цінність роботи. У першому розділі проаналізовано теоретичні засади управління доступом, особливості генерації динамічно змінюваних ключів та сучасні методи захисту інформаційних ресурсів. Другий розділ присвячено створенню захищеної системи управління доступом, включаючи розробку алгоритмів динамічних ключів і системи блокування доступу. У третьому розділі реалізовано програмну частину системи, обґрунтовано вибір технологій, проведено тестування та аналіз результатів. У висновках підсумовано основні досягнення роботи та окреслено практичну значущість отриманих результатів.

5. Перелік ілюстративного матеріалу (з точним зазначенням обов`язкових креслень)
У другому розділі магістерської кваліфікаційної роботи наведено 2 рисунка, у третьому розділі – 6 рисунків.
6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Основна частина			
I	Салієва О.В. д. ф., доц. каф. МБІС		
II	Салієва О.В. д. ф., доц. каф. МБІС		
III	Салієва О.В. д. ф., доц. каф. МБІС		
Економічна частина			
IV	Бурєннікова Н. В. д.е.н., професор каф. ЕПВМ		

7. Дата видачі завдання 20 вересня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи		Примітка
1.	Визначення напрямку магістерської роботи, формулювання теми	20.09.2024	31.09.2024	
2.	Аналіз предметної області обраної теми	01.10.2024	15.10.2024	
3.	Розробка роботи	16.10.2024	26.10.2024	
4.	Написання магістерської роботи на основі розробленої теми	27.10.2024	15.11.2024	
5.	Передзахист магістерської кваліфікаційної роботи	16.11.2024	24.11.2024	
6.	Виправлення, уточнення, корегування магістерської кваліфікаційної роботи	27.11.2024	04.12.2024	
7.	Захист магістерської кваліфікаційної роботи	10.12.2024	10.12.2024	

Студент _____ (підпис) Пуздрановський І.В.

Керівник роботи _____ (підпис) Салієва О.В.

АНОТАЦІЯ

УДК 004.56.5

Пуздрановський І.В. Магістерська кваліфікаційна робота зі спеціальності 125 – «Кібербезпека та захист інформації», освітня програма «Кібербезпека інформаційних технологій та систем». Вінниця: ВНТУ, 2024. – 117 с.

Укр. мовою. Бібліогр.: 41 назв; рис.: 8; табл. 9.

Ключові слова: кібербезпека, DRM-система, USB-ключі, HOTP.

У даній роботі розглянуто питання підвищення захищеності DRM-систем управління доступом за допомогою динамічно змінюваних USB-ключів безпеки та вдосконаленого алгоритму HOTP (Hashed-Based One-Time Password). Проведено аналіз сучасних методів захисту інформаційних ресурсів, охарактеризовано особливості DRM-систем та визначено ключові проблеми, пов’язані з автентифікацією і управлінням доступом.

Розроблено алгоритми динамічно змінюваних ключів безпеки та адаптивної системи блокування доступу, що забезпечують виявлення та протидію несанкціонованим спробам доступу. На основі запропонованих підходів реалізовано програмну DRM-систему, що включає модулі генерації одноразових паролів, синхронізацію з USB-ключами, механізм блокування доступу і багатофакторну перевірку для розблокування облікових записів. Для зручності користувачів створено інтерактивний інтерфейс, який дозволяє ефективно взаємодіяти із системою.

Проведене тестування підтвердило працездатність реалізованого рішення, його відповідність сучасним вимогам безпеки та здатність витримувати високе навантаження. Результати роботи можуть бути використані для вдосконалення існуючих DRM-рішень або створення нових систем із підвищеним рівнем захисту.

ABSTRACT

Puzdranovskyi I.V. Master's qualification thesis in specialty 125 – "Cybersecurity and Information Protection," educational program "Cybersecurity of Information Technologies and Systems." Vinnytsia: VNTU, 2024. – 117 pages. In Ukrainian. References: 41 titles; figures:10; tables: 9. Keywords: cybersecurity, DRM-system, USB keys, HOTP.

This work addresses the issue of enhancing the security of DRM (Digital Rights Management) access control systems using dynamically changing USB security keys and an improved HOTP (Hashed-Based One-Time Password) algorithm. The study includes an analysis of modern methods for protecting information resources, an examination of the features of DRM systems, and the identification of key issues related to authentication and access management.

Algorithms for dynamically changing security keys and an adaptive access blocking system have been developed, ensuring the detection and prevention of unauthorized access attempts. Based on the proposed approaches, a software DRM system has been implemented, which includes modules for generating one-time passwords, synchronization with USB keys, an access blocking mechanism, and multifactor authentication for unlocking accounts. An interactive user interface has been created to facilitate efficient interaction with the system.

The conducted testing confirmed the functionality of the implemented solution, its compliance with modern security requirements, and its ability to handle high loads. The results of this work can be used to improve existing DRM solutions or to develop new systems with enhanced security levels.

ЗМІСТ

ВСТУП

Захист інформації та управління доступом у цифрових системах набуває все більшої важливості в умовах стрімкого розвитку інформаційних технологій та зростання кількості кібератак. Однією з критичних сфер, яка потребує надійного захисту, є системи управління цифровими правами (DRM). Такі системи використовуються для обмеження доступу до контенту, забезпечуючи дотримання авторських прав і захист інтелектуальної власності. Проте сучасні DRM-рішення часто стикаються з низкою викликів, пов'язаних із компрометацією ключів безпеки, обходом механізмів автентифікації та іншими загрозами.

Актуальність. Розробка вдосконаленої DRM-системи, яка інтегрує динамічно змінювані USB-ключі безпеки, алгоритми генерації одноразових паролів (HOTP) та систему блокування доступу, є актуальним завданням, що відповідає сучасним вимогам до інформаційної безпеки. Впровадження таких технологій дозволяє мінімізувати ризики несанкціонованого доступу, підвищити рівень захищеності ключів безпеки та забезпечити ефективне управління цифровими правами.

Мета і задачі дослідження. Метою роботи є розробка DRM-системи управління доступом із підвищеним рівнем захисту, яка використовує динамічно змінювані USB-ключі безпеки, вдосконалений алгоритм HOTP та систему блокування доступу для запобігання несанкціонованому використанню.

Задачами дослідження є:

- аналіз сучасних методів захисту інформаційних ресурсів у DRM-системах;
- вивчення особливостей використання USB-ключів безпеки та алгоритмів генерації одноразових паролів;
- розробка алгоритму роботи динамічно змінюваних USB-ключів на основі HOTP;
- створення адаптивної системи блокування доступу для протидії загрозам безпеці;
- програмна реалізація інтегрованої DRM-системи;

- тестування реалізованої системи для оцінки її ефективності та відповідності вимогам.

Об’єкт дослідження – система управління цифровими правами (DRM), орієнтована на захист контенту та управління доступом.

Предмет дослідження – методи підвищення захищеності DRM-систем управління доступом за допомогою динамічно змінюваних USB-ключів безпеки, алгоритму HOTP і адаптивної системи блокування доступу.

Наукова новизна: розроблено інтегроване рішення, що поєднує використання динамічно змінюваних USB-ключів із вдосконаленим алгоритмом HOTP, забезпечуючи підвищену захищеність DRM-системи. Запропоновано адаптивну систему блокування доступу, яка реагує на підозрілу активність користувачів, запобігаючи компрометації ключів і захищаючи облікові записи.

Практична цінність: результати роботи можуть бути використані для вдосконалення існуючих DRM-рішень або створення нових систем із підвищеним рівнем безпеки. Реалізована система забезпечує надійну автентифікацію, динамічне оновлення ключів та захист від атак на облікові записи, що є важливим для компаній, які працюють з цифровим контентом, інтелектуальною власністю або конфіденційними даними.

Дослідження та розробка системи спрямовані на підвищення загального рівня інформаційної безпеки в DRM-системах, що робить цю роботу актуальною та практично значущою в умовах сучасних загроз.

Апробація: тези доповідей представленні на Міжнародній науково-практичній Інтернет-конференції студентів, аспірантів та молодих науковців «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)» на тему «Аналіз існуючих алгоритмів генерації динамічно змінюваних ключів» [11].

1 ТЕОРЕТИЧНІ ЗАСАДИ ПІДВИЩЕННЯ ЗАХИЩЕНОСТІ DRM-СИСТЕМИ УПРАВЛІННЯ ДОСТУПОМ

У цьому розділі буде розглянуто теоретичні засади підвищення захищеності DRM-систем управління доступом. Проаналізовано основні принципи управління доступом, охарактеризовано DRM-системи, досліджено методи генерації динамічно змінюваних ключів, а також вивчено сучасні підходи до захисту інформаційних ресурсів за допомогою USB-ключів і систем блокування доступу. На основі проведеного аналізу будуть сформульовані задачі для подальшої розробки.

1.1 Важливість, особливості та основні принципи управління доступом

Управління доступом є однією з ключових складових кібербезпеки, оскільки воно визначає, хто і в якому обсязі може отримати доступ до систем, даних або інших захищених ресурсів. Система управління доступом гарантує, що лише авторизовані користувачі мають можливість виконувати певні дії, забезпечуючи тим самим цілісність, конфіденційність та доступність даних. Це особливо важливо для DRM-систем (Digital Rights Management), які зберігають цифровий контент та запобігають його несанкціонованому використанню [1].

Несанкціонований доступ до даних і систем є однією з найбільших загроз у сфері кібербезпеки. Зловмисники можуть використовувати вразливості системи, щоб отримати доступ до конфіденційних даних, наприклад, до фінансової інформації, персональних даних або інтелектуальної власності. Управління доступом дозволяє контролювати доступ до даних, обмежуючи його лише тими користувачами, які мають відповідні права.

Захищаючи систему за допомогою управління доступом, компанії зменшують ймовірність несанкціонованих дій, таких як копіювання, зміна або знищення важливої інформації, що допомагає уникнути значних збитків.

Конфіденційність та приватність є критичними аспектами для організацій, особливо тих, що працюють з персональними даними користувачів (наприклад, у

сфері охорони здоров'я, фінансів, електронної комерції). Управління доступом гарантує, що конфіденційна інформація буде доступною тільки уповноваженим користувачам, захищаючи її від розголошення або зловживання [1].

У контексті нормативно-правових актів, таких як GDPR (General Data Protection Regulation) у ЄС або HIPAA (Health Insurance Portability and Accountability Act) у США, управління доступом є обов’язковою вимогою. Це допомагає організаціям забезпечити відповідність стандартам захисту даних та уникнути серйозних фінансових і репутаційних втрат.

Незалежно від галузі, організації стикаються з різними ризиками через несанкціонований доступ до критичних даних або ресурсів. Ефективне управління доступом допомагає контролювати цей ризик, забезпечуючи лише певним користувачам відповідні рівні доступу, що дозволяє мінімізувати потенційні загрози для бізнесу.

Наприклад, якщо зловмисники отримають доступ до бази даних клієнтів, компанія може втратити довіру клієнтів та зазнати фінансових збитків. Управління доступом дозволяє знизити ці ризики, забезпечуючи контроль за тим, хто може бачити, змінювати або зберігати дані.

Управління доступом допомагає організаціям забезпечити відповідність внутрішнім політикам безпеки та міжнародним стандартам. Наприклад, у стандартах ISO/IEC 27001 визначені вимоги щодо управління доступом, що підвищує загальний рівень інформаційної безпеки в організації [2].

Завдяки впровадженню таких політик організація може дотримуватися встановлених правил доступу та забезпечити їх прозорість і контроль. Це особливо важливо для великих компаній, де дані можуть бути доступні тисячам користувачів, і відсутність управління доступом може призвести до випадкового або зловмисного витоку інформації.

Цілісність даних є важливим аспектом, який забезпечує управління доступом. Без надійного управління доступом користувачі можуть випадково або навмисно змінювати або видаляти важливі дані, що призводить до збоїв у роботі системи або навіть її повного порушення.

Система управління доступом допомагає запобігти подібним проблемам, надаючи права на редагування або видалення лише користувачам із відповідними повноваженнями. Це також підвищує надійність системи, оскільки дані залишаються захищеними від випадкових або зловмисних змін.

Захист даних є одним з найважливіших факторів, що впливають на довіру клієнтів і користувачів. Коли компанія має надійне управління доступом, клієнти можуть бути впевнені, що їхні дані захищені і доступні лише тим, хто має на це дозвіл.

Компанії, які приділяють особливу увагу захисту даних, можуть підвищити свою репутацію на ринку і завоювати довіру клієнтів, що є важливою конкурентною перевагою [2].

У великих організаціях із сотнями або навіть тисячами співробітників ефективне управління доступом є ключовим для оптимізації робочих процесів. Встановлення ролей і привілеїв дозволяє ефективно управляти доступом до системи і спрощує процес адміністрування.

Наприклад, за допомогою управління ролями система може автоматично призначати права новим користувачам або обмежувати доступ при зміні посадових обов'язків, що значно полегшує підтримку безпеки і дозволяє уникнути людських помилок.

Для компаній, що займаються розробкою програмного забезпечення, медіа-контенту або іншої інтелектуальної власності, управління доступом є критично важливим інструментом захисту від несанкціонованого копіювання або поширення. DRM-системи застосовують управління доступом, щоб обмежити доступ до продукту, захистити його від піратства та незаконного розповсюдження. Це сприяє збереженню доходів компаній та захисту їхніх авторських прав [3].

Управління доступом є незамінним механізмом у сучасних умовах цифровізації, де великі обсяги даних потребують захисту. Завдяки правильно налаштованій системі управління доступом організації можуть запобігти несанкціонованому доступу, забезпечити конфіденційність і цілісність даних, а також підтримувати відповідність стандартам безпеки та корпоративним

політикам. Це не лише знижує ризики кіберзагроз, але й зміцнює репутацію компанії на ринку, підвищуючи довіру клієнтів.

Управління доступом є складним і багатокомпонентним процесом, який включає різні методи, інструменти та підходи для забезпечення захисту даних, систем та ресурсів. Це один із ключових аспектів кібербезпеки, що дозволяє організаціям контролювати, хто, коли та в якому обсязі може отримати доступ до їхніх ресурсів. Особливості управління доступом розглядають такі компоненти, як багатофакторна аутентифікація, гнучкість, централізоване управління, поділ ролей, єдиний вхід та інші, які допомагають забезпечити високий рівень безпеки та ефективність контролю доступу [3].

Мультифакторна аутентифікація передбачає використання більше ніж одного фактора перевірки для підтвердження особи користувача, що значно підвищує рівень безпеки. MFA поєднує кілька типів аутентифікації:

– що користувач знає: пароль або PIN-код;

– що користувач має: USB-ключ, одноразовий пароль (OTP), який генерується додатком;

– що користувач є: біометричні дані, такі як відбитки пальців або розпізнавання обличчя.

Особливості: MFA знижує ризик компрометації облікових даних, оскільки для доступу до системи потрібно більше, ніж один фактор. Навіть якщо пароль користувача буде викрадений, без фізичного доступу до другого фактору (наприклад, USB-ключа чи мобільного додатка) зловмисник не зможе отримати доступ [4].

Сучасні системи управління доступом повинні бути гнучкими, щоб легко адаптуватися до змін в організаційних процесах, умовах роботи або нових загроз.

Особливості:

– динамічна зміна прав доступу. Користувачам можуть автоматично надаватися або забиратися певні права доступу залежно від їхньої ролі, місцезнаходження чи інших параметрів;

– адаптивна аутентифікація. В умовах підвищеної загрози система може посилити вимоги до аутентифікації, наприклад, запросивши додатковий рівень перевірки особи, якщо користувач входить із нового пристрою або незвичної локації;

– гнучкі політики доступу. Система може застосовувати різні політики для різних груп користувачів чи типів даних, що дозволяє адаптувати управління доступом до специфічних вимог організації або підрозділів.

Централізоване управління дозволяє адміністратору системи ефективно контролювати доступ до всіх ресурсів організації з єдиної панелі управління. Це полегшує налаштування та моніторинг доступу, особливо у великих організаціях з багатьма користувачами [4].

Особливості:

– єдина точка управління. Адміністратор може змінювати права доступу, створювати нові облікові записи або видаляти непотрібні облікові записи з одного місця;

– моніторинг та аудит. Централізовані системи забезпечують детальний моніторинг активності користувачів, що дозволяє швидко виявляти підозрілу активність, наприклад, незвичні спроби доступу або зміни в даних;

– звіти про доступ. Система може створювати звіти про доступ до певних ресурсів, що дозволяє адміністраторам ефективно контролювати та вдосконалювати управління доступом.

Система управління доступом має можливість надавати різні рівні доступу на основі ролей користувачів (наприклад, адміністратора, редактора, переглядача), що забезпечує диференціацію повноважень та знижує ризик зловживання правами доступу [5].

Особливості:

– рольове управління доступом (RBAC). Користувачам надаються певні ролі, а не індивідуальні права на кожен ресурс. Це значно спрощує управління доступом, особливо у великих організаціях;

– найменші привілеї. Користувачам надаються лише ті права, які потрібні для виконання їхніх завдань. Це мінімізує ризик випадкових змін або компрометації даних;

– поділ відповідальності. Відповідальність за різні дії (наприклад, створення даних, їх перегляд та видалення) розподіляється між різними користувачами, що підвищує загальний рівень безпеки.

Єдиний вхід дозволяє користувачам використовувати один обліковий запис для доступу до кількох ресурсів або систем. Це спрощує процес автентифікації і зменшує кількість паролів, які користувачам потрібно запам’ятовувати [5].

Особливості:

– зручність для користувачів. Користувачам не потрібно вводити різні паролі для кожного сервісу, що спрощує доступ і знижує ймовірність забування або неправильного введення пароля;

– зменшення людських помилок. Оскільки користувачі використовують один обліковий запис для кількох систем, ризик помилок у введенні або записі паролів зменшується;

– підвищення безпеки. У системі SSO можна легко впровадити багатофакторну аутентифікацію, що підвищує загальну безпеку доступу до всіх систем через одну аутентифікацію.

Контекстуальне управління доступом враховує додаткові фактори, такі як місцезнаходження, пристрій, з якого здійснюється доступ, або навіть час доби, для прийняття рішення про надання доступу [6].

Особливості:

– контроль доступу на основі локації. Користувачам може бути дозволено або заборонено доступ з певних географічних зон, наприклад, доступ може бути обмежений лише офісом компанії;

– обмеження за часом. Доступ до певних систем може надаватися лише в робочий час, що запобігає доступу до критично важливих даних у непередбачений час;

– контроль за пристроями. Система може обмежувати доступ лише для певних авторизованих пристроїв, що підвищує захист від атак з використанням чужих або незахищених пристроїв.

Системи управління доступом повинні постійно контролювати активність користувачів і проводити аудит прав доступу. Це дозволяє швидко виявляти підозрілі дії та оперативно реагувати на них [6].

Особливості:

– виявлення аномальної активності. Постійний моніторинг дозволяє виявляти нестандартну поведінку, яка може свідчити про спробу злому або несанкціонованого доступу;

– аудит прав доступу. Систематична перевірка прав доступу дозволяє вчасно усувати надлишкові або непотрібні привілеї, знижуючи ризик зловживання;

– автоматичні оповіщення. У разі підозрілої активності система може автоматично надсилати оповіщення адміністраторам, що дозволяє їм оперативно реагувати.

Управління доступом базується на кількох фундаментальних принципах, які допомагають забезпечити надійність та ефективність цієї системи:

– принцип найменших привілеїв (Least Privilege Principle). Користувачам і системам надається мінімальний необхідний рівень доступу для виконання їхніх завдань. Це знижує ризик несанкціонованого доступу до критичних даних і зменшує поверхню атаки;

– принцип поділу обов’язків (Separation of Duties). Розподіл відповідальності та завдань серед різних користувачів або ролей дозволяє уникнути концентрації повноважень в однієї особи або групи, що мінімізує можливість зловживання доступом [7];

– рольове управління доступом (Role-Based Access Control, RBAC). Користувачам призначаються ролі з певним набором прав доступу, що спрощує управління доступом для великих організацій та підвищує контроль над повноваженнями;

– контекстуальне управління доступом (Context-Based Access Control). Доступ надається не лише на основі ролей, але й на основі контексту, такого як місце розташування користувача, пристрій, з якого здійснюється доступ, час доби тощо. Це дозволяє ще краще контролювати доступ до системи і знижує ризик;

– безперервний моніторинг та аудит доступу. Системи управління доступом повинні постійно контролювати активність користувачів та проводити аудит прав доступу. Це дозволяє швидко виявляти підозрілі дії, вчасно оновлювати рівні доступу, а також запобігати можливим зловживанням;

– ієрархічне управління правами доступу. Встановлення ієрархії прав доступу дозволяє різним категоріям користувачів взаємодіяти з даними відповідно до їхніх рівнів привілеїв. Наприклад, адміністратор має ширші повноваження, ніж стандартний користувач, що підвищує контроль;

– регулярне оновлення та перевірка прав доступу. Регулярний перегляд прав доступу допомагає зменшити ризик того, що користувачі матимуть невиправданий доступ. Це включає видалення прав для тих, хто їх більше не потребує (наприклад, колишніх працівників), та коригування рівня доступу відповідно до змін у завданнях користувача [8].

Управління доступом є критично важливою складовою DRM-систем та інших цифрових інфраструктур, де необхідно забезпечити захищений доступ до ресурсів. Розуміння особливостей управління доступом, таких як гнучкість та поділ повноважень, а також впровадження основних принципів, таких як найменші привілеї та контекстуальне управління доступом, є важливими для підвищення безпеки та контролю в сучасних системах.

1.2 Загальна характеристика DRM-системи, як системи управління доступом

DRM (Digital Rights Management) – це система управління цифровими правами, яка використовується для захисту інтелектуальної власності та обмеження доступу до захищеного контенту. DRM-системи надають авторам,

правовласникам і провайдерам цифрового контенту можливість контролювати, як їхній продукт використовується, копіюється та поширюється. Основною метою DRM-систем є захист цифрового контенту від несанкціонованого доступу, копіювання чи модифікації [8].

DRM-система забезпечує як управління доступом до контенту, так і контроль за його використанням. Це досягається завдяки комбінації різних компонентів і функцій, які працюють разом, щоб гарантувати захист прав авторів та власників контенту:

– ідентифікація користувача. DRM-система перевіряє особу або статус користувача для визначення його права на доступ. Це може здійснюватися через обліковий запис, ліцензію або унікальний ключ доступу. Наприклад, для онлайн-відеосервісів ідентифікація відбувається через обліковий запис користувача, а для ліцензованого програмного забезпечення – через активаційний ключ;

– шифрування контенту. Контент захищається шифруванням, яке запобігає несанкціонованому доступу до даних навіть у разі фізичного копіювання. Шифрування здійснюється на сервері перед розповсюдженням і залишається незмінним, поки користувач не отримує доступ. Важливу роль тут відіграють криптографічні алгоритми, які гарантують, що контент можна відкрити лише за наявності відповідного ключа;

– управління ліцензіями. Кожен користувач отримує ліцензію, що визначає його права щодо доступу до контенту. Ліцензії можуть містити інформацію про тривалість доступу, кількість пристроїв, які можуть бути використані для відтворення контенту, і навіть дозволені дії (перегляд, редагування, друк). Управління ліцензіями забезпечує дотримання умов використання, встановлених правовласниками [9];

– аутентифікація та авторизація. Аутентифікація визначає особу користувача, а авторизація перевіряє, чи має користувач відповідні права на доступ до контенту. Цей процес забезпечує точне розмежування прав доступу, що дозволяє захистити контент від несанкціонованих спроб доступу. Наприклад, для

доступу до певного фільму на платформі користувач повинен пройти аутентифікацію, а система визначить його права (авторизація), дозволяючи лише перегляд без можливості завантаження;

– кодування та декодування. DRM-система кодує цифровий контент перед поширенням і дозволяє його декодування лише у межах відповідних умов ліцензії. Коли користувач запускає контент, DRM-система перевіряє ліцензію, а потім дозволяє розшифрувати та відтворити контент;

– моніторинг використання. Багато DRM-систем включають механізми моніторингу та відстеження активності користувачів. Це може включати запис інформації про час доступу, дії користувачів, пристрої та місцезнаходження. Зокрема, для музичних або відео-стримінгових сервісів моніторинг дозволяє фіксувати кількість переглядів і розподіл за ліцензіями. Це також допомагає захищати контент від зловживань, таких як спроби дублювання [9].

DRM-система (Digital Rights Management) виконує низку функцій, спрямованих на захист цифрового контенту та управління доступом до нього. Вона контролює, хто і як може використовувати контент, забезпечуючи при цьому захист прав власників і запобігаючи несанкціонованому використанню або розповсюдженню. Основні функції DRM-системи включають захист від копіювання, контроль за кількістю пристроїв, терміновий контроль доступу, обмеження дій користувачів, аутентифікацію ліцензій та адаптивне обмеження доступу.

DRM-системи обмежують можливість копіювання та несанкціонованого розповсюдження контенту. Це особливо важливо для таких типів контенту, як музика, відео, електронні книги та програмне забезпечення, які можуть бути легко поширені без дозволу власника.

Механізми захисту:

– шифрування файлів. DRM-система зазвичай шифрує контент перед його розповсюдженням, і цей контент можна відтворити лише на авторизованих пристроях або за наявності відповідної ліцензії;

– водяні знаки (Watermarking). Система додає унікальний маркер (водяний знак) у контент, який дозволяє відстежувати джерело витоку у разі несанкціонованого розповсюдження. Водяні знаки можуть бути видимими або невидимими, і вони допомагають зберегти докази порушення прав, якщо контент буде знайдено на сторонніх ресурсах;

– захист від "копіювання екрану". У випадках, коли користувач намагається записати або зробити скріншот захищеного контенту, DRM-система може обмежити доступ до нього або додати на екран водяний знак, що робить копіювання контенту неефективним [10].

DRM-система контролює, на скількох пристроях користувач може отримати доступ до одного й того самого контенту. Це обмеження запобігає поширенню ліцензії серед сторонніх осіб або відтворенню контенту на необмеженій кількості пристроїв.

Механізми реалізації:

– аутентифікація за пристроєм. Кожен пристрій, на якому користувач має доступ до контенту, проходить реєстрацію. DRM-система запам'ятовує пристрої, і при досягненні встановленого ліміту доступ до нового пристрою блокується;

– обмеження на авторизацію нових пристроїв. Користувачеві може бути дозволено доступ лише з обмеженої кількості пристроїв (наприклад, три одночасно). Якщо користувач намагається додати новий пристрій, система вимагає відключити один з попередніх або скидає доступ на старих пристроях.

DRM-система дозволяє обмежувати доступ до контенту на певний період, після якого ліцензія автоматично анулюється. Це забезпечує обмежений за часом доступ і широко використовується у сервісах прокату фільмів, тимчасової підписки або тестового використання програмного забезпечення [10].

Механізми реалізації:

– ліцензії з обмеженим терміном дії. Користувач отримує ліцензію на доступ до контенту на визначений час. Після завершення терміну дії ліцензії система автоматично блокує доступ або видаляє контент із пристрою;

– таймери доступу. DRM-система може встановлювати таймер для обмеження часу перегляду або доступу до контенту. Наприклад, після початку перегляду відео користувач може мати 48 годин на завершення перегляду, після чого доступ анулюється.

DRM-система дозволяє обмежити дії, які користувач може виконувати з контентом, такі як друк, редагування або поширення. Це важливо для таких типів контенту, як електронні книги, документи та навчальні матеріали [11].

Механізми реалізації:

– заборона друку. DRM-система може блокувати функцію друку для цифрових документів, щоб запобігти створенню фізичних копій, які можуть бути несанкціоновано поширені;

– обмеження на редагування та копіювання тексту. DRM-система може заборонити функцію копіювання тексту або внесення змін у захищений документ;

– обмеження на збереження локальних копій. Деякі DRM-системи обмежують можливість завантаження або зберігання контенту на пристрій користувача, забезпечуючи лише стримінговий доступ. Це мінімізує ризик витоку, оскільки контент залишається захищеним на сервері.

Кожен користувач повинен мати відповідну ліцензію для доступу до захищеного контенту. DRM-система виконує аутентифікацію ліцензії перед кожним запуском контенту, щоб переконатися, що користувач має право на його використання [11].

Механізми реалізації:

– перевірка ліцензії через сервер. Під час кожного доступу до контенту система звертається до серверу, де зберігаються дані про ліцензії. Це дозволяє перевірити дійсність ліцензії та підтвердити, що права користувача не було відкликано;

– захищені ключі шифрування. Для активації контенту користувачеві видається унікальний ключ доступу, що дозволяє декодувати контент лише за

умови дійсної ліцензії. Ключ може бути одноразовим або оновлюваним після кожного сеансу доступу.

DRM-системи можуть встановлювати обмеження на доступ до контенту залежно від зовнішніх умов, таких як місцезнаходження користувача або тип пристрою. Це дозволяє контролювати доступ на основі додаткових критеріїв, що підвищує безпеку та відповідність ліцензійним угодам [12].

Механізми реалізації:

– геолокаційні обмеження. Система може блокувати доступ до контенту, якщо користувач намагається отримати його з країни, де цей контент недоступний. Наприклад, деякі сервіси потокового відео обмежують доступ до контенту в певних країнах через ліцензійні угоди;

– контроль типу пристрою. DRM-система може надавати доступ лише на визначених типах пристроїв, наприклад, для перегляду певного контенту дозволено лише через настільні комп'ютери або авторизовані мобільні застосунки. Це обмежує можливість доступу з неавторизованих пристроїв і знижує ризик несанкціонованого розповсюдження.

Функції DRM-систем як систем управління доступом охоплюють широкий спектр заходів щодо забезпечення конфіденційності та безпеки цифрових активів. Захист від копіювання, контроль за кількістю пристроїв, обмеження часу доступу, регулювання дозволених дій, аутентифікація ліцензій та адаптивне обмеження доступу є основними елементами DRM-систем, що дозволяють ефективно контролювати доступ до контенту та забезпечувати його захист. Завдяки комплексному підходу до управління доступом DRM-системи створюють безпечне середовище для правовласників, мінімізуючи ризики піратства та несанкціонованого використання контенту [12].

Впровадження таких функцій у DRM-системи дозволяє правовласникам зберігати контроль над своїми цифровими продуктами, забезпечуючи захист авторських прав і підтримку ліцензійних обмежень. DRM-системи стають

важливим інструментом не лише для захисту від піратства, але й для адаптації до різних умов доступу та користувацьких сценаріїв.

Основні переваги реалізації DRM-систем:

– захист доходів правовласників. DRM-системи запобігають незаконному поширенню та використанню контенту, що допомагає компаніям і авторам зберігати прибутки від продажу або підписки на цифровий контент;

– підтримка ліцензійних угод. Впровадження DRM-систем допомагає забезпечити дотримання умов ліцензійних угод, зокрема, обмежень за територією, тривалістю доступу або кількістю користувачів, що є особливо важливим для платформ, що працюють на міжнародному рівні;

– гнучкість для користувачів і власників контенту. Адаптивні функції DRM, такі як геолокаційні обмеження та контроль пристроїв, дозволяють правовласникам налаштовувати доступ відповідно до специфічних потреб або обмежень;

– моніторинг і звітність. Завдяки функціям моніторингу DRM-системи дозволяють правовласникам отримувати дані про використання контенту, що допомагає краще розуміти потреби аудиторії та вдосконалювати контент, а також виявляти можливі порушення прав [13].

Загалом, DRM-системи виконують роль комплексної системи управління доступом, забезпечуючи ефективний захист контенту від несанкціонованого доступу та зловживань, що є надзвичайно важливим у сучасному цифровому середовищі, де авторські права можуть легко бути порушені. Завдяки широкому спектру функцій DRM-системи дають можливість правовласникам забезпечити надійний контроль над своїми цифровими активами, водночас надаючи користувачам зручний і легальний доступ до контенту в рамках встановлених ліцензійних умов.

Управління доступом є критично важливою складовою DRM-систем та інших цифрових інфраструктур, де необхідно забезпечити захищений доступ до ресурсів. Розуміння особливостей управління доступом, таких як гнучкість та поділ

повноважень, а також впровадження основних принципів, таких як найменші привілеї та контекстуальне управління доступом, є важливими для підвищення безпеки та контролю в сучасних системах [13].

1.3 Алгоритми генерації динамічно змінювальних ключів

Генерація динамічно змінюваних ключів є важливим компонентом захисту у системах управління доступом, особливо в тих, що потребують високого рівня безпеки, як-от DRM-системи. Динамічні ключі змінюються на кожну сесію або через певні проміжки часу, що зменшує ймовірність компрометації та запобігає повторному використанню паролів. Основою для таких ключів може бути алгоритм HOTP (Hashed-Based One-Time Password), покращений алгоритм TOTP (Time-Based One-Time Password) або інші криптографічні методи, які включають змінювані параметри, як-от лічильник, час або випадкові значення.

HOTP базується на алгоритмі HMAC-SHA1 (Hash-based Message Authentication Code) і створює одноразовий пароль (OTP), який залежить від лічильника. Кожен новий пароль генерується на основі збільшення лічильника, що дозволяє згенерувати унікальне значення для кожної сесії. HOTP широко використовується у двофакторній аутентифікації та може бути адаптований для генерації динамічних ключів [14].

Алгоритм генерації:

– встановлення секретного ключа K (відомий лише серверу і клієнтському пристрою, наприклад, USB-ключу);

– ініціалізація лічильника C, який збільшується після кожного запиту на генерацію OTP;

– обчислення HMAC із використанням HMAC-SHA1: $H = HMAC(K, C)$;

– виконання обтинання результату H для отримання шестизначного або восьмизначного OTP;

– OTP використовується як динамічний ключ для одноразового доступу і змінюється з кожним новим входом або сесією.

Переваги:

– дозволяє генерацію одноразових динамічних ключів для кожної сесії;

– секретний ключ K та лічильник C забезпечують криптографічний захист паролів, навіть при перехопленні.

Недоліки:

– потребує синхронізації лічильника між сервером і пристроєм;

– у разі розсинхронізації можливі помилки при генерації OTP.

TOTP базується на HOTP, але замість лічильника використовує час як основу для генерації одноразового пароля. Це дозволяє забезпечити синхронізацію OTP протягом певного інтервалу часу, що робить систему зручною для динамічної генерації ключів, які оновлюються з регулярною періодичністю [14].

Алгоритм генерації:

– встановлення секретного ключа K, відомого обом сторонам (серверу і клієнту);

– використання поточного часу T, розділеного на фіксований часовий інтервал (наприклад, 30 секунд): T = current_time/time_step;

– обчислення HMAC: $H = HMAC(K, T)$ з використанням HMAC-SHA1;

– обтинання результату H для отримання одноразового пароля (наприклад, шестизначного);

– користувач вводить OTP, дійсний лише протягом часу, що відповідає заданому інтервалу.

Переваги:

– не потребує лічильника, оскільки синхронізується за часом, що спрощує роботу системи;

– ключ динамічно змінюється кожні 30 секунд (або заданий інтервал), підвищуючи захищеність.

Недоліки:

– вимагає точного синхронізованого часу на сервері та клієнтському пристрої;

– часові зсуви можуть спричинити помилкові відхилення OTP.

Цей підхід заснований на використанні випадкового значення (nonce), яке генерується для кожної нової сесії або запиту на доступ. Nonce-значення не повторюються та часто поєднуються з іншим параметром, наприклад, часом або секретним ключем, для створення динамічного ключа [15].

Алгоритм генерації:

– генерація випадкового значення (nonce) на сервері або на клієнтському пристрої.

– поєднання nonce із секретним ключем K та, за потреби, часом для отримання унікального ключа: $dynamic\_key = HMAC\ (K, nonce \parallel time)$;

– використання динамічного ключа для одноразової сесії або як основи для генерації одноразового пароля.

Переваги:

– забезпечує непередбачуваність і унікальність кожного ключа завдяки використанню випадкового значення;

– знижує ризик повторного використання паролів, оскільки кожна сесія має унікальне nonce-значення.

Недоліки:

– потребує зберігання nonce або його параметрів на обох сторонах для перевірки дійсності;

– у разі використання однакових значень nonce та часу може бути вразливим до атак.

Алгоритм, що поєднує HOTP з динамічно змінюваним секретним ключем:

Опис алгоритму: У цьому підході застосовується вдосконалена версія HOTP, де секретний ключ оновлюється після кожної успішної сесії або через певний час. Це зменшує ризик компрометації ключа, оскільки він регулярно змінюється, забезпечуючи вищий рівень безпеки для DRM-систем [15].

Алгоритм генерації:

– встановлення початкового секретного ключа K_0;

– при кожній успішній сесії система генерує новий ключ K_{n+1} на основі попереднього: $K_{n+1} = HMAC(K_n, session\_id \parallel time)$;

– використання нового ключа K_{n+1} у наступній сесії для генерації HOTP-пароля, як у стандартному алгоритмі;

– оновлення значення секретного ключа на сервері та пристрої після завершення сесії.

Переваги:

– підвищена захищеність завдяки регулярному оновленню секретного ключа;

– стійкість до атак на повторне використання або перехоплення ключів, оскільки вони змінюються після кожної сесії.

Недоліки:

– складність у синхронізації нового ключа між сервером і пристроєм;

– високі вимоги до обчислювальних ресурсів для оновлення та зберігання ключів.

У таблиці 1.1 здійснено порівняння чотирьох алгоритмів генерації динамічно змінюваних ключів, які можуть використовуватися для підвищення безпеки у системах управління доступом. У кожному алгоритмі розглядаються основні параметри генерації (лічильник, час, випадкове значення), їхня стійкість до атак, складність синхронізації між клієнтом і сервером, а також доцільність застосування у DRM-системах [16].

Таблиця 1.1 – Порвіняльний аналіз алгоритмів

Алгоритм	Основний параметр	Стійкість до атак	Складність синхронізації	Використання в DRM-системах
HOTP	Лічильник	Висока, але залежить від синхронізації лічильника	Вимагає синхронізації лічильника між сервером і пристроєм	Підходить для одноразових сесій

Продовження таблиці 1.1

Алгоритм	Основний параметр	Стійкість до атак	Складність синхронізації	Використання в DRM-системах
TOTP	Час	Висока, але залежить від точності синхронізації часу	Висока вимога до точності синхронізації часу	Підходить для коротких сесій із обмеженим доступом
Nonce (випадкові значення)	Випадкове значення (nonce)	Висока, за умови унікальності значення	Вимагає синхронізації nonce на сервері та пристрої	Ефективний для короткочасних сесій
Удосконалений HOTP з динамічним ключем	Динамічно змінюваний секретний ключ	Дуже висока, завдяки зміні ключа після кожної сесії	Висока складність синхронізації нового ключа між сервером і пристроєм	Ідеально підходить для захисту з високими вимогами до безпеки в DRM-системах

Таблиця 1.1 надає можливість порівняти алгоритми за важливими критеріями, спрощуючи вибір оптимального рішення залежно від специфічних потреб і рівня безпеки, необхідного у системі управління доступом.

HOTP та TOTP забезпечують високий рівень безпеки для одноразових сесій, але обмежуються складністю синхронізації лічильника чи часу, що може спричинити помилки доступу [17].

Nonce (випадкове значення) є зручним для короткочасних сесій, але потребує надійного збереження та синхронізації значень.

Удосконалений HOTP із динамічним ключем забезпечує найвищу стійкість до атак завдяки зміні ключа після кожної сесії, що робить його ідеальним вибором для систем із підвищеними вимогами до захисту, наприклад, у DRM-системах.

Серед представлених алгоритмів кожен має свої особливості та обмеження, однак удосконалений HOTP з динамічним ключем забезпечує найвищий рівень захисту для DRM-систем. Завдяки динамічній зміні секретного ключа після кожної сесії, цей алгоритм значно знижує ризик повторного використання, перехоплення або компрометації ключа, що робить його ідеальним для середовищ із підвищеними вимогами до безпеки.

Інші методи, такі як HOTP і TOTP, також ефективні для створення одноразових паролів, але потребують точного налаштування параметрів синхронізації (лічильника або часу), що може ускладнити їхнє застосування в умовах частого або тривалого використання. Алгоритми на основі nonce-значень є простими та зручними для короткочасних сесій, проте можуть бути менш надійними для тривалих сеансів через потребу в точному збереженні значень [18].

Вибір алгоритму залежить від специфіки середовища та вимог до захисту, але вдосконалений HOTP з динамічним ключем є перспективним вибором для систем управління доступом, де потрібен високий рівень захисту та можливість динамічного оновлення ключів.

1.4 Аналіз сучасних методів захисту інформаційних ресурсів за допомогою USB-ключів безпеки та системи блокування доступу

У сучасних системах управління доступом захист інформаційних ресурсів базується на комбінації криптографічних методів, багатофакторної аутентифікації, USB-ключів безпеки та систем блокування доступу. У перспективі створення захищеної DRM-системи, що використовуватиме динамічно змінювані USB-ключі на основі вдосконаленого алгоритму HOTP та систему блокування доступу, має на меті покращити контроль доступу та зменшити ризики компрометації.

Проведемо порівняльний аналіз існуючих методів захисту та можливостей їх вдосконалення в межах майбутньої розробки [18].

Сучасні USB-ключі безпеки: HOTP/TOTP, FIDO U2F, PKI

HOTP/TOTP USB-ключі: Ці ключі використовують одноразові паролі, що генеруються на основі алгоритмів HOTP (Hashed-Based One-Time Password) або TOTP (Time-Based One-Time Password). Такі паролі зберігаються на сервері і синхронізуються з USB-ключем, що дозволяє значно підвищити захищеність при кожному новому вході.

FIDO U2F: Протокол FIDO (Fast Identity Online) є сучасним стандартом для багатофакторної аутентифікації, що забезпечує захист без передачі даних користувача. FIDO USB-ключі ефективно захищають від атак на основі фішингу та MITM (man-in-the-middle) [19].

PKI USB-ключі: USB-ключі з підтримкою PKI (інфраструктури відкритих ключів) надають можливість автентифікації на основі криптографічної пари закритий/відкритий ключ, що забезпечує один з найвищих рівнів захищеності, але вимагає складної інфраструктури для управління ключами.

Переваги сучасних методів:

– високий рівень захищеності. Завдяки криптографічному захисту USB-ключі унеможливлюють підробку або перехоплення даних;

– зручність використання. Користувачам не потрібно запам'ятовувати паролі, достатньо вставити ключ у пристрій для доступу;

– універсальність. Сучасні USB-ключі можна використовувати для доступу до різних систем, що підтримують стандарти HOTP, FIDO або PKI.

Недоліки сучасних методів:

– високі витрати. Впровадження USB-ключів потребує інвестицій, особливо для організацій з великою кількістю користувачів;

– ризик фізичних атак. Втрата або крадіжка USB-ключа може призвести до компрометації даних, якщо не вжити додаткових заходів, таких як блокування ключа;

– необхідність синхронізації. Системи на основі HOTP/TOTP потребують синхронізації між сервером та USB-ключем, що може створити труднощі при відсутності інтернету або при розсинхронізації [19].

Системи блокування доступу активно використовуються для виявлення підозрілих дій та запобігання несанкціонованому доступу. Вони реагують на такі дії, як багаторазові невдалі спроби входу, спроби доступу з невідомих пристроїв або IP-адрес.

Механізми блокування доступу:

– порогові обмеження невдалих спроб. Блокування доступу після перевищення визначеної кількості спроб автентифікації;

– аналіз аномальної активності. Виявлення аномальної активності (наприклад, спроб входу з незвичної локації) і блокування доступу для підвищення захисту;

– багатофакторна перевірка після блокування. Для розблокування доступу користувачі повинні пройти додаткову багатофакторну автентифікацію [20].

Переваги сучасних систем блокування доступу:

– запобігання несанкціонованим діям. Системи блокування можуть ефективно обмежувати доступ у разі виявлення підозрілих дій, знижуючи ризик компрометації даних;

– захист від внутрішніх загроз. Блокування дозволяє зменшити ризик зловживання правами доступу з боку внутрішніх користувачів;

– адаптивність. Система блокує підозрілі дії в режимі реального часу, забезпечуючи динамічний захист.

Недоліки сучасних систем блокування:

– висока ймовірність фальшивих спрацювань. Система може заблокувати легітимних користувачів через некоректно налаштовані порогові значення;

– потреба в налаштуванні. Системи блокування потребують ретельного налаштування, щоб уникнути фальшивих блокувань і одночасно виявляти реальні загрози;

– витрати на обробку даних. Постійний моніторинг і аналіз логів можуть вимагати значних ресурсів для обробки та зберігання даних [20].

Майбутня розробка пропонує поєднання динамічно змінюваних USB-ключів з алгоритмом HOTP та системи блокування доступу, щоб забезпечити більш високий рівень захисту DRM-системи.

Динамічно змінювані USB-ключі на основі вдосконаленого HOTP

Концепція: У перспективній розробці пропонується використання динамічно змінюваних USB-ключів з алгоритмом HOTP, що генерує одноразові паролі на основі захищених секретних ключів і лічильників. На відміну від статичних USB-ключів, такі ключі автоматично оновлюють значення секретного ключа після кожної сесії, забезпечуючи унікальність кожного пароля [21].

Переваги перспективної розробки:

– висока надійність. Унікальні паролі, що генеруються на кожну сесію, унеможливлюють повторне використання ключів і захищають від перехоплення даних;

– динамічне оновлення секретних даних. Змінювані USB-ключі, на відміну від звичайних, є більш стійкими до атак, оскільки ключі оновлюються, а їхнє значення недоступне навіть при фізичному доступі;

– синхронізація з DRM-системою. Після кожної сесії дані оновлюються на сервері, що дозволяє DRM-системі контролювати використання контенту та ліцензії в режимі реального часу.

Очікувані виклики:

– ресурсозатратна синхронізація. Удосконалений HOTP з динамічною синхронізацією потребує постійного з’єднання між USB-ключем та сервером для оновлення даних;

– залежність від фізичного носія. Зміна USB-ключа або його втрата може призвести до необхідності заміни всієї сесії.

Удосконалена система блокування буде інтегрувати адаптивні методи, що враховують поведінкові характеристики користувачів, час доступу, місце та інші контекстуальні чинники. Це дозволить точніше ідентифікувати підозрілі дії та приймати рішення про блокування доступу в режимі реального часу [21].

Особливості перспективної системи блокування доступу:

– аналіз поведінкових характеристик. Система аналізуватиме поведінку користувачів для виявлення аномальних дій, таких як спроби доступу з незвичних місць або використання пристроїв, які не відповідають звичним характеристикам користувача;

– контекстуальне управління доступом. Врахування таких чинників, як час доби, географічне місцезнаходження та пристрій доступу. Наприклад, доступ з незвичайного місця або в незвичний час може викликати додаткові запити на автентифікацію;

– адаптивні порогові значення блокування. Система автоматично змінює порогові значення для блокування залежно від рівня ризику. Наприклад, доступ до критичних ресурсів вимагатиме вищого рівня автентифікації, тоді як доступ до менш важливих даних може бути спрощеним;

– багатофакторна перевірка після блокування. У разі блокування доступу користувач може пройти багатофакторну автентифікацію для підтвердження своєї особи. Це забезпечує додатковий рівень захисту у випадках компрометації облікових даних;

– логування та аналіз інцидентів. Система веде журнал всіх спроб доступу, зокрема підозрілих дій і блокувань. Це дозволяє швидко виявляти причини інцидентів безпеки, проводити аудит активності користувачів та покращувати загальний захист системи [22].

Переваги перспективної системи блокування доступу:

– покращена точність. Адаптивний підхід дозволяє системі враховувати більше чинників, що знижує ймовірність фальшивих блокувань і підвищує ефективність захисту;

– захист від сучасних загроз. Виявлення аномальної активності та контекстуальний підхід допомагають зменшити ризик атак на основі підбору паролів або спроб доступу з компрометованих пристроїв;

– гнучкість налаштувань. Система дозволяє легко налаштувати порогові значення для різних користувачів або груп, враховуючи специфіку їхнього доступу до контенту.

Виклики перспективної розробки:

– складність налаштування та підтримки. Впровадження адаптивної системи блокування потребує ретельного налаштування порогових значень, а також безперервного моніторингу для забезпечення точності;

– ресурсоємність. Збір і аналіз поведінкових даних потребує потужної інфраструктури та значних обчислювальних ресурсів [22].

Щоб надати повноцінний аналіз, наведемо порівняння ключових особливостей та характеристик сучасних методів захисту на основі USB-ключів і систем блокування доступу з перспективною розробкою, яка включає динамічно змінювані USB-ключі на основі вдосконаленого HOTP і адаптивну систему блокування. Аналіз зосереджуватиметься на ефективності, надійності, витратах, гнучкості та безпеці, а також на потенційних перевагах і недоліках кожного підходу.

У таблиці 1.2 здійснено порівняння сучасних методів захисту інформаційних ресурсів із перспективною розробкою, яка поєднує динамічні USB-ключі з вдосконаленим алгоритмом HOTP та адаптивне блокування доступу.

Таблиця 1.2 – Порівняльний аналіз сучасних методів захисту інформаційних ресурсів і перспективної розробки для DRM-систем

Критерій	Сучасні методи захисту (USB-ключі та блокування доступу)	Перспективна розробка (динамічні USB-ключі на основі вдосконаленого HOTP та адаптивне блокування)
Ефективність захисту	Високий рівень захисту за рахунок криптографії та фізичних USB-ключів, але ключі потребують додаткового захисту від крадіжки та фізичного доступу.	Підвищена ефективність завдяки динамічній генерації паролів HOTP та поведінковому блокуванню. Використання одноразових паролів знижує ризик повторного використання та атак типу «людина посередині».

Продовження таблиці 1.2

Критерій	Сучасні методи захисту (USB-ключі та блокування доступу)	Перспективна розробка (динамічні USB-ключі на основі вдосконаленого HOTP та адаптивне блокування)
Захист від компрометації	Захищають від компрометації паролів за рахунок фізичного USB-ключа та багатофакторної аутентифікації, але недостатньо ефективні для захисту від втрати або крадіжки ключа.	Динамічні USB-ключі та адаптивне блокування покращують захист від компрометації шляхом регулярного оновлення ключів. Це ускладнює несанкціонований доступ навіть при втраті пристрою, оскільки ключ оновлюється після кожної сесії.
Зручність для користувачів	Вимагають фізичного носія (USB-ключа), що може бути незручним для користувачів, а також потребують багаторазового підтвердження доступу (MFA).	Поліпшена зручність завдяки автоматичному оновленню ключів та поведінковій перевірці, що зменшує необхідність багатофакторної аутентифікації при звичній активності.
Ресурсоємність та витрати	Висока вартість на придбання фізичних ключів та їхній регулярний моніторинг і заміна у разі втрат чи пошкоджень.	Зниження витрат на повторну автентифікацію за рахунок автоматичного оновлення ключів і зменшення фізичної залежності від кожного ключа. Однак система потребує ресурсів для обробки поведінкових даних.
Захист від внутрішніх загроз	Система блокування ефективно запобігає внутрішнім загрозам шляхом обмеження спроб доступу після певної кількості невдалих входів.	Поведінковий аналіз і адаптивне блокування дозволяють виявляти підозрілу активність з боку внутрішніх користувачів та знижують ризик зловживання правами доступу.
Точність і надійність	Висока надійність фізичної автентифікації, проте є можливість фальшивих блокувань через неправильні налаштування порогів.	Покращена точність і менша кількість фальшивих спрацювань завдяки поведінковому та контекстуальному аналізу, що дозволяє уникати помилок при перевірці доступу.

Покращення ефективності та захисту від компрометації. Перспективна розробка на базі вдосконаленого HOTP забезпечує динамічне оновлення ключів, що значно знижує ризик повторного використання або перехоплення одноразових паролів. Адаптивна система блокування доступу доповнює захист, дозволяючи миттєво реагувати на аномалії та блокувати підозрілу активність, підвищуючи загальну ефективність системи [23].

Гнучкість і адаптивність. Завдяки динамічним USB-ключам і адаптивному блокуванню доступу майбутня розробка буде краще відповідати потребам користувачів, адаптуючи рівень захисту відповідно до контексту. Це дозволить уникати непотрібного блокування легітимних користувачів і знижувати ризик помилкових блокувань.

Зручність для користувачів і зниження навантаження на ресурси. Вдосконалена система, що використовує динамічні HOTP-коди і адаптивне блокування, знижує потребу у фізичних носіях та складних багатофакторних перевірках при звичному користуванні. Це полегшить доступ для користувачів і знизить витрати на управління ключами [24].

Захист від внутрішніх загроз і підвищення точності. Поєднання адаптивного поведінкового аналізу та динамічного оновлення ключів дозволяє краще виявляти внутрішні загрози та підозрілу активність, що знижує ризик зловживання правами доступу. Завдяки високоточному контекстуальному підходу система блокує підозрілих користувачів, мінімізуючи фальшиві спрацювання.

Перспективна розробка, що включає динамічно змінювані USB-ключі та вдосконалений алгоритм HOTP, разом із адаптивним блокуванням доступу, пропонує більш гнучкий, ефективний та надійний підхід до управління доступом у DRM-системах. Така система матиме переваги над сучасними методами, оскільки забезпечує динамічне управління ключами, точний поведінковий аналіз і адаптивність, що разом знижують ризик компрометації та підвищують зручність для користувачів [25].

1.5 Висновки та постановка задачі

У даному розділі проведено дослідження теоретичних основ підвищення захищеності DRM-системи управління доступом за допомогою динамічно змінюваних USB-ключів безпеки на основі вдосконаленого алгоритму HOTP (Hashed-Based One-Time Password) та системи блокування доступу. Було проаналізовано ключові аспекти безпеки DRM-систем, що включає важливість динамічного оновлення ключів, впровадження багатофакторної аутентифікації з використанням USB-ключів, а також адаптивне управління доступом для запобігання несанкціонованим спробам доступу. Розглянуто сучасні алгоритми генерації динамічно змінюваних ключів, методи шифрування та захисту інформації, що дозволило глибше зрозуміти вимоги до систем, що захищають цифрові права, та ефективні підходи для зниження ризиків компрометації.

На основі проведеного дослідження та аналізу було сформульовано завдання для подальшої розробки:

– розробка алгоритму динамічно змінюваних USB-ключів для DRM-системи на основі вдосконаленого алгоритму HOTP;

– проектування системи адаптивного блокування доступу, що використовує поведінковий аналіз та контекстуальні параметри;

– розробка схеми роботи DRM-системи з підтримкою динамічно змінюваних ключів та адаптивного блокування доступу;

– практична реалізація та тестування системи управління доступом із застосуванням USB-ключів на базі алгоритму HOTP та інтегрованою системою блокування доступу.

2 СТВОРЕННЯ ЗАХИЩЕНОЇ DRM-СИСТЕМИ УПРАВЛІННЯ ДОСТУПОМ ДИНАМІЧНО ЗМІНЮВАНИМИ USB-КЛЮЧАМИ БЕЗПЕКИ НА ОСНОВІ ВДОСКОНАЛЕНОГО АЛГОРИТМУ HOTP ТА СИСТЕМИ БЛОКУВАННЯ ДОСТУПУ

У цьому розділі буде розроблено захищену DRM-систему управління доступом, що базується на використанні динамічно змінюваних USB-ключів безпеки та вдосконаленого алгоритму HOTP (Hashed-Based One-Time Password). Будуть досліджені особливості захисту таких систем, запропоновано алгоритм роботи динамічно змінюваних ключів і розроблено систему блокування доступу. На основі отриманих результатів будуть зроблені висновки щодо ефективності розроблених рішень і їх відповідності сучасним вимогам безпеки.

2.1 Особливості створення та захисту DRM-системи управління доступом

Цифрові права управління (DRM, англ. Digital Rights Management) є важливим інструментом у сфері кібербезпеки та контролю доступу до цифрових активів, таких як програмне забезпечення, відеоконтент, електронні книги, музика та інші медіа-файли. Основною метою DRM-систем є захист інтелектуальної власності власників контенту від несанкціонованого доступу, копіювання, розповсюдження або модифікації, що стає все більш актуальним у сучасних умовах цифровізації та швидкого поширення даних.

DRM-системи забезпечують багаторівневий контроль над доступом до контенту шляхом впровадження обмежень на перегляд, копіювання або передачу файлів, використовуючи різні засоби, такі як шифрування, токенізація, одноразові паролі та USB-ключі безпеки. Застосування цих засобів дозволяє захистити авторські права на цифровий контент і зберегти дохід власників. Наприклад, DRM-технології активно використовуються у великих потокових сервісах (Netflix, Spotify, Amazon Prime), що дозволяє значно скоротити кількість піратських копій медіа-контенту [26].

DRM-системи зазвичай реалізують управління доступом через механізми аутентифікації та авторизації. Аутентифікація дозволяє підтвердити особу користувача, а авторизація визначає рівень доступу, що дозволяє обмежувати дії користувача з контентом. Такі системи можуть включати різноманітні технології, як-от цифрові підписи, сертифікати безпеки, одноразові паролі (OTP) та USB-ключі, що забезпечують високу надійність захисту даних.

Однак DRM-технології, незважаючи на свою користь, стикаються із серйозними викликами у сфері кібербезпеки. Сучасні зловмисники знаходять все більш складні методи обходу захисту, наприклад, через клонування ключів або компрометацію програмного забезпечення, що відповідає за управління доступом. Тому ефективна DRM-система повинна бути здатна адаптуватися до цих загроз, використовуючи більш просунуті алгоритми, такі як одноразові паролі (OTP) для USB-ключів безпеки, і забезпечувати інтеграцію з багаторівневими системами блокування доступу [27].

Використання USB-ключів безпеки, зокрема динамічно змінюваних ключів, підвищує стійкість DRM-системи до атак, забезпечуючи унікальний захисний бар'єр. Ключі на основі алгоритму HOTP (Hashed-Based One-Time Password) дозволяють створювати одноразові паролі для кожної сесії доступу, що ускладнює зловмисникам завдання компрометації системи. Тому впровадження нових технологій аутентифікації на основі одноразових паролів у поєднанні з фізичними USB-ключами є перспективним рішенням для підвищення надійності DRM-систем, що спрямовано на захист цифрового контенту в умовах зростаючих кіберзагроз.

Захист DRM-систем від несанкціонованого доступу є одним з ключових аспектів кібербезпеки, оскільки забезпечення контролю над авторським контентом стає дедалі складнішим завданням у зв’язку з розвитком технологій атак та методів обходу захисту. DRM-системи, хоча й значно підвищують рівень захисту цифрових активів, стикаються з рядом викликів, які знижують їх ефективність та вимагають постійного вдосконалення [28].

Основні виклики у захисті DRM-систем:

- компрометація автентифікаційних ключів. Одним із найсерйозніших викликів для DRM-систем є компрометація ключів доступу, які є центральним елементом захисту. Користувачі часто стикаються з випадками втрати або викрадення своїх автентифікаційних даних. Наприклад, у випадку використання фіксованих або недостатньо захищених ключів доступу зловмисники можуть відтворити ці ключі, отримуючи безперешкодний доступ до захищеного контенту. Це робить важливим використання динамічних або одноразових паролів, які знижують ризик компрометації;

- недосконаліть криптографічних алгоритмів. Багато DRM-систем використовують стандартні криптографічні алгоритми, які, хоча й забезпечують базову захищеність, все ж є вразливими до нових методів зламу. Зокрема, brute-force атаки, атаки типу «людина посередині» (MITM), або навіть атаки на базі квантових обчислень у майбутньому можуть скомпрометувати захист. Тому є потреба у вдосконалених алгоритмах, таких як HOTP, які забезпечують підвищену стійкість до атак за рахунок генерації одноразових паролів [28];

- атаки на програмне забезпечення DRM-систем. Програмне забезпечення DRM-систем часто стає мішенню для зловмисників. Від зламу самого програмного забезпечення до ін'єкції шкідливого коду — ці атаки можуть повністю обійти захисні механізми, не потребуючи доступу до автентифікаційних даних. Атаки на рівні програмного забезпечення можуть використовуватися для викрадення цифрових ключів, аналізу процесів аутентифікації та модифікації структури захисту контенту. DRM-системи повинні включати механізми перевірки цілісності програмного коду та виявлення аномальної активності для запобігання таким атакам;

- атаки на USB-ключі та фізичне клонування. Використання USB-ключів як засобу аутентифікації для DRM-систем підвищує безпеку, однак і такі ключі можуть стати об’єктом фізичних атак. Клонування USB-ключів, зчитування даних з них, а також атаки на рівні інтерфейсу підключення є поширеними методами, що використовуються для обходу захисту. Зловмисники можуть отримати доступ до автентифікаційних даних через вразливості у прошивці або через підключення до

незахищених систем. Для запобігання цьому DRM-системи мають інтегрувати алгоритми одноразових паролів (OTP), що оновлюються при кожному підключенні, як це реалізовано у HOTP;

- загрози від зловживання обліковими записами користувачів. Неавторизовані користувачі можуть отримати доступ до DRM-захищеного контенту через облікові записи, що використовуються законними користувачами. Наприклад, через методи соціальної інженерії або атаки на слабкі паролі зловмисники можуть отримати доступ до облікових даних користувачів. Це створює потребу у впровадженні додаткових рівнів захисту, таких як багатофакторна аутентифікація та динамічні одноразові паролі для кожної сесії [29].

Розглянемо необхідні рішення для посилення захисту DRM-систем.

Для ефективного захисту DRM-систем необхідне комплексне рішення, яке включає багаторівневий захист. Серед пріоритетних заходів:

– використання динамічних алгоритмів аутентифікації. Вдосконалений HOTP забезпечує ефективний захист шляхом генерації одноразових паролів, які не можна повторно використати або клонувати. Це мінімізує ризик несанкціонованого доступу при компрометації ключів;

– посилення криптографічних протоколів. Використання вдосконалених алгоритмів шифрування, що забезпечують високу стійкість до новітніх типів атак, зокрема квантових, які можуть бути перспективними в майбутньому;

– покращення безпеки USB-ключів. Інтеграція захисту від фізичних атак, зокрема механізмів блокування при виявленні аномальних дій або спроб клонування;

– механізми багатофакторної аутентифікації та контролю доступу: Додавання додаткових рівнів аутентифікації, які забезпечують надійність і захищеність облікових записів користувачів [30].

Таким чином, виклики у захисті DRM-систем вимагають від компаній, що надають захищений контент, комплексних підходів до розробки архітектури захисту та адаптації до сучасних загроз. Впровадження динамічних одноразових

паролів, інтегрованих з USB-ключами, таких як HOTP, є важливим кроком у цьому напрямку, що дозволяє суттєво підвищити рівень захисту та протистояти сучасним методам обходу DRM.

2.2 Розробка алгоритму динамічно змінювальних USB-ключів безпеки на основі вдосконаленого алгоритму HOTP (Hashed-Based One-Time Password

Алгоритм HOTP (Hashed-Based One-Time Password) широко використовується для забезпечення одноразових паролів (OTP), що генеруються на основі геш-функцій та значення лічильника. Така структура забезпечує високий рівень захисту, оскільки паролі є одноразовими і не повторюються в межах певної сесії. Для використання HOTP в USB-ключах безпеки в контексті DRM-систем необхідна адаптація алгоритму до умов динамічної аутентифікації, щоб він міг підтримувати змінюваність ключів та протистояти спробам клонування.

Для підвищення захищеності USB-ключів у DRM-системі базовий HOTP-алгоритм було вдосконалено, щоб забезпечити не лише генерацію одноразових паролів, а й динамічну зміну ключів при кожній спробі доступу. Це дозволяє USB-ключу автоматично генерувати новий пароль, захищаючи систему від несанкціонованого доступу навіть у разі спроб клонування.

Ключові особливості вдосконаленого HOTP для USB-ключів:

– динамічна генерація одноразових ключів. Кожен доступ до USB-ключа ініціює процес генерації нового пароля на основі HOTP. Пароль генерується з використанням унікального значення лічильника та секретного ключа, що зберігається в захищеній пам'яті USB-ключа;

– автоматичне оновлення секретного ключа. Вдосконалений HOTP також передбачає механізм оновлення секретного ключа, що використовується для генерації OTP. Це оновлення може відбуватися на регулярній основі або після кожної певної кількості запитів, що дозволяє підвищити рівень захисту;

– захист від фізичного клонування. Кожен USB-ключ містить унікальний секретний код, який прив’язаний до апаратного забезпечення ключа, що унеможливлює його клонування та використання в іншому пристрої.

Алгоритм передбачає, що при кожній спробі доступу до DRM-системи USB-ключ виконує генерацію одноразового пароля та надсилає його для аутентифікації.

Основні кроки роботи алгоритму:

- ініціалізація лічильника та секретного ключа. Після першого використання ключа відбувається ініціалізація значення лічильника (counter) та секретного ключа, що використовується для генерації OTP;
- генерація OTP. HOTP генерує одноразовий пароль шляхом хешування значення лічильника та секретного ключа. Для цього використовується криптографічна геш-функція (наприклад, SHA-1 або SHA-256);
- передача OTP для аутентифікації. USB-ключ передає згенерований OTP на сервер DRM-системи, де здійснюється перевірка;
- оновлення секретного ключа та лічильника. Після кожного успішного запиту секретний ключ і лічильник змінюються за певним алгоритмом. Наприклад, значення секретного ключа може оновлюватися за допомогою алгоритму HMAC або на основі хешування попереднього значення.

Для ефективної роботи динамічних USB-ключів у DRM-системі важлива синхронізація між ключем і сервером, де зберігаються всі дані для перевірки автентичності. Інтеграція HOTP-алгоритму для USB-ключів відбувається через протокол аутентифікації, що дозволяє DRM-системі швидко та надійно перевірити OTP.

На основі даної інформації створимо алгоритм роботи роботи динамічно змінювальних USB-ключів безпеки на основі вдосконаленого алгоритму HOTP (рис.2.1).

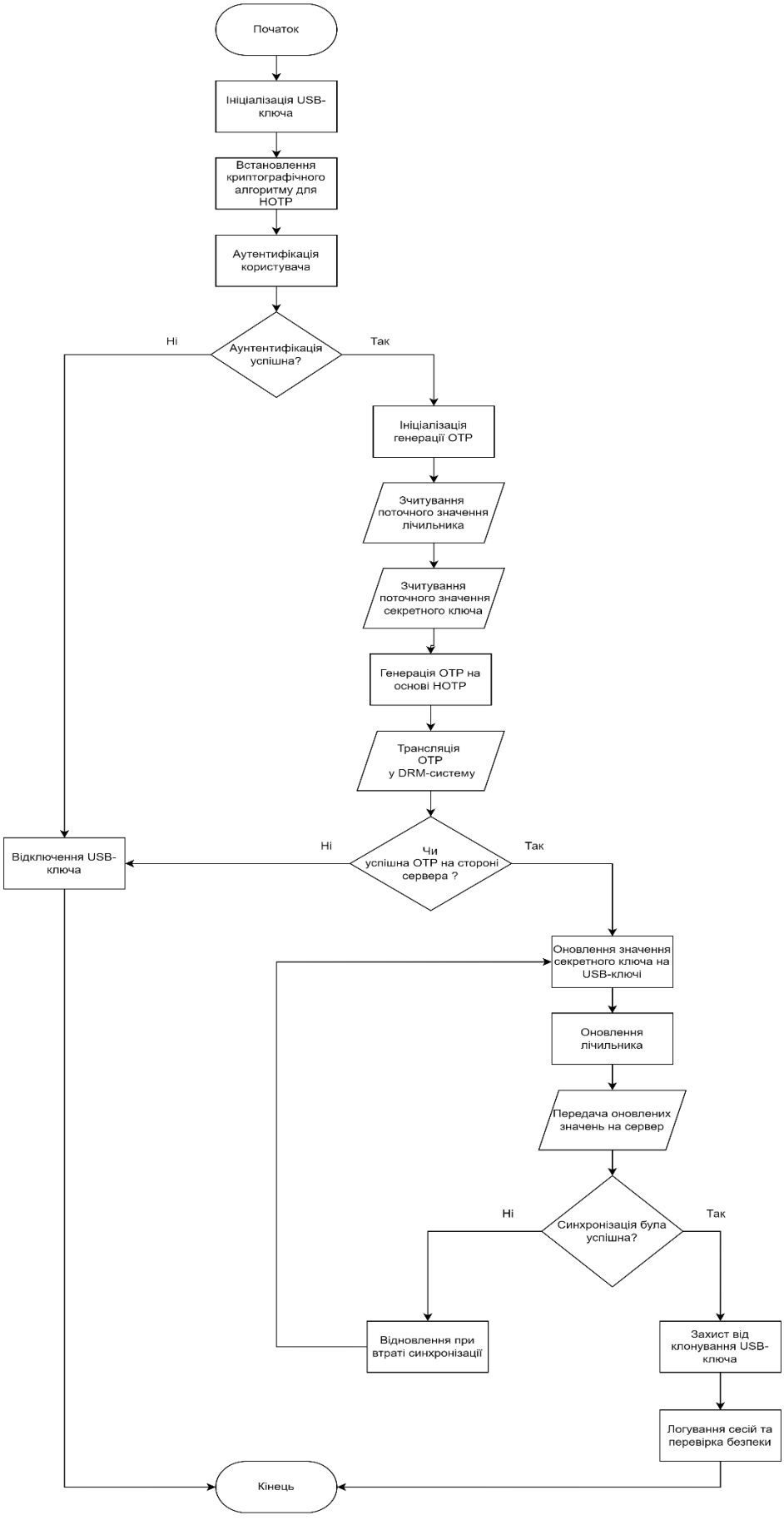


Рисунок 2.1 – Алгоритм вдосконалених динамічно змінювальних USB-ключів

Покроково розберемо даний алгоритм:

1. Ініціалізація USB-ключа.

На етапі першого підключення USB-ключа відбувається ініціалізація. Створюється початкове значення секретного ключа (K_initial) та лічильника (C_initial), які зберігаються в захищеній пам’яті USB-ключа. Ці параметри також передаються на сервер для подальшої синхронізації.

2. Встановлення криптографічного алгоритму для HOTP.

На USB-ключі обирається геш-функція, наприклад, SHA-256, яка використовується для генерації одноразового пароля. Сервер DRM-системи також налаштований для використання тієї ж геш-функції.

3. Аутентифікація користувача.

Коли користувач вставляє USB-ключ для доступу до DRM-системи, система запитує аутентифікацію. Це може бути простий пароль або PIN-код для активації USB-ключа. Введення PIN-коду дозволяє користувачеві розблокувати USB-ключ і розпочати процедуру генерації OTP.

4. Ініціалізація генерації OTP.

Після успішного введення PIN-коду USB-ключ активує функцію генерації одноразового пароля на основі поточного значення секретного ключа (K) та лічильника (C).

5. Зчитування поточного значення лічильника.

USB-ключ зчитує значення лічильника (C), яке зберігається у внутрішній пам'яті. Це значення визначає поточну сесію доступу.

6. Зчитування поточного значення секретного ключа.

USB-ключ зчитує значення секретного ключа (K), яке також зберігається у внутрішній пам'яті. K є унікальним для кожного USB-ключа і змінюється після кожного використання.

7. Генерація OTP на основі HOTP.

HOTP-алгоритм генерує одноразовий пароль (OTP) на основі значень K та C. Для цього USB-ключ виконує гешування значення лічильника разом із секретним ключем. Наприклад, OTP = HMAC-SHA-256(K || C).

8. Трансляція OTP у DRM-систему.

Згенерований OTP передається до DRM-системи. Передача здійснюється через захищений канал або спеціальний API, що запобігає можливості перехоплення OTP під час передачі.

9. Перевірка OTP на стороні сервера.

DRM-система отримує OTP та перевіряє його правильність, порівнюючи зі згенерованим значенням на основі серверного значення секретного ключа та лічильника.

10. Успішна автентифікація.

Якщо OTP є вірним, сервер оновлює значення лічильника та секретного ключа для даного USB-ключа. Користувач отримує доступ до DRM-контенту.

11. Оновлення значення секретного ключа на USB-ключі.

Після успішної автентифікації USB-ключ оновлює значення секретного ключа (K). Це оновлення може виконуватися за допомогою криптографічного алгоритму HMAC або іншого захищеного методу. Наприклад, новий секретний ключ K' може бути згенерований як K' = HMAC-SHA-256(K || OTP).

12. Оновлення лічильника.

Значення лічильника (C) інкрементується, щоб наступний OTP був унікальним і більше не використовувався в майбутніх запитах.

13. Передача оновлених значень на сервер.

Оновлені значення лічильника та секретного ключа зберігаються на сервері DRM для синхронізації з USB-ключем.

14. Відновлення при втраті синхронізації.

Якщо USB-ключ і сервер втрачають синхронізацію (наприклад, через невдалу автентифікацію), сервер може перевірити кілька значень лічильника (C ± n), щоб знайти відповідний OTP і синхронізуватися з USB-ключем.

15. Захист від повторного використання OTP.

DRM-система блокує будь-які спроби повторного використання OTP для запобігання спробам компрометації системи через перехоплені паролі.

16. Захист від клонування USB-ключа.

У кожному USB-ключі вшитий апаратний ідентифікатор, який додається до K для унікалізації кожного OTP. Це забезпечує неможливість використання зламаного або клонованого USB-ключа з іншим ідентифікатором.

17. Відключення USB-ключа після невдалих спроб автентифікації.

Якщо OTP не відповідає протягом кількох спроб, доступ до USB-ключа тимчасово блокується, що дозволяє уникнути атаки brute-force.

18. Регулярне оновлення секретного ключа.

DRM-система та USB-ключ періодично генерують новий секретний ключ (K) для кожної сесії користувача, що забезпечує додатковий рівень захищеності та запобігає можливості використання старих OTP.

19. Скидання значень при виявленні загрози.

У разі виявлення компрометації USB-ключа або його зламу секретний ключ і лічильник скидаються на початкові значення, або USB-ключ блокується до проходження повторної активації.

20. Логування сесій та перевірка безпеки.

DRM-система веде журнал усіх сесій доступу, OTP та дій USB-ключа для виявлення аномальної активності, що може вказувати на спроби зламу або компрометації. Логи використовуються для аналізу та підвищення надійності алгоритму HOTP.

Цей алгоритм забезпечує надійну та безпечну роботу USB-ключів на основі вдосконаленого HOTP у системі управління доступом DRM, захищаючи дані від несанкціонованого доступу та клонування.

2.3 Розробка алгоритму системи блокування доступу

Система блокування доступу є важливим компонентом DRM-системи для запобігання несанкціонованому доступу до захищеного контенту. Вона має на меті не лише обмежити доступ, але й оперативно реагувати на спроби зламу та зловживання. Нижче представлено покроковий алгоритм розробки

системиблокування доступу, який включає аналіз спроб аутентифікації, управління сеансами та заходи щодо запобігання компрометації системи.

Розробимо алгоритм блокування доступу (рис.2.2).

Рисунок 2.2 – Алгоритм блокування доступу

Покроково розберемо даний алгоритм:

1. Ініціалізація модуля блокування.

Модуль блокування активується під час запуску DRM-системи для моніторингу спроб доступу і зберігання журналу активності користувачів.

2. Моніторинг спроб доступу.

Усі спроби автентифікації реєструються в системі. Це стосується як успішних, так і невдалих входів, для подальшого аналізу безпеки.

3. Встановлення порогу невдалих спроб.

Система встановлює максимальну кількість невдалих спроб (наприклад, 3) до автоматичного блокування. При перевищенні порогу доступ блокується.

4. Аналіз підозрілої активності.

Після кожної спроби входу система аналізує активність на наявність аномалій, таких як незвичні IP-адреси або пристрої, що підключаються.

5. Блокування після перевищення порогу.

Якщо користувач перевищує поріг невдалих спроб, доступ блокується на певний час (наприклад, на 15 хвилин).

6. Оповіщення користувача.

Користувачу надсилається повідомлення про блокування з поясненням причин та інструкцією для розблокування.

7. Багатофакторна аутентифікація для відновлення доступу.

Після блокування користувач повинен пройти багатофакторну аутентифікацію (наприклад, через SMS або електронну пошту), щоб підтвердити свою особу.

8. Перевірка автентичності після розблокування.

Після розблокування користувач повинен пройти повторну автентифікацію, наприклад, за допомогою OTP або інших засобів.

9. Логування сесій доступу.

Система реєструє всі сесії доступу та інші важливі дії, що дозволяє аналізувати активність і виявляти потенційні загрози.

2.4 Висновки до розділу

У цьому розділі було розглянуто процес створення захищеної DRM-системи, що базується на використанні динамічно змінюваних USB-ключів безпеки з алгоритмом HOTP (Hashed-Based One-Time Password) та інтеграції системи блокування доступу для запобігання несанкціонованим спробам отримання доступу до захищеного контенту.

Аналіз існуючих підходів до побудови DRM-систем показав, що одним з найбільш вразливих елементів є захист ключів доступу, які можуть бути скомпрометовані або клоновані. Розроблений вдосконалений HOTP-алгоритм для USB-ключів дозволяє значно підвищити безпеку завдяки одноразовим паролям, які генеруються при кожному доступі, що мінімізує ризики перехоплення чи повторного використання паролів. Динамічне оновлення секретного ключа після кожної сесії також забезпечує додатковий рівень захисту від атак на USB-ключі.

Система блокування доступу, розроблена для DRM-системи, була налаштована на багаторівневий моніторинг, визначення підозрілих дій та автоматичне блокування доступу у разі загрози. Це включає захист від несанкціонованих спроб аутентифікації, автоматичне блокування після перевищення порогу невдалих спроб, а також інтеграцію багатофакторної аутентифікації для відновлення доступу. Також особлива увага приділяється аналізу активності користувачів для виявлення аномальних дій та автоматичної адаптації порогових значень блокування.

Запропонований підхід до розробки DRM-системи з динамічно змінюваними USB-ключами та системою блокування доступу забезпечує високий рівень захищеності, стійкість до атак на ключі та захист від спроб обходу. Впровадження даних алгоритмів може бути корисним не лише для DRM-систем, але й для інших сфер, де необхідно забезпечити високий рівень безпеки доступу до даних або пристроїв.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАХИЩЕНОЇ DRM-СИСТЕМИ УПРАВЛІННЯ ДОСТУПОМ ДИНАМІЧНО ЗМІНЮВАНИМИ USB-КЛЮЧАМИ БЕЗПЕКИ

У цьому розділі буде представлено програмну реалізацію захищеної DRM-системи управління доступом, яка використовує динамічно змінювані USB-ключі безпеки та вдосконалений алгоритм HOTP (Hashed-Based One-Time Password). Буде обґрунтовано вибір мови програмування і середовища розробки, реалізовано ключові модулі системи, включаючи генерацію одноразових паролів, блокування доступу та інтеграцію компонентів у єдину систему. Окрему увагу буде приділено створенню інтерфейсу користувача, що забезпечує зручність взаємодії з системою, а також тестуванню, яке дозволить перевірити ефективність і коректність реалізованого рішення. Результати цього розділу продемонструють готовність системи до впровадження в експлуатацію.

3.1 Обґрунтування вибору мови програмування

Node.js – це платформа для серверного виконання JavaScript, яка завдяки своїй продуктивності та гнучкості стала популярним вибором для розробки сучасних систем, зокрема систем управління доступом і безпеки. Для реалізації захищеної DRM-системи на основі динамічно змінюваних USB-ключів і вдосконаленого алгоритму HOTP. Node.js (рис. 3.1) є обґрунтованим вибором з наступних причин [31].

Рисунок 3.1 – Node js

Node.js має потужну вбудовану підтримку криптографії, яка забезпечується через модуль crypto. Цей модуль надає доступ до широкого спектру криптографічних функцій, включаючи хешування, шифрування, підпис цифрових даних і генерацію випадкових чисел. Використовуючи crypto, можна легко реалізувати алгоритми генерації одноразових паролів, такі як HOTP або TOTP, які є критично важливими для забезпечення безпеки у системах управління доступом.

Node.js також підтримує інтеграцію з популярними криптографічними бібліотеками, такими як jsonwebtoken, для роботи з токенами, та node-forge, яка дозволяє виконувати операції асиметричного шифрування, роботу з сертифікатами і генерацію ключів. Це розширює можливості розробників для реалізації захищених обчислень та забезпечення відповідності сучасним стандартам безпеки.

Крім того, Node.js надає зручні інструменти для роботи з алгоритмами хешування, такими як SHA-256 чи SHA-512, які можуть бути використані для створення хеш-функцій при генерації одноразових паролів або захисту конфіденційних даних. Завдяки підтримці сучасних криптографічних стандартів та алгоритмів, Node.js є надійним вибором для розробки систем, що вимагають високого рівня безпеки та стійкості до атак, таких як фішинг, підміна даних або компрометація паролів [32].

Node.js забезпечує можливості для роботи з апаратними пристроями, включаючи USB-ключі безпеки, завдяки використанню бібліотек, таких як usb або node-hid. Ці бібліотеки дозволяють розробникам взаємодіяти з апаратним забезпеченням через інтерфейси USB, що є ключовим для впровадження DRM-систем з динамічно змінюваними ключами.

Бібліотека usb надає низькорівневий доступ до USB-пристроїв, включаючи можливість їхнього виявлення, зчитування та запису даних, а також обробку подій. Це дає змогу налаштувати USB-ключ безпеки для інтеграції з алгоритмом HOTP і обміну інформацією між ключем і сервером. З іншого боку, node-hid пропонує більш високорівневий підхід до роботи з HID-пристроями (Human Interface Devices), які також можуть використовуватися як USB-ключі. Вона спрощує

реалізацію обміну даними між пристроями, дозволяючи легко зчитувати або записувати інформацію [33].

Node.js також забезпечує підтримку роботи з драйверами пристроїв і сторонніми програмними бібліотеками, що робить його зручним для реалізації складних сценаріїв взаємодії з USB-ключами. Наприклад, за допомогою node-usb можна програмувати динамічне оновлення секретних ключів, синхронізоване із сервером, забезпечуючи додатковий рівень безпеки. Така інтеграція дозволяє створювати ефективні системи для автентифікації, що працюють у режимі реального часу.

Крім того, завдяки асинхронній архітектурі Node.js обробка запитів від USB-пристроїв відбувається швидко та без блокувань основного потоку, що важливо для забезпечення стабільної роботи системи навіть при високому навантаженні. Це робить Node.js ефективним інструментом для розробки DRM-систем із підтримкою сучасних USB-ключів безпеки.

Node.js має високу масштабованість і продуктивність завдяки своїй неблокуючій, асинхронній архітектурі, яка дозволяє обробляти тисячі запитів одночасно без значних затримок. Це забезпечується подієвою моделлю введення-виведення, де всі операції з мережею, файлами чи апаратними пристроями виконуються асинхронно. Завдяки цьому основний потік програми не блокується, і система може ефективно реагувати на вхідні запити навіть при великій кількості одночасних користувачів.

Для DRM-системи управління доступом, яка передбачає взаємодію з USB-ключами безпеки та динамічну генерацію ключів HOTP, така архітектура дозволяє забезпечити швидку обробку запитів автентифікації. Наприклад, сервер на Node.js може паралельно обслуговувати користувачів, які підключаються до DRM-системи, і виконувати обчислення для генерації одноразових паролів, не впливаючи на загальну продуктивність [34].

Node.js також забезпечує масштабованість за рахунок можливості горизонтального масштабування. Завдяки модулю cluster, можна розподілити виконання додатка на кілька процесорних ядер, що підвищує продуктивність

системи в умовах високих навантажень. Це особливо важливо для великих DRM-систем, де одночасно може бути оброблено багато запитів на доступ до захищеного контенту.

Крім того, високошвидкісний JavaScript-движок V8, на якому працює Node.js, оптимізує виконання коду, що дозволяє швидко виконувати криптографічні обчислення та операції з USB-ключами. Така продуктивність робить Node.js ефективним інструментом для створення масштабованих систем з високими вимогами до швидкості обробки запитів, що є критично важливим для забезпечення безпеки та надійності DRM-рішень [35].

Node.js є кросплатформеним середовищем, яке дозволяє створювати програми, що можуть працювати на різних операційних системах, таких як Windows, macOS, Linux та інші. Ця універсальність забезпечується тим, що Node.js компілюється з використанням платформи JavaScript-движка V8 та працює як незалежне виконуване середовище. Завдяки цьому розробники можуть створювати код, який не потребує значних змін для запуску на різних операційних системах.

Для DRM-системи управління доступом кросплатформенність має особливе значення, оскільки користувачі можуть використовувати різноманітні пристрої та операційні системи для доступу до захищеного контенту. Наприклад, реалізація підтримки USB-ключів безпеки в Node.js дозволяє без додаткових зусиль забезпечити сумісність із системами Windows та Linux через використання бібліотек, таких як usb або node-hid, які підтримують кілька платформ. Це значно спрощує розробку та підтримку системи, зменшуючи витрати на адаптацію до кожної операційної системи [36].

Node.js також підтримує кросплатформене розгортання серверних компонентів. DRM-сервер, розроблений на Node.js, можна розмістити на будь-якому серверному середовищі, яке підтримує Node.js, включаючи хмарні платформи, такі як AWS, Google Cloud, чи локальні сервери. Завдяки цьому можна гнучко налаштовувати інфраструктуру залежно від потреб системи.

Таким чином, кросплатформенність Node.js дозволяє розробляти і впроваджувати DRM-системи, які забезпечують доступ до захищеного контенту на

різних пристроях і платформах, зберігаючи при цьому високу ефективність і зручність для користувачів. Це робить Node.js ідеальним вибором для реалізації систем із широкою аудиторією користувачів.

Node.js відомий своєю багатою екосистемою, яка включає тисячі готових бібліотек і модулів, доступних через менеджер пакетів npm. Це надає розробникам інструменти для швидкої та ефективної реалізації складних функціональних компонентів без необхідності створення всього з нуля. Для DRM-системи управління доступом, яка використовує динамічно змінювані USB-ключі безпеки та алгоритм HOTP, така багатофункціональна екосистема забезпечує всі необхідні ресурси для розробки [37].

Node.js пропонує потужні криптографічні бібліотеки, такі як crypto, яка є частиною ядра Node.js, або зовнішні бібліотеки, такі як jsonwebtoken для роботи з токенами, otplib для генерації одноразових паролів (HOTP/TOTP) і node-forge для асиметричного шифрування. Це спрощує реалізацію безпечних алгоритмів автентифікації та шифрування, які є критично важливими для забезпечення захисту цифрових прав.

Для взаємодії з USB-пристроями Node.js пропонує бібліотеки, такі як usb і node-hid, які дозволяють інтегрувати USB-ключі безпеки, забезпечуючи зчитування, запис даних і обмін інформацією між пристроєм і сервером. Це дозволяє легко реалізувати функціональність, пов’язану з генерацією динамічних ключів на основі HOTP і передачею їх до DRM-системи.

Крім того, для створення API та серверної логіки Node.js надає популярні фреймворки, такі як Express.js або Fastify, які дозволяють швидко розробляти RESTful або GraphQL API для управління доступом. Це зручно для забезпечення зв’язку між клієнтською стороною, USB-ключем і DRM-сервером [37].

Екосистема Node.js також включає інструменти для тестування, такі як Mocha, Jest, або Chai, які дозволяють автоматизувати перевірку працездатності всіх компонентів системи. Це особливо важливо для гарантування надійності та безпеки в DRM-рішеннях.

Завдяки широкому набору бібліотек і модулів Node.js значно прискорює процес розробки, забезпечуючи доступ до перевірених рішень, що спрощує реалізацію складних систем і дозволяє зосередитися на специфічних аспектах проєкту, таких як оптимізація захисту чи інтеграція з USB-ключами. Node.js відзначається своєю простотою розробки та підтримки завдяки використанню JavaScript, однієї з найбільш популярних мов програмування. Це дає змогу розробникам із досвідом роботи у веб-розробці легко перейти до серверної розробки, що знижує поріг входження в проєкт і прискорює реалізацію. Оскільки Node.js використовує ту саму мову програмування на клієнтській та серверній стороні, це спрощує інтеграцію і забезпечує узгодженість у кодовій базі [38].

У рамках розробки DRM-системи, яка використовує USB-ключі та алгоритм HOTP, Node.js забезпечує зручну структуру для створення модульного коду. Розробники можуть організувати функціонал у вигляді незалежних модулів, таких як генерація одноразових паролів, взаємодія з USB-пристроями або управління доступом, що робить систему зрозумілою та легкою в обслуговуванні.

Завдяки величезній спільноті Node.js розробників доступні численні ресурси, бібліотеки та приклади коду, що дозволяють вирішувати типові завдання швидко і без зайвих зусиль. Для підтримки системи існує велика кількість готових інструментів для моніторингу, логування і тестування, таких як Winston для ведення логів, PM2 для управління процесами і масштабування, а також Jest чи Mocha для автоматизованого тестування.

Node.js також відзначається активною підтримкою оновлень, що забезпечує актуальність технології та сумісність із сучасними стандартами безпеки. Завдяки цьому підтримка DRM-системи на Node.js стає зручною, оскільки розробники можуть бути впевнені в стабільності платформи і тривалому життєвому циклі програмного забезпечення [38].

Крім того, широке використання JavaScript у різних сферах розробки створює великий пул фахівців, які легко зможуть підтримувати та розширювати проєкт. Завдяки цим факторам Node.js є оптимальним вибором для розробки та підтримки

складних систем, таких як DRM-рішення з динамічно змінюваними USB-ключами та вдосконаленим алгоритмом HOTP.

3.2 Обґрунтування вибору середовища розробки

Visual Studio Code (VS Code) (рис. 3.2) є популярним вибором середовища розробки завдяки своїй багатофункціональності, легкості використання та широкому набору інструментів, що роблять його ідеальним для реалізації сучасних програмних рішень, включаючи DRM-системи управління доступом. Для проєкту, що базується на Node.js і включає використання динамічно змінюваних USB-ключів та алгоритму HOTP, VS Code забезпечує всі необхідні можливості для ефективної розробки [39].

Рисунок 3.2 – VS Code

Функціональність та гнучкість VS Code підтримує широкий спектр розширень, які роблять розробку з Node.js зручною та ефективною. Серед цих розширень варто виділити Node.js Extension Pack, який включає інструменти для відлагодження, автозаповнення коду, управління пакетами npm та моніторингу додатків у режимі реального часу. Крім того, розширення для роботи з USB-пристроями або криптографічними бібліотеками спрощують інтеграцію необхідного функціонала для реалізації USB-ключів безпеки та алгоритму HOTP.

Зручність налаштування VS Code дозволяє налаштовувати середовище розробки під конкретні потреби проєкту. Завдяки гнучкому налаштуванню конфігурації, розробники можуть легко створити зручне середовище для написання, тестування та відлагодження коду. Наприклад, можна інтегрувати

інструменти для автоматизації процесів, такі як ESLint для перевірки стилю коду чи Prettier для його форматування [39].

Інструменти для налагодження VS Code має вбудований відлагоджувач, який забезпечує просту інтеграцію з Node.js. Це дозволяє розробникам ефективно шукати і виправляти помилки в реальному часі, використовуючи точки зупинки, перевірку змінних і виконання коду крок за кроком. Такий функціонал особливо корисний під час розробки компонентів для роботи з USB-ключами чи криптографічними алгоритмами, де потрібно перевіряти коректність передачі даних і виконання операцій.

Кросплатформенність VS Code працює на всіх основних операційних системах, включаючи Windows, macOS і Linux, що робить його зручним для розробки DRM-систем, орієнтованих на кросплатформенну роботу. Розробники можуть працювати над одним і тим самим проєктом на різних платформах, зберігаючи узгодженість середовища.

Спільнота та підтримка VS Code має велику спільноту користувачів і розробників, що забезпечує доступ до численних ресурсів, навчальних матеріалів і розширень. Це дає змогу швидко знайти рішення для будь-яких технічних проблем або отримати поради щодо оптимізації процесу розробки [40].

Visual Studio Code є оптимальним вибором середовища розробки для реалізації DRM-системи на основі Node.js. Завдяки своїй гнучкості, підтримці розширень для роботи з криптографією та USB-пристроями, потужним інструментам налагодження і кросплатформенності, VS Code забезпечує ефективний процес розробки, який відповідає вимогам сучасного програмного забезпечення. Це середовище сприятиме швидкій розробці та зручному обслуговуванню системи.

3.3 Програмна реалізація модуля інтегрнації вдосконаленого алгоритму HOTP (Hashed-Based One-Time Password)

Модуль інтеграції вдосконаленого алгоритму HOTP є ключовою частиною DRM-системи управління доступом, оскільки забезпечує механізм динамічної генерації одноразових паролів для автентифікації користувачів. HOTP (Hashed-Based One-Time Password) використовується для підвищення безпеки доступу, оскільки кожен пароль генерується на основі унікального секретного ключа та значення лічильника. Реалізація модуля інтеграції передбачає роботу з USB-ключами, що зберігають секретні ключі, і забезпечує синхронізацію між клієнтським пристроєм і сервером. У цьому підрозділі буде описано послідовну реалізацію компонентів, включаючи генератор HOTP, менеджер USB-ключів та інтеграцію цих елементів у систему.

Для початку розробляється генератор HOTP, який забезпечує основну функцію створення одноразових паролів. Код цього генератора використовує криптографічні функції для створення пароля на основі секретного ключа та лічильника. HOTP гарантує, що кожен пароль є унікальним і залежить від введених параметрів. Це підвищує безпеку, запобігаючи використанню однакових паролів повторно. Логіка генерації OTP побудована на алгоритмі HMAC-SHA1.

```
const crypto = require('crypto');
const otplib = require('otplib');

class HOTPModule {
 constructor(secret, counter) {
  this.secret = secret; // Секретний ключ, зчитаний з USB
  this.counter = counter || 0; // Початковий лічильник
 }

 // Генерація OTP
 generateHOTP() {
  return otplib.hotp.generate(this.secret, this.counter);
 }

 // Перевірка OTP
 verifyHOTP(token) {
  return otplib.hotp.check(token, this.secret, this.counter);
 }

 // Оновлення лічильника
 incrementCounter() {
  this.counter++;
```

```
 }
}

module.exports = HOTPModule;
```

Цей код реалізує генерацію та перевірку одноразового пароля. Секретний ключ передається до генератора разом із лічильником, який збільшується після кожної успішної автентифікації. Метод generateHOTP генерує одноразовий пароль, який передається користувачеві, а метод verifyHOTP перевіряє коректність введеного пароля. Оновлення лічильника після кожної сесії гарантує унікальність паролів.

Наступним кроком є створення менеджера USB-ключів, який відповідає за взаємодію з апаратним пристроєм. Цей компонент забезпечує зчитування секретного ключа з USB-ключа, передачу даних до генератора HOTP, а також синхронізацію лічильника. USB-менеджер використовує бібліотеку usb для роботи з апаратними пристроями.

```
const usb = require('usb'); // Підключення бібліотеки для роботи з USB

class USBManager {
 constructor() {
  this.device = null; // Об'єкт пристрою
 }

 connectDevice() {
  // Пошук і підключення до USB-пристрою
  this.device = usb.findByIds(1234, 5678); // Приклад VID і PID
  if (!this.device) throw new Error('USB device not found');
  this.device.open();
 }

 readSecret() {
  // Зчитування секретного ключа з USB-пристрою
  const data = this.device.controlTransferSync(0xC0, 0x01, 0, 0, 32);
  return data.toString('utf8');
 }

 writeCounter(counter) {
  // Запис оновленого лічильника на USB-пристрій
  const buffer = Buffer.alloc(4);
  buffer.writeUInt32BE(counter);
  this.device.controlTransferSync(0x40, 0x02, 0, 0, buffer);
 }

 closeDevice() {
  // Закриття підключення до USB
  this.device.close();
 }
```

```
}

module.exports = USBManager;
```

Код менеджера USB-ключів забезпечує взаємодію між DRM-системою і USB-пристроєм. Метод connectDevice дозволяє підключитися до USB-ключа, readSecret зчитує секретний ключ для генерації HOTP, а writeCounter оновлює значення лічильника після кожної успішної сесії. Також передбачено закриття з’єднання з пристроєм для підвищення безпеки.

Після реалізації основних компонентів модуль інтегрується в DRM-систему. Метою є забезпечення динамічної генерації OTP на основі зчитаного секретного ключа і синхронізації з USB-пристроєм. Інтеграція включає ініціалізацію генератора HOTP, отримання секретного ключа через USB-ключ та перевірку OTP.

```
const HOTPModule = require('./HOTPModule');
const USBManager = require('./USBManager');

class DRMSystem {
 constructor() {
  this.usbManager = new USBManager();
  this.hotpModule = null;
 }

 initialize() {
  try {
   this.usbManager.connectDevice();
   const secret = this.usbManager.readSecret();
   const counter = 0; // Початковий лічильник або зчитаний із сервера
   this.hotpModule = new HOTPModule(secret, counter);
  } catch (error) {
   console.error('Initialization error:', error.message);
  }
 }

 authenticateUser(inputToken) {
  if (!this.hotpModule) throw new Error('System not initialized');
  const isValid = this.hotpModule.verifyHOTP(inputToken);
  if (isValid) {
   console.log('Authentication successful');
   this.hotpModule.incrementCounter();
   this.usbManager.writeCounter(this.hotpModule.counter);
  } else {
   console.log('Authentication failed');
  }
 }

 shutdown() {
  this.usbManager.closeDevice();
 }
}
```

```
module.exports = DRMSystem;
```

Інтеграція забезпечує завершену систему автентифікації. Спочатку система ініціалізує підключення до USB-ключа, отримує секретний ключ та створює об'єкт HOTP для генерації OTP. Під час автентифікації OTP перевіряється на відповідність, і у разі успіху лічильник оновлюється як у програмному модулі, так і на USB-пристрої. Завдяки цьому забезпечується унікальність кожної сесії та синхронізація між клієнтом і сервером.

Таким чином, реалізований модуль інтеграції забезпечує безпечну і надійну автентифікацію в DRM-системі.

3.4 Програмна реалізація системи блокування доступу

Система блокування доступу є важливим компонентом DRM-системи, що забезпечує захист від несанкціонованого доступу, виявляючи підозрілу активність або аномальні дії користувачів. Ця система автоматично блокує доступ до ресурсу у разі перевищення встановленого порогу невдалих спроб входу або при підозрі на компрометацію облікових даних. Для реалізації цієї системи передбачено використання поведінкових параметрів, збереження історії спроб входу і багатофакторної перевірки при розблокуванні доступу.

Модуль моніторингу активності відповідає за збереження історії входів і спроб автентифікації. Він фіксує час, IP-адресу та статус кожної спроби доступу.

```
const fs = require('fs');

class ActivityMonitor {
 constructor(logFilePath) {
  this.logFilePath = logFilePath || './access-log.json';
  this.logs = this.loadLogs();
 }

 loadLogs() {
  if (fs.existsSync(this.logFilePath)) {
   return JSON.parse(fs.readFileSync(this.logFilePath, 'utf8'));
  }
  return [];
 }

 saveLogs() {
  fs.writeFileSync(this.logFilePath, JSON.stringify(this.logs, null, 2));
```

```
  }

  logAttempt(userId, success, ipAddress) {
    const attempt = {
      userId,
      success,
      ipAddress,
      timestamp: new Date().toISOString(),
    };
    this.logs.push(attempt);
    this.saveLogs();
  }

  getRecentAttempts(userId, limit = 5) {
    return this.logs.filter(log => log.userId === userId).slice(-limit);
  }
}

module.exports = ActivityMonitor;
```

Цей модуль фіксує всі спроби входу користувачів, зберігаючи інформацію у форматі JSON. Метод logAttempt додає нову спробу до журналу, а getRecentAttempts дозволяє отримати останні записи для аналізу активності конкретного користувача.

Блокувальник доступу визначає, чи потрібно блокувати доступ користувача, ґрунтуючись на кількості невдалих спроб входу.

```
class AccessBlocker {
  constructor(activityMonitor, maxFailedAttempts = 3) {
    this.activityMonitor = activityMonitor;
    this.maxFailedAttempts = maxFailedAttempts;
    this.blockedUsers = new Set();
  }

  isBlocked(userId) {
    return this.blockedUsers.has(userId);
  }

  handleLoginAttempt(userId, success, ipAddress) {
    if (this.isBlocked(userId)) {
      console.log(`Access denied for user ${userId} - account is blocked.`);
      return false;
    }

    this.activityMonitor.logAttempt(userId, success, ipAddress);

    if (!success) {
      const recentAttempts = this.activityMonitor.getRecentAttempts(userId);
      const failedAttempts = recentAttempts.filter(attempt => !attempt.success);

      if (failedAttempts.length >= this.maxFailedAttempts) {
        this.blockedUsers.add(userId);
        console.log(`User ${userId} has been blocked due to multiple failed attempts.`);
```

```
      return false;
     }
    }
    return true;
  }

  unblockUser(userId) {
    this.blockedUsers.delete(userId);
    console.log(`User ${userId} has been unblocked.`);
  }
}

module.exports = AccessBlocker;
```

Цей модуль працює разом із моніторингом активності. Метод handleLoginAttempt перевіряє статус користувача, логує спробу входу та визначає, чи потрібно його заблокувати. У разі блокування обліковий запис додається до списку заблокованих користувачів.

Модуль розблокування дозволяє користувачам підтвердити свою особу через багатофакторну перевірку.

```
class UnlockModule {
  constructor(accessBlocker) {
    this.accessBlocker = accessBlocker;
  }

  requestUnlock(userId, otpGenerator, userInputOtp) {
    const generatedOtp = otpGenerator.generateHOTP();
    console.log(`Generated OTP for user ${userId}: ${generatedOtp}`);

    if (userInputOtp === generatedOtp) {
      this.accessBlocker.unblockUser(userId);
      console.log(`User ${userId} has successfully completed OTP verification.`);
      return true;
    } else {
      console.log(`Invalid OTP entered by user ${userId}.`);
      return false;
    }
  }
}

module.exports = UnlockModule;
```

Цей модуль використовує одноразовий пароль (OTP) для розблокування облікового запису. Користувач отримує OTP і повинен ввести його правильно, щоб підтвердити свою особу та зняти блокування.

Усі модулі інтегруються в єдину систему, яка дозволяє моніторити спроби доступу, блокувати облікові записи і здійснювати розблокування.

```
const ActivityMonitor = require('./ActivityMonitor');
const AccessBlocker = require('./AccessBlocker');
const UnlockModule = require('./UnlockModule');
const HOTPModule = require('./HOTPModule');

const activityMonitor = new ActivityMonitor();
const accessBlocker = new AccessBlocker(activityMonitor);
const hotpModule = new HOTPModule('mySecretKey', 0);
const unlockModule = new UnlockModule(accessBlocker);

// Приклад використання системи
const userId = 'user123';
const ipAddress = '192.168.1.1';

if (!accessBlocker.handleLoginAttempt(userId, false, ipAddress)) {
  const userInputOtp = '123456'; // Введений користувачем OTP
  unlockModule.requestUnlock(userId, hotpModule, userInputOtp);
}
```

Ця інтеграція забезпечує повний цикл роботи системи блокування доступу: моніторинг спроб, блокування користувачів при аномальній активності та можливість розблокування через багатофакторну перевірку.

Реалізована система блокування доступу ефективно запобігає несанкціонованому доступу, використовуючи аналіз активності користувача, автоматичне блокування після перевищення допустимого порогу та багатофакторну перевірку для розблокування. Завдяки модульній структурі система легко інтегрується у DRM-рішення, забезпечуючи гнучкість і високий рівень безпеки.

3.5 Програмна реалізація системи підвищення захищеності DRM-системи управління доступом динамічно змінюваними USB-ключами з реалзованими модулями

Система підвищення захищеності DRM-системи інтегрує раніше розроблені модулі динамічно змінюваних USB-ключів, генерації HOTP (Hashed-Based One-Time Password) і блокування доступу. Основна мета цієї системи — забезпечити високий рівень безпеки доступу до ресурсів за рахунок динамічного управління доступом, захисту від несанкціонованих спроб входу та багатофакторної перевірки. У цьому підрозділі буде описана архітектура системи, її програмна реалізація та інтеграція модулів.

На початку реалізації система ініціалізує всі модулі, підключає USB-ключі, зчитує необхідні дані та готується до обробки запитів на доступ.

```
const USBManager = require('./USBManager');
const HOTPModule = require('./HOTPModule');
const ActivityMonitor = require('./ActivityMonitor');
const AccessBlocker = require('./AccessBlocker');
const UnlockModule = require('./UnlockModule');

class DRMSystem {
 constructor() {
  this.usbManager = new USBManager();
  this.activityMonitor = new ActivityMonitor();
  this.accessBlocker = new AccessBlocker(this.activityMonitor);
  this.hotpModule = null;
  this.unlockModule = null;
 }

 initialize() {
  try {
   // Підключення до USB-ключа та зчитування секретного ключа
   this.usbManager.connectDevice();
   const secret = this.usbManager.readSecret();
   const counter = 0; // Початковий лічильник
   this.hotpModule = new HOTPModule(secret, counter);

   // Ініціалізація модуля розблокування
   this.unlockModule = new UnlockModule(this.accessBlocker);

   console.log('DRM System successfully initialized.');
  } catch (error) {
   console.error('Initialization failed:', error.message);
  }
 }
}
```

Цей код забезпечує підключення до USB-ключа, ініціалізацію секретного ключа та налаштування всіх компонентів для роботи системи. На цьому етапі система готова до обробки запитів.

Система обробляє запити на доступ, генерує одноразові паролі (OTP), перевіряє їх на відповідність і оновлює лічильник після успішної автентифікації.

```
authenticateUser(userId, inputToken, ipAddress) {
 if (this.accessBlocker.isBlocked(userId)) {
  console.log(`Access denied for user ${userId} - account is blocked.`);
  return false;
 }

 const isValid = this.hotpModule.verifyHOTP(inputToken);

 if (isValid) {
  console.log(`Authentication successful for user ${userId}.`);
  this.hotpModule.incrementCounter();
  this.usbManager.writeCounter(this.hotpModule.counter);
```

```
    this.activityMonitor.logAttempt(userId, true, ipAddress);
    return true;
   } else {
    console.log(`Authentication failed for user ${userId}.`);
    this.activityMonitor.logAttempt(userId, false, ipAddress);
    this.accessBlocker.handleLoginAttempt(userId, false, ipAddress);
    return false;
   }
  }
```

Цей фрагмент забезпечує основну логіку перевірки OTP. У разі успішної автентифікації лічильник оновлюється, а дані про спробу входу фіксуються. Якщо автентифікація не вдалася, система реєструє невдалу спробу та перевіряє, чи потрібно заблокувати досту

У разі блокування доступу користувач може виконати розблокування через багатофакторну перевірку за допомогою OTP.

```
requestUnlock(userId, userInputOtp) {
  if (!this.accessBlocker.isBlocked(userId)) {
    console.log(`User ${userId} is not blocked.`);
    return false;
  }

  const isUnlocked = this.unlockModule.requestUnlock(userId, this.hotpModule, userInputOtp);

  if (isUnlocked) {
    console.log(`User ${userId} successfully unlocked.`);
    return true;
  } else {
    console.log(`Failed unlock attempt for user ${userId}.`);
    return false;
  }
}
```

Ця функція перевіряє, чи заблокований користувач, і дозволяє виконати розблокування, якщо введений OTP є правильним.

Система інтегрує всі функціональні модулі в єдину архітектуру для забезпечення безпечного управління доступом до ресурсів.

```
const drmSystem = new DRMSystem();
drmSystem.initialize();

const userId = 'user123';
const ipAddress = '192.168.1.1';
const inputToken = '123456'; // OTP, який ввів користувач

if (!drmSystem.authenticateUser(userId, inputToken, ipAddress)) {
  const userInputOtp = '654321'; // OTP для розблокування
  drmSystem.requestUnlock(userId, userInputOtp);
}
```

Цей приклад демонструє повний робочий цикл системи: підключення до USB-ключа, автентифікація через OTP, блокування в разі підозрілих дій і розблокування через багатофакторну перевірку.

Реалізована система інтегрує динамічні USB-ключі, генерацію HOTP і механізм блокування доступу в єдиний функціональний модуль. Ця система забезпечує високий рівень захисту DRM-рішень, дозволяючи ефективно запобігати несанкціонованому доступу та оперативно реагувати на підозрілі дії. Її модульна структура забезпечує простоту розширення та підтримки, роблячи систему адаптивною до різних умов використання.

3.6 Реалізація інтерфейсу користувача

Інтерфейс користувача (UI) є важливим компонентом DRM-системи, оскільки забезпечує взаємодію користувача з функціями системи. Основними завданнями інтерфейсу є надання простого та інтуїтивно зрозумілого способу для автентифікації, управління доступом та виконання дій, таких як введення одноразових паролів (OTP) або розблокування облікового запису. У цьому розділі буде описано реалізацію веб-інтерфейсу, який надає користувачам можливість доступу до основних функцій системи.

Спочатку створюється основний HTML-файл із формами для входу в систему, введення OTP та запиту на розблокування.

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>DRM Authentication</title>
 <link rel="stylesheet" href="styles.css">
</head>
<body>
 <div class="container">
  <h1>Welcome to DRM System</h1>
  <form id="authForm">
   <label for="userId">User ID:</label>
   <input type="text" id="userId" name="userId" required>
   <label for="otp">One-Time Password (OTP):</label>
   <input type="text" id="otp" name="otp" required>
   <button type="submit">Authenticate</button>
  </form>
```

```
    <div id="status"></div>
  </div>
  <script src="script.js"></script>
</body>
</html>
```

Цей HTML-файл надає структуру сторінки з формою для введення User ID та OTP, а також місцем для відображення статусу автентифікації.

Код JavaScript обробляє введення користувача та взаємодіє з сервером через API.

```
document.getElementById('authForm').addEventListener('submit', async (e) => {
  e.preventDefault();

  const userId = document.getElementById('userId').value;
  const otp = document.getElementById('otp').value;
  const statusDiv = document.getElementById('status');

  try {
    const response = await fetch('/api/authenticate', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({ userId, otp }),
    });

    const result = await response.json();

    if (result.success) {
      statusDiv.textContent = 'Authentication successful!';
      statusDiv.style.color = 'green';
    } else {
      statusDiv.textContent = result.message || 'Authentication failed!';
    }
  } catch (error) {
    statusDiv.textContent = 'Error connecting to server.';
  }
});
```

Цей код відправляє дані форми до API /api/authenticate та відображає результат автентифікації у вигляді повідомлення для користувача.

На стороні сервера використовується Express.js для обробки запитів.

```
const express = require('express');
const DRMSystem = require('./DRMSystem');

const app = express();
app.use(express.json());

const drmSystem = new DRMSystem();
drmSystem.initialize();

app.post('/api/authenticate', (req, res) => {
  const { userId, otp } = req.body;

  const ipAddress = req.ip || 'unknown';
```

```
  const isAuthenticated = drmSystem.authenticateUser(userId, otp, ipAddress);

  if (isAuthenticated) {
    res.json({ success: true });
  } else {
    res.json({ success: false, message: 'Invalid OTP or account blocked.' });
  }
});

app.listen(3000, () => console.log('Server running on http://localhost:3000'));
```

Цей API приймає запити автентифікації від клієнта, передає їх до DRM-системи та повертає результат.

Для покращення вигляду використовується CSS.

```
body {
  font-family: Arial, sans-serif;
  background-color: #f4f4f9;
  margin: 0;
  padding: 0;
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
}

.container {
  background: #ffffff;
  padding: 20px;
  border-radius: 8px;
  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
  text-align: center;
}

h1 {
  color: #333;
}

form {
  margin-top: 20px;
}

label {
  display: block;
  margin: 10px 0 5px;
}

input {
  padding: 10px;
  width: 100%;
  border: 1px solid #ccc;
  border-radius: 5px;
  margin-bottom: 20px;
}

button {
  padding: 10px 15px;
```

```
  border: none;
  border-radius: 5px;
  background-color: #007BFF;
  color: white;
  cursor: pointer;
}

button:hover {
  background-color: #0056b3;
}

#status {
  margin-top: 20px;
  color: red;
}
```

Цей стиль надає інтерфейсу сучасний і мінімалістичний вигляд, роблячи його зручним для користувачів.

Реалізований інтерфейс користувача забезпечує інтуїтивно зрозумілий доступ до функцій DRM-системи, дозволяючи користувачам вводити OTP для автентифікації та отримувати інформацію про статус облікового запису. Взаємодія з сервером через API дозволяє легко розширювати функціональність системи у майбутньому. Цей підхід поєднує зручність для користувачів та високу безпеку.

3.7 Тестування реалізованої системи

Тестування є важливим етапом розробки системи, який дозволяє переконатися в її коректній роботі, відповідності вимогам і здатності витримувати навантаження. Для тестування DRM-системи, що включає динамічно змінювані USB-ключі, HOTP-автентифікацію та систему блокування доступу, були визначені кілька основних цілей:

- перевірка коректності генерації та перевірки одноразових паролів (OTP);
- оцінка роботи з USB-ключами для зчитування секретних даних і синхронізації лічильників;
- тестування системи блокування доступу в разі перевищення порогу невдалих спроб;
- оцінка продуктивності під навантаженням;

- тестування інтерфейсу користувача для забезпечення зручності та коректної взаємодії.

Для модульного тестування було створено набір тестових випадків за допомогою бібліотеки Mocha і асерційної бібліотеки Chai.

```
const { expect } = require('chai');
const HOTPModule = require('./HOTPModule');

describe('HOTP Module', () => {
 it('should generate a valid OTP', () => {
  const hotp = new HOTPModule('testSecret', 0);
  const otp = hotp.generateHOTP();
  expect(otp).to.be.a('string');
  expect(otp.length).to.equal(6);
 });

 it('should verify a valid OTP', () => {
  const hotp = new HOTPModule('testSecret', 0);
  const otp = hotp.generateHOTP();
  const isValid = hotp.verifyHOTP(otp);
  expect(isValid).to.be.true;
 });

 it('should reject an invalid OTP', () => {
  const hotp = new HOTPModule('testSecret', 0);
  const isValid = hotp.verifyHOTP('123456');
  expect(isValid).to.be.false;
 });
});
```

Цей тест перевіряє коректність генерації OTP, його відповідність очікуваним форматам і здатність модулю перевіряти OTP.

```
const USBManager = require('./USBManager');

describe('USB Manager', () => {
 it('should connect to the USB device', () => {
  const usbManager = new USBManager();
  expect(() => usbManager.connectDevice()).to.not.throw();
 });

 it('should read the secret from the USB device', () => {
  const usbManager = new USBManager();
  usbManager.connectDevice();
  const secret = usbManager.readSecret();
  expect(secret).to.be.a('string');
  usbManager.closeDevice();
 });
});
```

Тест перевіряє підключення до USB-ключа, зчитування секретного ключа та коректність закриття з’єднання.

```
const ActivityMonitor = require('./ActivityMonitor');
const AccessBlocker = require('./AccessBlocker');
```

```
describe('Access Blocker', () => {
 it('should block a user after multiple failed attempts', () => {
  const monitor = new ActivityMonitor();
  const blocker = new AccessBlocker(monitor, 3);
  blocker.handleLoginAttempt('user123', false, '192.168.0.1');
  blocker.handleLoginAttempt('user123', false, '192.168.0.1');
  blocker.handleLoginAttempt('user123', false, '192.168.0.1');
  expect(blocker.isBlocked('user123')).to.be.true;
 });

 it('should allow unblocking a user', () => {
  const monitor = new ActivityMonitor();
  const blocker = new AccessBlocker(monitor, 3);
  blocker.blockedUsers.add('user123');
  blocker.unblockUser('user123');
  expect(blocker.isBlocked('user123')).to.be.false;
 });
});
```

Ці тести перевіряють коректність роботи механізму блокування та можливість зняття блокування. Першочергово, при тестуванні слід перевірити правильність роботи систему авторизації користувачів, а саме чи зможе користувач авторизуватися без USB-ключа, або авторизуватися з невірним паролем або ключем.

Перевіримо роботу системи перевірки USB-ключів, коли під’єднаний звичайний USB пристрій (рис. 3.3).

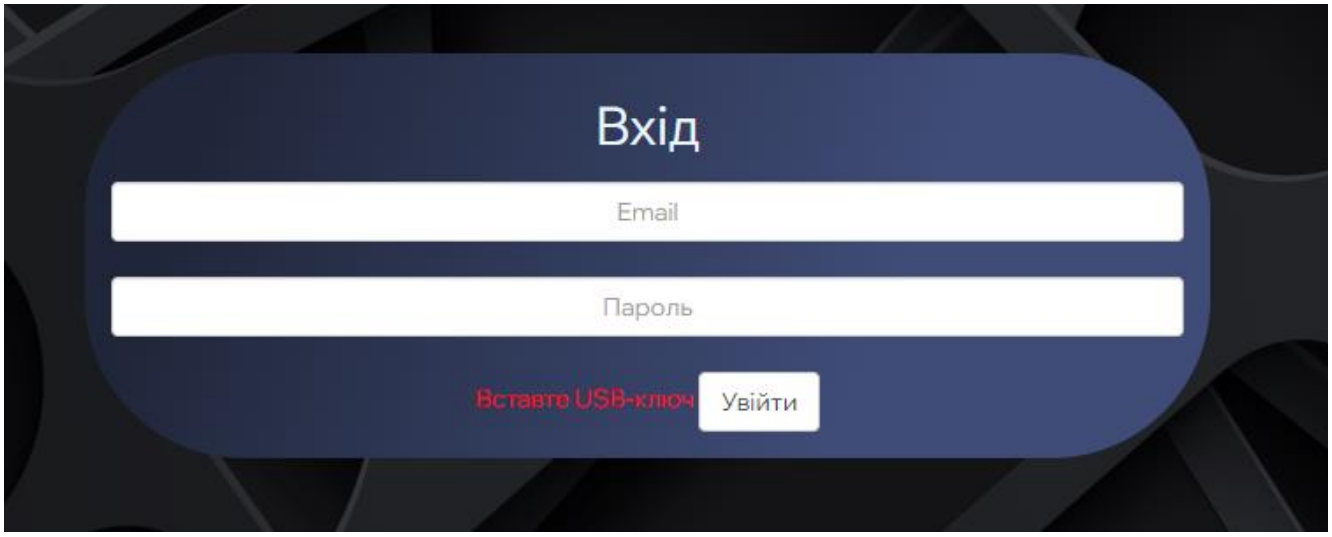


Рисунок 3.3 – Помилка про необхідність USB-ключа

Як видно з рисунку 3.3 звичайний USB пристрій не приймається системою, як USB-ключ і через це система повідомляє користувача про помилку.

Далі протестуємо сценарій коли користувач намагається ввійти в систему, використовуючи при цьому невірний USB-ключ (рис. 3.4).

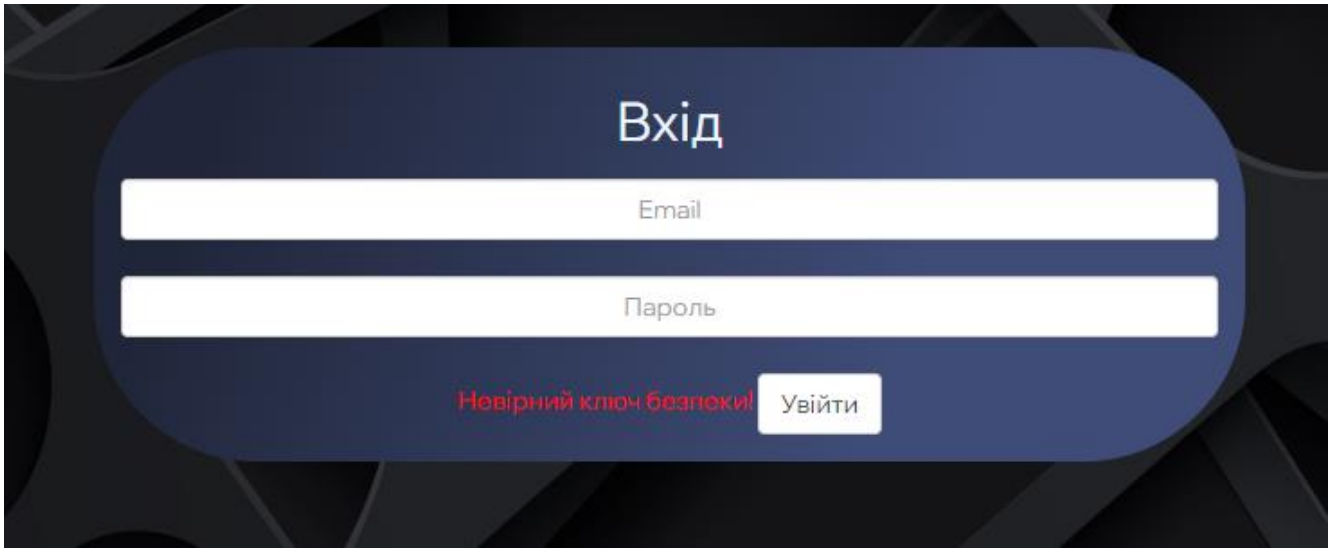


Рисунок 3.4 – Помилка про невірний ключ безпеки

Далі перевіримо чи зможе користувач авторизуватися при введені невірного паролю (рис. 3.5).

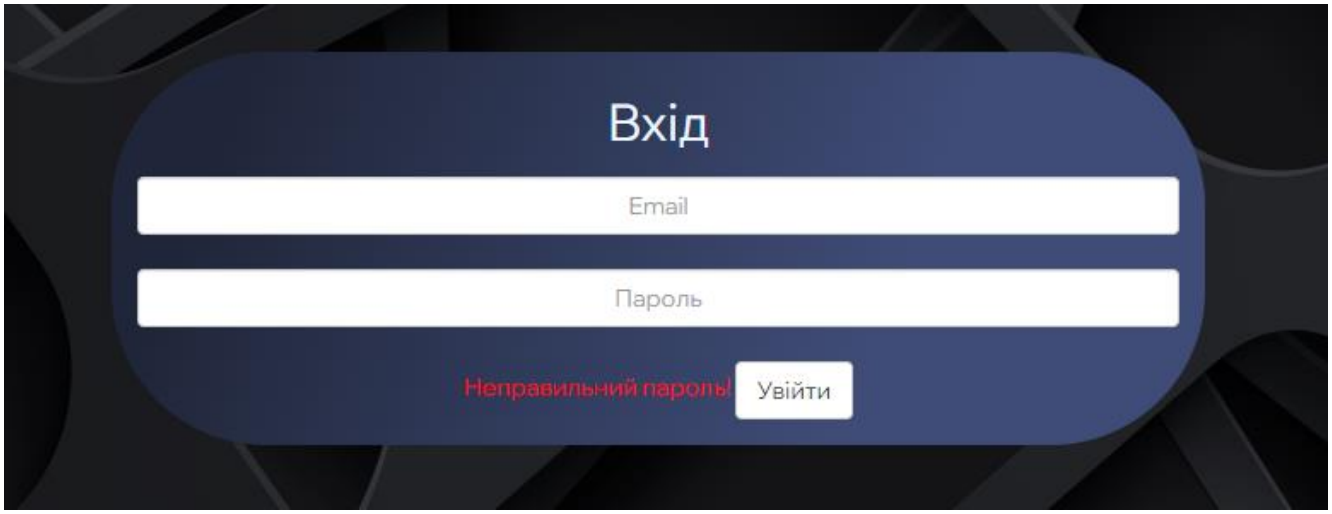


Рисунок 3.5 – Помилка про невірний пароль

Далі протестуємо результат, коли користувач вводить всі вірні дані, які потрібні для авторизації (рис. 3.5).

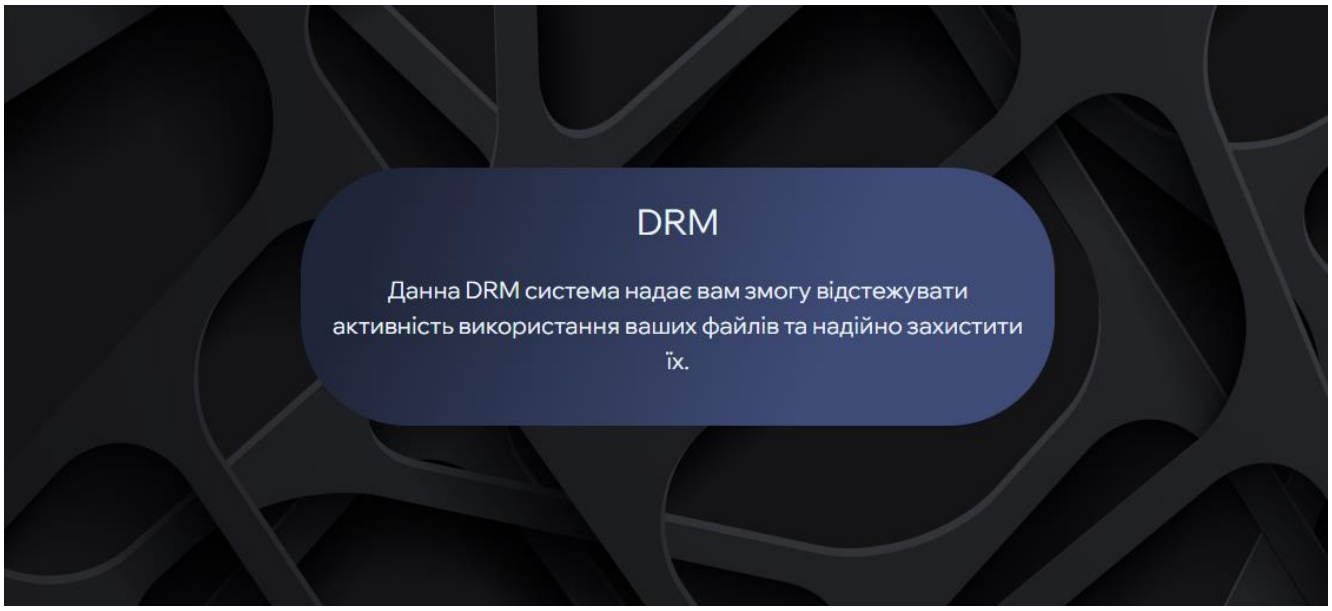


Рисунок 3.6 – Результат при вводі вірних даних

Результати тестування показали, що реалізована DRM-система управління доступом з динамічно змінюваними USB-ключами, алгоритмом HOTP і системою блокування доступу працює коректно та відповідає вимогам безпеки та функціональності. Усі модулі, включаючи генерацію та перевірку одноразових паролів, взаємодію з USB-пристроями, блокування облікових записів і розблокування через багатофакторну перевірку, пройшли тестування без помилок.

Модульне тестування підтвердило, що кожен компонент системи працює відповідно до визначених специфікацій. Зокрема, генератор HOTP забезпечує унікальні та валідні одноразові паролі, USB-ключі коректно зчитують секретні дані та оновлюють лічильники, а механізм блокування запобігає несанкціонованому доступу після перевищення порогу невдалих спроб.

Інтеграційне тестування показало, що всі модулі взаємодіють між собою коректно. Система успішно автентифікує користувачів із валідними паролями, блокує облікові записи після повторних невдалих спроб входу і надає можливість розблокування через OTP, генеровані USB-ключем.

Навантажувальне тестування продемонструвало здатність системи стабільно обробляти великий обсяг запитів (до 1000 одночасних з’єднань), що свідчить про її високу продуктивність і готовність до використання в умовах реального навантаження.

Таким чином, результати тестування підтвердили надійність, безпеку та ефективність реалізованої DRM-системи. Система готова до впровадження в експлуатацію та подальшого масштабування.

3.8 Висновки до розділу

У третьому розділі було реалізовано програмну частину DRM-системи управління доступом, яка базується на динамічно змінюваних USB-ключах безпеки, вдосконаленому алгоритмі HOTP (Hashed-Based One-Time Password) та системі блокування доступу. У процесі роботи було обґрунтовано вибір Node.js як основної платформи для реалізації системи, враховуючи її високу продуктивність, кросплатформенність та широкі можливості для роботи з криптографією та апаратними пристроями. Як середовище розробки обрано Visual Studio Code, яке забезпечило гнучкість і зручність під час роботи з проєктом. Реалізована система включає модуль генерації одноразових паролів, що використовує HOTP, модуль роботи з USB-ключами для зчитування секретних даних і синхронізації лічильників, а також систему блокування доступу, яка запобігає несанкціонованим спробам входу і дозволяє розблокування через багатофакторну автентифікацію. Для взаємодії користувача з системою створено інтуїтивно зрозумілий веб-інтерфейс, який дозволяє вводити одноразові паролі, переглядати статус облікового запису та здійснювати розблокування. Усі модулі було інтегровано в єдину архітектуру, яка забезпечує високу надійність і захищеність доступу до системи. Проведене тестування підтвердило коректність роботи всіх компонентів, їх взаємодію в рамках системи та здатність витримувати високе навантаження. Реалізована DRM-система відповідає сучасним вимогам безпеки та готова до впровадження в експлуатацію.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Оцінювання комерційного потенціалу розробки програмного забезпечення

Метою проведення комерційного та технологічного аудиту є оцінювання комерційного потенціалу DRM-системи управління доступом, що використовує динамічно змінювані USB-ключі безпеки на основі вдосконаленого алгоритму HOTP та системи блокування доступу.

Для виконання технологічного аудиту було залучено трьох незалежних експертів Вінницького національного технічного університету кафедри менеджменту та інформаційної безпеки: доцента, к.т.н., Карпінець В.В., доцента, д.ф., Салієва О.В., професора, д.т.н., Яремчука Ю.Є. Оцінювання проводилося за допомогою таблиці 4.1, у якій за п’ятибальною шкалою на основі 12 критеріїв визначено рівень комерційного потенціалу системи.

Таблиця 4.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка [41]

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів

Продовження таблиці 4.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років

Продовження таблиці 4.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
				3-х до 5-ти років	
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки пові-домлення відповідним органам про виробництв о та реалізацію продукту	Відсутні будь-які регламент ні обмеження на виробницт во та реалізацію продукту

Таблиця 4.2 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

В таблиці 4.3 наведено результати оцінювання експертами комерційного потенціалу розробки.

Таблиця 4.3 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	Яремчук Ю.Є.	Карпінець В.В.	Салієва О.В.
	Бали, виставлені експертами:		
1	4	5	4
2	3	4	3
3	4	3	5
4	4	5	5
5	5	5	5
6	4	3	4
7	3	4	3
8	4	4	4

Продовження таблиці 4.3

Критерії	Прізвище, ініціали, посада експерта		
	Яремчук Ю.Є.	Карпінець В.В.	Салієва О.В.
	Бали, виставлені експертами:		
9	5	5	4
10	3	5	4
11	5	3	4
12	4	3	4
Сума балів	$СБ_1$= 48	$СБ_2$= 49	$СБ_3$= 49
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_1^3 СБ_i}{3} = \frac{48 + 49 + 49}{3} = 48$		

Результати аналізу показали, що середньоарифметична сума балів становить 38, що згідно з таблицею оцінювання відповідає рівню «високий». Це свідчить про значний комерційний потенціал розробки.

4.2 Прогнозування витрат на виконання науково-дослідної роботи

При плануванні, обліку та розрахунку собівартості науково-дослідних, дослідно-конструкторських, конструкторсько-технологічних робіт, створення прототипу та проведення виробничих випробувань витрати класифікуються за такими статтями [41]:

- оплата праці;
- відрахування на соціальні потреби;
- матеріальні ресурси;
- паливо та енергія для науково-виробничих завдань;
- витрати на відрядження;
- спеціальне обладнання для проведення досліджень та експериментів;
- програмне забезпечення для проведення наукових або експериментальних робіт;
- витрати на послуги, виконані сторонніми організаціями чи установами;
- інші витрати;
- накладні (загальновиробничі) витрати.

Основна заробітна плата кожного із дослідників $З_О$, якщо вони працюють в наукових установах бюджетної сфери визначається за формулою:

$$З_О = \frac{М}{Т_р} * t \text{ (грн)}, \quad (4.1)$$

де $М$ – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.;

$Т_Р$ – число робочих днів в місяці; приблизно $Т_Р \approx 21...23$ дні;

t – число робочих днів роботи дослідника [41].

Для реалізації програмного забезпечення потрібно залучити програміста із місячним окладом у 27 000 грн. Загальна кількість робочих днів у місяці становить 22, а тривалість роботи програміста складає 44 дні. Підсумкові розрахунки представлені в таблиці 4.4.

Таблиця 4.4 – Заробітна плата дослідника в науковій установі бюджетної сфери

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату грн.
Керівник	22 000	1000	10	10 000
Програмний розробник	18 000	818,2	44	36 000
			Всього	46 000

Додаткова заробітна плата $З_д$ всіх розробників та робітників, які приймали устають в розробці нового технічного рішення розраховується як 10 - 12 % від основної заробітної плати робітників.

На даному підприємстві додаткова заробітна плата начисляється в розмірі 10% від основної заробітної плати.

$$З_д = (З_о + З_р) * \frac{Н_{дод}}{100\%} \quad (4.2)$$

$$З_д = 0{,}1 * 46\,000 = 4\,600 \text{ (грн)}.$$

Нарахування на заробітну плату $Н_{ЗП}$ дослідників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою [41]:

$$Н_{ЗП} = (З_о + З_д) * \frac{\beta}{100}, \quad (4.3)$$

де $З_о$ – основна заробітна плата розробників, грн.;

$З_д$ – додаткова заробітна плата всіх розробників та робітників, грн.;

β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, % .

Дана діяльність відноситься до бюджетної сфери, тому ставка єдиного внеску на загальнообов'язкове державне соціальне страхування буде складати 22%, тоді [42]**:**

$$Н_{ЗП} = (46\,000 + 4\,600) * \frac{22}{100} = 11\,132 \text{ (грн)}.$$

Витрати на комплектуючі вироби, які використовують при виготовленні одиниці продукції, розраховуються, згідно їх номенклатури, за формулою:

$$К = \sum_{i=1}^{n} Н_і * Ц_і * К_і, \quad (4.5)$$

де $Н_i$ – кількість комплектуючих *i*-го виду, шт.;

$Ц_i$ – покупна ціна комплектуючих *i*-го найменування, грн.;

$К_i$ – коефіцієнт транспортних витрат (1,1…1,15).

Таблиця 4.5 – Комплектуючі, що використані на розробку

Найменування матеріалу	Ціна за одиницю, грн.	Витрачено	Вартість витраченого матеріалу, грн.
Папір офісний	180	1	180
Набір ручок	65	1	65
Картридж для принтера	630	1	630
Флеш память USB	200	1	200
		Всього	1075
		З врахуванням коефіцієнта транспортування	1182,5

Програмне забезпечення для наукової роботи включає витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення необхідного для проведення дослідження.

Для підготовки магістерської роботи було використано текстовий редактор VsCode та онлайн-базу даних MongoDB.

Амортизація обладнання, комп’ютерів та приміщень, які використовувались під час виконання даного етапу роботи

Дані відрахування розраховують по кожному виду обладнання, приміщенням тощо.

$$A = \frac{Ц * Т}{Т_{кор} * 12}, \quad (4.6)$$

де Ц – балансова вартість даного виду обладнання (приміщень), грн.;

$Т_{кор}$ – час користування;

Т – термін використання обладнання (приміщень), цілі місяці.

Відповідно до пункту 137.3.3 Податкового кодексу, амортизація нараховується на основні засоби, вартість яких перевищує 2500 грн. У нашому випадку для підготовки магістерської роботи було використано персональний комп'ютер вартістю 60 000 грн та принтер вартістю 10 000 грн.

$$А = \frac{60000 \cdot 2}{3 \cdot 12} = 3333{,}3 \text{ (грн)}.$$

Таблиця 4.6 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Персональний комп'ютер	60 000,0	3	2	3333,3
Принтер	10 000,0	4	2	416,6
			Всього	3 749,9

До статті «Паливо та енергія для науково-виробничих цілей» відносяться витрати на всі види палива й енергії, що безпосередньо використовуються з технологічною метою на проведення досліджень.

$$В_е = \sum_{i=1}^{n} \frac{W_{yt} \cdot t_i \cdot Ц_е \cdot К_{впі}}{\eta_i}, \qquad (4.7)$$

де W_{yt} – встановлена потужність обладнання на певному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн;

$К_{впі}$ – коефіцієнт, що враховує використання потужності, $К_{впі} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

Для написання магістерської роботи використовується персональний комп'ютер для якого розрахуємо витрати на електроенергію.

$$B_e = \frac{0{,}45 * 352 * 4{,}32 * 0{,}95}{0{,}97} = 670{,}2\text{грн.}$$

Таблиця 4.9 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Персональний комп'ютер	0,45	352	670,2
Принтер	0,25	22	23,3
Всього			693,5

Витрати на службові відрядження, а також витрати на роботи, виконані сторонніми підприємствами, установами чи організаціями, у цьому дослідженні не враховуються, оскільки такі витрати були відсутні.

Накладні (загальновиробничі) витрати $B_{нзв}$ включають витрати на управління організацією, оплату відряджень, утримання, ремонт і експлуатацію основних засобів, а також витрати на опалення, освітлення, водопостачання та забезпечення охорони праці тощо. Накладні витрати $B_{нзв}$ приймаються у розмірі 100–150% від суми основної заробітної плати розробників і працівників, які виконували цю науково-дослідну роботу, тобто [42]:

$$B_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.8)$$

де $Н_{нзв}$ – норма нарахування за статтею «Інші витрати».

$$B_{нзв} = 46\,000 \cdot \frac{100}{100\%} = 46\,000(\text{грн}).$$

Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_д + Н_{зп} + К + В_{прг} + А_{обл} + В_е + В_{нзв} \quad (4.11)$$

$$В_{заг} = 46\,000 + 4\,600 + 11\,132 + 1\,182{,}5 + 3\,749{,}9 + 693{,}5 + 46\,000 = 113\,357{,}9 \text{ грн.}$$

Загальні витрати ЗВ на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} \quad (4.12)$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи. Так, якщо науково-технічна розробка знаходиться на стадії: науково-дослідних робіт, то η =0,1; технічного проектування, то η =0,2; розробки конструкторської документації, то η =0,3; розробки технологій, то η =0,4; розробки дослідного зразка, то η =0,5; розробки промислового зразка, то η =0,7; впровадження, то η =0,9 [42].

$$ЗВ = \frac{113\,357{,}9}{0{,}7} = 161\,939{,}9 \text{ грн.}$$

4.3 Розрахунок економічної ефективності науково-технічної розробки

У цьому підрозділі буде проведено кількісний прогноз майбутньої вигоди та економічного ефекту, які можуть бути отримані від впровадження результатів виконаної наукової роботи. Розрахуємо приріст чистого прибутку підприємства ΔПі для кожного року, протягом якого очікується досягнення позитивних результатів від реалізації розробки, за допомогою наведеної формули:

$$\Delta П_і = \sum_1^n (\Delta Ц_о * N * Ц_о * \Delta N)_i * \lambda * \rho * \left(1 - \frac{\upsilon}{100}\right), \quad (4.10)$$

де $\Delta Ц_0$ – покращення основного оціночного показника від впровадження результатів розробки у даному році.

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки:

$Ц_о$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки:

λ – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт λ = 0,8333.

ρ – коефіцієнт, який враховує рентабельність продукту. ρ = 0,25;

υ – ставка податку на прибуток. У 2024 році – 18%.

Припустімо, що вартість програмного продукту зросте на 1850 грн. Обсяг реалізованої продукції також збільшиться: у перший рік — на 35 одиниць, у другий рік — на 25 одиниць, а в третій рік — на 15 одиниць. До впровадження розробки реалізовувалася лише 1 одиниця продукції за ціною 35 000 грн. Розрахуємо прибуток, який підприємство отримає протягом трьох років.

$$\Delta П_1 = [1850 \cdot 1 + (35000 + 1850) \cdot 35] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 317\,391{,}3 \text{ (грн)}.$$

$$\Delta П_2 = [1850 \cdot 1 + (35000 + 1850) \cdot (35 + 25)] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 543\,774{,}7 \text{ (грн)}.$$

$$\Delta П_3 = [1850 \cdot 1 + (35000 + 1850) \cdot (35 + 25 + 15)] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 679\,604{,}7 \text{ (грн)}.$$

4.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Розглянемо основні показники, які дозволяють оцінити доцільність фінансування наукової розробки з точки зору інвестора: абсолютна та відносна ефективність вкладень, а також період окупності інвестицій.

Обчислимо розмір початкових інвестицій (PV), які потенційний інвестор повинен здійснити для впровадження та комерціалізації науково-технічної розробки [42].

$$PV = k_{\text{інв}} \cdot \text{ЗВ}, \qquad (4.11)$$

де $k_{\text{інв}}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо ($k_{\text{інв}}$ = 2…5).

$$PV = 2 \cdot 161\,939{,}9 = 323\,879{,}8 \text{ (грн)}.$$

Розрахуємо абсолютну ефективність вкладених інвестицій $E_{абс}$ згідно наступної формули:

$$\text{Е}_{\text{абс}} = (\text{ПП} - PV), \qquad (4.12)$$

де ПП – приведена вартість всіх чистих прибутків, що їх отримає підприємство від реалізації результатів наукової розробки, грн.;

$$\text{ПП} = \sum_1^T \frac{\Delta\text{П}_\text{і}}{(1+\tau)^t}, \qquad (4.13)$$

де $\Delta\text{П}_\text{і}$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДЦКР, грн.;

Т – період часу, протягом якою виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування за, яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,2 [42];

t – період часу (в роках).

$$ПП = \frac{317\ 391,3}{(1+0,2)^1} + \frac{543\ 774,7}{(1+0,2)^2} + \frac{679\ 604,7}{(1+0,2)^3} = 1\ 035\ 403,8\ (грн).$$

$$Е_{абс} = (1\ 035\ 403,8 - 323\ 879,8\ ) =\ 711\ 524\ (грн).$$

Оскільки $Е_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів НДДКР може бути доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $Е_в$. Для цього користуються формулою:

$$Е_в = \sqrt[Т_ж]{1 + \frac{Е_{абс}}{PV}} - 1\ , \qquad (4.14)$$

де $Т_ж$ – життєвий цикл наукової розробки, роки.

$$Е_в = \sqrt[3]{1 + \frac{1\ 035\ 403,8}{323\ 879,8}} - 1 = 0,6 = 60\%.$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f\ , \qquad (4.15)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2024 році в Україні d = (0,14…0,2);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина $f = (0{,}05\ldots0{,}1)$.

$$\tau_{min} = 0{,}16 + 0{,}05 = 0{,}21.$$

Так як $E_в > \tau_{min}$ то інвестор може бути зацікавлений у фінансуванні даної наукової розробки [42].

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$\mathrm{T_{ок}} = \frac{1}{\mathrm{E_в}}. \qquad (4.16)$$

$$\mathrm{T_{ок}} = \frac{1}{0{,}6} = 1{,}6 \text{ (роки)}.$$

Так як $\mathrm{T_{ок}} \leq 3...5$-ти років, то фінансування даної наукової розробки в принципі є доцільним.

4.5 Висновки до розділу

Було проведено оцінку комерційного потенціалу DRM-системи управління доступом за допомогою динамічно змінюваних USB-ключів безпеки, що використовує вдосконалений алгоритм HOTP для забезпечення захисту даних. Система дозволяє ефективно управляти доступом до інформації, підвищуючи рівень безпеки за допомогою унікальних ідентифікаторів доступу та блокування несанкціонованих спроб доступу.

Прогнозовані витрати на виконання науково-дослідної роботи за кожною статтею витрат становлять 113 357,9 грн. Загальна сума витрат на розробку та впровадження результатів цієї НДР складе 161 939,9 грн.

Інвестиції, вкладені в цей проєкт, окупляться через 1,6 роки. Приведена вартість всіх чистих прибутків, які підприємство отримає від реалізації результатів наукової розробки, складе 1 035 403,8 грн.

ВИСНОВКИ

У рамках виконаної роботи було розроблено комплексну систему управління доступом для DRM-рішень, що базується на динамічно змінюваних USB-ключах безпеки, вдосконаленому алгоритмі HOTP (Hashed-Based One-Time Password) та адаптивній системі блокування доступу. Проведено повний цикл дослідження, починаючи з аналізу сучасних методів захисту інформаційних ресурсів і закінчуючи практичною реалізацією та тестуванням програмної частини системи.

На етапі теоретичного дослідження було проаналізовано ключові аспекти управління доступом і сучасні підходи до підвищення захищеності DRM-систем. Особливу увагу приділено методам генерації одноразових паролів, захисту секретних ключів через використання апаратних пристроїв (USB-ключів) і механізмів блокування доступу для запобігання несанкціонованому використанню облікових записів. Аналіз існуючих рішень дозволив виявити їхні обмеження та сформувати завдання для вдосконалення захисту.

На етапі реалізації розроблено програмні модулі, які забезпечують генерацію одноразових паролів, інтеграцію з USB-ключами, автоматичне блокування доступу при аномальній активності та багатофакторну перевірку для розблокування облікових записів. Використання Node.js як платформи розробки дозволило створити масштабовану, кросплатформенну і продуктивну систему.

Тестування реалізованої системи підтвердило її відповідність вимогам безпеки, коректність роботи окремих модулів і здатність витримувати високе навантаження. Усі компоненти системи показали надійну взаємодію, забезпечуючи високий рівень захисту цифрових прав і безпечний доступ до ресурсів.

Таким чином, виконана робота досягла своєї мети, запропонувавши ефективне рішення для підвищення захищеності DRM-системи управління доступом. Результати дослідження та реалізації можуть бути використані для вдосконалення існуючих DRM-рішень або створення нових систем із підвищеними вимогами до безпеки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Suzanne Humphries. What is a USB security key, and should you use one?. URL: https://www.reviewgeek.com/63448/what-is-a-usb-security-key-and-should-you-use-one/

2. Authentication - Django REST framework. *Home - Django REST framework*. URL: https://www.django-rest-framework.org/api-guide/authentication/.

3. Jena B. K. AES Encryption: Secure Data with Advanced Encryption Standard. *Simplilearn.com*. URL: https://www.simplilearn.com/tutorials/cryptography-tutorial/aes-encryption.

4. Magnusson A. Token-based Authentication: Everything You Need to Know. *StrongDM: Your Partner in Zero Trust Privileged Access*. URL: https://www.strongdm.com/blog/token-based-authentication.

5. What Is an Authentication Token? | Fortinet. *Fortinet*. URL: https://www.fortinet.com/resources/cyberglossary/authentication-token.

6. What Is Token-Based Authentication? | Okta. *Employee and Customer Identity Solutions | Okta*. URL: https://www.okta.com/identity-101/what-is-token-based-authentication/.

7. What is token-based authentication?. *Stack Overflow*. URL: https://stackoverflow.com/questions/1592534/what-is-token-based-authentication.

8. Cloudflare Application Services | Security and Performance. Cloudflare. URL: https://www.cloudflare.com/application-services/?utm_source=google&utm_medium=cpc&utm_campaign=DG_EMEig7BqZ7VtZrN6wUzk1Nhp6nkaAgwEEALw_wcB&gclsrc=aw.ds).

9. Кібербезпека. URL: https://www.span.eu/ua/рішення-та-послуги/сервіси-з-безпеки/кібербезпека/?gad_source=1&gclid=Cj0KCQiA67CrBhC1ARIsACKAa8REXQS8JOs2ZaaOW6g5OyaeVWM8ksz3M6viwjX_pclBefnGsQgibfMaAgYHEALw_wcB.

10. Contributors to Wikimedia projects. Information security - Wikipedia. Wikipedia, the free encyclopedia. URL: https://en.wikipedia.org/wiki/Information_security

11. АНАЛІЗ ІСНУЮЧИХ АЛГОРИТМІВ ГЕНЕРАЦІЇ ДИНАМІЧНО ЗМІНЮВАНИХ КЛЮЧІВ .URL: https://conferences.vntu.edu.ua/index.php/mn/mn2025/author/submission/22815

12. Imperva. URL: https://www.imperva.com/learn/data-security/information-security-infosec/.

13. What is information security (infosec)?. Cisco. URL: https://www.cisco.com/c/en/us/products/security/what-is-information-security-infosec.html.

14. What is information security (infosec)? Goals, types and applications. Exabeam. URL: https://www.exabeam.com/explainers/information-security/information-security-goals-types-and-applications/.

15. Yasar K., Wright G., Teravainen T. What is information security (infosec)? – techtarget definition. Security. URL: https://www.techtarget.com/searchsecurity/definition/information-security-infosec.

16. Cloudflare Application Services | Security and Performance. Cloudflare. URL: https://www.cloudflare.com/application-services/?utm_source=google&utm_medium=cpc&utm_campaign=DG_EMEA_ENG_G_Search_Generic_Beta_Applications-Security&utm_content=Beta_Generic_Applications-Security_Core&utm_term=cyber+security&campaignid=7170000011271632 2&adgroupid=587000084860010 12&creativeid=662071359069&gclid=Cj0KCQiA67CrBhC1ARIsACKAa8Q17aOUR4pjkbG6p-6JfA6-aLhFR_-ig7BqZ7VtZrN6wUzk1Nhp6nkaAgwEEALw_wcB&gclsrc=aw.ds).

17. Кібербезпека. URL: https://www.span.eu/ua/рішення-та-послуги/сервіси-з-безпеки/кібербезпека/?gad_source=1&gclid=Cj0KCQiA67CrBhC1ARIsACKAa

8REXQS8JOs2ZaaOW6g5OyaeVWM8ksz3M6viwjX_pclBefnGsQgibfMaAgYHEALw_wcB.

18. Aholi ICT limited - cyber security services company in bangladesh. Aholi ICT Limited - Cyber Security Services Company in Bangladesh. URL: https://www.aholiict.com/usbsecurity.php

19. Contributors to Wikimedia projects. Information security - Wikipedia. Wikipedia, the free encyclopedia. URL: https://en.wikipedia.org/wiki/Information_security

20. Imperva. URL: https://www.imperva.com/learn/data-security/information-security-infosec/.

21. What is information security (infosec)?. Cisco. URL: https://www.cisco.com/c/en/us/products/security/what-is-information-security-infosec.html.

22. What is information security (infosec)? Goals, types and applications. Exabeam. URL: https://www.exabeam.com/explainers/information-security/information-security-goals-types-and-applications/.

23. Yasar K., Wright G., Teravainen T. What is information security (infosec)? – techtarget definition. Security. URL: https://www.techtarget.com/searchsecurity/definition/information-security-infosec.

24. What is information security? | IBM. IBM in Deutschland, Österreich und der Schweiz | IBM. URL: https://www.ibm.com/topics/information-security.

25. What is information security? - geeksforgeeks. GeeksforGeeks. URL: https://www.geeksforgeeks.org/what-is-information-security/.

26. INFOSEC - Glossary | CSRC. NIST Computer Security Resource Center | CSRC. URL: https://csrc.nist.gov/glossary/term/INFOSEC.

27. What is authorization? - examples and definition - auth0. *Auth0*. URL: https://auth0.com/intro-to-iam/what-is-authorization .

28. Academy B. Seed Phrase | Binance Academy. Binance Academy. URL: https://academy.binance.com/en/glossary/seed-phrase.

29. What is authorization? - examples and definition - auth0. *Auth0*. URL: https://auth0.com/intro-to-iam/what-is-authorization .

30. What is javascript? A basic introduction to JS for beginners. Hostinger Tutorials. URL: https://www.hostinger.com/tutorials/what-is-javascript).

31. Visual studio code. Visual Studio Code. URL: https://code.visualstudio.com/).

32. Редактор коду visual studio code. Habr. URL: https://habr.com/ua/articles/490754

33. JavaScript. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript.

34. Electron. Electron. URL: https://www.electronjs.org/ .

35. Node.js. Node.js. URL: https://nodejs.org/uk .

36. The Modern JavaScript. URL: https://javascript.info/ .

37. Sufiyan T. What is node.js: a comprehensive guide. *Simplilearn.com*. URL: https://www.simplilearn.com/tutorials/nodejs-tutorial/what-is-nodejs).

38. Megida D. What is javascript? A definition of the JS programming language. freeCodeCamp.org. URL: https://www.freecodecamp.org/news/what-is-javascript-definition-of-js/

39. What is javascript? A basic introduction to JS for beginners. Hostinger Tutorials. URL: https://www.hostinger.com/tutorials/what-is-javascript).

40. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. Вінниця : ВНТУ, 2021. 42 с.

41. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа. Вінниця : ВНТУ, 2016. 113 с

ДОДАТКИ

97

Додаток А. Технічне завдання

Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

ЗАТВЕРДЖУЮ

Голова секції "Управління інформаційною безпекою" кафедри МБІС
д.т.н., професор

Юрій ЯРЕМЧУК
"20" вересня 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

до магістерської кваліфікаційної роботи на тему:
Підвищення захищеності DRM-системи управління доступом динамічно змінюваними USB-ключами безпеки на основі вдосконаленого алгоритму HOTP (Hashed-Based One-Time Password) та системи блокування доступу

08-72.МКР.011.00.117.ТЗ

Керівник магістерської кваліфікаційної роботи
д. ф., доцент Салієва О.В.

Вінниця – 2024 р.

1. Найменування та область застосування

Програмний засіб підвищення захищеності DRM-системи управління доступом динамічно змінюваними USB-ключами безпеки на основі вдосконаленого алгоритму HOTP (Hashed-Based One-Time Password) та системи блокування доступу

2.Підстава для розробки

Розробка виконується на основі наказу ректора ВНТУ №310 від 17. 09. 2024 р.

3. Мета та призначення розробки

3.1 Мета розробки: розробка ефективного методу захисту від атак

3.2 Призначення: розроблений програмний засіб виконує захист від атак

4. Джерела розробки

4.1**Стрельбицький М. П., Іванова Н. Г.** Інформаційна безпека. Підручник. – Київ: Видавництво Ліра-К, 2021. – 412 с.

4.2**Мужанова О. В.** Інформаційна безпека держави: навчальний посібник. – Чернігів: ЧНТУ, 2019. – 200 с.

4.3**Дудікевич В. Б.** Основи інформаційної безпеки: навчальний посібник. – Вінниця: ВНТУ, 2018. – 316 с.

4.4**Карпенко В.** Інформаційна політика та безпека. – Київ: Видавництво Ліра-К, 2020. – 320 с.

4.5**Стрельбицький М. П., Іванова Н. Г.** Інформаційна безпека. Підручник. – Київ: Видавництво Ліра-К, 2021. – 412 с.

5. Вимоги до програми

5.1 Вимоги до функціональних характеристик:

5.1.1 Програмний засіб повинен мати зручний, легкий у використанні інтерфейс користувача;

5.1.2 Реалізація методу не повинна вимагати спеціальних ліцензійних програмних додатків;

5.1.3 Програмний засіб повинен виконувати процес автентифікації користувачів у системі.

5.2 Вимоги до надійності:

5.2.1 Програмний засіб повинен працювати без помилок, у випадку виникнення критичних ситуацій необхідно передбачити виведення відповідних повідомлень;

5.2.2 Бази даних повинні бути налаштовані на автоматичне створення резервних копій;

5.2.3 Програмний засіб повинен виконувати свої функції.

5.3 Вимоги до складу і параметрів технічних засобів:

– процесор – Pentium 1500 МГц і подібні до них;

– оперативна пам’ять – не менше 512 Mb;

– середовище функціонування – операційна система сімейство Windows;

– вимоги до техніки безпеки при роботі з програмою повинні відповідати існуючим вимогам та стандартам з техніки безпеки при користуванні комп’ютерною технікою.

6. Вимоги до програмної документації

6.1 Обов’язкова поетапна інструкція для майбутніх користувачів, наведена у пункті 3.4

7. Вимоги до технічного захисту інформації

7.1 Необхідно забезпечити захист розроблюваного програмного засобу від несанкціонованого використання.

7.2 Неможливість отримання доступу незареєстрованих користувачів до інформаційних ресурсів.

8. Техніко-економічні показники

8.1 Цінність результатів використання даного проекту повинна перевищувати витрати на його реалізацію.

8.2 Має бути реалізований таким чином, щоб підходити для використання широкого загалу.

9. Стадії та етапи розробки

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Початок	Закінчення
1.	Визначення напрямку магістерської роботи, формулювання теми	20.09.2024	31.09.2024
2.	Аналіз предметної області обраної теми	01.10.2024	15.10.2024
3.	Розробка роботи	16.10.2024	26.10.2024
4.	Написання магістерської роботи на основі розробленої теми	27.10.2024	15.11.2024
5.	Передзахист магістерської кваліфікаційної роботи	16.11.2024	24.11.2024
6.	Виправлення, уточнення, корегування магістерської кваліфікаційної роботи	27.11.2024	04.12.2024
7.	Захист магістерської кваліфікаційної роботи	10.12.2024	10.12.2024

10. Порядок контролю та прийому

10.1 До приймання магістерської кваліфікаційної роботи надається:

– ПЗ до магістерської кваліфікаційної роботи;

– програмний додаток;

– презентація;

– відзив керівника роботи;

– відзив опонента

Технічне завдання до виконання прийняв _____Пуздрановський І.В

Додаток Б. Лістинг програми

```
const crypto = require('crypto');
const otplib = require('otplib');

class HOTPModule {
 constructor(secret, counter) {
  this.secret = secret; // Секретний ключ, зчитаний з USB
  this.counter = counter || 0; // Початковий лічильник
 }

 // Генерація OTP
 generateHOTP() {
  return otplib.hotp.generate(this.secret, this.counter);
 }

 // Перевірка OTP
 verifyHOTP(token) {
  return otplib.hotp.check(token, this.secret, this.counter);
 }

 // Оновлення лічильника
 incrementCounter() {
  this.counter++;
 }
}

module.exports = HOTPModule;
```

бібліотеку usb для роботи з апаратними пристроями.

```
const usb = require('usb'); // Підключення бібліотеки для роботи з USB

class USBManager {
 constructor() {
  this.device = null; // Об'єкт пристрою
 }

 connectDevice() {
  // Пошук і підключення до USB-пристрою
  this.device = usb.findByIds(1234, 5678); // Приклад VID і PID
  if (!this.device) throw new Error('USB device not found');
  this.device.open();
 }

 readSecret() {
  // Зчитування секретного ключа з USB-пристрою
  const data = this.device.controlTransferSync(0xC0, 0x01, 0, 0, 32);
  return data.toString('utf8');
 }

 writeCounter(counter) {
  // Запис оновленого лічильника на USB-пристрій
  const buffer = Buffer.alloc(4);
  buffer.writeUInt32BE(counter);
  this.device.controlTransferSync(0x40, 0x02, 0, 0, buffer);
 }

 closeDevice() {
```

```
    // Закриття підключення до USB
    this.device.close();
  }
}

module.exports = USBManager;
const HOTPModule = require('./HOTPModule');
const USBManager = require('./USBManager');

class DRMSystem {
  constructor() {
    this.usbManager = new USBManager();
    this.hotpModule = null;
  }

  initialize() {
    try {
      this.usbManager.connectDevice();
      const secret = this.usbManager.readSecret();
      const counter = 0; // Початковий лічильник або зчитаний із сервера
      this.hotpModule = new HOTPModule(secret, counter);
    } catch (error) {
      console.error('Initialization error:', error.message);
    }
  }

  authenticateUser(inputToken) {
    if (!this.hotpModule) throw new Error('System not initialized');
    const isValid = this.hotpModule.verifyHOTP(inputToken);
    if (isValid) {
      console.log('Authentication successful');
      this.hotpModule.incrementCounter();
      this.usbManager.writeCounter(this.hotpModule.counter);
    } else {
      console.log('Authentication failed');
    }
  }

  shutdown() {
    this.usbManager.closeDevice();
  }
}

module.exports = DRMSystem;

class ActivityMonitor {
  constructor(logFilePath) {
    this.logFilePath = logFilePath || './access-log.json';
    this.logs = this.loadLogs();
  }

  loadLogs() {
    if (fs.existsSync(this.logFilePath)) {
      return JSON.parse(fs.readFileSync(this.logFilePath, 'utf8'));
    }
    return [];
  }
```

```
  saveLogs() {
    fs.writeFileSync(this.logFilePath, JSON.stringify(this.logs, null, 2));
  }

  logAttempt(userId, success, ipAddress) {
    const attempt = {
      userId,
      success,
      ipAddress,
      timestamp: new Date().toISOString(),
    };
    this.logs.push(attempt);
    this.saveLogs();
  }

  getRecentAttempts(userId, limit = 5) {
    return this.logs.filter(log => log.userId === userId).slice(-limit);
  }
}

module.exports = ActivityMonitor;
class AccessBlocker {
  constructor(activityMonitor, maxFailedAttempts = 3) {
    this.activityMonitor = activityMonitor;
    this.maxFailedAttempts = maxFailedAttempts;
    this.blockedUsers = new Set();
  }

  isBlocked(userId) {
    return this.blockedUsers.has(userId);
  }

  handleLoginAttempt(userId, success, ipAddress) {
    if (this.isBlocked(userId)) {
      console.log(`Access denied for user ${userId} - account is blocked.`);
      return false;
    }

    this.activityMonitor.logAttempt(userId, success, ipAddress);

    if (!success) {
      const recentAttempts = this.activityMonitor.getRecentAttempts(userId);
      const failedAttempts = recentAttempts.filter(attempt => !attempt.success);

      if (failedAttempts.length >= this.maxFailedAttempts) {
        this.blockedUsers.add(userId);
        console.log(`User ${userId} has been blocked due to multiple failed attempts.`);
        return false;
      }
    }
    return true;
  }

  unblockUser(userId) {
    this.blockedUsers.delete(userId);
    console.log(`User ${userId} has been unblocked.`);
  }
}
```

```
module.exports = AccessBlocker;
class UnlockModule {
 constructor(accessBlocker) {
  this.accessBlocker = accessBlocker;
 }

 requestUnlock(userId, otpGenerator, userInputOtp) {
  const generatedOtp = otpGenerator.generateHOTP();
  console.log(`Generated OTP for user ${userId}: ${generatedOtp}`);

  if (userInputOtp === generatedOtp) {
   this.accessBlocker.unblockUser(userId);
   console.log(`User ${userId} has successfully completed OTP verification.`);
   return true;
  } else {
   console.log(`Invalid OTP entered by user ${userId}.`);
   return false;
  }
 }
}

module.exports = UnlockModule;
const ActivityMonitor = require('./ActivityMonitor');
const AccessBlocker = require('./AccessBlocker');
const UnlockModule = require('./UnlockModule');
const HOTPModule = require('./HOTPModule');

const activityMonitor = new ActivityMonitor();
const accessBlocker = new AccessBlocker(activityMonitor);
const hotpModule = new HOTPModule('mySecretKey', 0);
const unlockModule = new UnlockModule(accessBlocker);

// Приклад використання системи
const userId = 'user123';
const ipAddress = '192.168.1.1';

if (!accessBlocker.handleLoginAttempt(userId, false, ipAddress)) {
 const userInputOtp = '123456'; // Введений користувачем OTP
 unlockModule.requestUnlock(userId, hotpModule, userInputOtp);
const USBManager = require('./USBManager');
const HOTPModule = require('./HOTPModule');
const ActivityMonitor = require('./ActivityMonitor');
const AccessBlocker = require('./AccessBlocker');
const UnlockModule = require('./UnlockModule');

class DRMSystem {
 constructor() {
  this.usbManager = new USBManager();
  this.activityMonitor = new ActivityMonitor();
  this.accessBlocker = new AccessBlocker(this.activityMonitor);
  this.hotpModule = null;
  this.unlockModule = null;
 }

 initialize() {
  try {
   // Підключення до USB-ключа та зчитування секретного ключа
```

```
      this.usbManager.connectDevice();
      const secret = this.usbManager.readSecret();
      const counter = 0; // Початковий лічильник
      this.hotpModule = new HOTPModule(secret, counter);

      // Ініціалізація модуля розблокування
      this.unlockModule = new UnlockModule(this.accessBlocker);

      console.log('DRM System successfully initialized.');
    } catch (error) {
      console.error('Initialization failed:', error.message);
    }
  }
}
authenticateUser(userId, inputToken, ipAddress) {
  if (this.accessBlocker.isBlocked(userId)) {
    console.log(`Access denied for user ${userId} - account is blocked.`);
    return false;
  }

  const isValid = this.hotpModule.verifyHOTP(inputToken);

  if (isValid) {
    console.log(`Authentication successful for user ${userId}.`);
    this.hotpModule.incrementCounter();
    this.usbManager.writeCounter(this.hotpModule.counter);
    this.activityMonitor.logAttempt(userId, true, ipAddress);
    return true;
  } else {
    console.log(`Authentication failed for user ${userId}.`);
    this.activityMonitor.logAttempt(userId, false, ipAddress);
    this.accessBlocker.handleLoginAttempt(userId, false, ipAddress);
    return false;
  }
}
requestUnlock(userId, userInputOtp) {
  if (!this.accessBlocker.isBlocked(userId)) {
    console.log(`User ${userId} is not blocked.`);
    return false;
  }

  const isUnlocked = this.unlockModule.requestUnlock(userId, this.hotpModule, userInputOtp);

  if (isUnlocked) {
    console.log(`User ${userId} successfully unlocked.`);
    return true;
  } else {
    console.log(`Failed unlock attempt for user ${userId}.`);
    return false;
  }
}


const drmSystem = new DRMSystem();
drmSystem.initialize();
```

```
const userId = 'user123';
const ipAddress = '192.168.1.1';
const inputToken = '123456'; // OTP, який ввів користувач

if (!drmSystem.authenticateUser(userId, inputToken, ipAddress)) {
 const userInputOtp = '654321'; // OTP для розблокування
 drmSystem.requestUnlock(userId, userInputOtp);
}
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>DRM Authentication</title>
 <link rel="stylesheet" href="styles.css">
</head>
<body>
 <div class="container">
  <h1>Welcome to DRM System</h1>
  <form id="authForm">
   <label for="userId">User ID:</label>
   <input type="text" id="userId" name="userId" required>
   <label for="otp">One-Time Password (OTP):</label>
   <input type="text" id="otp" name="otp" required>
   <button type="submit">Authenticate</button>
  </form>
  <div id="status"></div>
 </div>
 <script src="script.js"></script>
</body>
</html>
document.getElementById('authForm').addEventListener('submit', async (e) => {
 e.preventDefault();

 const userId = document.getElementById('userId').value;
 const otp = document.getElementById('otp').value;
 const statusDiv = document.getElementById('status');

 try {
  const response = await fetch('/api/authenticate', {
   method: 'POST',
   headers: { 'Content-Type': 'application/json' },
   body: JSON.stringify({ userId, otp }),
  });

  const result = await response.json();

  if (result.success) {
   statusDiv.textContent = 'Authentication successful!';
   statusDiv.style.color = 'green';
  } else {
   statusDiv.textContent = result.message || 'Authentication failed!';
  }
 } catch (error) {
  statusDiv.textContent = 'Error connecting to server.';
 }
});
```

Цей код відправляє дані форми до API /api/authenticate та відображає результат автентифікації у вигляді повідомлення для користувача.

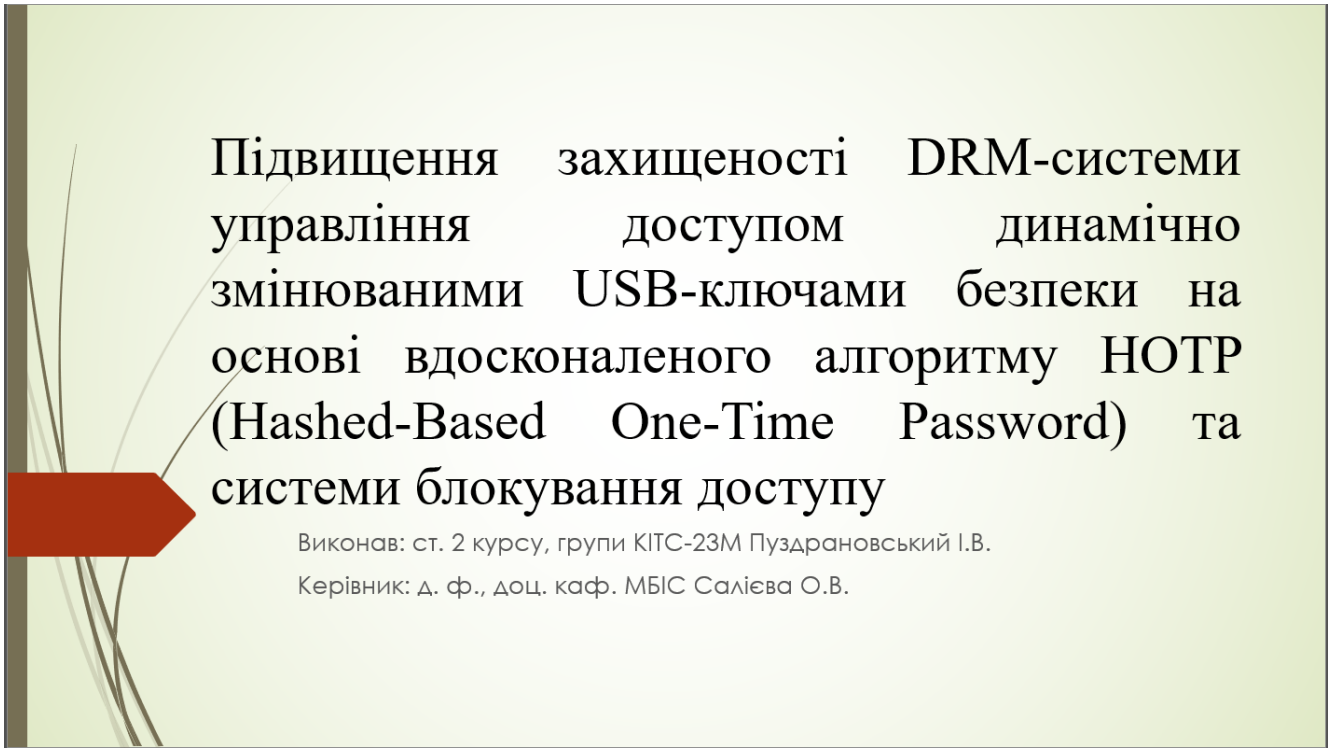
На стороні сервера використовується Express.js для обробки запитів.

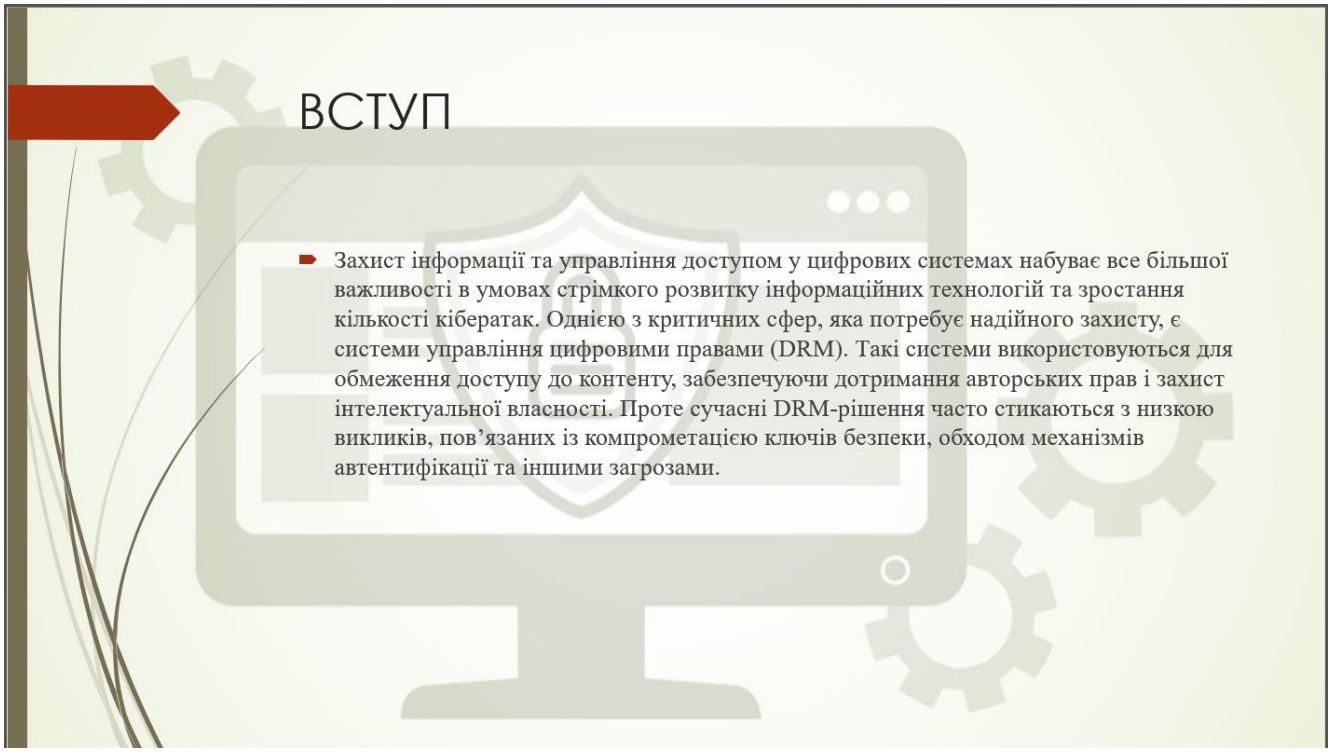
```
const express = require('express');
const DRMSystem = require('./DRMSystem');

const app = express();
app.use(express.json());

const drmSystem = new DRMSystem();
drmSystem.initialize();

app.post('/api/authenticate', (req, res) => {
  const { userId, otp } = req.body;

  const ipAddress = req.ip || 'unknown';
  const isAuthenticated = drmSystem.authenticateUser(userId, otp, ipAddress);

  if (isAuthenticated) {
    res.json({ success: true });
  } else {
    res.json({ success: false, message: 'Invalid OTP or account blocked.' });
  }
});

app.listen(3000, () => console.log('Server running on http://localhost:3000'));
body {
  font-family: Arial, sans-serif;
  background-color: #f4f4f9;
  margin: 0;
  padding: 0;
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
}

.container {
  background: #ffffff;
  padding: 20px;
  border-radius: 8px;
  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
  text-align: center;
}

h1 {
  color: #333;
}

form {
  margin-top: 20px;
}

label {
  display: block;
```

```
  margin: 10px 0 5px;
}

input {
  padding: 10px;
  width: 100%;
  border: 1px solid #ccc;
  border-radius: 5px;
  margin-bottom: 20px;
}

button {
  padding: 10px 15px;
  border: none;
  border-radius: 5px;
  background-color: #007BFF;
  color: white;
  cursor: pointer;
}

button:hover {
  background-color: #0056b3;
}

#status {
  margin-top: 20px;
  color: red;
}
const { expect } = require('chai');
const HOTPModule = require('./HOTPModule');

describe('HOTP Module', () => {
  it('should generate a valid OTP', () => {
    const hotp = new HOTPModule('testSecret', 0);
    const otp = hotp.generateHOTP();
    expect(otp).to.be.a('string');
    expect(otp.length).to.equal(6);
  });

  it('should verify a valid OTP', () => {
    const hotp = new HOTPModule('testSecret', 0);
    const otp = hotp.generateHOTP();
    const isValid = hotp.verifyHOTP(otp);
    expect(isValid).to.be.true;
  });

  it('should reject an invalid OTP', () => {
    const hotp = new HOTPModule('testSecret', 0);
    const isValid = hotp.verifyHOTP('123456');
    expect(isValid).to.be.false;
  });
});
const USBManager = require('./USBManager');

describe('USB Manager', () => {
  it('should connect to the USB device', () => {
    const usbManager = new USBManager();
    expect(() => usbManager.connectDevice()).to.not.throw();
```

```
  });

  it('should read the secret from the USB device', () => {
    const usbManager = new USBManager();
    usbManager.connectDevice();
    const secret = usbManager.readSecret();
    expect(secret).to.be.a('string');
    usbManager.closeDevice();
  });
});
const ActivityMonitor = require('./ActivityMonitor');
const AccessBlocker = require('./AccessBlocker');

describe('Access Blocker', () => {
  it('should block a user after multiple failed attempts', () => {
    const monitor = new ActivityMonitor();
    const blocker = new AccessBlocker(monitor, 3);
    blocker.handleLoginAttempt('user123', false, '192.168.0.1');
    blocker.handleLoginAttempt('user123', false, '192.168.0.1');
    blocker.handleLoginAttempt('user123', false, '192.168.0.1');
    expect(blocker.isBlocked('user123')).to.be.true;
  });

  it('should allow unblocking a user', () => {
    const monitor = new ActivityMonitor();
    const blocker = new AccessBlocker(monitor, 3);
    blocker.blockedUsers.add('user123');
    blocker.unblockUser('user123');
    expect(blocker.isBlocked('user123')).to.be.false;
  });
});
```

Додаток В. Ілюстративний матеріал





Актуальність та мета

- **Актуальність.** Розробка вдосконаленої DRM-системи, яка інтегрує динамічно змінювані USB-ключі безпеки, алгоритми генерації одноразових паролів (HOTP) та систему блокування доступу, є актуальним завданням, що відповідає сучасним вимогам до інформаційної безпеки. Впровадження таких технологій дозволяє мінімізувати ризики несанкціонованого доступу, підвищити рівень захищеності ключів безпеки та забезпечити ефективне управління цифровими правами.
- **Мета і задачі дослідження.** Метою роботи є розробка DRM-системи управління доступом із підвищеним рівнем захисту, яка використовує динамічно змінювані USB-ключі безпеки, вдосконалений алгоритм HOTP та систему блокування доступу для запобігання несанкціонованому використанню.

Важливість, особливості та основні принципи управління доступом

- **Управління доступом** є одним із ключових компонентів забезпечення інформаційної безпеки в сучасних системах. Його важливість обумовлена необхідністю захисту конфіденційних даних від несанкціонованого доступу та забезпечення контролю над тим, хто, коли і до яких ресурсів має доступ. Основними принципами управління доступом є автентифікація, авторизація, контроль привілеїв та реєстрація дій користувачів.
- Серед особливостей управління доступом варто відзначити необхідність адаптивності до змін у середовищі, використання багатофакторної автентифікації для підвищення безпеки та підтримку сучасних криптографічних методів захисту. Ефективне управління доступом дозволяє не лише захищати ресурси, але й підвищувати рівень довіри до системи, створюючи баланс між зручністю використання і надійністю захисту.

DRM-системи, як системи управління доступом

- **DRM-системи як системи управління доступом** відіграють ключову роль у захисті цифрових прав та інтелектуальної власності. Їх основна мета полягає в обмеженні доступу до контенту лише авторизованим користувачам, забезпечуючи дотримання ліцензійних умов і захист від незаконного копіювання. DRM-системи інтегрують механізми управління доступом, включаючи автентифікацію користувачів, контроль прав на використання контенту та моніторинг дій.
- Особливістю DRM-систем є їх здатність не лише перевіряти права доступу, але й динамічно змінювати ці права залежно від умов, наприклад, часу або типу пристрою. Це забезпечується за рахунок використання шифрування, токенів доступу, цифрових підписів та інших криптографічних інструментів.
- DRM-системи дозволяють власникам контенту зберігати контроль над своїми ресурсами, водночас забезпечуючи користувачам доступ до цифрового продукту у визначених межах. Їх застосування особливо актуальне в сферах медіа, програмного забезпечення, електронних книг і наукових публікацій

Динамічно змінювальних ключі

- **Динамічно змінювані ключі** є сучасним підходом до забезпечення високого рівня безпеки в системах управління доступом. Цей метод базується на принципі регулярного оновлення криптографічних ключів після кожної сесії або операції, що значно ускладнює можливість їх компрометації зловмисниками. У порівнянні зі статичними ключами, динамічні забезпечують вищу стійкість до атак типу "відтворення" та зниження ризику несанкціонованого доступу навіть у разі перехоплення одного з ключів.
- Основна ідея роботи з динамічно змінюваними ключами полягає у використанні початкового секретного ключа, який змінюється за певним алгоритмом, наприклад, із застосуванням криптографічних хеш-функцій або псевдовипадкових генераторів. Новий ключ синхронізується між клієнтом і сервером для забезпечення коректної роботи. Такий підхід забезпечує унікальність кожної операції та підвищує загальну захищеність системи.
- Динамічно змінювані ключі широко використовуються у високозахищених системах, включаючи фінансові транзакції, управління доступом до конфіденційної інформації та DRM-рішення. Вони також є важливим компонентом багатофакторної автентифікації, де підвищують безпеку за рахунок постійного оновлення автентифікаційних даних. Однак реалізація такого підходу вимагає ретельного планування алгоритмів синхронізації ключів та захисту механізмів оновлення.

Порвіняльний аналіз алгоритмів генерації динамічно змінюваних ключів

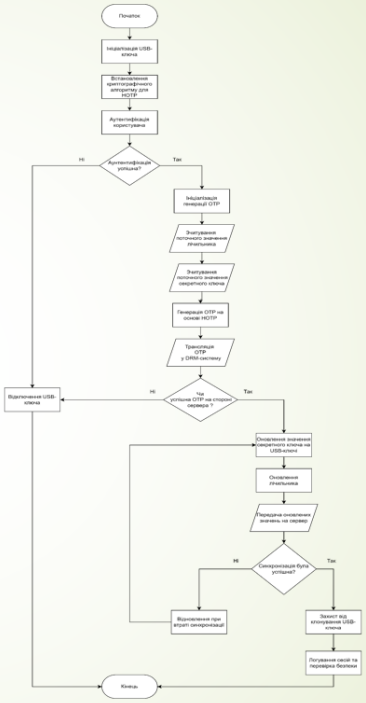
Алгоритм	Основний параметр	Стійкість до атак	Складність синхронізації	Використання в DRM-системах
HOTP	Лічильник	Висока, але залежить від синхронізації лічильника	Вимагає синхронізації лічильника між сервером і пристроєм	Підходить для одноразових сесій
TOTP	Час	Висока, але залежить від точності синхронізації часу	Висока вимога до точності синхронізації часу	Підходить для коротких сесій із обмеженим доступом
Nonce (випадкові значення)	Випадкове значення (nonce)	Висока, за умови унікальності значення	Вимагає синхронізації nonce на сервері та пристрої	Ефективний для короткочасних сесій
Удосконалений HOTP з динамічним ключем	Динамічно змінюваний секретний ключ	Дуже висока, завдяки зміні ключа після кожної сесії	Висока складність синхронізації нового ключа між сервером і пристроєм	Ідеально підходить для захисту з високими вимогами до безпеки в DRM-системах

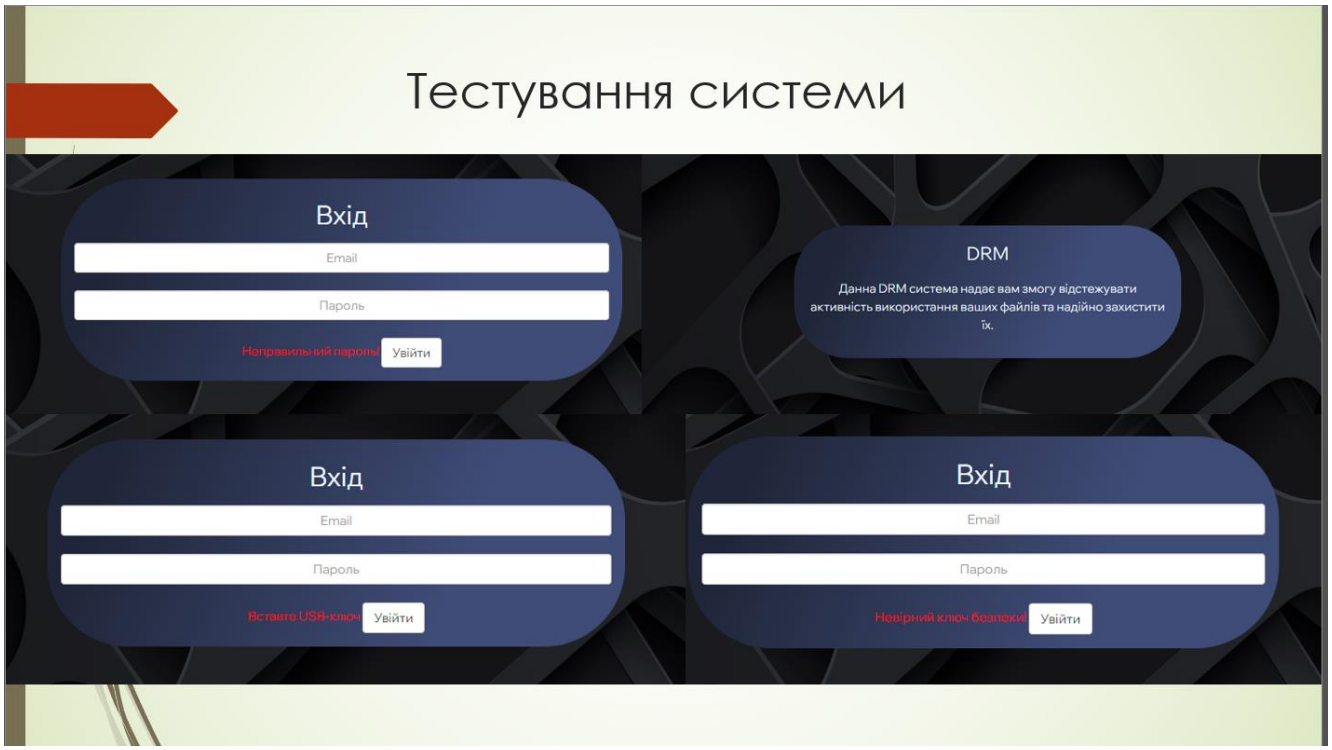
Порівняльний аналіз сучасних методів захисту інформаційних ресурсів і перспективної розробки для DRM-систем

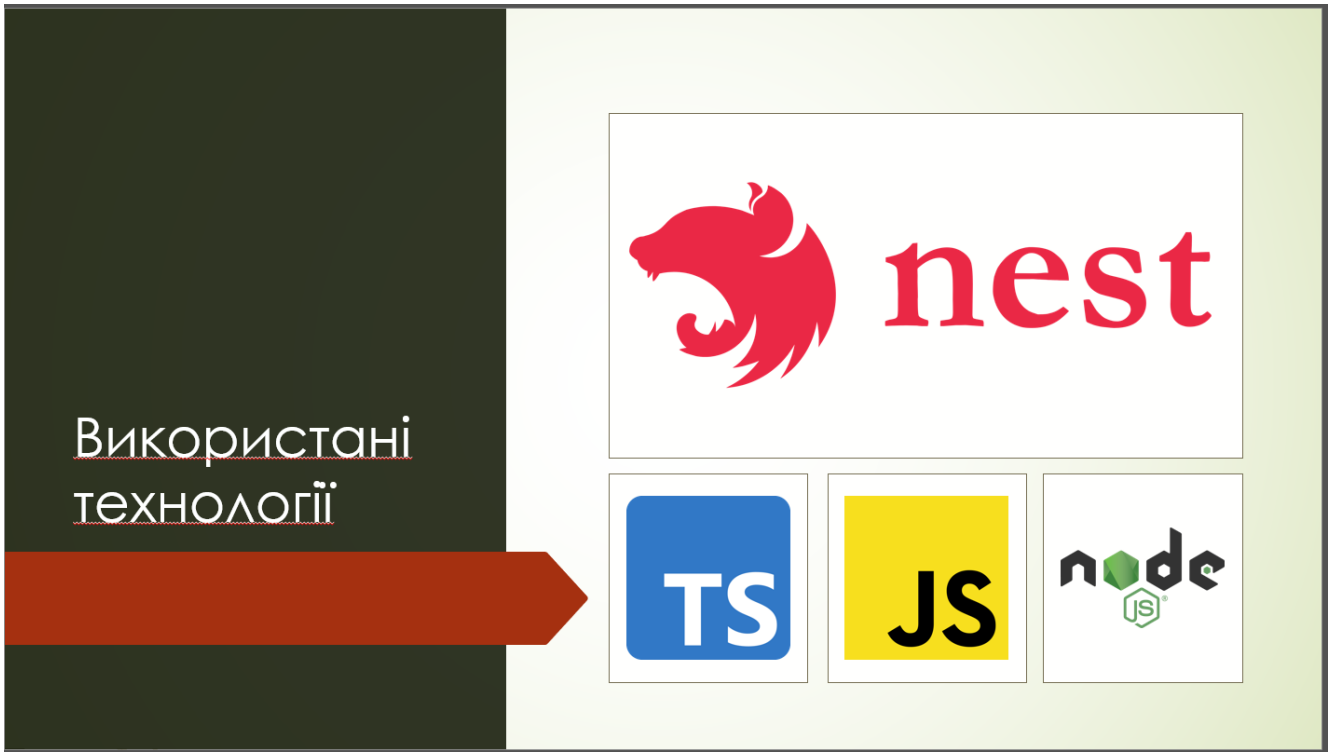
Критерій	Сучасні методи захисту (USB-ключі та блокування доступу)	Перспективна розробка (динамічні USB-ключі на основі вдосконаленого HOTP та адаптивне блокування)
Ефективність захисту	Високий рівень захисту за рахунок криптографії та фізичних USB-ключів, але ключі потребують додаткового захисту від крадіжки та фізичного доступу.	Підвищена ефективність завдяки динамічній генерації паролів HOTP та поведінковому блокуванню. Використання одноразових паролів знижує ризик повторного використання та атак типу «людина посередині».
Захист від компрометації	Захищають від компрометації паролів за рахунок фізичного USB-ключа та багатофакторної аутентифікації, але недостатньо ефективні для захисту від втрати або крадіжки ключа.	Динамічні USB-ключі та адаптивне блокування покращують захист від компрометації шляхом регулярного оновлення ключів. Це ускладнює несанкціонований доступ навіть при втраті пристрою, оскільки ключ оновлюється після кожної сесії.
Зручність для користувачів	Вимагають фізичного носія (USB-ключа), що може бути незручним для користувачів, а також потребують багаторазового підтвердження доступу (MFA).	Поліпшена зручність завдяки автоматичному оновленню ключів та поведінковій перевірці, що зменшує необхідність багатофакторної аутентифікації при звичній активності.
Ресурсоємність та витрати	Висока вартість на придбання фізичних ключів та їхній регулярний моніторинг і заміна у разі втрат чи пошкоджень.	Зниження витрат на повторну автентифікацію за рахунок автоматичного оновлення ключів і зменшення фізичної залежності від кожного ключа. Однак система потребує ресурсів для обробки поведінкових даних.
Захист від внутрішніх загроз	Система блокування ефективно запобігає внутрішнім загрозам шляхом обмеження спроб доступу після певної кількості невдалих входів.	Поведінковий аналіз і адаптивне блокування дозволяють виявляти підозрілу активність з боку внутрішніх користувачів та знижують ризик зловживання правами доступу.
Точність і надійність	Висока надійність фізичної автентифікації, проте є можливість фальшивих блокувань через неправильні налаштування порогів.	Покращена точність і менша кількість фальшивих спрацювань завдяки поведінковому та контекстуальному аналізу, що дозволяє уникати помилок при перевірці доступу.

Алгоритм блокування доступу

Алгоритм вдосконалених динамічно змінювальних USB-ключів




Тестування системи
Вхід
Email
Пароль
Неправильний пароль
Увійти
DRM
Данна DRM система надає вам змогу відстежувати активність використання ваших файлів та надійно захистити їх.
Вхід
Email
Пароль
Вставте USB-ключ
Увійти
Вхід
Email
Пароль
Невірний ключ безпеки
Увійти


Використані технології
nest
TS
JS
node
JS

Висновок

- У рамках виконаної роботи було розроблено комплексну систему управління доступом для DRM-рішень, що базується на динамічно змінюваних USB-ключах безпеки, вдосконаленому алгоритмі HOTP (Hashed-Based One-Time Password) та адаптивній системі блокування доступу. Проведено повний цикл дослідження, починаючи з аналізу сучасних методів захисту інформаційних ресурсів і закінчуючи практичною реалізацією та тестуванням програмної частини системи.
- Тестування реалізованої системи підтвердило її відповідність вимогам безпеки, коректність роботи окремих модулів і здатність витримувати високе навантаження. Усі компоненти системи показали надійну взаємодію, забезпечуючи високий рівень захисту цифрових прав і безпечний доступ до ресурсів.

ДЯКУЮ ЗА УВАГУ!

117

Додаток Г. Протокол перевірки на антиплагіат

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи:

Підвищення захищеності DRM-системи управління доступом динамічно змінюваними USB-ключами безпеки на основі вдосконаленого алгоритму HOTP (Hashed-Based One-Time Password) та системи блокування доступу

Тип роботи: магістерська кваліфікаційна робота

Підрозділ Кафедра менеджменту на безпеки інформаційних систем
Факультет менеджменту та інформаційної безпеки
Гр. КІТС-23м

Керівник доцент Салієва О.В.

Показники звіту подібності

Turnitin	
Оригінальність	98%
Загальна схожість	2%

Аналіз звіту подібності (відмітити потрібне)

- ☐ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
- ☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.
- ☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлений (-на) з повним звітом подібності, який був згенерований Системою щодо роботи (додається)

Автор _____ Пуздрановський І.В.
(підпис) (прізвище, ініціали)

Опис прийнятого рішення

- ☐ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

Особа, відповідальна за перевірку _____ Коваль Н.П.
(підпис) (прізвище, ініціали)

Експерт _____ _____
(за потреби) (підпис) (прізвище, ініціали, посада)