НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет прикладної математики
Кафедра системного програмування
і спеціалізованих комп'ютерних систем

«На правах рукопису»
УДК 004.056.52

До захисту допущено:
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
"___"_____2024 р.

Магістерська дисертація

на здобуття ступеня магістра
за освітньо-професійною програмою
«Системне програмування і спеціалізовані комп'ютерні системи»
зі спеціальності 123 «Комп'ютерна інженерія»
на тему: «Способи створення та розповсюдження одноразових паролів у комп'ютерних системах»

Виконав:
студент ІІ курсу, групи КВ-21мп
Бікерей Олексій Ігорович _____

Науковий керівник:
Доцент кафедри СПСКС, к.т.н., доцент,
Клятченко Ярослав Михайлович

_____

Рецензент:
_____
_____ _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент ___

Київ 2024 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра системного програмування і спеціалізованих комп‘ютерних систем

Рівень вищої освіти – другий (магістерський)
Спеціальність – 123 «Комп‘ютерна інженерія»
Освітньо-професійна програма «Системне програмування і спеціалізовані комп‘ютерні системи»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
«___»_____2024 р.

ЗАВДАННЯ
на магістерську дисертацію студента
Бікерея Олексія Ігоровича

1. Тема дисертації «Способи створення та розповсюдження одноразових паролів у комп’ютерних системах», науковий керівник дисертації Клятченко Ярослав Михайлович, к.т.н., доцент, затверджено наказом № 5217-с по університету від «9» листопада 2023 року.

2. Термін подання студентом дисертації 10.01.2024

3. Об’єкт дослідження: використання алгоритму генерації одноразового пароля на основі часу для обмеження доступу до ресурсів у високонавантажених системах.

4. Вихідні дані: розроблений модифікований алгоритм генерації одноразового пароля на основі часу, що дозволяє контролювати доступ до певного ресурсу з плином часу.

5. Перелік завдань, які потрібно розробити:

1) Проаналізувати методи генерації одноразових паролів та методи контролю доступу до ресурсу з обмеженим часом.

2) Модифікувати ефективний алгоритм обмеження доступу до ресурсу на основі одноразових паролів.
3) Протестувати ефективність порівняно з стандартною реалізацією.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: презентація.

7. Перелік публікацій:
1) CXXXIV Міжнародній інтернет-конференції «Сучасні аспекти розвитку науки і техніки в умовах війни» (24 листопада 2023 р.).
2) LVVI Міжнародної інтернет-конференції «Наука 2023: Дослідження та інновації» (2 - 3 грудня 2023 р.).

8. Дата видачі завдання 03.09.2023

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертаці	Примітка
1.	Вивчення літератури за тематикою МД	12.05.2023	
2.	Розроблення та узгодження технічного завдання	07.09.2023	
3.	Аналіз існуючих рішень	11.09.2023	
4.	Підготовка матеріалів розділів МД	12.10.2023	
5.	Розроблення програмного забезпечення	10.11.2023	
6.	Тестування розробленого програмного забезпечення	01.12.2023	
7.	Оформлення документації дипломного МД	15.12.2023	
8.	Попередній розгляд магістерської дисертації на кафедрі	20.12.2023	

Студент Бікерей Олексій

Науковий керівник Ярослав КЛЯТЧЕНКО

РЕФЕРАТ

Актуальність теми. Магістерська дисертація присвячена розробці та аналізу модифікованого алгоритму Time-Based One-Time Password для обмеження доступу до ресурсів у високонавантажених системах, важлива у сфері кібербезпеки та технологій. У сучасному цифровому середовищі, захист інформації від кіберзагроз є надзвичайно актуальним. Високонавантажені системи потребують ефективного та швидкого методу автентифікації, щоб забезпечити безпеку та уникнути перевантаження ресурсів. Модифікований алгоритм TOTP може стати ефективним інструментом для забезпечення безпеки та зниження навантаження на систему. Таке дослідження має велике значення для поліпшення методів захисту та забезпечення стійкості та швидкості доступу у високонавантажених системах

Об’єктом дослідження є використання алгоритму Time-Based One-Time Password (TOTP) для обмеження доступу до ресурсів у високонавантажених системах.

Предметом дослідження є розробка та аналіз модифікованого алгоритму Time-Based One-Time Password (TOTP) для оптимізації та підвищення безпеки обмеження доступу до ресурсів у високонавантажених системах.

Мета роботи: полягає у вивченні, розробці та оцінці модифікованого алгоритму Time-Based One-Time Password (TOTP) для підвищення ефективності та безпеки обмеження доступу до ресурсів у високонавантажених системах. Основними цілями є аналіз існуючих методів обмеження доступу, розробка модифікованого алгоритму TOTP, виявлення та оцінка його переваг у контексті ефективності, надійності та стійкості до потенційних кіберзагроз. Мета також включає в себе визначення оптимальних умов та сценаріїв застосування цього алгоритму в

високонавантажених системах, забезпечуючи високу захищеність та зниження негативного впливу на продуктивність системи.

Наукова новизна цієї роботи полягає у розробці та аналізі модифікованого алгоритму TOTP з метою покращення процесу обмеження доступу до ресурсів у високонавантажених системах. Цей алгоритм включає в себе нові методи генерації та верифікації одноразових паролів на основі часу, що сприяє підвищенню ефективності, стійкості та безпеки систем у порівнянні зі стандартними методами аутентифікації. Дослідження також враховує оптимізацію алгоритму для роботи у високонавантажених середовищах, що відрізняється від попередніх підходів та є актуальною новизною у сфері кібербезпеки та застосування алгоритмів аутентифікації в сучасних інформаційних системах.

Практична цінність отримані результати дослідження мають велику практичну цінність у сферах бізнесу, фінансів, електронної комерції та інших сферах. Вони можуть служити основою для покращення інформаційної безпеки через розробку та впровадження більш надійних систем захисту даних та методів автентифікації.

Апробація роботи. Основні положення були представлені та обговорювались на CXXXIV Міжнародній інтернет-конференції «Сучасні аспекти розвитку науки і техніки в умовах війни» та LVVI Міжнародної інтернет-конференції «Наука 2023: Дослідження та інновації»

Структура та обсяг роботи. Магістерська дисертація складається з вступу, чотирьох розділів та висновків.

У *вступі* проведено огляд загальної характеристики роботи, яка зорієнтована на важливі аспекти інформаційної безпеки та наявні методи захисту інформації.

Перший розділ дослідження присвячено аналізу ключових аспектів кібербезпеки та ролі паролів у захисті особистих даних. Аналізуючи ці аспекти, було виявлено потенціал у токенах, заснованих на часі, як

ефективному інструменті забезпечення тимчасової та унікальної ідентифікації.

У другому розділі проведено аналіз різноманітних методів створення одноразових паролів, виокремлено їх переваги та недоліки. Одноразові паролі виступають важливим інструментом у захисті даних, а вибір конкретного методу генерації залежить від контексту та вимог безпеки системи.

Розділ третій складається з дослідження та аналізу різних методів імплементації алгоритму обмеження доступу до ресурсів за часовим параметром. Основна увага приділена модифікації токена Time-Based One-Time Password (TOTP) та обґрунтуванню вибору конкретних інструментів та методів для його модифікації.

Четвертий розділ містить результати тестування модифікованого алгоритму та показав його високу швидкодію, ефективність у використанні ресурсів та стійкість до атак.

У *висновках* представлені результати проведеної роботи.

Робота представлена на 103 аркушах, містить посилання на список використаних літературних джерел.

Ключові слова: одноразові паролі, кібербезпека, автентифікація, захист інформації, криптографія, генерація паролів, розповсюдження паролів, TOTP.

ABSTRACT

Actuality of theme. The master's dissertation is dedicated to the development and analysis of the modified Time-Based One-Time Password algorithm for restricting access to resources in high-load systems, significant in the realm of cybersecurity and technology. In the modern digital environment, safeguarding information against cyber threats is exceedingly relevant. High-load systems require an efficient and rapid authentication method to ensure security and avoid resource overload. The modified TOTP algorithm could become an effective tool for ensuring security and reducing system load. Such research holds great significance in enhancing protection methods and ensuring the resilience and speed of access in high-load systems.

The object of the study is the use of the Time-Based One-Time Password (TOTP) algorithm for restricting access to resources in high-load systems.

The subject of the research is the development and analysis of the modified Time-Based One-Time Password (TOTP) algorithm to optimize and enhance security in restricting access to resources in high-load systems.

The purpose of the work: is to study, develop, and evaluate the modified Time-Based One-Time Password (TOTP) algorithm to enhance the efficiency and security of access restriction to resources in high-load systems. The main goals include analyzing existing access restriction methods, developing the modified TOTP algorithm, identifying and assessing its advantages concerning efficiency, reliability, and resilience against potential cyber threats. Additionally, the aim involves defining optimal conditions and scenarios for implementing this algorithm in high-load systems to ensure high security levels and mitigate negative impacts on system productivity.

The scientific novelty is the purpose of this work is to develop and analyze a modified TOTP algorithm aimed at improving the process of restricting access to resources in high-load systems. This algorithm incorporates novel methods for generating and verifying one-time passwords based on time, contributing to

increased efficiency, resilience, and security of systems compared to standard authentication methods. The research also considers optimizing the algorithm to operate in high-load environments, which distinguishes it from previous approaches and presents significant novelty in the field of cybersecurity and the application of authentication algorithms in modern information systems.

The practical value is the obtained research outcomes hold significant practical value across various domains such as business, finance, e-commerce, and others. They can serve as the foundation for enhancing information security by developing and implementing more reliable data protection systems and authentication methods.

Approbation of work. The main findings and ideas were presented and discussed at the CXXXIV International Internet Conference "Current Aspects of Science and Technology Development in Times of War" and the LVVI International Internet Conference "Science 2023: Research and Innovation".

Structure and scope of work. The master's thesis comprises an introduction, four chapters, and conclusions.

The *introduction* provides an overview of the general characteristics of the work, focusing on crucial aspects of information security and existing methods of information protection.

The first chapter delves into analyzing key aspects of cybersecurity and the role of passwords in safeguarding personal data. Through this analysis, the potential of time-based tokens as an effective tool for temporary and unique identification was identified.

The second chapter involves an analysis of various methods for generating one-time passwords, highlighting their advantages and disadvantages. One-time passwords play a significant role in data protection, and the selection of a specific generation method depends on the context and security requirements of the system.

The third chapter comprises the exploration and analysis of different methods for implementing access restriction algorithms based on time parameters. It focuses primarily on the modification of the Time-Based One-Time Password (TOTP) token and justifies the selection of specific tools and methods for its modification.

The fourth chapter presents the results of testing the modified algorithm, demonstrating its high speed, efficiency in resource utilization, and resistance to attacks.

The *conclusions* section encapsulates the findings of the conducted research.

The thesis consists of 103 pages and includes a reference list of utilized literary sources.

Keywords: one-time passwords, cybersecurity, authentication, information security, cryptography, password generation, password distribution, TOTP.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

OTP - Одноразовий пароль

TOTP - Часовий Чотириразовий Одноразовий Пароль

HOTP - Хеш-повідомлення на основі HMAC для Одноразових Паролів

HMAC - Код аутентифікації повідомлення за допомогою ключа

SHA-1 - Безпечний Хеш-алгоритм 1

SHA-256 - Безпечний Хеш-алгоритм з довжиною хеша 256 біт

SMS - Коротке повідомлення

IoT - Інтернет Речей

CSPRNG - Криптографічно стійкий псевдовипадковий генератор

PRNG - Псевдовипадковий генератор

MD5 - Алгоритм хешування повідомлень 5

AES - Розширений стандарт шифрування

CTR - Режим лічильника

CFB - Режим зворотного зв'язку шифру

ECB - Режим електронної кодової книги

DES - Стандарт шифрування даних

OFB - Режим зворотного зв'язку по вихідним байтам

XOR - Виключне "або"

RC4 - Шифр Рівеста

TLS/SSL - Протокол захищеного з'єднання/Протокол захищеного сокету

HTTP - Протокол перенесення гіпертексту

JWT - JSON Веб-токен

API - Інтерфейс програмування застосунків

ВСТУП

В сучасному світі високонавантажені системи стають ключовим елементом у багатьох сферах, таких як фінанси, інформаційні технології, медицина та багато інших. Забезпечення безпеки та обмеження доступу до цих систем стає дедалі важливішим завданням через зростання кількості загроз кібербезпеці та несанкціонованого доступу до конфіденційної інформації.

Високонавантажені системи є магнітом для кіберзлочинців через велику кількість цінної інформації, яку вони містять або обробляють. Проблеми, пов'язані з обмеженням доступу до ресурсів у таких системах, стають критичними в аспекті кібербезпеки. Неправильно налаштована система обмеження доступу може призвести до збільшення ризику кібератак та несанкціонованого доступу, витоку конфіденційної інформації або навіть порушення цілісності даних. Крім того, відсутність ефективних методів обмеження може викликати перевантаження системи, що призведе до недоступності для користувачів або зниження продуктивності.

Застосування спеціалізованих методів обмеження доступу є важливим у розв’язанні цих проблем. Такі методи дозволяють точно налаштовувати правила доступу, забезпечуючи захист від кіберзагроз, ефективне використання ресурсів і збереження конфіденційності даних. Вони створюють можливість гнучко керувати доступом до ресурсів, спрощуючи адміністрування і моніторинг, тим самим забезпечуючи безпеку і ефективність у високонавантажених системах.

Основна мета моєї роботи полягає в розробці та аналізі модифікованого алгоритму Time-Based One-Time Password (TOTP) з метою підвищення рівня безпеки та ефективності обмеження доступу до ресурсів у високонавантажених системах. Це дослідження спрямоване на створення покращеного механізму, який забезпечить високий рівень безпеки під час

доступу до цінних ресурсів, знизить ймовірність несанкціонованого доступу та оптимізує використання системних ресурсів.

Основні завдання дослідження включають аналіз існуючих методів обмеження доступу до ресурсів, зокрема, огляд алгоритмів аутентифікації на основі TOTP. Крім цього, робота передбачає розробку модифікованого алгоритму TOTP, який буде оптимізований для використання в високонавантажених системах. Це включатиме створення та реалізацію алгоритму генерації та перевірки токенів, які забезпечують не лише високий рівень безпеки, але й ефективність у процесі доступу до ресурсів.

Моє дослідження відображає актуальність у сучасному світі технологій та кібербезпеки через різко зростаючий обсяг цифрових атак та несанкціонованого доступу до важливих ресурсів. У високонавантажених системах, де доступ до цінних даних та послуг здійснюється через мережі, стає ключовою проблемою забезпечення ефективного та безпечного обмеження доступу. Застосування модифікованого алгоритму Time-Based One-Time Password (TOTP) в цьому контексті є важливим, оскільки цей підхід спрямований на покращення безпеки та ефективності автентифікації.

До того ж, у високонавантажених системах, де швидкість та ефективність мають велике значення, застосування модифікованого алгоритму TOTP може сприяти оптимізації використання ресурсів, так як цей підхід дозволяє забезпечити безпеку доступу зі значно меншими витратами ресурсів системи порівняно з традиційними методами автентифікації. Така система може забезпечити швидкий та надійний доступ до ресурсів у високонавантажених умовах, зменшуючи навантаження на сервери та забезпечуючи високий рівень безпеки.

Отже, впровадження модифікованого алгоритму TOTP в високонавантажених системах може зробити процес автентифікації більш ефективним та безпечним, що є критичним у сучасному цифровому середовищі.

1. ОСНОВНІ ПОНЯТТЯ ТА ТЕОРЕТИЧНІ АСПЕКТИ

1.1 Основні поняття кібербезпеки

Сьогоднішній світ інформації вражає своєю доступністю та розмаїттям, але разом з цим виникають питання, пов'язані з контролем за доступом до цих даних. Стрімкий розвиток технологій та цифровізація усіх сфер життя породжують проблеми з відсівом інформації.

Проблема з доступністю матеріалів часто полягає у величезній кількості інформації, доступ до якої може бути складно врегулювати. Це може призвести до перенасичення даними, де користувачі мають проблеми із визначенням релевантної та достовірної інформації. Однак, існує також потреба у контролі за доступом до інформації з питань безпеки та приватності. Одноразові паролі, як один із засобів захисту, стають необхідними для забезпечення конфіденційності та безпеки особистих даних.

Для вирішення цих проблем важливо розглядати методи обмеження доступу до інформації. Використання технологій шифрування, систем автентифікації та авторизації, а також управлінням доступом допомагає регулювати, хто має доступ до певної інформації та яким чином вона використовується. Такий підхід до обмеження доступу до інформації стає важливим у забезпеченні конфіденційності, захисту приватності та збереження якості та достовірності інформації. Тож сучасному цифровому віці, де комп'ютери та інтернет стали неодмінною частиною повсякденного життя, поняття кібербезпеки набуло вельми важливого значення.

Експерти з кіберзахисту української компанії Datami у своїй статті пояснили, чим небезпечний взлом облікового запису. Фахівці вважають, що облікові записи соціальних мереж містять чимало важливих особистих даних. Люди часто надсилають одне одному номери банківських карток чи

іншу приватну інформацію. Хакери можуть легко знайти цю інформацію і використати у своїх інтересах [5].

Кібербезпека охоплює комплекс процесів, рекомендацій та технологічних рішень, спрямованих на захист важливих систем і мережі від кібератак. За останні роки об'єм інформації став значно більшим, а користувачі здійснюють взаємодію з даними з різних джерел. У зв'язку з цим кіберзлочинці використовують складні та вдосконалені методи для отримання несанкціонованого доступу до цих ресурсів, викрадення інформації, порушення роботи компаній або вимагання викупу. Кількість кібератак постійно зростає, і зловмисники постійно вдосконалюють свої методи, щоб уникнути виявлення та протидії [1].

Однією з ключових мет кібербезпеки є забезпечення конфіденційності, цілісності та доступності інформації. Конфіденційність гарантує, що інформація не може бути доступна несанкціонованим користувачам або процесам. Цілісність стежить за тим, щоб інформація залишалася недоторканною та не зазнавала неправомірних змін. Доступність забезпечує, що авторизовані користувачі мають можливість отримати доступ до необхідних ресурсів у відповідний час [2].

Цілісність інформації гарантує, що дані залишаються непошкодженими та правильними, убезпечуючи їх від будь-яких несанкціонованих змін або порушень. Забезпечення доступності означає, що користувачі мають можливість отримати необхідну інформацію у відповідний час та в необхідному обсязі для здійснення їх робочих або особистих завдань. Ці три принципи - конфіденційність, цілісність та доступність (відомі як принципи Конфіденційності Інформації) - утворюють основу безпеки інформації в сучасних системах та мережах [3].

1.2. Технології автентифікації

Крім того, аспекти автентифікації та авторизації відіграють важливу роль у забезпеченні безпеки. Ідентифікація (identification) – процедура присвоєння ідентифікатора об'єкту комп'ютерної системи або встановлення відповідності між об'єктом і його ідентифікатором; впізнання .

Автентифікація є процедурою перевірки та підтвердження відповідності пред'явленого ідентифікатора об'єкта комп'ютерної системи та його правом на доступ чи взаємодію з системою чи ресурсами. Це важливий етап у визначенні та встановленні достовірності особи чи об'єкта перед наданням прав доступу або виконанням певних операцій у системі. Автентифікація включає в себе перевірку достовірності ідентифікатора та забезпечення, що він належить відповідній особі чи суб'єкту, що намагається отримати доступ [4]. Це важлива складова у забезпеченні безпеки в інформаційних системах та мережах.

Основна мета автентифікації - це перевірка та підтвердження, що особа, яка намагається отримати доступ, дійсно тим, за кого себе видає. Цей процес зазвичай передбачає представлення ідентифікатора (наприклад, ім'я користувача) та відповідного пароля, а також може включати інші методи аутентифікації, такі як використання біометричних даних (відбиток пальця, розпізнавання обличчя, голосове визначення) або токенів безпеки (які генерують одноразові паролі).

В інформаційних технологіях є такі методи автентифікації:

- однобічна, коли клієнт веб сервісу для доступу до даних доводить свою автентичність;
- двобічна, коли, для доступу до інформації повинен доводити автентичність не тільки клієнт, а й веб сервіс;
- трибічна, коли для підтвердження кожного з партнерів в обміні даними, використовується так звана нотаріальна служба автентифікації [9].

Авторизація - це процедура, під час якої визначається дозвіл на виконання певних дій або доступ до конкретних ресурсів. Вона встановлює відповідність між окремим повідомленням чи об'єктом та джерелом, яке його створило або користується ним. Цей процес передбачає надання прав або повноважень об'єкту чи користувачеві для виконання певних дій в системі. Авторизація визначає, які конкретні дії чи операції може виконувати користувач, відповідний процес чи об'єкт у межах системи чи програми, забезпечуючи тим самим контроль над доступом до ресурсів [4].

Сьогодні існує ряд різних технологій, які використовуються для ідентифікації та автентифікації користувачів в інформаційно-комунікаційних системах. Кожна з них має свої переваги та обмеження, що робить їх придатними для використання в певних типах комп'ютерних систем. Однак не завжди можна однозначно вибрати оптимальний метод, оскільки в деяких випадках немає чіткого рішення щодо найкращої технології. Тому як розробники програмного забезпечення, так і користувачі мають самостійно приймати рішення про те, який метод ідентифікації використовувати у своїх інформаційно-комунікаційних системах.

Звичайно, існують різноманітні методи автентифікації, які використовуються для підтвердження особистості користувачів. Наведемо основні з них на рисунку 1.1.

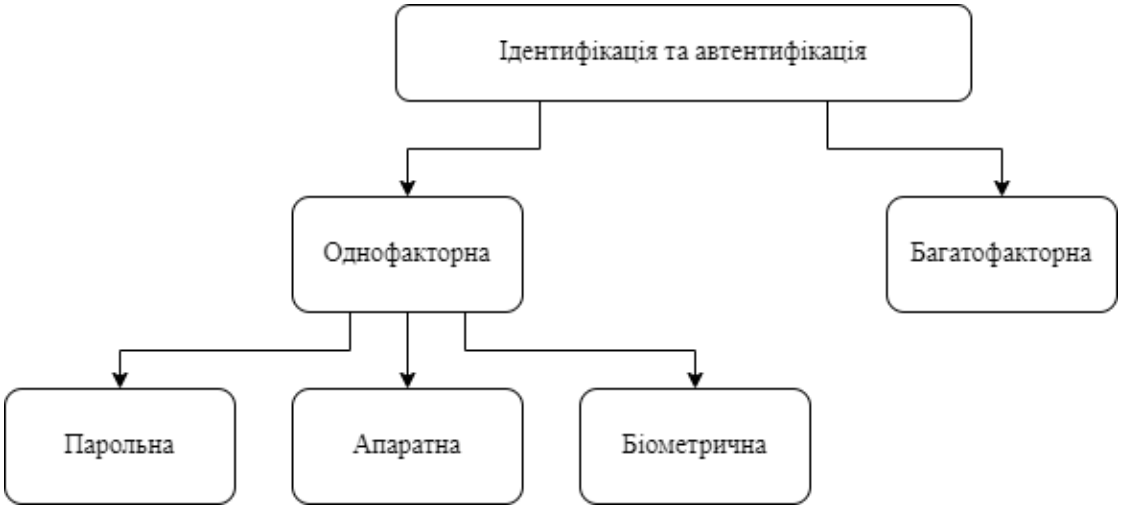


Рисунок 1.1 - види автентифікації

Однофакторна ідентифікація і автентифікація базується на використанні лише одного елемента для перевірки особи. Найпоширенішим прикладом однофакторної системи є введення пароля. Однак такий підхід може бути вразливим, оскільки зловмисники можуть зламати чи отримати доступ до пароля.

У свою чергу, багатофакторна ідентифікація використовує комбінацію двох чи більше елементів для перевірки особи. Це може бути поєднання пароля разом з фізичним пристроєм (таким як токен) або біометричними даними (відбитком пальця чи розпізнаванням обличчя). Цей підхід надійніший, оскільки вимагає наявності декількох незалежних елементів для підтвердження особи.

Парольна ідентифікація базується на введенні спеціальної комбінації символів, які встановлюють ідентичність користувача. Апаратна ідентифікація використовує спеціальні пристрої (токени, смарт-карти) для забезпечення доступу. Біометрична ідентифікація використовує унікальні фізіологічні характеристики (відбиток пальця, розпізнавання обличчя), що є важкими для підробки або втрати. Використання різних методів ідентифікації та автентифікації в комплексі може значно підвищити рівень безпеки систем та даних.

1.2.1 Парольна автентифікація

До недавнього часу парольна ідентифікація була одним з основних методів підтвердження особистості користувача. Однак паролі були простим і широко використовуваним засобом, оскільки були легкі у використанні та імплементації. Тим не менш, через недоліки парольної системи, такі як вразливість до атак, слабка безпека в разі витоку або втрати паролів, виникла потреба у більш безпечних та надійних методах автентифікації [7].

Коли користувач намагається увійти до свого облікового запису, він вводить свій ідентифікатор (наприклад, ім'я користувача чи електронну пошту) разом зі своїм паролем на відповідній сторінці входу. Система обробляє введені дані: вона витягує хешоване значення збереженого пароля у базі даних для введеного ідентифікатора, а потім обчислює хеш введеного пароля. Якщо отриманий хеш співпадає з збереженим в базі, система дозволяє користувачеві увійти до облікового запису.

Цей процес автентифікації ґрунтується на тому, що система не зберігає фактичний пароль, який вводить користувач, а лише його захешовану версію. Такий підхід забезпечує додатковий рівень безпеки, оскільки навіть адміністратор системи не може переглянути фактичний пароль користувача, який зберігається у вигляді хеша.

Ці хеші отримуються шляхом застосування криптографічних хеш-функцій до вихідних паролів.

Хеш-функція – це процес перетворення будь-яких вхідних даних різної довжини в фіксовану послідовність бітів. Ця функція враховує всю інформацію, що ідентифікує певний об'єкт, і створює хеш-код на основі цих даних. Отриманий код може використовуватися для порівняння з іншими хеш-кодами, якщо потрібно. Хеш-функції застосовуються для числового

представлення різних типів мультимедійних даних. Вони також корисні для виявлення порушень авторських прав в Інтернеті та цифрової криміналістики, оскільки дозволяють знайти схожість між хеш-кодами та встановити зв'язок між ними [8].

Парольна аутентифікація має кілька переваг:

- Простота використання: Паролі легко створювати, змінювати та використовувати. Користувачам зручно запам'ятовувати паролі, особливо якщо вони використовують логічні та легкі для них комбінації символів.
- Вартість імплементації: Це один з найдешевших та найбільш доступних методів автентифікації, оскільки не вимагає спеціального обладнання або складних технологій.
- Гнучкість в управлінні: Паролі можна легко змінювати, скасовувати чи оновлювати для кожного користувача окремо без значних труднощів.
- Широке застосування: Парольна аутентифікація використовується практично у всіх сферах, включаючи веб-сайти, мобільні додатки, комп'ютерні системи та інші області.
- Різноманітність політик безпеки: Можливість встановлення складних паролів, вимоги до частоти їх зміни, а також використання додаткових заходів безпеки, наприклад, двофакторної аутентифікації, для підвищення рівня безпеки.
- Здатність до індивідуалізації: Кожен користувач може вибрати унікальний пароль, що дозволяє підвищити безпеку вхідних даних.

Незважаючи на ці переваги, важливо зауважити, що парольна аутентифікація має і свої недоліки.

Часто користувачі вибирають слабкі або легко вгадуванні паролі, такі як "password123", що робить їх вразливими до атак перебору паролів або словникових атак. Наприклад, користувач використовує явно слабкий

пароль (наприклад, "123456") для доступу до свого електронного банкінгу, що робить його обліковий запис вразливим до зловмисницьких атак, але існують різні методи покращення. Один з них це запровадження вимог до складності паролів для цього треба налаштування системи, щоб вимагати від користувачів використання складних паролів, які складаються з різних символів (букви верхнього та нижнього регістрів, цифри, символи) та є достатньо довгими. Також зміна політики паролів що вимагає періодичної зміни паролів (наприклад, кожні 60-90 днів), щоб зменшити ризик в разі можливого компрометації пароля. Допустиме використання парольних фраз, замість простих паролів користувачі можуть використовувати фрази, які легше запам'ятовувати та є більш безпечними. Можна запропонувати використання парольних менеджерів. Користувачі можуть використовувати програми-менеджери паролів, що генерують та зберігають складні та унікальні паролі для різних облікових записів. Необхідне проведення навчання та інформування користувачів про важливість використання міцних паролів, а також про наслідки використання слабких паролів. Один з найдієвіших є двофакторна аутентифікація але це потребує застосування додаткових методів аутентифікації, таких як SMS-повідомлення, аплікації або апаратні токени, разом із паролями таких як OTP (одноразові паролі) для підвищення рівня безпеки.

Ці методи допомагають уникнути використання слабких паролів та підвищують безпеку облікових записів користувачів.

Під час передачі через мережу паролі можуть бути перехоплені, особливо на незахищених або вразливих мережах. Якщо користувач використовує публічну Wi-Fi мережу, його пароль може бути перехоплений зловмисниками, які моніторять мережевий трафік.

Щоб запобігти перехопленню паролю можливе використання шифрування трафіку між користувачем та сервером (наприклад, за допомогою протоколу HTTPS), щоб ускладнити можливість перехоплення

паролів під час їх передачі через мережу. Використання захищених Wi-Fi мереж із зашифрованим з'єднанням (WPA/WPA2) для уникнення перехоплення паролів у відкритих або незахищених мережах. Знову один з найдієвіших способів двофакторна аутентифікація, також допустиме створення складних паролів та регулярна зміна паролів. Важливим є забезпечення своєчасного оновлення програмного забезпечення та використання оновлених версій програм для уникнення використання вразливостей, які можуть бути використані для перехоплення даних, включаючи паролі.

Ці методи допомагають ускладнити атаки на перехоплення паролів та забезпечують додатковий рівень безпеки для захисту конфіденційності та цілісності парольної інформації.

Використання одного паролю для кількох сервісів може призвести до проблем у випадку, якщо один з облікових записів стає скомпрометований , що загрожує безпеці інших сервісів, які використовують той самий пароль.

Наведемо кілька методів захисту. По-перше використання унікальних паролів для кожного облікового запису. Це ускладнює завдання зловмисників, які можуть намагатися отримати доступ до інших сервісів, якщо вони мають доступ до одного паролю. Або використання менеджерів паролів, двофакторна аутентифікація, створення парольних фраз, проведення навчань та інформування користувачів про потенційні ризики використання одного паролю для декількох облікових записів та навчання навичкам створення сильних паролів. Такі методи допомагають уникнути проблем, пов'язаних з використанням одного паролю для багатьох сервісів, та забезпечують додатковий рівень безпеки облікових записів.

Атаки на паролі можуть включати методи соціальної інженерії, які змушують людей розкривати свої паролі або надавати доступ до облікових записів. Наприклад зловмисники можуть використовувати фішингові атаки (наприклад, підроблені сторінки входу) для переконання користувачів

розкрити свої паролі або особисту інформацію. Такий метод можна попередити наданням навчання та інструкцій користувачам про типові методи атак соціальної інженерії, такі як фішинг, перехоплення ідентифікаторів, шахрайство по телефону тощо.

Ці недоліки підкреслюють важливість вибору міцних та унікальних паролів, а також використання додаткових заходів безпеки, таких як двофакторна аутентифікація, для захисту від можливих атак.

Недоліком парольної ідентифікації є значна залежність надійності ідентифікації від користувачів, точніше від обраних ними паролів. Справа в тому, що більшість людей використовують ненадійні ключові слова, які легко підбираються. Фахівці в області інформаційної безпеки радять використати довгі паролі, що складаються з безладного сполучення букв, цифр і різних символів [7].

1.2.2. Апаратна автентифікація

Апаратна або електронна ідентифікація - це метод, який базується на визначенні особистості користувача за допомогою конкретного предмету або ключа, який належить лише йому. Цей підхід не ґрунтується на типових ключах, які використовують більшість людей, а на спеціальних електронних пристроях [10].

Апаратна ідентифікація може включати в себе використання різноманітних технологій, таких як біометричні сканери (відбиток пальця, розпізнавання обличчя, сканери радужки ока), апаратні токени (такі як USB-ключі або смарт-карти), або інші фізичні пристрої, що дозволяють підтвердження ідентифікації.

Апаратна ідентифікація, заснована на використанні фізичних пристроїв для підтвердження особистості або системи, має декілька важливих переваг. Вона забезпечує високий рівень безпеки, оскільки

вимагає наявності фізичного пристрою для доступу. Це ускладнює спроби несанкціонованого входу до системи через необхідність наявності апаратних засобів. Порівняно з паролями, які можуть бути скомпрометовані або вкрадені, апаратні пристрої стійкіші до кібератак.

Крім того, апаратна ідентифікація має певну зручність та простоту використання: деякі методи, такі як біометричні сканери, можуть бути швидкими та зручними у використанні, не потребуючи запам'ятовування складних паролів. Це сприяє індивідуальності, оскільки біометричні дані є унікальними для кожної особи. Менша ймовірність забуття чи втрати також є однією з переваг апаратної ідентифікації: вона може бути менше схильною до забуття, оскільки апаратні пристрої можуть бути прикріплені до особи чи зберігатися в безпечному місці. Такі переваги роблять апаратну ідентифікацію важливим методом забезпечення безпеки в різних сферах. Апаратна ідентифікація, хоча і є ефективним методом забезпечення безпеки, має кілька недоліків. Один з найбільших недоліків полягає у можливості втрати або пошкодження фізичних пристроїв, що використовуються для апаратної ідентифікації. Наприклад, якщо відбиток пальця, який використовується для аутентифікації, пошкодиться або стане недоступним через травму, це може призвести до ускладнення процедури доступу.

Іншим недоліком є витрати на впровадження та підтримку апаратних пристроїв для ідентифікації, таких як біометричні сканери або апаратні токени. Ці пристрої потребують фінансових витрат на придбання, установку та обслуговування, що може бути високим заходом витрат для організацій.

Крім того, застосування біометричних даних для ідентифікації може створювати певні проблеми в забезпеченні приватності та безпеки. Якщо біометричні дані потрапляють у несанкціоновані руки або стають

доступними для зловмисників, це може призвести до серйозних наслідків для конфіденційності користувача.

Для запобігання недолікам апаратної ідентифікації, можна виконати кілька заходів. Наприклад, для уникнення втрати або пошкодження фізичних пристроїв, можна створити резервні копії або альтернативні методи ідентифікації. Також важливо вести контроль за фізичними засобами, щоб уникнути їхнього неправильного використання чи втрати.

Управління та захист біометричних даних, включаючи шифрування та забезпечення безпеки під час їх зберігання та передачі, є важливими аспектами для запобігання витоку цих даних. Проведення аудиту безпеки та встановлення строгих правил доступу до біометричних даних також може сприяти уникненню ризиків.

Цей метод ідентифікації надає високий рівень безпеки, оскільки вимагає наявності фізичного пристрою для доступу, і у разі втрати пристрою або несанкціонованого доступу до нього ідентифікація стає значно складнішою. Апаратна ідентифікація широко використовується в банківській сфері, управлінні доступом до приміщень, а також в комп'ютерних системах для підвищення рівня безпеки.

1.2.3. Біометрична аутентифікація

Біометрична ідентифікація ґрунтується на унікальних біологічних ознаках людини, які є властивими лише їй. Технології біометрії завжди розвивалися з метою точного підтвердження особистості конкретної людини [11].

Замість використання паролів або карток, біометрична ідентифікація використовує такі ознаки, як відбиток пальця, розпізнавання обличчя, структура райдужки ока, голосові зразки, ходьба тощо.

Цей метод ідентифікації використовує унікальні анатомічні або поведінкові особливості, які важко підробити чи скопіювати. Біометричні дані зазвичай конвертуються у цифровий формат та зберігаються в безпечних базах даних для подальшого порівняння при ідентифікації особи.

Біометрична ідентифікація має кілька значних переваг, які роблять її привабливим методом в багатьох галузях.

Висока стійкість до шахрайства: Однією з ключових переваг є висока стійкість до шахрайства та підробки. Оскільки біометричні дані базуються на унікальних фізіологічних чи поведінкових характеристиках кожної особи, їх значно складніше підробити чи скопіювати, порівняно з паролями або картками доступу.

Біометричні дані унікальні для кожного: Кожна людина має свої унікальні біометричні дані, що робить цей метод ідентифікації надзвичайно індивідуалізованими. Відбиток пальця, розпізнавання обличчя чи інші біометричні дані є унікальними для кожної людини, що підвищує точність ідентифікації.

Зручність використання та швидкий доступ: Використання біометричних методів ідентифікації може бути швидким та зручним для користувачів. Наприклад, відбиток пальця чи розпізнавання обличчя можуть бути легко використані без необхідності запам'ятовування складних паролів і номерів.

Зниження ризику втрати паролів чи карток доступу: Оскільки біометричні дані є частиною фізичного тіла, вони менш схильні до втрати чи забуття, порівняно з картками доступу чи паролями, що може спростити процес ідентифікації.

Підвищення рівня безпеки: Використання біометричних даних може підвищити рівень безпеки в різних сферах, таких як фінанси, медицина, управління доступом, оскільки цей метод ідентифікації забезпечує високий

рівень впевненості у правильності особи, яка намагається отримати доступ до системи чи послуги.

Хоча біометрична ідентифікація має багато переваг, вона також має деякі недоліки, на які варто звернути увагу.

Недостатня надійність: Певні біометричні системи можуть мати недоліки в точності та надійності. Наприклад, розпізнавання обличчя може бути менш ефективним в умовах поганого освітлення або при зміні зовнішності особи (зміна зачіски, окулярів тощо). Це може призвести до помилкових відмов у доступі або спричинити потребу в додатковій автентифікації.

Викрадення та злам біометричних даних: Якщо біометричні дані не зберігаються або не обробляються належним чином, вони можуть бути вкрадені чи скомпрометовані. Наприклад, виявлені випадки, коли сканери відбитків пальців або дані про обличчя були вкрадені з метою несанкціонованого доступу. Це ставить під загрозу конфіденційність та приватність користувачів.

Проблеми з приватністю: Використання біометричних даних може породжувати проблеми з приватністю. Збір, зберігання та обробка таких особистих даних може порушити приватність особи, особливо якщо ці дані потрапляють у несанкціоновані руки або використовуються без дозволу.

Методи запобігання: Для запобігання цим недолікам, важливо вжити заходів забезпечення безпеки. Це включає в себе застосування криптографічних методів для захисту біометричних даних, встановлення багаторівневих систем аутентифікації, використання додаткових методів перевірки особи (наприклад, підтвердження біометричних даних за допомогою пароля) та забезпечення високої захищеності систем зберігання біометричних даних.

Біометрична ідентифікація є потужним інструментом, проте важливо розуміти та усувати можливі ризики та недоліки для забезпечення максимальної безпеки та захисту приватності користувачів.

Цей метод ідентифікації використовується в багатьох сферах, включаючи доступ до комп'ютерів, смартфонів, фізичних приміщень, фінансових установ, медичних закладів тощо. Біометрична ідентифікація має великий потенціал у полегшенні ідентифікації осіб та підвищенні рівня безпеки в різних сферах життя.

1.2.4. Багатофакторна аутентифікація

Багатофакторна аутентифікація — це процес перевірки особистості користувача, який використовує два або більше різних методів підтвердження. Ці фактори можуть поєднуватися у будь-якому порядку. Зазвичай використовується одна пара: пароль та токен. Це дозволяє зменшити ризик, оскільки зловмиснику потрібно мати як пароль, так і токен для несанкціонованого доступу. Однак у деяких системах використовуються більш надійні методи ідентифікації, які включають паролі, токени і біометричні характеристики користувача одночасно [7].

Багатофакторна аутентифікація має декілька значних переваг, які роблять її привабливим та ефективним методом забезпечення безпеки.

Високий рівень безпеки: Однією з основних переваг є вищий рівень безпеки порівняно з однофакторною аутентифікацією. Комбінування кількох різних факторів аутентифікації (наприклад, пароль, біометричні дані, мобільний пристрій) робить процес доступу складнішим для несанкціонованих осіб.

Менша ймовірність несанкціонованого доступу: Завдяки використанню кількох етапів підтвердження особистості, багатофакторна аутентифікація знижує ймовірність несанкціонованого доступу.

Інтегруючи різні методи аутентифікації, цей підхід важче обійти зловмисникам, які намагаються отримати доступ до системи чи пристрою.

Захист конфіденційності та особистої інформації: Багатофакторна аутентифікація дозволяє зберігати конфіденційні дані у безпечних умовах, оскільки вона вимагає використання різних видів ідентифікації. Це забезпечує більшу захищеність особистої інформації користувача.

Забезпечення додаткової гнучкості та зручності: Іноді користувачі мають можливість вибору методу аутентифікації, який їм зручний. Наприклад, вони можуть обрати між використанням біометричних даних та підтвердженням через мобільний пристрій, що дозволяє відчувати більшу гнучкість у виборі способу доступу.

Загалом, багатофакторна аутентифікація є ефективним засобом забезпечення безпеки, що знижує ризик несанкціонованого доступу та підвищує рівень захисту конфіденційної інформації користувачів.

Багатофакторна аутентифікація, хоча й є ефективним засобом захисту, все ж має кілька недоліків та викликів, на які важливо звернути увагу.

- Складність впровадження для деяких користувачів: Для деяких користувачів, особливо тих, хто менше знайомий з технологіями або має обмежені навички, процес багатофакторної аутентифікації може бути складним. Наприклад, вимагається використання додаткових пристроїв або методів, що може бути заплутано для окремих користувачів.
- Можливість втрати або крадіжки пристроїв: Оскільки деякі методи багатофакторної аутентифікації базуються на володінні певними пристроями (наприклад, мобільні телефони або токени), втрата чи крадіжка цих пристроїв може стати проблемою. Це може призвести до неспроможності отримати доступ до системи чи послуг, а також до

можливості доступу зловмисників до важливої інформації, якщо пристрій буде використано для аутентифікації.

- Потреба у додаткових ресурсах та інфраструктурі: Впровадження багатофакторної аутентифікації може потребувати додаткових фінансових ресурсів для покупки пристроїв чи розвитку необхідної інфраструктури. Наприклад, для реалізації біометричної аутентифікації можуть знадобитися спеціальні сканер або програмне забезпечення.
- Методи запобігання: Для уникнення недоліків багатофакторної аутентифікації, важливо вживати заходів запобігання. Це може включати резервне копіювання чи віддалене блокування пристроїв у випадку їх втрати або крадіжки, надання інструкцій та підтримки користувачам щодо використання нових технологій, а також регулярне оновлення систем безпеки для запобігання можливих загроз.

Хоча багатофакторна аутентифікація є потужним інструментом забезпечення безпеки, важливо усвідомлювати та управляти можливими викликами та недоліками, щоб максимально використовувати її переваги.

Приклад першого використання багатофакторної авторизації від Google який ввів опційну функцію двоетапної перевірки. Як показано на рисунку 1.2.

Рисунок 1.2 – перше використання двофакторної авторизації

Користувачі, увійшовши вперше до системи Google з нового пристрою за допомогою звичайної автентифікації через ім'я користувача та пароль, отримують запит на введення шестизначного коду перевірки. Цей код можна отримати через SMS-повідомлення або голосовий дзвінок на зазначений телефон, за допомогою офлайн-додатка на смартфоні, або скориставшись одноразовим "scratch code", що був заздалегідь згенерований і збережений на сторінці налаштувань [12].

Цей підхід дозволяє підвищити рівень безпеки, оскільки для отримання доступу необхідно успішно пройти кілька різних етапів аутентифікації. Такий метод стає складнішим для зловмисників, які намагаються отримати несанкціонований доступ до системи.

1.3 Роль паролів у забезпеченні безпеки

Зважаючи на сутність цифрової епохи, паролі відіграють важливу роль у забезпеченні безпеки. Вони стали ключовим елементом автентифікації, який дозволяє системам перевіряти ідентичність користувача перед наданням доступу до акаунтів, пристроїв чи інших

ресурсів, паролі виконують кілька важливих функцій у забезпеченні безпеки.

По-перше, вони служать засобом аутентифікації користувача, ідентифікації його права на доступ до системи або аккаунта. Пароль перевіряє ідентичність особи, що намагається увійти, і у разі успішної перевірки дозволяє отримати доступ.

По-друге, паролі допомагають обмежити доступ до конфіденційної інформації чи ресурсів лише тим особам, які мають право на цей доступ. Це сприяє забезпеченню приватності та безпеки даних.

Третя функція паролів полягає у захисті від несанкціонованого доступу. Використання складних та надійних паролів ускладнює спроби зламу системи або аккаунта, зменшуючи ризик несанкціонованого входу. Паролі також є ключовим елементом у впровадженні багатофакторної аутентифікації, яка забезпечує додатковий рівень безпеки. Вони можуть використовуватися разом з іншими методами, такими як біометричні дані чи одноразові паролі, щоб зробити процес аутентифікації більш надійним.

У цілому, паролі відіграють ключову роль у забезпеченні безпеки, забезпечуючи захист від несанкціонованого доступу, контролюючи доступ до інформації та даних, а також створюючи додатковий шар безпеки через багатофакторну аутентифікацію.

Традиційні текстові паролі є найпоширенішим типом, що використовується для захисту доступу до цифрових ресурсів. Вони можуть бути складними комбінаціями літер, цифр та спеціальних символів. Біометричні паролі, основані на унікальних біологічних характеристиках користувача, таких як відбитки пальців чи розпізнавання обличчя, використовуються для входу в пристрої або системи, де важлива висока ступінь безпеки.

Одноразові паролі (OTP) стали популярними для додаткового рівня безпеки. Вони є унікальними та діють лише один раз, часто

використовуються для багатофакторної аутентифікації. Ключі та сертифікати використовуються для захисту даних, шифрування і підтвердження ідентичності, часто в корпоративних середовищах.

Ці різні види паролів відіграють важливу роль у забезпеченні безпеки. Вони допомагають уникнути несанкціонованого доступу, захищають особисті дані та конфіденційну інформацію, а також сприяють створенню більш надійних систем безпеки за допомогою багатофакторної аутентифікації.

Парольна безпека в сучасному світі є невід'ємною складовою цифрового простору, впливаючи на різні сфери та аспекти нашого життя. Вона відіграє важливу роль у захисті особистих даних, де цифрове зберігання інформації стало стандартом. Паролі виступають не лише як перший захист особистих даних від несанкціонованого доступу, а й як перешкода для кіберзлочинців, що ускладнює спроби вторгнутися у системи та облікові записи.

У бізнесі, паролі відіграють ключову роль у захисті конфіденційної інформації компаній, допомагаючи у запобіганні кібератак та збереженні цылысносты бізнесу. Збільшення уваги до парольної безпеки також сприяє підвищенню свідомості про кібербезпеку серед користувачів, стимулює їх до створення складних та надійних паролів, а також усвідомлення важливості захисту власної особистої інформації.

Додатково, у багатьох галузях існують правила та вимоги до безпеки паролів, що змушує компанії більше уважно ставитися до захисту конфіденційної інформації та використовувати ефективні механізми безпеки. Узагальнюючи, парольна безпека має величезний вплив на забезпечення приватності, запобігання кіберзлочинності та створення свідомої культури кібербезпеки в сучасному суспільстві.

Нижче наведено кілька ключових аспектів її впливу:

- Захист особистих даних: У нашому цифровому світі, де особиста інформація зберігається у електронному форматі, паролі стали першим бар'єром захисту. Вони є основним засобом захисту особистих даних користувачів від несанкціонованого доступу, крадіжки та злому.
- Кібербезпека та протидія кіберзлочинності: Міцні та надійні паролі створюють важкі умови для кіберзлочинців. Вони ускладнюють спроби отримання несанкціонованого доступу до систем, облікових записів чи мереж.
- Безпека в корпоративному середовищі: У бізнесі, паролі відіграють ключову роль у захисті конфіденційної інформації компаній. Забезпечення безпеки паролів у великих організаціях має величезне значення для запобігання кібератак та захисту бізнесу.
- Підвищення свідомості про кібербезпеку: Збільшення уваги до парольної безпеки сприяє підвищенню загальної свідомості про кібербезпеку серед користувачів. Люди вчаться створювати складні паролі та розуміють важливість захисту своєї особистої інформації.
- Регулювання та відповідальність: У багатьох галузях, особливо у фінансовій та медичній сферах, існують правила та вимоги до безпеки паролів, що стимулює компанії приділяти більше уваги захисту інформації.

Узагальнюючи, парольна безпека впливає на різні аспекти життя. Вона є ключовим елементом у протидії кіберзагрозам, захисті особистих та конфіденційних даних, а також у формуванні свідомої культури кібербезпеки в суспільстві.

Паролі відіграють критичну роль у забезпеченні безпеки в цифровому світі. Вони є основним інструментом аутентифікації, захисту особистих даних та запобігання несанкціонованому доступу до різноманітних систем та ресурсів.

Важливість парольної безпеки полягає у їхній здатності захищати конфіденційні дані та особисту інформацію від кіберзагроз, що постійно зростають. Міцні та унікальні паролі є першою лінією оборони в онлайн-світі, вони визначають надійність захисту інформації.

Майбутні перспективи парольної безпеки включають постійний пошук нових методів аутентифікації, які будуть більш безпечними та зручними для користувачів. Застосування біометричних даних, технології розпізнавання особистості, а також багатофакторна аутентифікація стають більш популярними та важливими методами у галузі кібербезпеки. Додатково, розробка більш прогресивних систем безпеки та посилення усвідомлення користувачів щодо створення сильних паролів також відіграють ключову роль у подальшому розвитку парольної безпеки.

1.4 Основні принципи одноразових паролів

Одноразові паролі (OTP) - це форма автентифікаційного методу, призначена для забезпечення безпеки доступу до різних систем, облікових записів або сервісів. Основний принцип роботи одноразових паролів полягає у створенні та використанні унікального парольного коду, який діє лише один раз для автентифікації користувача під час входу в систему чи сервіс.

Цей парольний код генерується спеціальними алгоритмами або пристроями та відображається користувачу на декілька секунд, хвилин або до моменту використання. Після того, як він використаний для автентифікації в системі, він стає недійсним для подальшого використання, навіть якщо зловмисники спробують його перехопити.

Одноразові паролі можуть бути створені різними способами, такими як:

- SMS повідомлення з унікальним кодом, який відправляється на мобільний телефон користувача.
- Генерація парольного коду за допомогою спеціальних мобільних додатків або програм для аутентифікації.
- Використання фізичних пристроїв (апаратних токенів), що генерують унікальні паролі.

Одноразові паролі є ефективним засобом захисту від кіберзлочинності, оскільки навіть якщо пароль буде перехоплений, його короткочасна дія та одноразовий характер роблять його непридатним для подальшого використання зловмисниками.

Принцип генерації одноразових паролів (OTP) полягає у створенні унікальних та одноразових кодів, які використовуються для автентифікації користувача. Цей процес може відрізнятися залежно від методу генерації.

Генерація OTP може ґрунтуватися на різних алгоритмах, таких як алгоритм HOTP (HMAC-based One-Time Password) або TOTP (Time-Based One-Time Password). HOTP використовує хеш-функцію для генерації одноразового паролю на основі секретного ключа та лічильника, тоді як TOTP використовує хеш-функцію разом зі значенням часу для створення OTP, який змінюється кожну певну періодичність (наприклад, кожні 30 секунд).

Головна ідея полягає в тому, щоб генерувати унікальний пароль або код, який буде дійсним лише протягом короткого періоду часу (наприклад, кілька секунд або хвилин). Це забезпечує одноразовість та унікальність пароля і ускладнює його використання для атак. Генерація OTP використовує секретний ключ, який відомий лише серверу автентифікації та у пристрої користувача. Цей ключ використовується разом з алгоритмом генерації для створення OTP. Генерація OTP зазвичай базується на використанні хеш-функцій, таких як SHA-1, SHA-256 тощо, які перетворюють вхідні дані (наприклад, секретний ключ і, у випадку TOTP,

поточний час) в унікальний вихідний код. Отриманий OTP може бути представлений користувачу через спеціальний додаток, SMS, або інші засоби зв'язку, і використаний ним для входу в систему або підтвердження транзакцій.

Головна мета генерації одноразових паролів - забезпечити безпеку шляхом створення тимчасових та унікальних кодів, які важко перехопити або використати зловмисниками.

Одноразові паролі (OTP) можуть бути використані через різноманітні технології та методи. OTP може бути надісланий користувачеві у вигляді SMS повідомлення на їхній мобільний телефон.

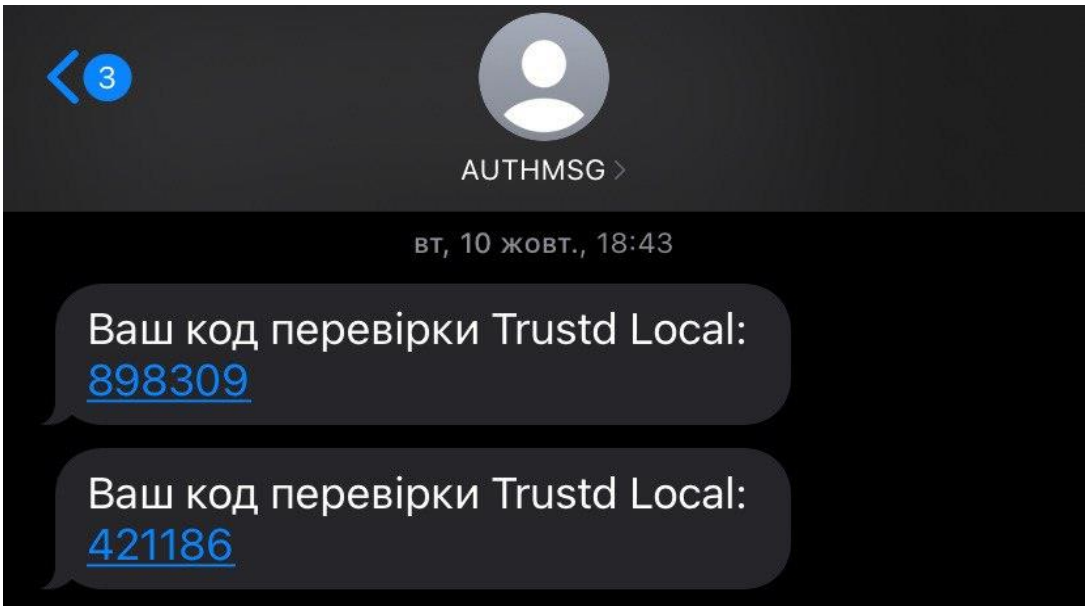


Рисунок 1.3 – приклад OTP через SMS

Цей метод є одним із найпоширеніших, оскільки більшість користувачів мають доступ до мобільних пристроїв. Існують спеціальні мобільні додатки, такі як Google Authenticator, Authy, Microsoft Authenticator та інші, які генерують одноразові паролі безпосередньо на мобільному пристрої користувача.

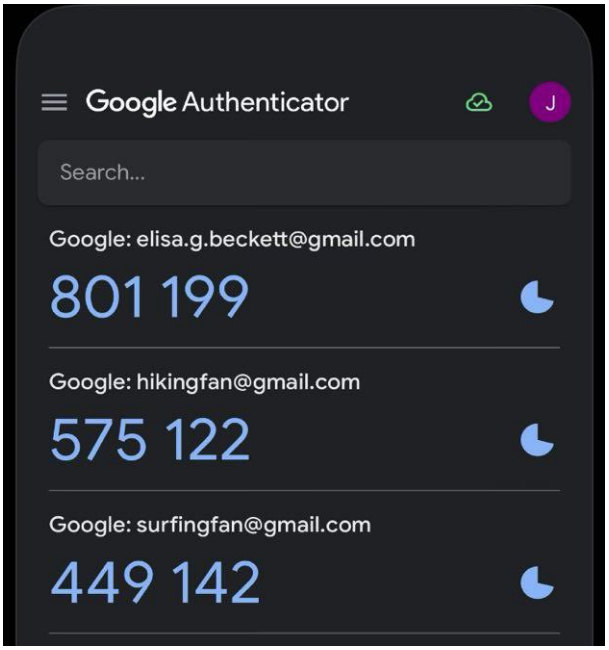


Рисунок 1.4 – приклад OTP через Google Authenticator

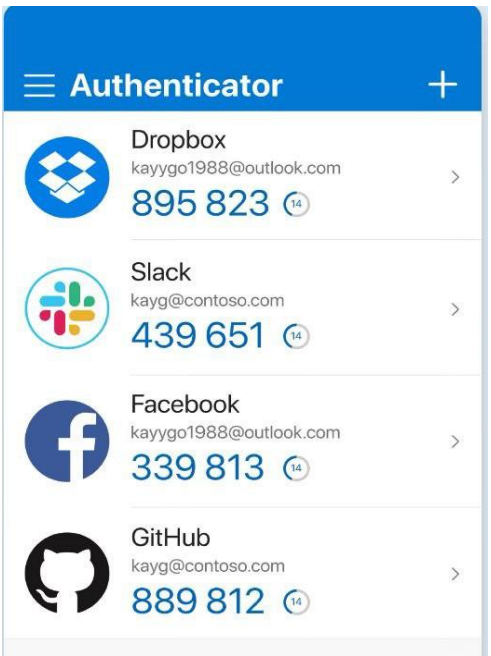


Рисунок 1.5 – приклад OTP через Microsoft Authenticator

Ці додатки зберігають секретні ключі та генерують нові OTP кожні 30 секунд.

Фізичні пристрої (апаратні токени), які генерують і відображають одноразові паролі, ці токени можуть бути пластиковими картками або

ключами, які користувачі мають при собі і використовують для генерації OTP.

Рисунок 1.6 – приклад фізичних генераторів OTP

Програмні токени, технологія яких базується на створенні та управлінні цифровими кодами або програмними пристроями в комп'ютері або на мобільних пристроях користувача.

Кожен із цих методів має свої переваги та недоліки. Вибір конкретного методу може залежати від потреб безпеки, зручності для користувачів, вартості реалізації та інших факторів. Наприклад, SMS може бути дешевшим в реалізації, але менш безпечним через можливість перехоплення повідомлень, у той час як мобільні додатки або апаратні токени можуть бути більш безпечними, але вимагати більшого зусилля для впровадження.

Звичайні статичні паролі можуть мати кілька обмежень, в той час як одноразові паролі (OTP) пропонують додаткові переваги в плані безпеки. Одноразовість та тимчасовість є ключовою особливістю одноразових паролів. Вони діють лише один раз або протягом короткого періоду часу,

після чого стають недійсними. Це ускладнює їх перехоплення та подальше використання зловмисниками. Відмінність полягає у захисті від перехоплення, оскільки одноразові паролі мають короткий термін дії, зменшуючи можливість їхнього використання для несанкціонованого доступу. Це також сприяє зниженню ризику повторного використання. Одноразові паролі втрачають свою дійсність відразу після використання, запобігаючи їх повторне використання для входу. Додатковий рівень безпеки стає можливим через тимчасовий та унікальний характер одноразових паролів, що ускладнює їх передбачення чи використання зловмисниками. Крім того, одноразові паролі можуть використовуватися як складова частина багатофакторної аутентифікації, забезпечуючи більш високий рівень безпеки завдяки використанню кількох методів підтвердження.

При використанні одноразових паролів (OTP) існують деякі обмеження та недоліки. Одним з обмежень є залежність від спеціальних технічних засобів, таких як мобільні додатки чи апаратні токени. Це може ускладнити використання для користувачів, які не мають доступу до цих пристроїв. Втрата доступу до технічних пристроїв, на яких зберігаються одноразові паролі, може призвести до проблем зі входом у систему. Також важливо усвідомити ризик втрати доступу, якщо ці пристрої будуть втрачені чи пошкоджені. Одноразові паролі можуть бути вразливі до фішингових атак та соціального інжинірингу, які можуть призвести до незаконного доступу до облікових записів користувачів. Технічні проблеми, такі як затримки у надходженні SMS або проблеми з мобільними додатками, можуть спричинити ускладнення отримання одноразових паролів. Іноді виникають проблеми з безпекою в самому методі генерації одноразових паролів, що може підірвати довіру до цього методу. Щодо майбутніх перспектив та можливостей вдосконалень, деякі напрями включають розвиток біометричних методів автентифікації, використання

блокчейну для забезпечення безпеки, а також поліпшення технологій для отримання одноразових паролів, щоб спростити процес їх отримання та використання.

Технологія одноразових паролів продовжує розвиватися для покращення безпеки та зручності використання. Ось деякі перспективи та інновації в цій галузі:

- Біометрична аутентифікація: Інтеграція біометричних даних (відбитків пальців, розпізнавання обличчя, сканування райдужки ока) для генерації одноразових паролів може забезпечити ще більш високий рівень безпеки. Це може бути зручним та ефективним способом підтвердження особи.
- Використання блокчейну: Технологія блокчейну може використовуватися для зберігання та забезпечення безпеки унікальних ідентифікаторів, які використовуються для генерації одноразових паролів.
- Розширення IoT: Застосування технологій IoT може дозволити використовувати одноразові паролі для автентифікації в різних зв'язаних пристроях, що розширить їх застосування на побутовому та промисловому рівнях.
- Удосконалені мобільні додатки: Розвиток мобільних додатків для генерації OTP, які працюють на принципі безпосереднього з'єднання з безпровідними або Bluetooth-токенами для автентифікації.

У реальному житті, вдосконалення технологій одноразових паролів можна використовувати в багатьох сферах: від банківської сфери до корпоративного сектору та сфери медичних послуг. Багатофакторна аутентифікація з використанням одноразових паролів вже широко застосовується в бізнесі для підвищення рівня безпеки та захисту конфіденційної інформації. Одноразові паролі також можуть бути використані для захисту від кібератак у споживчих товарах, таких як

особисті акаунти в соціальних мережах, електронна пошта, інтернет-банкінг та інші сервіси онлайн. Застосування цієї технології надає більший рівень захисту та забезпечує безпеку особистих даних.

Одноразові паролі використовуються у різних галузях, щоб забезпечити безпеку та підтвердження ідентифікації користувачів. Наприклад, у фінансовому секторі вони використовуються для підтвердження фінансових транзакцій, дозволяючи клієнтам виконувати перекази чи платежі з використанням унікальних кодів. В медичній сфері одноразові паролі використовуються для доступу до електронних медичних записів та конфіденційних даних про пацієнтів, забезпечуючи безпеку та конфіденційність медичної інформації. У корпоративному секторі вони застосовуються для захисту конфіденційної інформації та доступу до корпоративних систем, забезпечуючи безпеку при роботі з важливими даними. Також, онлайн-сервіси та інтернет-платформи використовують одноразові паролі для підтвердження доступу до особистих облікових записів, підвищуючи рівень безпеки даних користувачів. У сфері безпеки даних та електронної пошти одноразові паролі застосовуються для захисту від несанкціонованого доступу та забезпечення конфіденційності особистої інформації. Ці приклади показують широке використання одноразових паролів у різних секторах для забезпечення безпеки та підтвердження ідентифікації користувачів у цифровому світі.

Одноразові паролі є важливим елементом сучасної кібербезпеки, який відіграє критичну роль у захисті конфіденційної інформації та персональних даних. Вони ґрунтуються на принципі тимчасової дії та унікальності, що робить їх важливим інструментом для підтвердження ідентифікації користувачів.

Застосування одноразових паролів має значний вплив у різних сферах, починаючи від фінансового сектору та закінчуючи медичною сферою та інформаційними технологіями. Вони забезпечують безпеку

фінансових транзакцій, захищають особисті та медичні дані, а також допомагають у захисті корпоративних систем та особистих облікових записів. Одноразові паролі мають свої переваги, такі як унікальність, тимчасовість та захист від фішингу, але також мають деякі недоліки, такі як залежність від технічних засобів та вразливість до втрати доступу до них. Загалом, одноразові паролі є важливим елементом у підвищенні рівня безпеки та захисту особистих даних в цифровому середовищі. їхнє використання сприяє покращенню безпеки в різних секторах та підвищує впевненість користувачів у захищеності їхніх інформаційних активів.
необхідні дані без необхідності постійної взаємодії з базою даних.

Висновки до першого розділу

У першому розділі були розглянуті основні поняття кібербезпеки, роль паролів у забезпеченні безпеки, основні принципи та технології створення одноразових паролів.

Основні поняття кібербезпеки надають розуміння загроз, що існують у цифровому середовищі та методів захисту від цих загроз. Роль паролів у забезпеченні безпеки велика, оскільки вони є першою лінією захисту особистих даних від несанкціонованого доступу. Одноразові паролі, будучи одним із методів аутентифікації, базуються на унікальних кодах, які використовуються лише один раз, підвищуючи безпеку та запобігаючи повторному використанню.

Технології створення одноразових паролів різноманітні: вони можуть базуватися на часових алгоритмах, хеш-функціях, SMS-повідомленнях, мобільних додатках чи апаратних пристроях. Кожен метод має свої переваги та особливості, але спільною рисою є створення тимчасових та унікальних паролів для забезпечення безпеки доступу.

Варто відзначити значний потенціал одноразових паролів, який в сучасному світі посилюється завдяки їхній зростаючій популярності. Це відбувається через їхню несталість, яка стає актуальною завдяки швидкому розвитку передачі даних та обчислювальних можливостей. Зараз існує безліч варіантів одноразових паролів, які надаються в різних форматах.

Вивчаючи ці аспекти, я виявив значний потенціал у токенах, що базуються на часі. Час стає універсальним маркером, за яким можна встановити ідентифікацію майже у будь-якому контексті.

Отже, цей розділ висвітлив ключові аспекти кібербезпеки, значення паролів у захисті даних та технології створення одноразових паролів, які використовуються для забезпечення надійного захисту в цифровому середовищі.

2. МЕТОДИ СТВОРЕННЯ ОДНОРАЗОВИХ ПАРОЛІВ

2.1. Генерація випадкових одноразових паролів

Генерація випадкових одноразових паролів - це процес створення унікальних послідовностей символів, які використовуються тимчасово для доступу до системи чи послуги та не можуть бути повторно використані. Цей метод має на меті забезпечити безпеку та захист користувачів від зловмисників.

Основні принципи генерації випадкових одноразових паролів включають:

- Випадковість: Пароль повинен бути повністю випадковим, тобто неможливим для передбачення. Це забезпечується використанням криптографічно стійких генераторів випадкових чисел або псевдовипадкових алгоритмів.
- Унікальність: Кожен створений пароль повинен бути унікальним, не повторюватися в межах системи чи певного часового періоду.
- Тимчасовість: Одноразовий пароль має бути дійсним лише для одного використання або обмеженого періоду часу, після чого втрачає свою дійсність.
- Безпека передачі та зберігання: Паролі повинні передаватися та зберігатися в зашифрованому або безпечному форматі, щоб уникнути несанкціонованого доступу до них.

Для генерації випадкових одноразових паролів використовуються різноманітні методи, включаючи використання криптографічних алгоритмів, генераторів випадкових чисел, комбінації різних символів (цифри, літери верхнього та нижнього регістрів, спеціальні символи) та інші техніки для створення відповідно до вимог безпеки системи чи сервісу.

Так, генерація випадкових одноразових паролів є важливим елементом забезпечення безпеки системи або послуги. Основна мета

полягає в створенні унікальних послідовностей символів, які є надійними та випадковими, щоб ускладнити їх передбачення зловмисниками.

Методи генерації випадкових одноразових паролів можуть включати наступне: використання криптографічно стійких генераторів випадкових чисел, використання псевдовипадкових алгоритмів, комбінація різних методів, забезпечення високої ентропії.

Використання криптографічно стійких генераторів випадкових чисел наприклад як інтегровані генератори випадкових чисел. В багатьох мовах програмування та криптографічних бібліотеках існують спеціальні функції, що надають доступ до високоякісних криптографічних генераторів випадкових чисел (CSPRNG), є ключовими компонентами в багатьох мовах програмування та криптографічних бібліотеках. Ці генератори забезпечують високий рівень випадковості та стійкості до прогнозування, які використовуються в широкому спектрі застосувань, включаючи генерацію випадкових одноразових паролів.

Високоякісна випадковість, стійкість до прогнозування, криптографічна стійкість, низький рівень кореляції це основні характеристики інтегрованих генераторів випадкових чисел (CSPRNG)

Інтегровані криптографічно стійкі генератори випадкових чисел (CSPRNG) в основному забезпечують ці характеристики за допомогою криптографічних алгоритмів та додаткових методів, що гарантують випадковість чисел. Ці генератори використовують криптографічно стійкі алгоритми для генерації випадкових чисел. Ці алгоритми базуються на складних математичних функціях, які забезпечують високий рівень стійкості до прогнозування. Такі алгоритми, наприклад, можуть використовувати хеш-функції, блочні шифри або інші криптографічні примітиви для створення випадкових чисел.

Для забезпечення випадковості чисел, CSPRNG можуть використовувати надійні джерела ентропії [23]. Ці джерела включають

фізичні процеси, такі як шуми в електричних ланцюгах, випадкові рухи миші або клавіатури, або дані про час та інші фізичні події, які є складними для передбачення. (CSPRNG) зазвичай використовують ітеративний процес генерації випадкових чисел. Це означає, що кожне нове випадкове число генерується на основі попередніх значень, але з використанням складних математичних або криптографічних операцій, що ускладнюють передбачення наступного числа. Криптографічні генератори випадкових чисел можуть мати внутрішні стани, які слід ініціалізувати та підтримувати в безпечному стані. Це означає, що кожне нове випадкове число залежить від попереднього стану генератора та його налаштувань.

Стійкість до прогнозування в інтегрованих криптографічно стійких генераторах випадкових чисел (CSPRNG) визначається декількома факторами, які забезпечують високий рівень випадковості та ускладнюють передбачення наступних значень. CSPRNG використовують надійні джерела ентропії для отримання початкових даних чи "насіння" (seed). Ці генератори призначені для криптографічних застосувань, де важлива стійкість до прогнозування. Такі генератори розроблені таким чином, щоб навіть за наявності деякої кількості попередніх чисел або знанні початкових умов, було дуже складно передбачити подальші числа. Криптографічні алгоритми, які використовуються у CSPRNG, мають велику обчислювальну складність. Це ускладнює злому алгоритму та передбачення наступних значень без знання внутрішніх параметрів. Випадкові числа, генеровані CSPRNG, мають відсутність закономірностей та низький рівень кореляції між собою. Це означає, що кожне наступне число повинне бути практично незалежним від попередніх чисел. Генератори CSPRNG мають велику довжину періоду, що означає, що вони можуть генерувати велику кількість унікальних чисел перед повторенням послідовності. Це важливо для того, щоб уникнути повторних значень та забезпечити випадковість. CSPRNG піддаються тестам на криптографічну стійкість, таким як тести

випадковості та криптоаналізу, для підтвердження їхньої безпеки та випадковості.

Загалом, високоякісна випадковість в інтегрованих криптографічно стійких генераторах випадкових чисел (CSPRNG) забезпечується за рахунок використання надійних алгоритмів, стійких до прогнозування, високоякісних джерел випадковості та відповідної обробки випадкових чисел для забезпечення безпеки та непередбачуваності у генерації чисел. Ці характеристики разом забезпечують високий рівень криптографічної стійкості в інтегрованих криптографічно стійких генераторах випадкових чисел (CSPRNG), роблячи їх важливими для застосувань, де потрібна велика безпека та випадковість.

2.1.1. Види криптографічних алгоритмів

У криптографічно стійких генераторах випадкових чисел (CSPRNG) використовуються різні криптографічні алгоритми та примітиви для забезпечення високої випадковості та стійкості до прогнозування випадкових чисел. Ось деякі з них:

1. Hash-функції: Вони використовуються для обчислення "відбитків" (hashes) вхідних даних. Криптографічно стійкі hash-функції, такі як SHA-256 (Secure Hash Algorithm 256-bit), SHA-3 (Secure Hash Algorithm 3), використовуються для генерації випадкових чисел на основі вхідних даних чи стану генератора. Hash-функції в криптографії використовуються для перетворення довільного вхідного повідомлення фіксованої довжини, яке називається хеш-значенням або хешем. Основна властивість криптографічно стійких hash-функцій - навіть невелика зміна у вхідних даних призводить до значно відмінного хеш-вектора. Також, важко знайти два різних вхідних повідомлення, які мають однаковий хеш-вектор.

Ось кілька прикладів криптографічно стійких hash-функцій та їх характеристик:

- SHA-256 (Secure Hash Algorithm 256-bit): Це один із варіантів функції хешування, що використовується для обчислення хеш-значення з вихідним розміром 256 біт [24].
- MD5 (Message Digest Algorithm 5): Це раніше популярний, але зараз вважається вразливим до колізій (колізія - ситуація, коли два різних вхідних повідомлення мають однаковий хеш).
- SHA-3 (Secure Hash Algorithm 3): Це нова серія хеш-функцій, що використовуються для генерації хешів різної довжини, замінюючи попередні версії SHA.

Звертаю увагу, що ці функції слід використовувати з усіма обережностями, оскільки деякі з них можуть бути вразливими або застарілими через появу нових атак. У важливих криптографічних застосунках рекомендується використовувати найбільш сучасні та перевірені криптографічні функції, щоб забезпечити високий рівень безпеки.

2. Блочні шифри: Деякі CSPRNG використовують блочні шифри, такі як AES (Advanced Encryption Standard), у режимі гамування (CTR, CFB), для генерації випадкових чисел з використанням ключа та вектора ініціалізації.

Блочні шифри - це криптографічні алгоритми, які шифрують дані у вигляді блоків фіксованої довжини (наприклад, 64 або 128 біт) за допомогою ключа. Одним із основних прикладів блочних шифрів є шифр DES (Data Encryption Standard) та його покращений варіант AES (Advanced Encryption Standard).

Основний принцип дії блочного шифру полягає у розбитті вхідного тексту на блоки, які потім шифруються одним ключем у вихідний шифрований текст. Кожен блок у вихідному тексті зазвичай залежить від попередніх блоків та ключа шифрування.

Ось приклад роботи блочного шифру на прикладі простого шифрування у режимі ECB (Electronic Codebook) [25]:

Шифрування: нехай у нас є вхідне повідомлення M та ключ шифрування K.

Кожен блок вхідного повідомлення M шифрується окремо за допомогою ключа K та алгоритму шифрування E. Формула для шифрування блоку може виглядати наступним чином:

$$C_i = E_k(M_i)$$

де C_i - зашифрований блок,

M_i - блок вхідного тексту,

E - функція шифрування з ключем K.

Дешифрування: для отримання вихідного тексту M з шифрованого тексту C використовується процес дешифрування за допомогою того ж самого ключа K та функції дешифрування D:

$$M_i = D_k(C_i)$$

де M_i - дешифрований блок,

C_i - зашифрований блок,

D - функція дешифрування з ключем K.

Режим роботи блочного шифру може бути різним. Наприклад, є режими CBC (Cipher Block Chaining), CFB (Cipher Feedback), а також більш сучасні та безпечні режими, такі як GCM (Galois/Counter Mode) для використання в криптографічних застосунках.Згаданий режим ECB (Electronic Codebook) вважається менш безпечним через властивість шифрування блоків однакових даних у той самий шифротекст, що може призвести до деяких атак. Тому його рідко використовують у практичних застосунках.

Блочні шифри мають різні параметри та режими роботи, проте базовий принцип шифрування блоків за допомогою ключа залишається схожим для багатьох з них.

3. Потікоподібні шифри: CSPRNG можуть використовувати потікоподібні шифри, наприклад, блочні шифри у режимі потоку (OFB, CTR), які перетворюють блоки даних в потік випадкових бітів.

Потокові шифри - це тип криптографічних шифрів, які шифрують дані поодинокими бітами або байтами (на відміну від блочних шифрів, які шифрують блоки даних) за допомогою ключа. Одним з прикладів потокового шифру є RC4. Призначенням потокового шифру є перетворення вхідного потоку даних у шифрований потік тієї ж довжини.

Основний принцип дії потокових шифрів полягає у генерації псевдовипадкової послідовності (потоку ключів), яка потім комбінується з вхідними даними за допомогою операції побітового XOR (ексклюзивне або) для отримання шифрованого потоку.

Ось приклад принципу роботи потокового шифру [26].

Шифрування: нехай у нас є вхідний потік даних *M* та ключ шифрування *K*. Потік ключів генерується за допомогою генератора ключів (псевдовипадковий генератор), який використовує ключ шифрування *K*. Отримана псевдовипадкова послідовність K_s потім комбінується з вхідним потоком даних *M* за допомогою операції XOR. Формула може виглядати так:

$$C = M \oplus K_s$$

де *C* - шифрований потік,

M - вхідний потік даних,

K_s - потік ключів,

$\oplus$ - операція XOR.

Дешифрування процес дешифрування відбувається аналогічно: шифрований потік *C* комбінується з потоком ключів K_s за допомогою операції XOR, щоб отримати вихідний потік даних *M*:

$M = C \oplus K_s$

де C - шифрований потік,

M - вхідний потік даних,

K_s - потік ключів,

$\oplus$ - операція XOR.

Цей принцип показує, як потоковий шифр застосовує потік псевдовипадкових ключів до вхідних даних для шифрування та як саме цей же потік ключів використовується для дешифрування того ж шифрованого потоку .

Хоча потокові шифри можуть бути швидкими та ефективними, вони також можуть бути вразливими до атак, особливо якщо використовуються неправильно або якщо ключі генеруються неякісними генераторами випадкових чисел. Тому важливо використовувати потокові шифри відповідно до рекомендацій та найкращих практик криптографії.

4. (HMAC): Вони використовуються для обчислення аутентифікованих хешів даних, що використовують ключ, і можуть бути використані у сполученні з іншими алгоритмами для створення CSPRNG.

HMAC (Hash-Based Message Authentication Code) - це криптографічний протокол аутентифікації повідомлень, який використовує хеш-функцію у поєднанні з секретним ключем для генерації аутентифікаційного коду для повідомлення. Він дозволяє перевірити цілісність повідомлення та переконатися, що воно не було змінено під час передачі. Основна ідея HMAC полягає у використанні хеш-функції (наприклад, SHA-256, SHA-512) в поєднанні з секретним ключем для створення аутентифікаційного коду для повідомлення. Цей код додається до повідомлення або передається разом з ним для подальшої перевірки цілісності.

Ось формула HMAC для створення аутентифікаційного коду:

$HMAC(K, M) = H((K \oplus opad)||H((K \oplus ipad)||M))$

де K - секретний ключ,

M – повідомлення,

H - хеш-функція,

$\oplus$ - операція XOR,

$opad$ - зовнішній ключ HMAC (визначений як константа, яка складається з байтів 0x5C),

$ipad$ - внутрішній ключ HMAC (визначений як константа, яка складається з байтів 0x36).

Процес створення HMAC може бути такий:

1. Ключ K доповнюється нулями або хешується, якщо він довший за розмір блоку хеш-функції.
2. Ключ K застосовується до $ipad$ та $opad$ через операцію XOR.
3. До результатів операції XOR застосовується хешування.
4. Результат першого хешування $(K \oplus ipad)||M)$ об'єднується з $opad$ через операцію XOR, після чого результат хешується ще раз.

Отриманий HMAC може бути використаний для перевірки цілісності та автентифікації повідомлення. Він перевіряється отримувачем, який має ключ K і може обчислити HMAC для прийнятого повідомлення для порівняння з тим, що було отримано разом із повідомленням.

HMAC широко використовується у протоколах аутентифікації та забезпечення цілісності, таких як TLS/SSL, IPsec, HTTP, і багатьох інших протоколах забезпечення безпеки [16].

5. Fortuna. Це конструкція для CSPRNG, яка поєднує в собі криптографічні алгоритми, такі як хеш-функції та блочні шифри, для генерації випадкових чисел. Fortuna використовує поновлення ключа з кожним використанням та може використовувати декілька генераторів паралельно.

Fortuna - це криптографічний алгоритм, який використовується для генерації високоякісних випадкових чисел, використовуючи комбінацію

криптографічних алгоритмів із поновленням ключа. Він був розроблений Брюсом Шнаєром та Нільсоном Фергюсоном і призначений для забезпечення безпеки та надійності генерації випадкових чисел у системах, які вимагають високої випадковості, таких як шифрування, аутентифікація та інші криптографічні застосування.

Основні особливості Fortuna.

Структура з декількох генераторів. Fortuna використовує декілька незалежних генераторів випадкових чисел. Кожен генератор може бути пов'язаний з різними джерелами випадковості.

Ієрархічна структура ключів. Fortuna використовує ієрархічну структуру ключів для зберігання випадкових даних різного рівня важливості та чутливості.

Регулярне поновлення ключів. Ключі генератора регулярно оновлюються з використанням спеціального алгоритму, який забезпечує поновлення ключів після кожного використання.

Безпечна стійкість до атак. Fortuna розроблений таким чином, щоб залишатися стійким до різних атак, включаючи атаки на ключі, попередні напади на випадкові дані, а також можливі збої генераторів.

Принцип роботи Fortuna можна уявити як систему, яка об'єднує криптографічні алгоритми та генератори випадкових чисел з різних джерел, а потім комбінує та поновлює їх ключі з використанням криптографічних методів.

Fortuna використовує 32 пули ентропії. Вибірки з будь-якого джерела подаються в ці пули через циклічне обертання. Перевантаження відбувається, коли перший пул стає достатньо великим. Перший пул завжди використовується для перевантаження генератора. Другий пул також використовується кожен другий раз. Третій пул також використовується кожен четвертий раз, і так далі. Розв'язок Fortuna в кінцевому рахунку переведе систему в безпечний стан. Fortuna розроблений таким чином, щоб

не мати можливості окремо виводити ентропію. Усю ентропію подають у ці пули, які використовуються лише для перевантаження PRNG. Більшість універсальних систем можуть вважати це неприйнятним використанням ентропії. Мета концепції Fortuna - уникнути оцінки ентропії. Ви можете просто вказати довжину, при якій перший пул виконає перевантаження, і система в кінцевому рахунку досягне безпечного стану. Немає можливості знати, коли це станеться без хорошого оцінювача, особливо коли деякі джерела можуть бути забрудненими. Важко оцінити, чи ймовірно, що PRNG буде безпечним, основуючись на кількості перевантажень. Наприклад, одне забруднене джерело може подати достатньо ентропії для 100 перевантажень до того, як перше джерело надасть будь-яку ентропію для першого пулу. Якщо забруднене джерело викликає багато перевантажень, то воно може забезпечити те, що перші кілька пулів завжди матимуть малі обсяги фактичної ентропії під час перевантаження, що означає, що злоумисник може завдовга перешкоджати досягненню PRNG безпечного стану. Тим не менш, Fortuna в кінцевому рахунку досягне цієї мети [14].

Це лише загальний огляд принципу роботи Fortuna. Для реалізації Fortuna в практичних застосунках використовуються конкретні криптографічні алгоритми, методи поновлення ключів та стратегії зберігання даних, що забезпечують високу безпеку та надійність системи генерації випадкових чисел [14].

6. Yarrow: Це ще одна конструкція для генерації випадкових чисел, що використовує блочні шифри та хеш-функції, включаючи криптографічні алгоритми, які генерують випадкові дані та реєструють певні події для покращення випадковості.

Yarrow є криптографічним алгоритмом генерації випадкових чисел (RNG), який був розроблений та представлений ще в 1999 році Брюсом Шнайером та Джоном Каменом. Yarrow є псевдовипадковим генератором,

який використовує асортимент криптографічних алгоритмів та методів для генерації надійних випадкових чисел.

Використання компонентів, незалежних від алгоритмів, у дизайні верхнього рівня є ключовою концепцією в Yarrow. Мета - не збільшити кількість базових криптографічних модулів, на яких ґрунтується криптографічна система, а максимально використовувати вже існуючі модулі. Тому вони спиралися на односторонні хеш-функції та блочні шифри - дві з найбільш вивчених та широко поширених базових криптографічних основ [15].

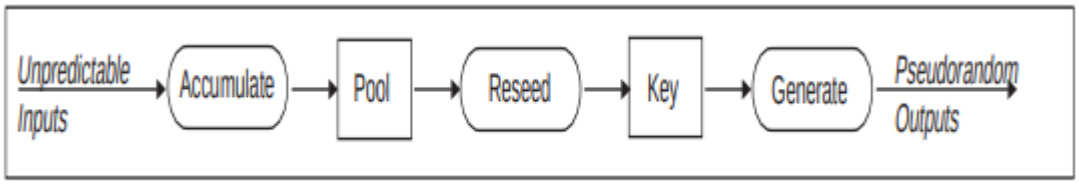


Рисунок 2.1 - загальна блок-схема Yarrow.

Є чотири основні складові:

1. Акумулятор ентропії, який збирає вибірки з джерел ентропії та накопичує їх у двох пулах. Це процес, за допомогою якого PRNG (генератор псевдовипадкових чисел) отримує новий, непередбачуваний внутрішній стан. Під час ініціалізації PRNG, а також для перевантаження під час роботи, важливо успішно накопичувати ентропію з вибірок.
2. Механізм перевантаження, який періодично оновлює ключ новою ентропією з пулів. Механізм перевантаження з'єднує акумулятор ентропії з механізмом генерації. Коли керування перевантаженням визначає необхідність перевантаження, компонент перевантаження повинен оновити ключ, який використовується механізмом генерації, інформацією з одного або обох пулів, які підтримує акумулятор

ентропії. Це повинно здійснюватися таким чином, що якщо ключ або пули були невідомі для зловмисника до перевантаження, то після перевантаження ключ також буде невідомим для зловмисника. Також повинна бути можливість зробити процес перевантаження обчислювально витратним, щоб ускладнити атаки, що ґрунтуються на вгадуванні невідомих вхідних вибірок.

3. Механізм генерації, який створює вихідні дані PRNG з ключа. Механізм генерації забезпечує вихід PRNG. Вихід має мати властивість таку, що якщо зловмисник не знає ключа PRNG, він не може відрізнити вихід PRNG від по-справжньому випадкової послідовності бітів. Механізм генерації повинен мати наступні властивості:
    - Стійкість до криптоаналітичних атак,
    - Ефективність,
    - Стійкість до відстеження після компрометації ключа,
    - Здатність генерувати дуже довгу послідовність виходів безпечно без перевантаження.
4. Керування перевантаженням, яке визначає, коли слід виконувати перевантаження. Механізм керування перевантаженням повинен враховувати різні обставини. Часте перевантаження бажане, але це збільшує ймовірність ітеративної атаки на вгадування. Рідке перевантаження надає зловмиснику, який компрометував ключ, більше інформації. Дизайн механізму керування перевантаженням є компромісом між цими цілями [15].

Генерація випадкових одноразових паролів використовується для створення унікальних та тимчасових паролів, які важко або неможливо передбачити, забезпечуючи тим самим високий рівень безпеки при доступі до системи чи послуги.

Також треба розуміти що використання асиметричного шифрування для одноразових паролів може бути важливим елементом забезпечення безпеки при аутентифікації користувачів. Асиметричне шифрування включає в себе пари ключів - публічний та приватний, які використовуються для зашифрування та розшифрування даних.

Ідея полягає в тому, що сервер може використовувати публічний ключ для шифрування одноразового пароля перед його надсиланням на пристрій або клієнтську програму. Пристрій клієнта володіє відповідним приватним ключем для розшифрування цього одноразового пароля. Цей процес може забезпечити безпеку при передачі чутливої інформації, оскільки навіть якщо зловмисник перехопить зашифрований одноразовий пароль, він не зможе його розшифрувати без приватного ключа, який знаходиться лише у власника пристрою.

Додатковим перевагою використання асиметричного шифрування є можливість відкриття та використання цих ключів лише тоді, коли це необхідно. Це забезпечує додатковий рівень безпеки, оскільки приватний ключ залишається захищеним та недоступним для повноцінного використання.

Загалом, використання асиметричного шифрування для одноразових паролів допомагає забезпечити безпеку та конфіденційність інформації, що передається, знижуючи ризик перехоплення та несанкціонованого доступу до конфіденційних даних.

2.2 Розгляд генерування OTP

Тож розглянемо алгоритм генерування одноразового пароля HOTP. Це алгоритм на базі алгоритму HMAC (Hash-Based Message Authentication Code). Одноразовий пароль (HOTP), безумовно, є одним із найпростіших і найпопулярніших популярні форми двофакторної автентифікації для

захисту доступу до мережі. Алгоритм HOTP базується на значенні монотонно зростаючого лічильника та статичному симетричному ключі, який відомий лише клієнту та серверу. Для створення значення HOTP використовується алгоритм HMAC-SHA-1. Кожен клієнт має унікальний спільний секрет, який, як правило, має довжину 128 або 160 біт. Спільний секрет поєднується з зростаючим лічильником, який також спільний між клієнтом і сервером, для генерації поточного парольного коду [17].

Алгоритм HOTP базується на значенні зростаючого лічильника та статичному симетричному ключі, який відомий тільки токену і службі перевірки. Для створення значення HOTP ми будемо використовувати алгоритм HMAC-SHA-1, як визначено в RFC 2104 [16].

Оскільки вихідне значення обчислення HMAC-SHA-1 складає 160 біт, нам необхідно обрізати це значення до чогось, що може бути легко введено користувачем.

HOTP(K,C) = Truncate(HMAC-SHA-1(K,C))

де C - Лічильник у вигляді 8-байтового значення, рухомий фактор.

Цей лічильник обов'язково повинен бути синхронізований між генератором HOTP (клієнтом) та HOTP (сервером), який перевіряє.

K - Спільний секрет між клієнтом та сервером; кожен генератор HOTP має різний та унікальний секрет K. Значення ключа (K), лічильника (C) та даних хешуються, починаючи з найстарших байтів. Значення HOTP, згенеровані генератором HOTP, обробляються як big endian (порядок байтів, де старший байт йде першим).

Truncate представляє функцію, яка перетворює значення HMAC-SHA-1 в значення HOTP. Дана функція обрізає значення HMAC-SHA-1, яке має 160 біт, до 6-ти цифр, використовуючи функцію обрізки згідно наведеного алгоритму. Ця операція використовується для отримання HOTP значення.

Алгоритм обрізки:

1. Отримання 4-байтового значення з результату HMAC-SHA-1. Для цього беруться 4 останні байти.
2. Отримання значення в 31-ому біті. Виконується побітове І-логічне множення останнього байта HMAC-SHA-1 із 0x7f.
3. Отримання значення в 32-ому біті. Виконується побітове І-логічне множення передостаннього байта HMAC-SHA-1 із 0xff.
4. Об'єднання отриманих значень для формування 4-байтового значення.
5. Отримання шестицифрового значення від цього 4-байтового значення. Для цього виконується побітове І-логічне множення отриманого 4-байтового значення із 0x7fffffff.
6. Повернення цього шестицифрового значення як результату функції обрізки [18].

Зразок обчислення HOTP для 6 цифр.

Наведений нижче приклад коду описує отримання динамічного бінарного коду за умови, що hmac_result - це масив байтів із результатом HMAC-SHA-1.

```
int offset   =  hmac_result[19] & 0xf ;
        int bin_code = (hmac_result[offset]  & 0x7f) << 24
           | (hmac_result[offset+1] & 0xff) << 16
           | (hmac_result[offset+2] & 0xff) <<  8
           | (hmac_result[offset+3] & 0xff) ;
```

Рисунок 2.2 – приклад псевдокоду

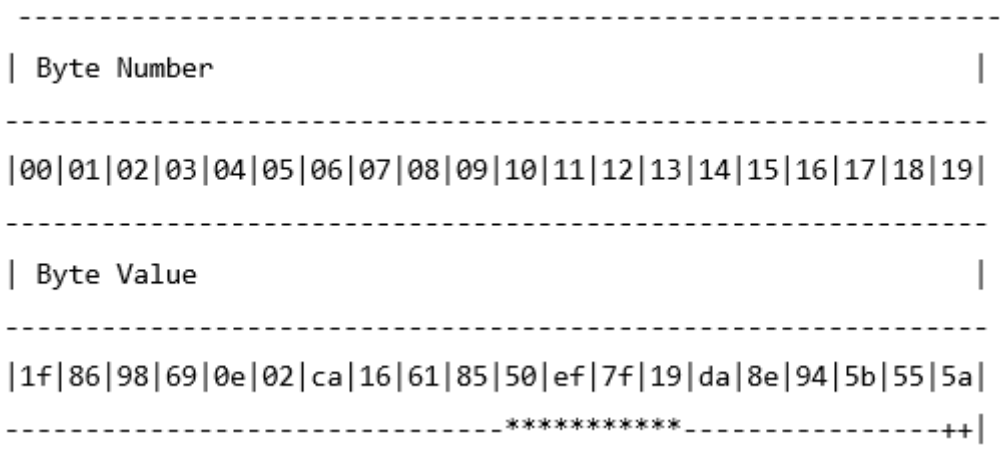


Рисунок 2.3 – приклад SHA-1 HMAC Bytes

- Останній байт (байт 19) має шістнадцяткове значення 0x5a.
- Значення менших 4 біт - 0xa (значення зміщення).
- Значення зміщення - байт 10 (0xa).
- Значення 4 байтів, починаючи з байта 10, становить 0x50ef7f19, яке є динамічним бінарним кодом DBC1.
- Старший біт DBC1 - 0x50, отже, DBC2 = DBC1 = 0x50ef7f19.
- HOTP = DBC2 за модулем 10^6 = 872921.

Ми розглядаємо динамічний бінарний код як 31-бітне, беззнакове, ціле число; перший байт маскується з 0x7f.

Далі беремо це число по модулю 1 000 000 (10^6), щоб згенерувати 6-значне значення HOTP 872921 в десятковому вигляді [18].

Також якщо проаналізувати безпеку цього алгоритму, можна зрозуміти що вона залежить від основного блоку побудови HOTP, який є конструкцією, заснованою на HMAC [RFC2104], використовуючи SHA-1 як хеш-функцію.

Також можна виділити вимоги щодо безпеки даного алгоритму:
1. Використання криптографічних ключів: Рекомендація використовувати криптографічно стійкі ключі, які вибираються випадковим чином або

генеруються за допомогою криптографічно стійкого псевдовипадкового генератора.

2. Довжина ключа: Рекомендація обирати довжину ключа, яка відповідає виводу хеш-функції HMAC для полегшення взаємодії між системами.

3. Використання безпечних каналів: Всі комунікації, пов'язані з HOTP, рекомендується проводити через безпечні канали, такі як SSL/TLS або IPsec, для захисту від перехоплення або зміни даних.

4. Зберігання ключів: Рекомендації щодо безпечного зберігання ключів, зокрема, зашифрування їх за допомогою захищеної апаратної шифрування та обмеження доступу до них лише необхідним процесам.

5. Перевірка випередження (look-ahead parameter): Рекомендація використовувати параметр випередження (look-ahead parameter), який дозволяє серверу перевіряти наступні значення HOTP для синхронізації, але при цьому обмежує можливості атак залежно від обраного значення параметра.

TOTP (Time-Based One-Time Password) токени є формою аутентифікації з двофакторною перевіркою, що базується на часі. Ці токени генерують одноразові паролі, які використовуються для доступу до різних систем та послуг. Основна їхність полягає в генерації одноразового коду, який діє лише обмежений час.

Процес використання TOTP-токенів зазвичай виглядає так: користувач збирається увійти до певної системи чи сервісу, де вимагається двофакторна перевірка. Перша частина аутентифікації - це щось, що користувач знає (наприклад, пароль). Друга частина - це TOTP-токен, який генерує одноразовий пароль. Цей одноразовий пароль змінюється кожні 30 секунд, заснований на поточному часі і спільному секретному ключі між сервером та токеном.

Такі токени зазвичай втілюються у фізичній формі, у вигляді пристроїв з дисплеєм, які генерують та відображають одноразові паролі.

Проте, останнім часом частіше використовуються мобільні додатки, що перетворюють смартфони на генератори TOTP-кодів.

У сучасному світі TOTP-токени широко використовуються для захисту особистих акаунтів у різних онлайн-сервісах, фінансових платформах, корпоративних системах та багатьох інших сферах. Їх висока популярність пояснюється надійністю та зручністю використання, оскільки вони забезпечують додатковий шар безпеки під час аутентифікації користувача.

Основною ідеєю TOTP є визначення його як TOTP = HOTP(K, T), де T є цілим числом, що представляє кількість кроків часу між початковим часом лічильника T0 та поточним часом Unix.

Конкретніше, T = (Поточний час Unix - T0) / X, де для обчислень використовується типова функція округлення до меншого цілого. Наприклад, з T0 = 0 та кроком часу X = 30, T = 1, якщо поточний час Unix складає 59 секунд, і T = 2, якщо поточний час Unix становить 60 секунд. Реалізація цього алгоритму обов'язково підтримує значення часу T, яке перевищує 32-бітне ціле число, коли воно виходить за межі року 2038. Значення системних параметрів X та T0 передбачено під час процесу підготовки і сповіщається між довірником та перевірником як частина етапу підготовки [19].

Крім того, токени OTP мають великий потенціал, оскільки можуть бути модифіковані різними способами.

Наприклад:

- Зміна часових кроків: Зміна регулярності та швидкості генерації токенів може збільшити або зменшити ступінь безпеки в залежності від потреб користувача.
- Додаткові аутентифікаційні параметри: Додавання до токенів додаткових параметрів, таких як біометричні дані (відбиток пальця,

розпізнавання обличчя тощо), може забезпечити двофакторну або багатофакторну аутентифікацію.

- Технологічні інтеграції: Інтеграція токенів OTP у різноманітні пристрої, такі як мобільні телефони або фізичні токенізовані пристрої, робить їх більш зручними для користувачів, спрощуючи процес авторизації.
- Керування часом дії: Застосування токенів з обмеженим терміном дії для конкретних операцій або певних рівнів доступу може допомогти вирішити проблеми, пов'язані з часом.
- Захист від повторного використання: Врахування часу створення або часу використання токенів для уникнення їх повторного використання також є важливим аспектом для забезпечення безпеки.

2.3. Безпека OTP

Хотів би розглянути питання безпеки алгоритму HOTP, так як ці алгоритми пов’язані, було детально описано за посиланням [18].

Висновок аналізу безпеки полягає в тому, що для всіх практичних цілей вихідні значення динамічного скорочення (DT) на різних вхідних значеннях лічильника рівномірно та незалежно розподілені 31-бітові рядки.

Далі у аналізі безпеки детально описано вплив перетворення з рядка в ціле число та остаточне зменшення за модулем 10^{Digit}, де Digit - це кількість цифр у значенні HOTP.

Аналіз показує, що ці остаточні кроки вводять знехтувану відхиленість, яка не впливає на безпеку алгоритму HOTP, в тому розумінні, що найкращий можливий атака на функцію HOTP - це атака брут-форс [18].

Брут-форс атака - це метод зламу, при якому атакуючий намагається вирішити проблему, перебираючи всі можливі варіанти шляхом систематичної спроби та помилки. Цей підхід не використовує ніякої

специфічної інформації про систему, а замість цього намагається відкрити захищену інформацію шляхом перебору всіх можливих комбінацій. Наприклад, у випадку брут-форс атаки на пароль, атакуючий спробує усі можливі комбінації символів (цифр, літер, спеціальних символів тощо), починаючи з коротких та закінчуючи довгими послідовностями символів, доти, доки не знайде правильний пароль. Цей метод атаки є простим та універсальним, але він може вимагати значних обчислювальних ресурсів та часу, особливо якщо пароль або ключ є достатньо довгими або складними. У сфері кібербезпеки брут-форс атаки можуть бути спрямовані не лише на перебір паролів, а й на перебір ключів шифрування, перевірку автентичності через повторні спроби входу, спроби розкриття даних шляхом випробовування всіх можливих комбінацій тощо [21].

Припускаючи, що злочинець здатен спостерігати численні обміни протоколом та збирати послідовності успішних значень аутентифікації, цей злочинець, намагаючись побудувати функцію F для генерації значень HOTP на основі своїх спостережень, не матиме значного переваги перед випадковим вгадуванням.

Логічним висновком є те, що найкращою стратегією знову стане виконання атаки брут-форс для переліку та спроб всіх можливих значень.

Розглядаючи аналіз безпеки в додатку цього документа [18], без втрати загальності, ми можемо досить точно наблизити безпеку алгоритму TOTP за наступною формулою:

$$Sec = sv/10^{Digit}$$

де, Sec - ймовірність успіху атаки,

s - розмір вікна синхронізації в перегляді вперед,

v - кількість спроб верифікації,

Digit - кількість цифр у значеннях TOTP.

Очевидно, ми можемо експериментувати з s, T (параметром обмеження, який обмежує кількість спроб атаки) та Digit, досягнувши певного рівня

безпеки, зберігаючи при цьому придатність системи [18].

Висновки до другого розділу

У розділі було розглянуто варіативність методів створення одноразових паролів, проаналізували їхні переваги та недоліки. Також, виділяється важливість використання не шаблонних токенів, які не повторюються та важко передбачувані, що підвищує рівень захисту даних. Підкреслюється, що різні методи генерації мають свої особливості, і вибір конкретного методу залежить від контексту використання та потреб безпеки системи.

Звичайно, OTP (одноразові паролі) стали надзвичайно важливими в наш час у зв'язку зі зростанням кількості онлайн сервісів та інформації, яка потребує захисту. Вони є простим та швидким способом авторизації, спрощуючи доступ до особистих облікових записів, банківських систем, а також для забезпечення безпеки та захисту даних у цифровому світі. Токени OTP вирішують проблему забезпечення безпеки у віртуальному просторі, оскільки вони надають додатковий шар захисту, який може бути змінений та налаштований згідно з вимогами конкретного користувача чи системи.

Моєю основною метою є створення механізму генерації посилань з обмеженим доступом за часом, не використовуючи для цього бази даних. Це поліпшення буде спрямоване на підвищення продуктивності, зменшення завантаження на базу даних та уникнення потреби в постійних операціях запису та перевірки інформації про надання конкретного доступу. Вихідні імплементації зазвичай вимагають зберігання даних в базі даних і подальшу їх верифікацію, що призводить до зв'язку з базою даних. Моя робота спрямована на усунення цієї залежності, створюючи спосіб генерації одноразових паролів, який був би більш автономним, швидким та масштабованим.

Остаточно, висновок полягає в тому, що подальше вдосконалення методів генерації одноразових паролів важливе для підвищення рівня захисту даних. Це може включати розробку нових алгоритмів, які б забезпечували випадковість та короткочасне існування даних, не вимагаючи записування їх в базу даних.

3. РЕАЛІЗАЦІЯ АЛГОРИТМУ ОБМЕЖЕННЯ ДОСТУПУ ДО РЕСУРСУ ПО ЧАСУ

3.1. Методи імплементації

Зважаючи на постійний розвиток технологій та зміну потреб користувачів, покращення методів доступу обмежених по часу є важливим кроком для забезпечення ефективності та безпеки систем. Вдосконалення методів доступу обмежених по часу допоможе підвищити рівень безпеки даних. Зменшення тривалості часу доступу може обмежити можливість несанкціонованого доступу до важливої інформації. Коротший, але ефективний час доступу може покращити швидкість роботи користувачів, оскільки вони матимуть доступ до необхідної інформації лише протягом визначеного періоду. Вдосконалені методи доступу можуть відповідати сучасним стандартам безпеки даних та робити систему більш сумісною з регуляторними вимогами. Якщо система використовує ефективні методи доступу обмеженого часу, це може спростити процес адміністрування системи, зменшивши час, витрачений на управління доступом користувачів.

Залежно від контексту і мети обмеження доступу до ресурсів по часу, існує кілька можливих способів реалізації цих методів. Ось деякі з них:

1. Часові обмеження сеансів:

- Налаштування серверних параметрів: В багатьох веб-серверах або програмних засобах можна встановити параметри сеансів, які автоматично завершують сеанс після заданого періоду бездіяльності.
- Використання middleware: Розробники можуть використовувати middleware в програмному забезпеченні для відстеження активності користувача та автоматичного виходу з системи після певного проміжку часу бездіяльності.

2. Розклад доступу:

- Керування правами доступу: Використання систем керування правами доступу для налаштування часових обмежень для різних груп користувачів або окремих користувачів.
- Задання графіку роботи: Реалізація логіки програми або системи, яка обмежує доступ до функцій або ресурсів протягом певного періоду.

3. Таймери сесій:

- Використання програмних бібліотек: Застосування спеціальних бібліотек або функцій, які встановлюють таймери для сесій користувачів та автоматично завершують їх після визначеного часу.
- Імплементація логіки в програмі: Програмісти можуть вбудувати логіку з використанням мов програмування, щоб встановити таймери сесій і автоматично виходити з аккаунту після певного часу.

4. Обмеження діапазону часу:

- Конфігурація серверів або додатків: Встановлення налаштувань на рівні системи для обмеження доступу протягом певних годин або діапазону днів.
- Використання планувальників завдань: Визначення розкладу для запуску або припинення роботи деяких сервісів чи додатків у визначений час.

5. Часові маркери аутентифікації:

- Застосування часових обмежень до токенів аутентифікації: При роботі з механізмами аутентифікації, можливо використовувати токени з обмеженим терміном дії, які автоматично стають недійсними після визначеного часу.

Це лише загальні приклади. Реалізація цих методів може варіюватися від конкретної системи до системи, в залежності від використовуваних технологій, мов програмування та архітектури системи.

Але якщо ми хочемо обмежити доступ анонімному користувачу ця задача вже є дещо складнішою. Оскільки відсутність ідентифікаційної

інформації про користувача ускладнює точне встановлення параметрів доступу. Таким чином, ці обмеження можуть бути менш точними і ефективними порівняно з обмеженнями, що базуються на ідентифікації конкретного користувача. Ось деякі можливі способи:

1. Використання сесійного обмеження: Навіть для анонімних користувачів може бути встановлено часові рамки для сесій. Це може бути реалізовано шляхом встановлення часу життя сесії, після якого доступ буде припинено, навіть без ідентифікації користувача.
2. IP-адресне обмеження: У деяких випадках адміністратори ресурсів можуть обмежувати доступ до вмісту з певних IP-адрес протягом певного періоду. Це може бути використано для обмеження доступу анонімних користувачів.
3. Обмеження на основі мови, локації або інших параметрів: В залежності від характеру ресурсу, його адміністратор може обмежувати доступ до контенту на основі різних параметрів, таких як мова браузера, локація користувача тощо. Це може бути корисно для визначення доступу анонімних користувачів у певний час або обставини.
4. Використання cookie-файлів: Хоча це не забезпечить абсолютне обмеження, але може бути використано для встановлення тимчасових обмежень на певний період часу через cookie-файли в браузері анонімного користувача.
5. Застосування загальних обмежень ресурсу: В деяких випадках може бути встановлено загальні обмеження доступу до певних ресурсів для всіх користувачів, навіть анонімних, протягом певного часу.

Це все не дуже прості в реалізації методи, та мають багато нюансів. Один з найпростіших та дієвих алгоритмів на мій погляд. Це використання бази даних для контролю доступу по часу для анонімних користувачів.

Припустимо, що кур'єр з служби доставки прибуває для забору відправлення. Однак для спокою користувача його необхідно чітко ідентифікувати та переконатися, що він віддає свою посилку саме уповноваженій особі. Для цього кур'єр передбачає процес створення спеціального ідентифікаційного знаку, який ми можемо назвати "id-card", у власному додатку.

З метою поширення цієї "id-card", ми генеруємо посилання або QR-код, що містить відповідне посилання на "id-card". Оскільки ця інформація є особистою, важливо, щоб вона не була постійно доступною користувачеві. Для цього потрібно якось ідентифікувати конкретне посилання, або ввести поняття запиту.

Стандартне втілення цього процесу вимагає створення таблиці в базі даних, в якій зберігається ідентифікатор даного запиту. Цей ідентифікатор служить параметром, що приєднується до посилання, та містить інформацію про час створення та той момент, коли цей запит втрачає валідність.

При запиті "id-card" необхідно зробити запит до бази даних та перевірити це ще не була втрачена валідність цього посилання. Це все дає навантаження на базу даних збільшує трафік.

Тому моєю метою є створення алгоритму який би виконував цю функцію але не потребував бази даних. Вирішив використати за основу алгоритм TOTP токена, та модифікувати його для того щоб запропонувати.

3.2. Обґрунтування інструментів та методів для модифікаціїї TOTP токена

Стандартний алгоритм TOTP токена базується, як TOTP = HOTP(K, T), як це вже було описано. Моя реалізація в дечому є гібридом між JWT та TOTP.

JWT (JSON Web Token) - це компактний та самостійний спосіб представлення інформації у вигляді токена, що забезпечує захищену обміненими між сторонами даними в JSON-форматі. JWT складається з трьох основних частин: заголовок (Header), твердження (Payload) та підпис (Signature).

Основні характеристики JWT:

1. Заголовок (Header): Містить метадані токена та тип підпису, зазвичай це містить тип (typ) та алгоритм підпису (alg). Наприклад: {"alg": "HS256", "typ": "JWT"}.
2. Твердження (Payload): Це корисна інформація або твердження, які зазвичай містять дані про користувача або будь-яку іншу інформацію. Ці дані можуть бути у форматі JSON. Наприклад: {"user_id": 123, "role": "admin"}.
3. Підпис (Signature): Заголовок та твердження об'єднуються та підписуються для перевірки цілісності та автентифікації токена. Це забезпечує впевненість, що токен не був змінений під час передачі між сторонами.

JWT може бути підписаним або зашифрованим. У випадку підпису токен може бути перевірений стороною, яка отримала його, за допомогою секретного ключа, який використовувався для підпису. Зашифрований JWT, навпаки, не може бути прочитаний стороною, яка отримала його, без відповідного ключа для розшифрування.

JWT використовується для забезпечення автентифікації та авторизації в додатках та API, а також для обміну даними між різними сервісами. Це дозволяє зберігати інформацію у форматі токенів, які можуть бути перевірені та використані різними компонентами системи [20].

Суть моєї модифікації полягає в тому щоб до коду TOTP, додатково додати значення часу коли він був створений, та час на скільки. Ці два

параметри будуть включені в генерування секрету, по якому генерується токен. Та змінити правила генерування та валідації.

Для модифікації алгоритма я обрав використання мови Java та пакету javax.crypto, це доцільно з кількох причин:

Підтримка вбудованих криптографічних функцій: Java має вбудовану підтримку криптографічних алгоритмів через пакет javax.crypto. Це дозволяє зручно використовувати різноманітні алгоритми шифрування, хешування та інших криптографічних операцій для реалізації безпечних методів створення та перевірки TOTP токенів.

Безпека і стандартизація: Java має вбудовані інструменти для роботи з криптографією, які відповідають стандартам безпеки. Це означає, що можна використовувати визнані криптографічні алгоритми та методи забезпечення безпеки для реалізації TOTP алгоритмів.

Кросплатформеність: Код, написаний на Java, є кросплатформеним і може працювати на різних операційних системах, які підтримують віртуальну машину Java (JVM). Це забезпечує переносність реалізації TOTP токенів між різними платформами.

Широкі можливості для розробки: В Java є багато інструментів і бібліотек для розробки програм, включаючи криптографічні операції. Це дає можливість розширити функціоналість та забезпечити безпеку інших складових системи, таких як зберігання ключів, обробка інформації користувачів тощо.

Підтримка алгоритмів TOTP: Java має можливість використовувати криптографічні функції для створення та перевірки TOTP токенів. Ви можете використовувати ці функції для впровадження механізму двофакторної аутентифікації на основі TOTP забезпечуючи високий рівень безпеки.

Загалом, Java та javax.crypto забезпечують потужний набір інструментів для роботи з криптографією, що може бути корисним для модифікації алгоритму TOTP токенів та забезпечення їх безпеки.

Пакет javax.crypto - це частина Java Cryptography Architecture (JCA), яка надає API для роботи з криптографією в програмах на мові Java. Цей пакет містить класи та інтерфейси для виконання криптографічних операцій, таких як шифрування, розшифрування, гешування та підписування даних [22].

Основні можливості пакету javax.crypto включають:

- Шифрування та розшифрування даних: Надає функції для зашифрування та розшифрування даних за допомогою різних криптографічних алгоритмів, таких як AES, DES, RSA та інших.
- Гешування даних: Дозволяє обчислювати хеш-коди (геші) для даних за допомогою різних алгоритмів гешування, таких як SHA-1, SHA-256, MD5 та інші.
- Підписування та перевірка підписів: Надає функції для створення та перевірки цифрових підписів для даних.
- Генерація випадкових чисел: Має інструменти для генерації випадкових чисел, які можуть бути використані для генерації ключів інших криптографічних операцій.
- Підтримка безпеки: Надає інструменти для роботи з криптографічними ключами, управління сертифікатами та іншими елементами безпеки.

Цей пакет дозволяє розробникам легко використовувати криптографічні функції в своїх програмах Java, забезпечуючи безпеку обробки даних та забезпечуючи конфіденційність, цілісність та аутентифікацію.

3.3. Опис модифікованого алгоритму

3.3.1 Генерація

У процесі вдосконалення продуктивності та оптимізації системи було виявлено, що надлишкові запити до бази даних спричиняють певні завади та сповільнення в роботі програми. Метою було знайти ефективний спосіб зменшення кількості запитів до бази даних, не ушкоджуючи безпеку чи надійність системи. Вивчення одноразових паролів та їх можливе використання привело до виявлення TOTP токена як потенційного рішення для даної проблеми.

Моє рішення стосується удосконалення стандартного алгоритму створення TOTP токенів. Однак, в процесі розробки виявилась одна особливість у базовому алгоритмі, яка не відповідає вимогам моєї реалізації. Саме, базова реалізація алгоритму TOTP використовує алгоритм HOTP (HMAC-based One-Time Password) для генерації токенів на основі часу, замість використання лічильника. Це означає, що крок часу для генерації токенів визначається як:

$$step = time / lifespan,$$

де step - це інтервал,

time - поточний час,

lifespan - тривалість життя токену у секундах.

Тож, допустимо, що lifespan становить 30 секунд, а час генерації токену - 12:15:13. В такому випадку, токен буде дійсним ще протягом 17 секунд до 12:15:30, після чого буде згенеровано наступний токен о 12:15:30, і його життєвий цикл буде до 12:16:00. Оскільки запити на генерацію токенів можуть надходити в будь-який момент, ми повинні гарантувати, що токен залишатиметься дійсним протягом усього часу, визначеного параметром lifespan. Такий підхід дозволяє забезпечити стабільність та

відповідність умовам життєвого циклу токенів, не залежно від моменту їхньої генерації.

Цю проблему я вирішив за допомогою певного підходу: я застосовую округлення часу до найближчої години для використання його як початкового часу. Тобто, коли запит на генерацію токену надходить о 19:15:60, мій алгоритм сформує токен на основі часу, округленого до 19:00:00. Тобто якщо взяти базову функцію TOTP = HOTP(K, T), то T представимо наступним чином:

$$T = Truncate(S),$$

де S це поточний час Unix,

Truncate це функція яка округлює час до найближчої години.

Отримаємо наступне TOTP = HOTP(K, Truncate(S)). Це дозволяє уникнути проблеми, пов'язаної з різними моментами генерації токенів у межах конкретної години.

Розглянувши вищезгадану проблему, виявилася інша складність, яка полягає в тому, що неможливо просто порівняти токени на предмет їхнього співпадіння, як це описано у документації RFC6238 [19]. Ця особливість призводить до потреби зберігати час генерації кожного токену.

Для подальшої перевірки токенів мною було запропоновано змінити правила порівняння токенів. Тепер необхідно враховувати час генерації токену, отримати актуальний поточний час при перевірці та підрахувати їхню різницю.

$$D = Unix - S$$

Ця різниця позначається літерою D і розраховується як різниця між часом Unix на момент перевірки та стартовим часом S, коли токен був згенерований.

Отже, для генерації нового токену для подальшої перевірки, я пропоную застосовувати функцію TOTP, яка використовує параметри, які включають початковий час, округлений до найближчої години, та різницю

між початковим часом та поточним часом. Функція матиме наступний вигляд:

$$TOTP = HOTP(K, Truncate(S) + D),$$

де TOTP - новий токен для подальшої перевірки,

HOTP - функція генерації одноразового пароля на основі HMAC,

K - ключ, Truncate(S) - відсічений початковий час,

D - різниця між часами, яка забезпечує унікальність та актуальність токену.

Зважаючи на особливості процесу генерації токенів для першого використання користувачем, необхідно почати з встановлення початкової різниці на нульовому рівні. Це важливо, оскільки відсутність попередніх токенів для відношення до поточного моменту часу вимагає ініціалізації значення різниці як нуль.

Такий підхід до початкового значення різниці є ключовим етапом у забезпеченні коректного розпочатку генерації токенів для нових користувачів. Це дозволяє системі вірно орієнтуватися в часових рамках та починати процес генерації токенів для користувача з правильним визначенням початкових значень для подальшого використання. Така початкова ініціалізація різниці забезпечує нульову точку для належного початку процесу створення та перевірки токенів у системі.

Щоб вирішити питання з відслідковуванням часу генерації токенів, я використовую простий, але ефективний метод - конкатенацію часу генерації токену з самим токеном за допомогою сепаратора '-'. Це дає змогу додати важливість та зрозумілість до самого токену, включаючи в нього інформацію про момент його створення.

Завдяки такому підходу до створення токенів з інтегрованим часовим відбитком, система отримує можливість зручно відстежувати, коли саме був згенерований токен. Це дозволяє користувачеві та системі точно визначати момент часу, з якого починається чинність токену і який період

він покриває. Такий підхід додає прозорості та дозволяє відслідковувати актуальність токену з позначенням часу його створення, що зробить процес перевірки токенів більш простим та ефективним.

Зокрема, для створення системи, що забезпечує динамічну зміну часу життя токенів, я впровадив ідентифікатор часу життя токену в секундах. Цей ідентифікатор дозволяє динамічно змінювати період чинності токену, надаючи більшу гнучкість в управлінні та налаштуванні терміну його дії.

Додавання часу життя токену в секундах стало значущим кроком у покращенні системи безпеки, оскільки це дозволяє змінювати термін дії токенів залежно від конкретних потреб користувачів чи умов використання. Такий підхід дозволяє гнучко контролювати термін чинності токену, забезпечуючи максимальну безпеку використання системи.

З урахуванням ідентифікатора часу життя токену, я впровадив можливість динамічно налаштовувати та пристосовувати період активності токенів, що стало ключовим аспектом в забезпеченні безпеки та ефективності системи.

В результаті такого підходу виникає проблема з безпекою, оскільки інформація про початковий час та тривалість життя токену стають доступними та вразливими до маніпулювання з боку користувачів. Це відкриває можливість для користувачів змінювати ці параметри, що може призвести до продовження терміну чинності токену та порушення безпеки системи.

Це важлива проблема, оскільки доступ до параметрів стартового часу та тривалості токену може стати підставою для впливу на життєвий цикл токенів, зокрема їхню дійсність та строк дії. Такі можливості необдуманої зміни параметрів токенів можуть викликати важливі проблеми з безпекою, оскільки користувачі можуть випадково або навмисно продовжувати строк дії токенів, знижуючи ефективність системи автентифікації та вразливість до атак.

Ця ситуація вимагає більш детального розгляду та удосконалення механізмів захисту та керування цими параметрами, щоб уникнути можливих ризиків, пов'язаних з недостовірними та маніпульованими значеннями часу життя токенів.

Зміна секретного ключа є важливою частиною безпечного управління токенами. Моя нова реалізація полягає в тому, щоб формувати секретний ключ на основі параметрів стартового часу та тривалості життя токену, які комбінуються зі змінним секретним ключем, який задається програмно.

Це дозволяє створювати унікальний ключ для кожного токену, що базується на конкретному часовому вікні та налаштуваннях тривалості життя. Відтак, кожен створений токен матиме свій унікальний секретний ключ, який не буде піддається маніпуляціям чи змінам з боку користувачів, забезпечуючи вищий рівень безпеки та надійності системи. Такий підхід дозволяє контролювати створення та використання токенів, унеможливлюючи неправомірні дії чи спроби маніпулювання параметрами токенів.

Зважаючи на те, що секретний ключ (K) об'єднується зі стартовим часом (S) та тривалістю життя токену (L), це гарантує унікальність кожного секретного ключа, який використовується для генерації кожного конкретного токену.

$$SECRET = S + K + L,$$

Звідси випливає, що генерований одноразовий код (TOTP) обчислюється за формулою

$$TOTP = HOTP(SECRET, Truncate(S) + D),$$

де SECRET - поєднання параметрів секретного ключа,

S - стартового часу,

D - різниці між поточним часом і стартовим часом токену.

Результатом цього удосконалення стало створення ідентифікатора, що має вигляд XXXX-YYYYY-ZZ, де XXXX представляє собою момент

часу у секундах. Цей момент часу відображає момент створення токену, надаючи можливість легко ідентифікувати час початку чинності токену.

У свою чергу, YYYYYY відображає згенерований токен, який представляє собою унікальну послідовність символів. Цей токен використовується для перевірки автентичності та ідентифікації користувача в системі.

Додатково, частину ZZ у ідентифікаторі, що визначається в секундах, відображає час життя токену. Ця характеристика відображає тривалість часу, протягом якого токен залишається дійсним. Такий підхід надає більш гнучкі умови управління терміном чинності токенів, що є важливим аспектом у забезпеченні оптимальної безпеки й ефективності системи.

Опис цього алгоритму можна відобразити наступною блок схемою:

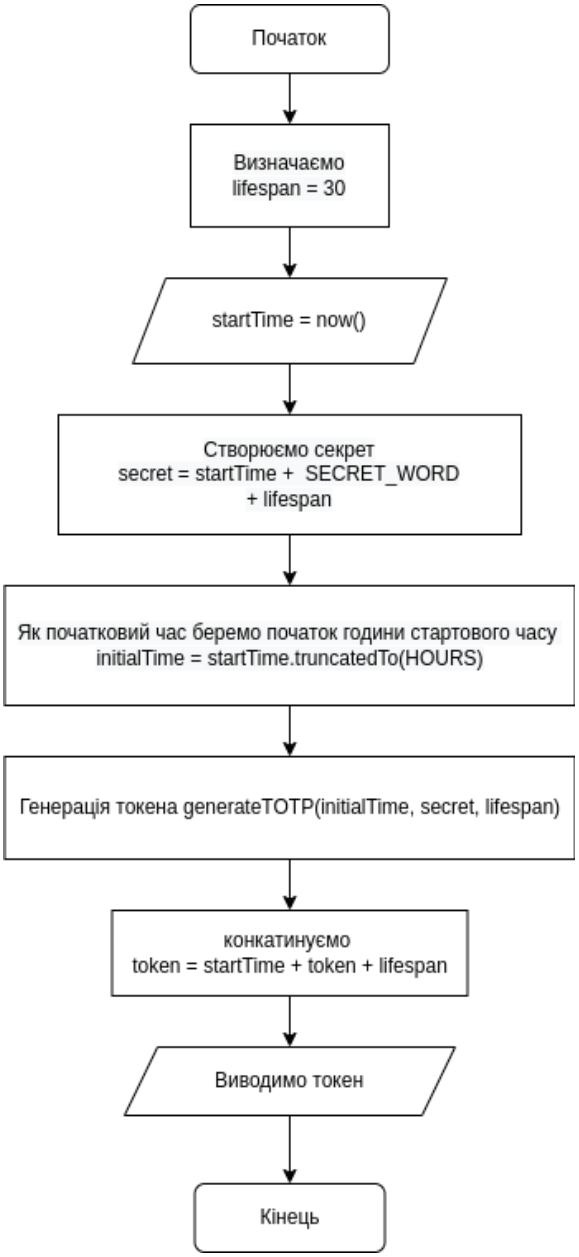


Рисунок 2.3 – блок схема генерації модифікованого токена

3.3.2. Перевірка

Зміни, що відбулися у процесі перевірки валідності токенів, варто детальніше розглянути. У вихідній версії перевірки використовувався лише час приходу токену для підтвердження його валідності. Однак завдяки нашим модифікаціям, сам процес перевірки зазнав суттєвих змін, які варто ретельно розглянути для з'ясування їх впливу та значення у загальному

контексті. Оскільки для створення токена ми використовуємо поточний час округлений до години, значення такого округлення варто відзначити.

Отже, важливо відзначити, що правильний вибір моменту часу для перевірки токена має велике значення. Це пояснюється тим, що процес генерації токена ґрунтується на поточному часі, але в обмеженому годинним форматом. Зважаючи на те, що ідентифікатор передає інформацію про час створення токена та його валідності, було вирішено враховувати різницю часу між моментом створення ідентифікатора та моментом перевірки токена. Ця різниця часу додається до округленого часу до години, який міститься в ідентифікаторі, для забезпечення відповідності поточному моменту перевірки.

За встановленими параметрами та відповідно до заданих вхідних умов ми генеруємо та формуємо новий токен. Особливості цих параметрів відображаються у процесі створення токену, який забезпечує певні властивості та функціональність новоствореного ідентифікатора.

Опис цього алгоритму можна відобразити такою блок схемою:

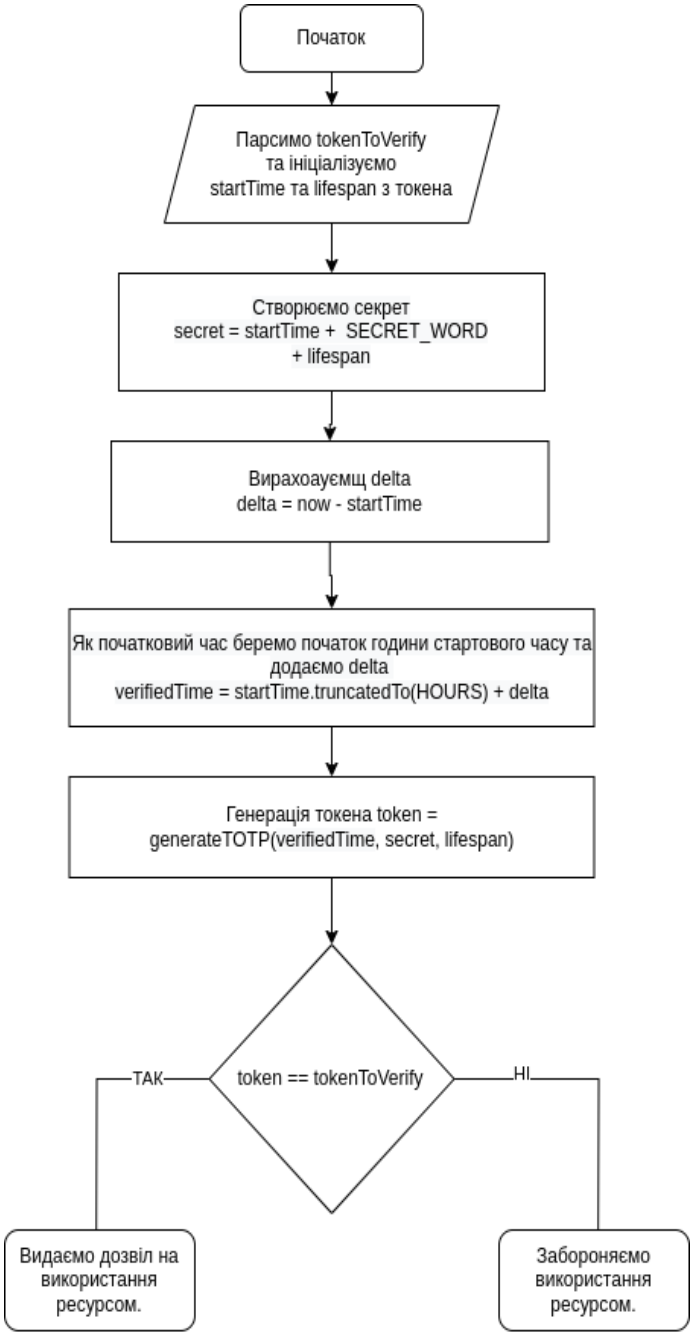


Рисунок 2.4 – блок схема перевірки модифікованого токена

Висновки до розділу третього

У розділі, присвяченому реалізації алгоритму обмеження доступу до ресурсу за допомогою часу, було проведено дослідження та аналіз різних

методів імплементації. Обґрунтовано вибір конкретних інструментів та методів для модифікації TOTP (Time-Based One-Time Password) токена.

У результаті розгляду було ретельно описано модифікований алгоритм, складений з двох основних етапів: генерація та перевірка.

В розділі були розглянуті різні підходи до реалізації алгоритму обмеження доступу за часом. Зроблено акцент на тому, як вибір конкретного методу може вплинути на ефективність та безпеку системи.

Обґрунтування вибору інструментів та методів для модифікації TOTP токена відображено з урахуванням їхньої придатності для задачі обмеження доступу до ресурсу по часу.

У розділі надано докладний опис модифікацій, що застосовуються до стандартного TOTP алгоритму. Проаналізовано дві ключові операції - генерацію та перевірку модифікованого токена з урахуванням їхньої точності, надійності та відповідності поставленим завданням.

Детально описано процес створення модифікованого токена, враховуючи специфіку його взаємодії з часовими параметрами.

Надано докладний опис процедури перевірки модифікованого токена під час доступу до ресурсу. Підкреслено важливість цього етапу у забезпеченні безпеки та ефективності алгоритму обмеження доступу за часом.

4. ТЕСТУВАННЯ МОДИФІКОВАНОГО АЛГОРИТМУ

У даному розділі моєю метою є перевірка та практичне тестування алгоритму для демонстрації його ефективності. Основний акцент буде зроблений на проведенні тестів для виявлення можливості атаки методом брут-форс, оскільки цей алгоритм є найбільш вразливим саме до такого типу атак. На підставі результатів тестування будуть надані рекомендації щодо максимального часу життя токена та його довжини, спрямовані на максимізацію безпеки та ефективності використання алгоритму.

4.1. Тестування ефективності

Для тестування ефективності алгоритму була розроблена програма, яка виконує функцію створення токенів за допомогою бази даних. Основна логіка полягає у створенні запису в базі даних MongoDB з використанням Java та фреймворка Spring Boot.

Ось як вона працює: при запиті на створення токена програма формує новий запис в базі даних. Таблиця містить три поля:

1. id, який генерується функцією випадкового генерування UUID і виступає як унікальний ідентифікатор запису.
2. createdAt, яке вказує час створення запису.
3. expiredAt, що визначає час, до якого токен буде дійсним.

Вибір мови програмування Java та використання фреймворка Spring Boot дозволяють зручно реалізувати цю функціональність. Використання бази даних MongoDB також може бути корисним, оскільки ця NoSQL база даних забезпечує гнучкість та швидкість в роботі з даними.

Програма, яка базується на цьому алгоритмі, може бути використана для генерації та керування токенами, які мають обмежений термін дії. Це може бути корисним для різних варіантів використання, таких як

автентифікація, авторизація або контроль доступу до ресурсів, які потребують обмеження за часом.

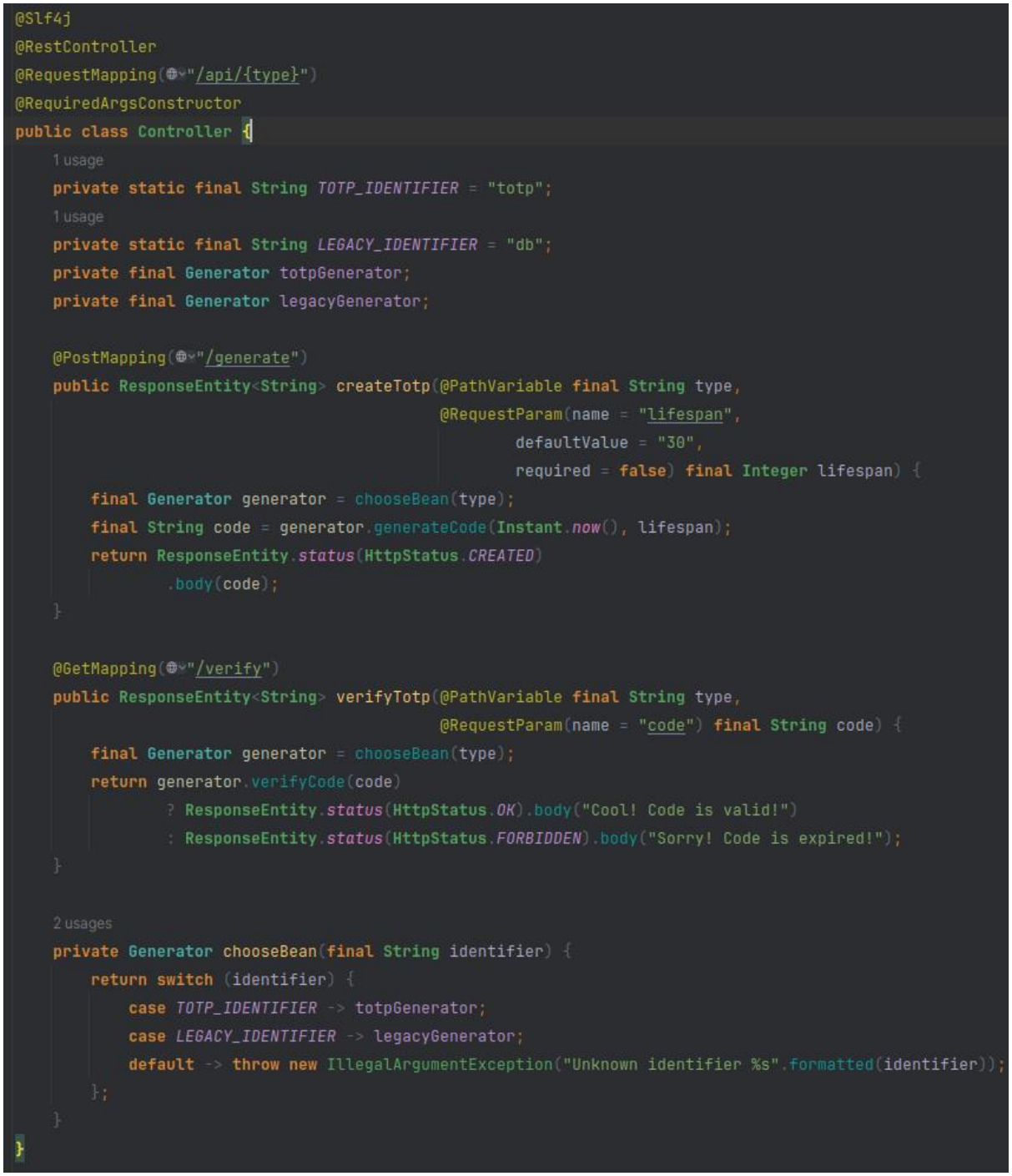
Цей підхід є ефективним для випробування ефективності алгоритму, оскільки база даних забезпечує збереження та керування токенами, а вибір Java та Spring Boot сприяє простоті та швидкості розробки.

Та програма з модифікованим алгоритмом генерації TOTP токенів, який був розглянутий раніше. Ці два модуля імплементують єдиний інтерфейс Generator, наведено лістинг цього інтерфейсу:

Рисунок 4.1 – лістинг інтерфейсу Generator

Метод generateCode генерує ідентифікатори на основі часу, та періоду валідності його а метод verifyCode перевіряє їх.

Для більш точного тестування було створено Controller наведено лістинг цього контролера



```java
@Slf4j
@RestController
@RequestMapping("/api/{type}")
@RequiredArgsConstructor
public class Controller {
    1 usage
    private static final String TOTP_IDENTIFIER = "totp";
    1 usage
    private static final String LEGACY_IDENTIFIER = "db";
    private final Generator totpGenerator;
    private final Generator legacyGenerator;

    @PostMapping("/generate")
    public ResponseEntity<String> createTotp(@PathVariable final String type,
                                             @RequestParam(name = "lifespan",
                                                     defaultValue = "30",
                                                     required = false) final Integer lifespan) {
        final Generator generator = chooseBean(type);
        final String code = generator.generateCode(Instant.now(), lifespan);
        return ResponseEntity.status(HttpStatus.CREATED)
                .body(code);
    }

    @GetMapping("/verify")
    public ResponseEntity<String> verifyTotp(@PathVariable final String type,
                                             @RequestParam(name = "code") final String code) {
        final Generator generator = chooseBean(type);
        return generator.verifyCode(code)
                ? ResponseEntity.status(HttpStatus.OK).body("Cool! Code is valid!")
                : ResponseEntity.status(HttpStatus.FORBIDDEN).body("Sorry! Code is expired!");
    }

    2 usages
    private Generator chooseBean(final String identifier) {
        return switch (identifier) {
            case TOTP_IDENTIFIER -> totpGenerator;
            case LEGACY_IDENTIFIER -> legacyGenerator;
            default -> throw new IllegalArgumentException("Unknown identifier %s".formatted(identifier));
        };
    }
}
```

Рисунок 4.3 – лістинг класу Controller

Цей контролер Controller відповідає за обробку запитів на генерацію та перевірку токенів на основі різних ідентифікаторів, таких як TOTP та ідентифікатори бази даних (Legacy). Основна мета полягає в тому, щоб відповідно до переданого типу, використовувати відповідний генератор для створення токенів або ідентифікаторів та їх верифікації.

Отже, основні аспекти цього контролера варто розглянути наступним чином:

- Мапування шляхів.

  Контролер мапує свої методи на шляхи /api/{type}/generate та /api/{type}/verify, де {type} є шляховим параметром, що вказує тип ідентифікатора (TOTP чи Legacy).

- Обробка запиті.

  Метод createTotp обробляє POST-запит на генерацію токену за допомогою відповідного генератора. Використовується переданий тип, щоб вибрати потрібний генератор.

  Метод verifyTotp обробляє GET-запит на перевірку токену, що передається як параметр code. Аналогічно, використовується переданий тип для вибору генератора для верифікації.

- Вибір генератора.

  Приватний метод chooseBean визначає, який генератор використовувати в залежності від переданого ідентифікатора (TOTP_IDENTIFIER або LEGACY_IDENTIFIER).

  За допомогою оператора switch вибирається відповідний генератор на основі переданого типу ідентифікатора.

- Відповіді.

  В залежності від результату верифікації коду, повертається відповідь зі статусом 200 OK, якщо код вірний, або 403 Forbidden, якщо код застарів або невірний.

- Використання інтерфейсу Generator.

  totpGenerator та legacyGenerator є об'єктами, які реалізують інтерфейс Generator. Це означає, що вони мають методи generateCode та verifyCode для генерації та верифікації токенів відповідно.

Цей контролер дозволяє створювати та перевіряти токени на основі двох різних ідентифікаторів за допомогою відповідних генераторів. Маю наступний тест план:

1. Тест на генерацію токенів:

Мета: Виміряти час, необхідний для генерації токенів обома генераторами.

Опис тесту: Запустити обидва генератори на генерацію токенів певної кількості разів. Виміряйте час, який займає генерація токенів кожним з генераторів.

Метрика успіху: Менше часу, необхідного для генерації токенів, означає вищу швидкість генератора.

2. Тест на верифікацію токенів:

Мета: Виміряти час, необхідний для верифікації токенів обома генераторами.

Опис тесту: Згенерувати токени з обох генераторів. Виміряти час, який займає верифікація токенів кожним з генераторів.

Метрика успіху: Менше часу, необхідного для верифікації токенів, свідчить про вищу швидкість верифікації генератора.

3. Тест на ресурсозатратність:

Мета: оцінити використання системних ресурсів кожним з генераторів під час генерації токенів.

Опис тесту: виміряти використання пам'яті та обсяг обчислювальних ресурсів під час роботи кожного з генераторів.

Метрика успіху: менша використана пам'ять та обчислювальні ресурси свідчать про більш ефективний генератор.

4. Тест на обсяг токенів за одиницю часу в залежності від кількості символів в токені:

Мета: визначити, як впливає більша кількість символів в токені на продуктивність.

Опис тесту: запустити генератор TOTP протягом фіксованого часу (наприклад, 1 хвилина) та міняти кількість символів від 6 до 18. Виміряти кількість токенів, які кожен генератор створив за цей період.

Метрика успіху: більше згенерованих токенів за одиницю часу означає більшу продуктивність генератора.

4.1.1.Тест на генерацію токенів

Для проведення цього тесту необхідно запустити обидва генератори токенів на генерацію певної кількості токенів(100, 1000, 10000, 100000). Під час цього експерименту важливо виміряти час, який займає генерація токенів кожним з генераторів. Проводити кілька запусків для отримання середнього часу генерації .

Будемо вважати що Станд. DB - це звичайний алгорим з використанням бази даних, а мод. TOTP – це модифікований алгоритм генерації токена.

Отримаємо наступні результати

Рисунок 4.4 – час генерації 100 токенів

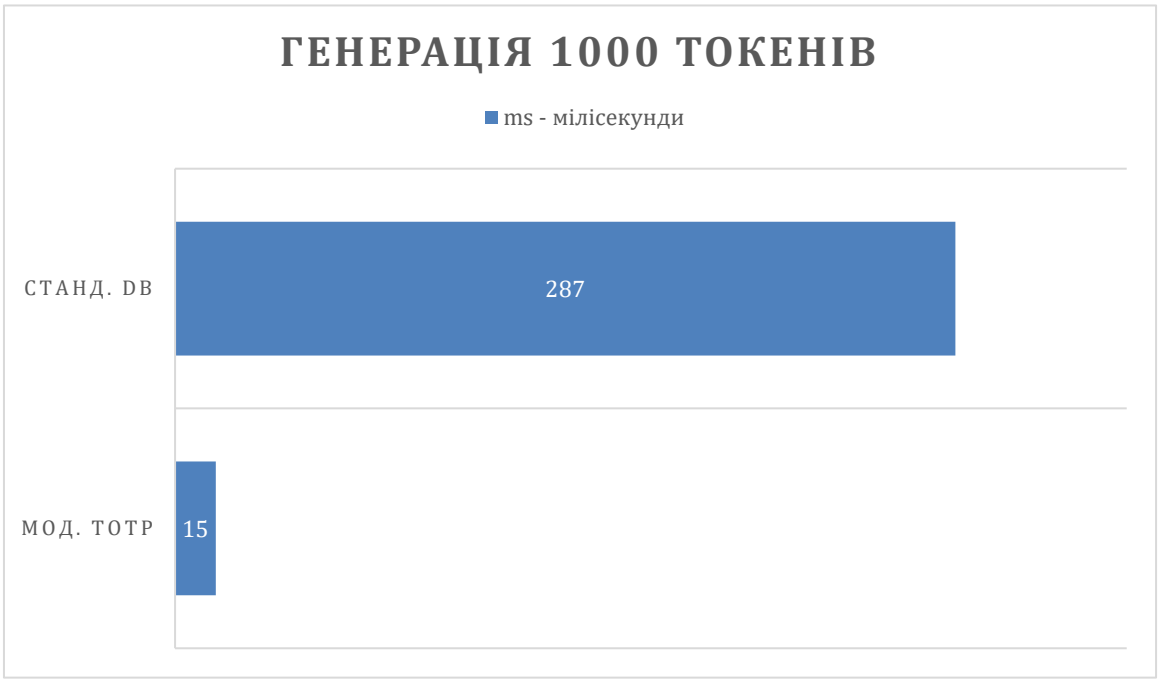


Рисунок 4.5 – час генерації 1000 токенів

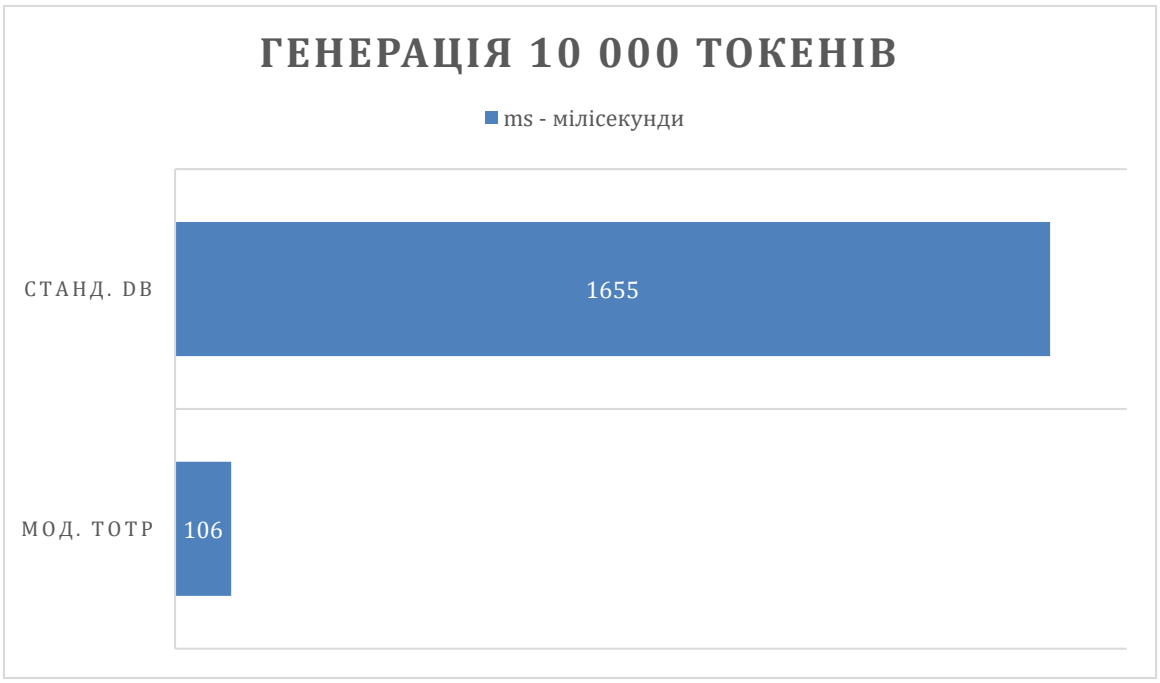


Рисунок 4.6 – час генерації 10 000 токенів

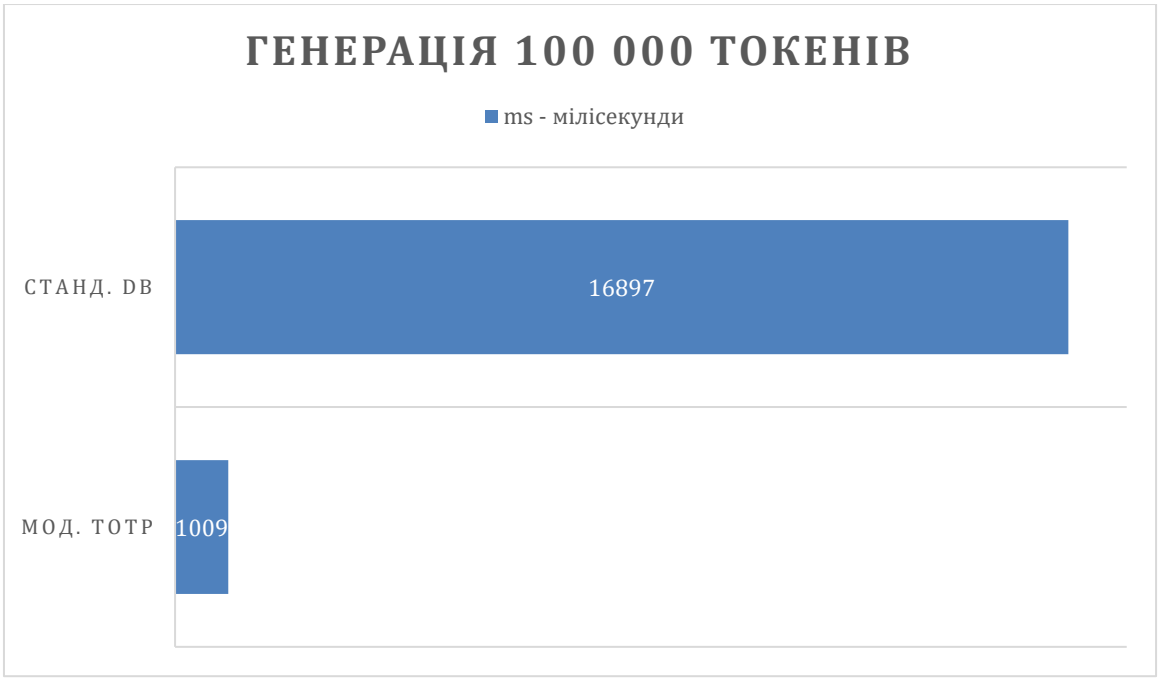


Рисунок 4.7 – час генерації 100000 токенів

Проведений тест на генерацію токенів виявив, що модифікований алгоритм TOTP демонструє значно кращу швидкість у порівнянні з стандартним алгоритмом DB. Незважаючи на те, що обидва алгоритми успішно генерували токени, TOTP виявився набагато ефективнішим у кожному обсязі спроб.

Під час тестування для 100, 1000, 10000 та 100000 спроб TOTP витрачав значно менше часу на генерацію токенів у порівнянні з DB. Це підтверджує перевагу TOTP щодо швидкості створення токенів незалежно від обсягу спроб.

Зокрема, зі збільшенням кількості спроб різниця у часі між алгоритмами також збільшувалась, показуючи перевагу TOTP навіть у випадку великої кількості спроб (100000), де він виявився значно швидшим і більш ефективним у порівнянні з DB.

Отже, отримані результати вказують на те, що алгоритм TOTP є більш ефективним у плані швидкості генерації токенів порівняно з алгоритмом

DB для різних обсягів спроб, що підтверджує його перевагу у реалізації алгоритму обмеження доступу за часом.

4.1.2.Тест на верифікацію токенів

Для тестування процедури верифікації токенів потрібно зробити схожі дії що і при генерації, але перед верифікацією створюємо тестовий токен для верифікації. Цю генерацію в час тесту не враховується, далі тестовий код перевіряємо певну кількість раз. Для об’єктивності візьмемо такий же набір кількості спроб верифікації токену, що і в попередньому тесті.

Звичайний DB використовує стандартний алгоритм з використанням бази даних, тоді як модифікований TOTP є модифікованим алгоритмом верифікації токенів.

Після проведення тестів очікуються наступні результати:

Рисунок 4.8 – час верифікації 100 токенів

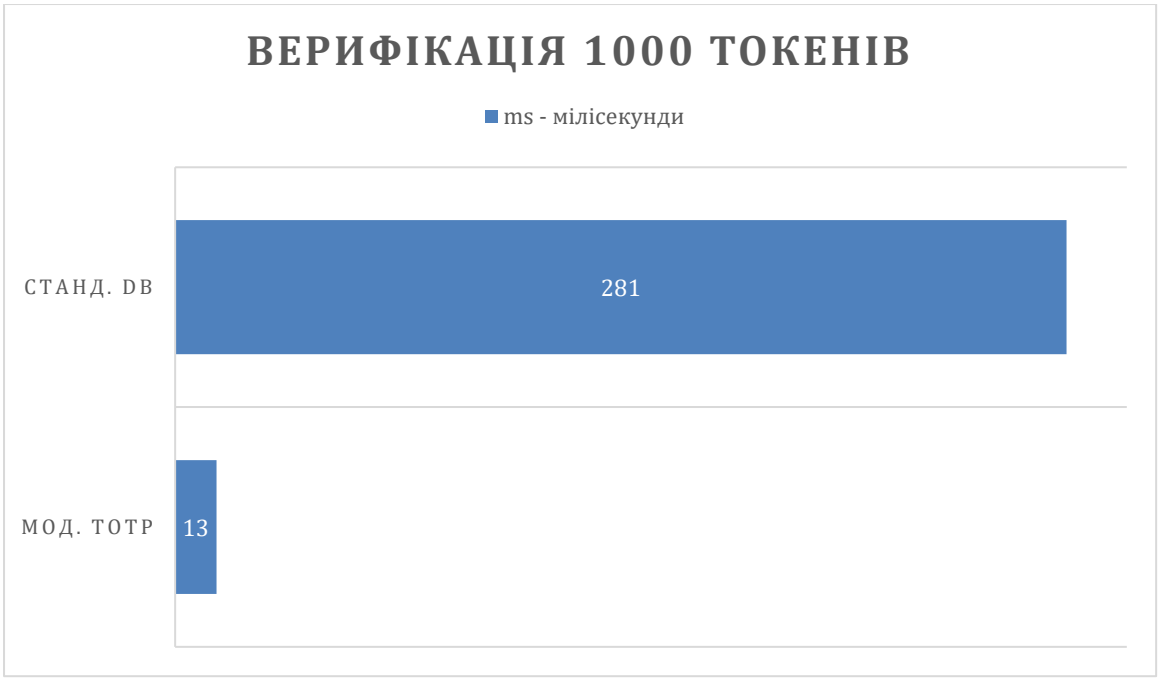


Рисунок 4.9 – час верифікації 1000 токенів

Рисунок 4.10 – час верифікації 10000 токенів

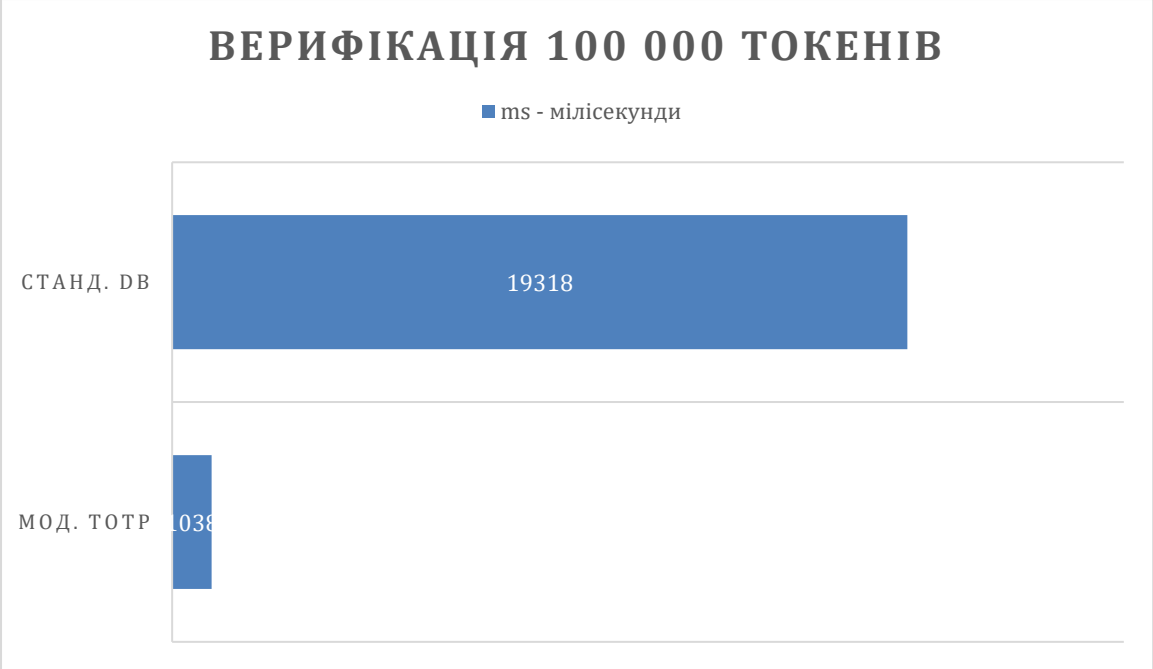


Рисунок 4.11 – час верифікації 100000 токенів

З результатів тестів видно, що процедура верифікації токенів виявилася схожою на процес їх генерації. Однак важливо відзначити, що стандартний алгоритм трошки уповільнився у порівнянні з процесом генерації для такої ж кількості даних.

Отже, слід підкреслити, що модифікований алгоритм виявився набагато швидшим під час верифікації токенів, в порівнянні зі стандартним алгоритмом. Це вказує на його високу продуктивність та ефективність у виконанні завдань верифікації та підтвердження доступу.

4.1.3. Тест на ресурсозатратність

Звісно, тест на ресурсозатратність має на меті виміряти, наскільки ефективно та оптимально використовуються ресурси (такі як час, пам'ять, обчислювальна потужність) під час виконання певних процесів або операцій. Цей вид тестування дозволяє оцінити рівень затрат ресурсів для виконання конкретних завдань у системі чи програмному забезпеченні.

Основна мета тесту на ресурсозатратність полягає у визначенні оптимальності функціонування системи або програми з точки зору використання ресурсів. В процесі тестування вимірюються ресурсозатрати під час виконання конкретних операцій або взаємодії з системою.

Цей тест дозволяє виявити можливі слабкі місця у програмному забезпеченні або системі, які можуть призвести до неефективного використання ресурсів, сповільнення роботи чи зайвих витрат. Наприклад, вимірюються час виконання певних операцій, обсяг використаної пам'яті, кількість процесорних ресурсів та інші параметри для оцінки ефективності системи або програми.

Загальна мета тесту на ресурсозатратність - забезпечити оптимальну роботу системи, уникнути надмірного споживання ресурсів та забезпечити їхню ефективну та раціональну використання під час виконання різноманітних завдань чи операцій.

Для тестування був використаний IntelliJ Profiler - це інструмент, який надає можливості для аналізу продуктивності програмного забезпечення в середовищі IntelliJ IDEA. Він дозволяє розробникам вимірювати та аналізувати різні параметри, такі як використання пам'яті, час виконання методів, обсяг CPU і інші характеристики програми під час її роботи. Профайлер надає графічний інтерфейс для візуалізації даних про продуктивність програми, що допомагає ідентифікувати проблемні місця в коді та вдосконалювати його ефективність.

Для тестування було згенеровано та верифіковано 100000 токенів використовуючи як стандартний так і модифікований алгоритм.

Розглянемо модифікований алгоритм TOTP. Після генерації 100000 токенів отримаємо такі результати.

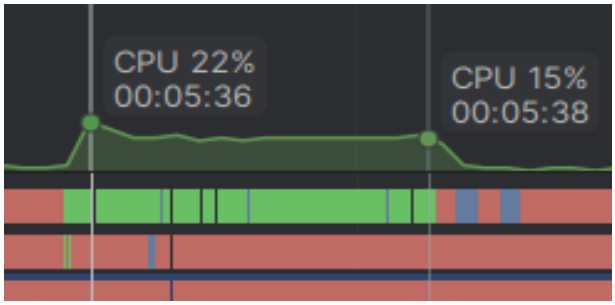


Рисунок 4.12 – обсяг CPU модифікованого алгоритму при генерації TOTP

Рисунок 4.13 – обсяг пам’яті модифікованого алгоритму при генерації TOTP

З результатів видно, що обсяг CPU сягає в своєму піку 22% але далі зменшується і залишається приблизно 15%. Ця операція займає приблизно 2 секунду, що вже було показано в одному з попередніх тестів. Але як можемо побачити обсяг пам’яті який використовує метод generateCode сягає 656.23 MB.

Результати для стандартного алгоритму який використовує базу даних показують наступне:

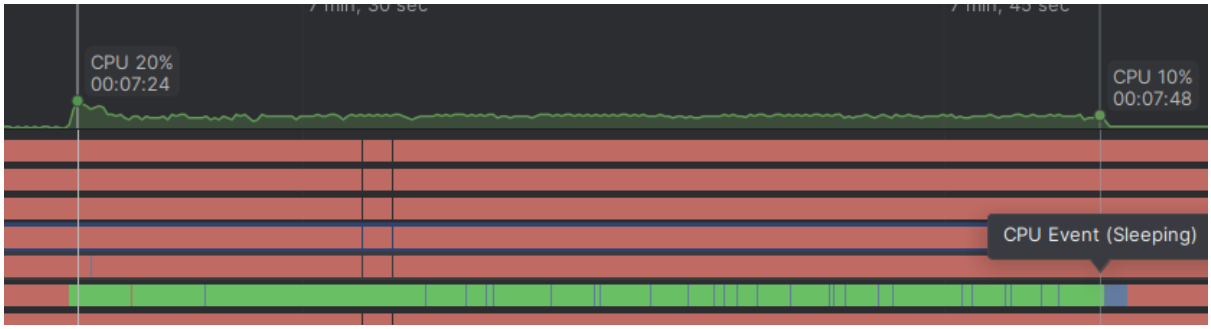


Рисунок 4.14 – обсяг CPU стандартного алгоритму DB при генерації

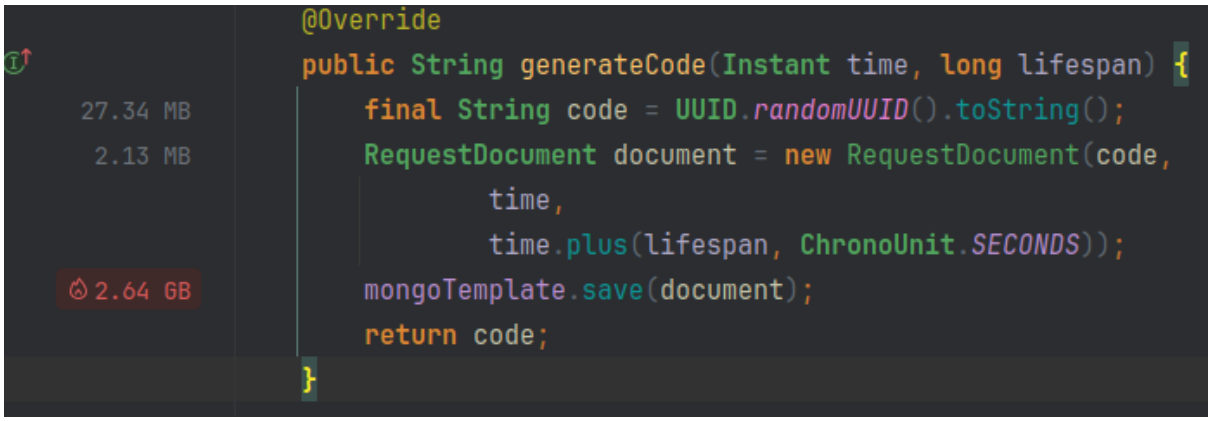


Рисунок 4.15 – обсяг пам’яті стандартного алгоритму DB при генерації

З результатів видно, що обсяг CPU сягає в своєму піку 20% але далі зменшується і залишається приблизно 10%. Треба відмітити що дана операція займає набагато більше часу ніж модифікований алгоритм. Ця операція займає приблизно 20 секунду, що вже було показано в одному з попередніх тестів. Але як можемо побачити обсяг пам’яті який використовує метод generateCode сягає 2.65 GB, що набагато більше ніж у модифікованого алгоритму.

Тож маємо що модифікований алгоритм використовує трошки більше обчислювальної спроможності, хоча час за який він це робить досить короткий, Це пояснюється використання складними алгоритмами хешування. В свою чергу стандартна реалізація яка використовує базу даних використовує в рази більше пам’яті, але пікове навантаження на CPU менше, хоча тривалість цього навантаження велика, що не є ефективним.

Пропоную розглянути ще верифікацію:

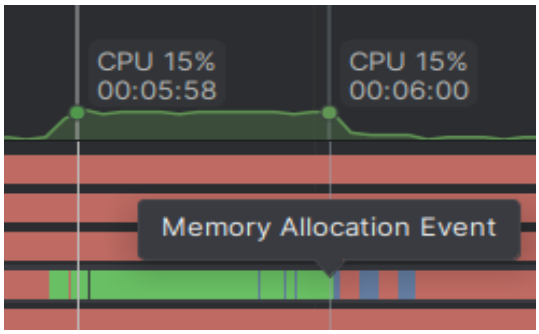


Рисунок 4.16 – обсяг CPU модифікованого алгоритму TOTP при верифікації

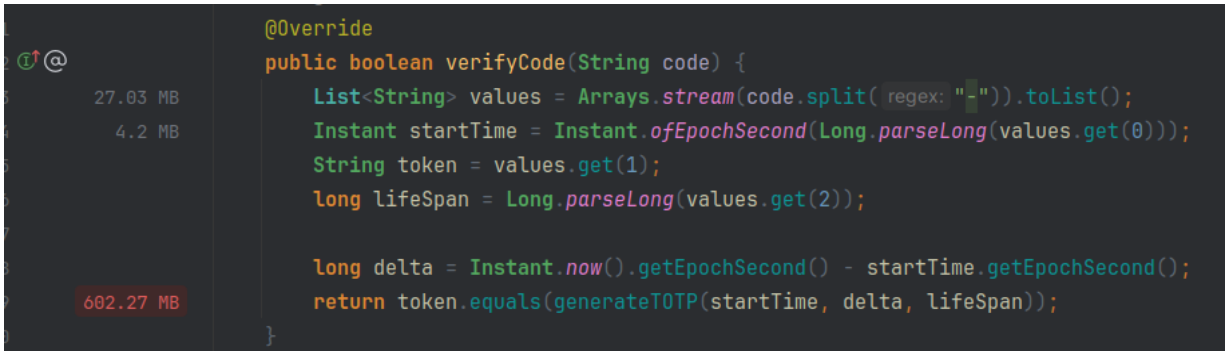


Рисунок 4.17 – обсяг пам’яті модифікованого алгоритму при верифікації TOTP

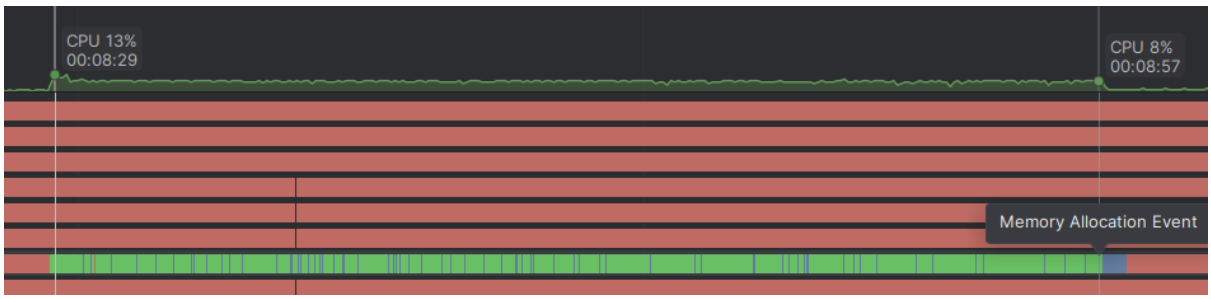


Рисунок 4.18 – обсяг CPU стандартного алгоритму DB при верифікації

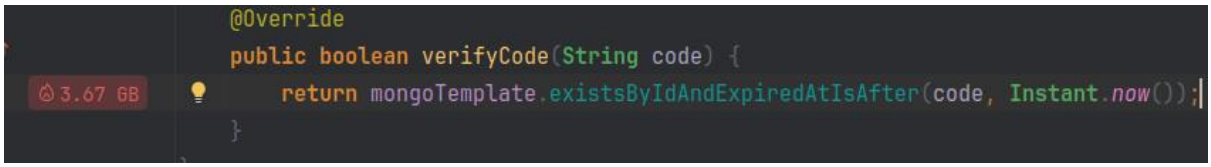


Рисунок 4.19 – обсяг пам’яті стандартного алгоритму DB при верифікації

Для модифікованого алгоритму отримаємо схожі дані, так як процес верифікації відрізняється від верифікації лише тим що ми отримаємо код парсимо його та вираховуємо дельту між часом коли був згенерований та коли він прийшов на перевірку. Далі виконується процес генерації токену, то порівняння його з тим що надійшов.

Стандартна реалізація трохи зменшила навантаження на CPU, але помітно збільшила у використанні пам’яті. Так як вона перевіряє наявність ідентифікатора що надійшов у базі даних, та чи не пройшов його термін валідності. І не генерує дані для запису в баз, тому маємо такі результати.

В загальному знову можна відмітити продуктивність та ефективність, а також економію даних при використані модифікованого алгоритму TOTP токені.

4.1.4. Тест на обсяг токенів за одиницю часу в залежності від кількості символів в токені.

Оскільки алгоритм, який використовує базу даних в якості ідентифікатора, генерує випадковий UUID і час його генерації залишається постійним, проведення такого тесту для цього алгоритму виявляється неефективним. Натомість, модифікований алгоритм може створювати токени різної довжини, забезпечуючи безпеку та ефективність. Таким чином, обумовлено проведення тестування генерації певної кількості токенів протягом фіксованого проміжку часу, наприклад, 60 секунд, що вважається достатнім для оцінки продуктивності алгоритму. Проведемо

тест в якому будемо генерувати токену довжиною від 6 до 18 символів, так як саме така довжина є безпечною при використанні даного алгоритму.

Рисунок 4.20 – Графік згенерованих токенів за 60 секунд в залежності від кількості символів в токені

Як можна побачити з графіка на продуктивність модифікованого алгоритму не сильно впливає кількість символів в токені. Беручи до уваги скільки токенів було згенеровано, відмінність між результатами можна вважати похибкою.

Та можна зробити висновок що кількість символів токенів, не впливає на продуктивність та ефективність алгоритму. Тож без втрати ефективності можна використовувати максимальну допустиму кількість символів в токені.

4.2. Тестування стійкості до брут-форс атак в залежності від часу та довжини токена.

Для проведення експерименту з визначення оптимальної тривалості життя токена відносно кількості цифр у TOTP можна створити програмне

забезпечення, що емулює брут-форс атаку. Це програмне забезпечення генеруватиме випадкові числа та намагатиметься перевірити TOTP коди.

Варто забезпечити унікальність кожної спроби, щоб уникнути повторних запитів, та обмежити тривалість життя токена, щоб він був вразливим саме на цей період.

Експеримент передбачає визначення оптимальних параметрів для безпеки системи. Для цього будуть встановлені різні значення кількості цифр у TOTP (від 6 до 18 цифр) та різна тривалість життя токена для кожної варіації (наприклад, 30 секунд, 60 секунд, 120 секунд і т. д.).

Після цього буде створено програмне забезпечення для емуляції брутфорс атаки, яке автоматично генеруватиме випадкові числа та відправлятиме запити на перевірку TOTP кодів. Кожна спроба атаки буде унікальною, без повторних запитів, та обмежена часом життя токена. Після запуску програми для кожної варіації кількості цифр у TOTP буде проведено брутфорс атаки з урахуванням обмеження часу життя токена.

Отримані результати дослідження дозволять зіставити час, необхідний для успішної атаки, для різних варіацій кількості цифр та тривалості життя токена. На основі аналізу отриманих даних будуть сформульовані рекомендації щодо оптимальної тривалості життя токена в контексті безпеки системи в залежності від кількості цифр у TOTP.

Час життя токена у системі важливий для забезпечення безпеки та зручності користувачів. Розглядаючи модифікований алгоритм генерації токенів, я приймаю рішення встановити його час життя на рівні 3 хвилин. Ця тривалість є відповідною для багатьох сучасних систем, що мають схожий функціонал автентифікації.

Вибір тривалості 3 хвилин для токена базується на балансі між безпекою та зручністю використання. Короткий час життя токена сприяє підвищенню безпеки системи, оскільки він скорочує період часу, протягом якого можлива атака на доступ до системи через витік аутентифікаційної

інформації. Такий підхід дозволяє зменшити час, протягом якого зловмисники можуть скористатися викраденою інформацією для несанкціонованого доступу.

З іншого боку, тривалість 3 хвилин є достатньою для забезпечення зручності користувачів. Вона дає достатньо часу для завершення процесу автентифікації без надмірних обмежень для користувачів системи. Користувачі можуть подивитися інформацію та завершити процедуру вчасно без великих незручностей, що додає певний рівень комфорту при використанні системи.

Отже, обрана тривалість життя токена у 3 хвилини є оптимальною, забезпечуючи баланс між безпекою та зручністю використання, що є важливим аспектом для багатьох подібних систем з такою функціональністю.

Для тестування встановив максимальну тривалість брутфорс атаки у 3 хв, та будемо збільшувати кількість цифр в токені. По завершенню проаналізуємо результати.

Виконавши таке тестування отримали наступні результати з логів.

```
- Initial code - 1702499130-469360-180
-- TOTP length 6. The code is CRACKED, the number of attempts is 377255 in 2 seconds
- Initial code - 1702499132-4033344-180
-- TOTP length 7. The code is CRACKED, the number of attempts is 3841018 in 15 seconds
- Initial code - 1702499148-56889947-180
-- TOTP length 8. The code is CRACKED, the number of attempts is 24804525 in 111 seconds
- Initial code - 1702499259-292658200-180
- TOTP length 9. The code DOESN'T cracked, the number of attempts is 37893520 in 180 seconds
- Initial code - 1702499439-3118658048-180
- TOTP length 10. The code DOESN'T cracked, the number of attempts is 40407474 in 180 seconds
```

Рисунок 4.21 – лог виконання емуляції брутфорс атаки.

З логів можемо побачити, що токен з довжиною 6, 7 та 8 було зламані за 2, 17 та 111 секунд відповідно. Подальші результати показали що токен довжиною від 9 є надійні для використання токена який має тривалість

життя 3 хвилини. Але хочу зазначити що це тестування не може бути супер о’бєктивним. Бо машина для тестування може бути не достатньо потужною. Хоча з іншого боку сучасні системи не будуть допускати так багато запитів від однієї машини. Але далі при тестуванні було виявлено що моя тестова машина не змогла зламати токен довжиною 10 символів навіть за 10 хвилин. На своїй локальній машині я так і не зміг зламати токен у 18 символів за 10 годин тестування.

Беручи до уваги той факт, що кількість символів не впливає на час генерування та верифікації, то рекомендую використовувати максимальну довжину токена у 18 символів.

Висновки до четвертого розділу

В результаті проведення тестування модифікованого алгоритму для генерації та верифікації токенів, були зроблені наступні висновки:

Проведений тест на генерацію токенів показав, що модифікований алгоритм TOTP виявився надзвичайно швидким та ефективним у порівнянні зі стандартним алгоритмом DB. Незалежно від обсягу спроб (100, 1000, 10000 та 100000), TOTP виявив набагато кращу продуктивність у порівнянні з DB, забезпечуючи високу швидкість генерації токенів.

Під час тестування верифікації токенів виявлено, що стандартний алгоритм уповільнюється у порівнянні з генерацією для такої ж кількості даних. З іншого боку, модифікований алгоритм проявив набагато кращі показники швидкості під час верифікації токенів, вказуючи на його високу продуктивність та ефективність.

Результати показали, що модифікований алгоритм має менші вимоги до CPU та пам'яті порівняно зі стандартним алгоритмом. Хоча використання модифікованого алгоритму дещо збільшує обсяг

використаної пам'яті, проте він все одно виявився більш ефективним у використанні ресурсів.

Тест на обсяг токенів за одиницю часу в залежності від кількості символів в токені, інформація з тесту показала, що кількість символів в токені не суттєво впливає на продуктивність алгоритму, дозволяючи використовувати максимальну довжину токену без втрати ефективності.

Тест на стійкість до брут-форс атак в залежності від часу та довжини токена, в ході тестів було виявлено, що модифікований алгоритм, особливо при довжині токена від 8 символів, є досить надійним для використання, оскільки навіть після тривалого тестування брут-форс атак не виявлено зламу токена.

Враховуючи всі вищезазначені результати тестування, рекомендується використовувати модифікований алгоритм TOTP для забезпечення високої швидкості, ефективності, мінімізації ресурсозатрат та стійкості до брут-форс атак у системах обмеження доступу за часом з використанням токенів

ВИСНОВКИ

В ході роботи над магістерською роботою було розглянуто проблеми надання обмеженого часу доступу до ресурсу по часу, за допомогою одноразових паролів які генеруються на основі поточного часу.

У першому розділі роботи було проаналізовано ключові аспекти кібербезпеки, роль паролів у забезпеченні безпеки та технології створення одноразових паролів. Поняття кібербезпеки розкривають ризики та методи захисту в цифровому середовищі. Паролі, як перший рівень захисту, залишаються ключовими в аспекті обмеження несанкціонованого доступу до особистих даних. Одноразові паролі, будучи одним із методів аутентифікації, гарантують тимчасову та унікальну ідентифікацію.

Вивчаючи ці аспекти, було виявлено важливий потенціал у токенах, заснованих на часі. Час стає універсальним маркером, який визначає ідентифікацію у різних контекстах. Цей розділ освітлив ключові аспекти кібербезпеки та методи захисту даних у цифровому середовищі.

У другому розділі були проаналізовані різноманітні методи створення одноразових паролів, їх переваги та недоліки. Виділено важливість використання непередбачуваних та унікальних токенів для підвищення рівня безпеки даних. Різні методи генерації мають свої особливості, і вибір конкретного методу залежить від контексту та потреб безпеки системи.

Одноразові паролі стають важливим інструментом у захисті інформації від несанкціонованого доступу. Моя робота спрямована на розробку механізму генерації посилань з обмеженим доступом, уникати потреби в базі даних та забезпечити вищий рівень безпеки.

Остаточно, подальше вдосконалення методів створення одноразових паролів є ключовим аспектом у підвищенні рівня захисту даних. Це може включати розробку нових алгоритмів, які гарантують випадковість та тимчасовість даних без необхідності записування їх в базу даних.

В третьому розділі проведено дослідження та аналіз різних методів імплементації алгоритму обмеження доступу до ресурсів за часовим параметром. Зосереджено увагу на модифікації Time-Based One-Time Password (TOTP) токена, обґрунтовано вибір конкретних інструментів та методів для його модифікації.

Проведено ретельний опис модифікованого алгоритму, який складається з генерації та перевірки. Проаналізовано різні підходи до реалізації алгоритму, враховуючи вплив конкретних методів на ефективність та безпеку системи.

Обґрунтувано вибір інструментів та методів модифікації TOTP токена з огляду на їхню придатність для обмеження доступу до ресурсу по часу. Докладно описано модифікації стандартного TOTP алгоритму, проаналізовано ключові операції - генерацію та перевірку модифікованого токена, враховуючи їхню точність, надійність та відповідність поставленим завданням.

Четвертий розділ містить в собі тестуванні модифікованого алгоритму та під час тестування модифікованого алгоритму для генерації та верифікації токенів отримані наступні результати:

- Тестування генерації токенів: модифікований алгоритм TOTP проявив надзвичайну швидкість та ефективність порівняно зі стандартним алгоритмом DB, незалежно від обсягу спроб.
- Тестування верифікації токенів: стандартний алгоритм уповільнюється порівняно з генерацією для такої ж кількості даних, у той час як модифікований алгоритм виявив кращі показники швидкості та ефективності.
- Вимоги до ресурсів: модифікований алгоритм виявив менші вимоги до CPU та пам'яті, залишаючись ефективним у використанні ресурсів.

- Кількість символів в токені: довжина токену майже не впливає на продуктивність алгоритму, що дозволяє використовувати максимальну довжину токену без втрати ефективності.
- Стійкість до брут-форс атак: модифікований алгоритм, особливо при довжині токену від 9 символів, є досить надійним для використання, оскільки після тривалого тестування брут-форс атак не виявлено зламу токена.

Отже, рекомендується використовувати модифікований алгоритм TOTP для забезпечення високої швидкості, ефективності, мінімізації ресурсозатрат та стійкості до брут-форс атак у системах обмеження доступу за часом з використанням токенів.

Так, використання модифікованого алгоритму TOTP (Time-Based One-Time Password) у високонавантажених сучасних системах може бути досить доцільним. Цей алгоритм може ефективно обмежувати доступ до ресурсів за допомогою часових обмежень, забезпечуючи при цьому економію ресурсів системи. Однак перед його впровадженням необхідно також враховувати конкретні особливості і вимоги самої системи, можливі загрози та необхідність підтримки та оновлення цього алгоритму в майбутньому.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Microsoft: Що таке кібербезпека? – [Електронний ресурс]. – Режим доступу: https://www.microsoft.com/uk-ua/security/business/security-101/what-is-cybersecurity#Cybersecuritydefined
2. Богуш В.М., Кривуца В.Г., Кудін А.М.. «Інформаційна безпека: Термінологічний навчальний довідник». - 2004.
3. Boritz, J. "IS Practitioners' Views on Core Concepts of Information Integrity". International Journal of Accounting Information Systems. Elsevier. – 2011.
4. Аналіз існуючих рішень автентифікації користувачів інформаційних систем та мереж спеціального призначення – [Електронний ресурс]. – Режим доступу: https://www.viti.edu.ua/files/zbk/2020/13_3_2020.pdf
5. Розробка рекомендацій щодо мінімізації ризиків зломів облікових записів на основі аналізу найпошинеріших методів злому – [Електронний ресурс]. – Режим доступу: https://csecurity.kubg.edu.ua/index.php/journal/article/view/300/257
6. Криптографія: загальні визначення класифікація. Асиметричні та симетричні криптоалгоритми, їх порівняння – [Електронний ресурс]. – Режим доступу: https://dehtyarov09.wordpress.com/2014/03/16/криптографія-загальні-визначення-кл-2/
7. Охота Д.Б. Технології комп’ютерної безпеки. Монографія. МЕГУ, Рівне, 2011. - 97 с
8. Cox, I.J., Miller, M.L., Bloom, J.A.: Digital Watermarking. Morgan Kaufmann, 2002, ISBN 1558607145
9. Прикладні науково-технічні дослідження : матеріали V міжнар. наук.-прак. конф., 5-7 квіт. 2021 р. – Академія технічних наук України. – Івано-Франківськ : Видавець Кушнір Г. М. – 2021. – 77 – 80 с.

10. Ідентифікація користувачів інформаційно-комп'ютерних системах – [Електронний ресурс]. – Режим доступу: https://ir.lib.vntu.edu.ua/bitstream/handle/123456789/33705/Колган.pdf?sequence=1&isAllowed=y
11. Kingston: Що таке шифрування та як воно працює? – [Електронний ресурс]. – Режим доступу: https://www.kingston.com/ua/blog/data-security/what-is-encryption
12. Authentication at Scale – [Електронний ресурс]. – Режим доступу: https://web.archive.org/web/20180427183929/https://www.computer.org/cms/Computer.org/ComputingNow/pdfs/AuthenticationAtScale.pdf
13. A One-Time Password System – [Електронний ресурс]. – Режим доступу: https://www.ietf.org/rfc/rfc2289.txt
14. Practical Random Number Generation in Software – [Електронний ресурс]. – Режим доступу: https://www.acsac.org/2003/papers/79.pdf
15. A Practice-Oriented Treatment of Pseudorandom Number Generators Software – [Електронний ресурс]. – Режим доступу: https://link.springer.com/content/pdf/10.1007/3-540-46035-7_24.pdf
16. HMAC: Keyed-Hashing for Message Authentication – [Електронний ресурс]. – Режим доступу: https://datatracker.ietf.org/doc/html/rfc2104
17. HOTP-Based User Authentication Scheme in Home Networks – [Електронний ресурс]. – Режим доступу: https://web.archive.org/web/20160304075138/http://netgna.it.ubi.pt/files/2009-ISA_NASSUE.pdf
18. HOTP: An HMAC-Based One-Time Password Algorithm – [Електронний ресурс]. – Режим доступу: https://datatracker.ietf.org/doc/html/rfc4226
19. TOTP: Time-Based One-Time Password Algorithm – [Електронний ресурс]. – Режим доступу: https://datatracker.ietf.org/doc/html/rfc6238

20. JSON Web Token (JWT) – [Електронний ресурс]. – Режим доступу: https://datatracker.ietf.org/doc/html/rfc7519
21. What is a brute force attack? – [Електронний ресурс]. – Режим доступу: https://www.cloudflare.com/learning/bots/brute-force-attack/
22. Package javax.crypto – [Електронний ресурс]. – Режим доступу: https://docs.oracle.com/javase/8/docs/api/javax/crypto/package-summary.html
23. Cryptanalytic Attacks on Pseudorandom Number Generators – [Електронний ресурс]. – Режим доступу: https://www.schneier.com/wp-content/uploads/2017/10/paper-prngs.pdf
24. Secure Hash Standard – [Електронний ресурс]. – Режим доступу: https://csrc.nist.gov/files/pubs/fips/180-3/final/docs/fips180-3_final.pdf
25. Технології захисту інформації – [Електронний ресурс]. – Режим доступу: https://ela.kpi.ua/bitstream/123456789/23896/1/TZI_book.pdf
26. Поремський Михайло Васильович – Дисертація – [Електронний ресурс]. – Режим доступу: https://ela.kpi.ua/bitstream/123456789/36454/1/Poremskyi_dys.pdf