Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
В.о. завідувача кафедри
кібербезпеки та захисту
інформації
_____ Іван ПАРХОМЕНКО
«___» червня 2025 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи

галузь знань 12 Інформаційні технології
(шифр і назва галузі знань)
спеціальність 125 Кібербезпека
(код і назва спеціальності)
освітній ступень бакалавр
освітня програма Кібербезпека
(назва освітньо-професійної програми)
на тему: «Система автентифікації користувачів на основі одноразових паролів»

Виконавець: студент IV курсу, групи КБ-42

Назар ТКАЧУК
(підпис) (ім’я, прізвище)

	Підпис	**Ім’я, прізвище**
Керівник		Юрій ЩЕБЛАНІН

Нормоконтроль		Яніна ШЕСТАК

Київ 2025

Міністерство освіти і науки України
«Київський національний університет імені Тараса Шевченка»

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
В.о. завідувача кафедрою
кібербезпеки та захисту
інформації
_____Іван ПАРХОМЕНКО
«29» листопада 2024 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності 125 Кібербезпека
(код і назва спеціальності)

освітньої програми Кібербезпека
(назва освітньої програми)

Студенту **КБ-42** **Ткачуку Назару Руслановичу**
(група) (прізвище ім'я по батькові)

Тема кваліфікаційної роботи Система автентифікації користувачів на основі одноразових паролів

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Тема кваліфікаційної роботи затверджена на засіданні кафедри кібербезпеки та захисту інформації протокол №6 від 28.11.2024 р.

2. ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Системи багатофакторної автентифікації, алгоритми RFC 4226 та RFC 6238.

3. ЗМІСТ РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Необхідно ознайомитись з сучасними методами автентифікації користувачів, також алгоритмами генерації одноразових паролів, дослідити вразливості та обмеження до автентифікації, розробити програмну реалізацію системи автентифікації користувачів на основі одноразових паролів.

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Практична цінність Розроблена система автентифікації користувачів

реалізує адаптивний захист та забезпечує багатоетапну перевірку.

5. ДАТА ВИДАЧІ ЗАВДАННЯ

Дата видачі завдання: 29 листопада 2024 року

Завдання видав _____ Юрій ЩЕБЛАНІН
(підпис) (ініціали, прізвище)

Завдання прийняв до виконання _____ Назар ТКАЧУК
(підпис) (ініціали, прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів робіт	Строки виконання робіт (початок-кінець)	Відмітка про виконання
1	Уточнення постановки задачі	29.11.2024 – 09.12.2024	*виконано*
2	Аналіз літератури	10.12.2024 – 13.01.2025	*виконано*
3	Обгрунтування вибору рішення	14.01.2025 – 03.02.2025	*виконано*
4	Аналіз сучасних систем автентифікації	04.02.2025 – 17.03.2025	*виконано*
5	Дослідження вразливостей та загроз	18.03.2025 – 14.04.2025	*виконано*
6	Розробка системи з компонентами аналізу ризику	15.04.2025 – 05.05.2025	*виконано*
7	Тестування та оцінка ефективності системи	06.05.2025 – 19.05.2025	*виконано*
8	Оформлення пояснювальної записки	20.05.2025 – 02.06.2025	*виконано*
9	Підготовка до захисту кваліфікаційної роботи	02.06.2025 – 13.06.2025	*виконано*

Завдання видав _____ Юрій ЩЕБЛАНІН
(підпис) (ініціали, прізвище)

Завдання прийняв до виконання _____ Назар ТКАЧУК
(підпис) (ініціали, прізвище)

Термін подання кваліфікаційної роботи до ЕК « 13» червня 2025 року

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел, 1 додатку, має 76 сторінок основного тексту, 2 таблиці та 15 рисунків. Список використаних джерел містить 31 найменування і займає 4 сторінки.

Метою роботи є розробка системи автентифікації користувачів на основі одноразових паролів з додатковим захистом.

Для досягнення зазначеної мети поставлено наступні завдання:

- Проаналізувати існуючі системи та методи автентифікації;
- Проаналізувати архітектури та механізми генерації одноразових паролів у системах автентифікації користувачів;
- Створити прототип системи автентифікації на основі одноразових паролів;
- Реалізувати адаптивну оцінку ризиків при автентифікації у систему автентифікації на основі одноразових паролів;

Об'єктом дослідження є процес автентифікації користувачів у систему на основі одноразових паролів.

Предметом дослідження є методи та засоби підвищення безпеки систем автентифікації на основі одноразових паролів.

Практична цінність отриманих результатів полягає у створенні повноцінної системи автентифікації, здатної забезпечити посилений рівень безпеки.

Ключові слова: автентифікація користувачів, одноразовий пароль, OTP, багатофакторна автентифікація, адаптивна безпека, TOTP, HOTP, система захисту, оцінка ризику, SMS, e-mail, QR-код, кібербезпека.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

2FA	–	Two-Factor Authentication
API	–	Application Programming Interface
HOTP	–	HMAC-Based One-Time Password
IP	–	Internet Protocol
MITM	–	Man-In-The-Middle
NIST	–	National Institute of Standards and Technology
OTP	–	One-Time Password
QR	–	Quick Response
SMS	–	Short Message Service
TOTP	–	Time-Based One-Time Password

ВСТУП

У зв’язку з швидким розвитком діджиталізації процесів у сферах соціальних, фінансових та адміністративних послуг та комерційного сектору із значним зростанням персоналізованих цифрових операцій, проблема надійності ідентифікація користувача набирають великого значення. Навіть при широкому застосуванні механізму одноразових паролей (OTP) у ряду інформаційних систем є певні вразливості цих технологій як таких. Це значно ускладнює досягнення належного рівня стійкості до несанкціонованого доступу та може створити потенційні вектори атак на критичні ресурсів.

Варто звернути увагу на той факт, що більшість існуючих втілень механізмів OTP не повністю ураховують ризики від таких явищ як міжсесійна компрометація даних, атаки типу “людина посередині” (MITM), перехоплення SMS-трафіку та експлуатація недоліків у протоколах синхронізації. Таким чином, формально високий рівень захисту автентифікаційного процесу не завжди корелює з фактичною безпечністю застосованої архітектури, що детермінує необхідність комплексного дослідження вразливих місць зазначених систем.

Крім того, у світлі актуальних тенденцій розвитку інформаційної безпеки, зокрема імплементації zero-trust-парадигм та багатофакторної автентифікації, вивчення слабких місць одноразових паролів дозволяє не лише оцінити ефективність поточних захисних підходів, але й розробити принципово нові рекомендації щодо побудови більш стійких схем ідентифікації, що враховують як специфіку загрозового ландшафту, так і обмеження кінцевих пристроїв користувачів.

З огляду на викладене, дослідження у сфері розробки систем автентифікації на основі одноразових паролів є вкрай актуальним як у теоретичному, так і в прикладному аспектах, оскільки безпосередньо впливає на

здатність інформаційних систем забезпечувати захист персональних і корпоративних даних в умовах постійної еволюції кіберзагроз.

Метою кваліфікаційної роботи є розробка системи автентифікації користувачів на основі одноразових паролів з додатковим захистом.

Досягнення поставленої мети потребує розв'язання таких задач:

- аналіз існуючих систем та методів автентифікації;
- аналіз архітектури та механізмів генерації одноразових паролів у системах автентифікації користувачів;
- створення прототипу системи автентифікації на основі одноразових паролів;
- реалізація адаптивної оцінки ризику при автентифікації у систему автентифікації на основі одноразових паролів;

Об'єкт дослідження: процес автентифікації користувачів у систему на основі одноразових паролів.

Предмет дослідження: Методи та засоби підвищення безпеки систем автентифікації на основі одноразових паролів.

Оцінка сучасного стану проблеми на основі вітчизняної та зарубіжної літератури. Проблематика забезпечення надійної автентифікації користувачів в інформаційних системах постійно перебуває в центрі уваги наукової спільноти та практиків у сфері інформаційної безпеки. Зокрема, значну увагу приділено використанню одноразових паролів .

У вітчизняних і зарубіжних дослідженнях активно вивчаються алгоритми формування та використання одноразових паролів. Так, у роботах українських науковців запропоновано методи формування псевдовипадкових послідовностей, що ґрунтуються на модулярних перетвореннях, які демонструють високу криптостійкість і адаптивність до сучасних викликів кібербезпеки Серед інших напрямів дослідження – створення інструментів для тестування випадковості OTP, розробка генераторів HOTP і TOTP, а також використання стандартів RFC 4226 та RFC 6238, які визначають загальні підходи до реалізації таких методів у сучасних IT-системах.

Галузь застосування. Системи автентифікації на основі одноразових паролів мають широке застосування в таких сферах, як електронне урядування, банківські системи, захищені корпоративні середовища, а також у побудові безпечних хмарних сервісів. Їхнє впровадження дозволяє істотно підвищити рівень безпеки користувацьких облікових записів без значного зниження зручності доступу.

Практична цінність дослідження полягає в тому, що запропонована система автентифікації може бути використана в інформаційних системах державних установ, банківських сервісах, приватних компаніях і корпоративних мережах. Її реалізація дозволяє підвищити рівень захисту користувацьких облікових даних, забезпечити ефективне реагування на підозрілі спроби входу та підтримати вимоги сучасних стандартів інформаційної безпеки.

Апробація роботи. Основні результати роботи доповідались на таких наукових конференціях:

— XI Міжнародна конференція «Information Technology and Implementation (Satellite)» (21 листопада 2024 р.) за темою «Vulnerrabilities of user authentication systems based on one-time passwords».

РОЗДІЛ 1
АНАЛІЗ СИСТЕМ АВНТЕНТИФІКАЦІЇ КОРИСТУВАЧІВ В ІНФОРМАЦІЙНІЙ БЕЗПЕЦІ

1.1 Класифікація систем автентифікації користувачів

Системи автентифікації, як функціональний компонент комплексного середовища інформаційної безпеки, призначені для встановлення достовірної відповідності між заявленою ідентичністю суб’єкта доступу та підтвердженням цієї ідентичності на основі наданої автентифікаційної інформації. Зважаючи на еволюцію цифрових загроз, розширення моделей доступу та багатоваріантність реалізації політик безпеки, класифікація систем автентифікації набуває дедалі більшого значення у контексті побудови масштабованих, адаптивних і стійких до компрометації рішень.

Систематизація автентифікаційних систем може здійснюватись за низкою класифікаційних ознак, зокрема за кількістю використовуваних факторів, за архітектурною моделлю розгортання, за методом ініціації, за контекстною чутливістю, а також за ступенем централізації управління.

Згідно з міжнародним стандартом ISO/IEC 29115:2013, системи автентифікації класифікуються за рівнями впевненості в автентичності (LoA), які визначаються на основі критеріїв, таких як методи перевірки ідентичності, стійкість до атак та надійність процесу автентифікації. Більш часто системи автентифікації поділяються за кількістю факторів на:

- Однофакторні системи — реалізують автентифікацію на основі одного фактору, що може належати до категорії знання, володіння або притаманності. Ці системи є мінімально ресурсозатратними, проте водночас демонструють найнижчий рівень стійкості до атак, зокрема фішингу та підбору паролів.

- Багатофакторні системи — використовують поєднання декількох незалежних факторів, що значно підвищує ефективність протидії несанкціонованому доступу. MFA системи можуть бути реалізовані як апаратно-програмні комплекси або як сервісні рішення у хмарній інфраструктурі.

В залежності від топології розгортання та організації взаємодії між компонентами, системи автентифікації поділяються на централізовані, децентралізовані та федеративні.

Перші здійснюють обробку облікових записів, ключів та логіки перевірки на одному сервері або у межах керованої доменної структури. Перевагами є уніфікація політик доступу та зручність адміністрування; однак такі системи мають єдину точку відмови. Другі ж базуються на розподіленій моделі управління, де кожен елемент інфраструктури реалізує автентифікаційні функції автономно або у межах довірчої мережі. Прикладом є peer-to-peer системи або блокчейн-орієнтовані моделі ідентифікації.

Федеративні системи автентифікації забезпечують взаємодію між незалежними організаційними доменами через встановлення довірчих відносин, використовуючи протоколи, такі як SAML, OpenID Connect та WS-Federation. Ці протоколи дозволяють обмінюватися інформацією про ідентичність та забезпечують єдиний вхід (SSO) у різних системах.

Залежно від принципів запуску процесу автентифікації та ступеня інтерактивності користувача, системи класифікують на:

- Активні системи автентифікації вимагають від суб’єкта самостійного виконання дії для ініціації перевірки ідентичності. Прикладами є введення логіна і пароля, сканування QR-коду, натискання підтвердження в push-сповіщенні. Такі системи є широко застосовуваними в інтерактивних сервісах з прямим залученням користувача.
- Пасивні системи автентифікації забезпечують перевірку ідентичності без явної участі користувача, на основі фонових характеристик (геолокація, пристрій, поведінкові шаблони, раніше збережені токени сесії).

Зазвичай використовуються як доповнення до активних методів або як частина Zero Trust архітектур.

- Гібридні системи об’єднують активні та пасивні механізми, дозволяючи адаптивне реагування на ризик на основі попереднього аналізу дій користувача.

Новітнім напрямом розвитку автентифікаційних рішень є впровадження адаптивних систем, які динамічно змінюють механізм перевірки ідентичності залежно від контексту та поведінки користувача. Такі системи динамічно коригують вимоги до автентифікації, виходячи з оцінки ризиків, що значно підвищує як безпеку, так і зручність для користувача [1].

Контекстно-адаптивні системи приймають рішення про необхідність автентифікації на основі сукупності змінних параметрів: поточне місцезнаходження, час доби, профіль пристрою, IP-адреса, історія входів тощо. Такі системи зазвичай інтегруються із системами управління ризиками та SIEM-рішеннями. Це дозволяє здійснювати автентифікацію з урахуванням поведінкових шаблонів користувача та характеристик середовища, підвищуючи точність і надійність процесу [2].

Поведінкові системи автентифікації здійснюють безперервний аналіз характерних ознак поведінки користувача під час сеансу: швидкість набору тексту, кути нахилу пристрою, шаблони переміщення курсору. Вони є ефективним додатковим захисним шаром, особливо у високоризикових транзакціях або мобільних додатках, забезпечуючи безперервну перевірку ідентичності на основі унікальних біометричних характеристик [3]. Інтелектуальні системи такого типу значно ускладнюють реалізацію класичних атак, водночас забезпечуючи прозору для користувача взаємодію.

У контексті управління обліковими записами та авторизаційною логікою розрізняють локальні, корпоративні та хмарні системи автентифікації.

Системи хмарного класу демонструють високу гнучкість та масштабованість, проте потребують жорсткого контролю за передачею даних і розмежуванням зон відповідальності між провайдером і кінцевим споживачем.

Підсумовуючи, класифікація систем автентифікації користувачів формується як багатовимірна матриця, де кожна система може одночасно належати до кількох категорій за різними критеріями. Такий підхід дозволяє формувати комплексні, контекстно чутливі системи автентифікації, здатні забезпечити як високий рівень безпеки, так і достатню гнучкість у різнорідних архітектурах.

1.2 Основні методи автентифікації

Методи автентифікації користувачів у цифрових середовищах формують основу будь-якої архітектури інформаційної безпеки, забезпечуючи процедуру перевірки відповідності заявленої ідентичності суб’єкта з його фактичними характеристиками, що підтверджують право доступу до визначених об’єктів. На відміну від класифікації систем автентифікації як інфраструктурних рішень, методи автентифікації представляють собою алгоритмічні, процедурні та технологічні механізми реалізації автентифікаційної логіки. Вибір методу автентифікації впливає не лише на рівень захищеності системи, але й на її продуктивність, масштабованість і сумісність із супутніми сервісами авторизації, моніторингу та управління доступом.

У сучасній практиці методи автентифікації поділяються на базові (однофакторні) та комбіновані (багатофакторні або адаптивні), серед яких виокремлюють низку ключових реалізацій — кожна з яких володіє власними перевагами, обмеженнями та специфічними векторами вразливості.

Парольна автентифікація є історично першим і досі найбільш розповсюдженим методом перевірки ідентичності, що базується на знанні секретної інформації. Механізм реалізується шляхом введення користувачем символьного рядка, який порівнюється із заздалегідь збереженим значенням, що, як правило, представлене у вигляді гешу. Незважаючи на свою простоту, метод демонструє низьку стійкість до цілого ряду атак — зокрема перебору, фішингу, перехоплення клавіш, а також атак через витік хешів або сесійні токени. Проте,

попри свою вразливість, паролі залишаються фундаментальним компонентом багатьох систем автентифікації, особливо у поєднанні з іншими факторами [4].

Криптографічна надійність методу залежить від алгоритму гешування (SHA-256, bcrypt, Argon2), довжини та ентропії пароля, політик ротації та використання сільових значень (salt). Уразливість парольних систем часто виникає не на рівні алгоритму, а внаслідок помилок реалізації — наприклад, зберігання паролів у відкритому вигляді, відсутність обмежень на кількість спроб, слабка система відновлення облікових записів.

Автентифікація за допомогою токенів передбачає використання фізичного або віртуального носія, який містить або генерує унікальний автентифікаційний маркер. Токен може бути реалізований у вигляді USB-ключа, смарт-картки, мобільного додатку або веб-токена типу JWT (JSON Web Token), що передається у заголовках запитів.

У криптографічному контексті токен містить зашифровану або підписану інформацію, що підтверджує автентичність користувача, а також строк його сесії. Такий підхід дозволяє розмежовувати процес автентифікації (отримання токена) та авторизації (перевірка прав на основі вмісту токена). Однак порушення цілісності транспортного каналу, зберігання токенів у незахищеній пам’яті клієнта або вразливості типу XSS/CSRF можуть спричинити несанкціонований доступ навіть без компрометації облікового запису.

Автентифікація на основі біометрії реалізується шляхом зіставлення унікальних фізіологічних або поведінкових характеристик користувача із заздалегідь збереженим шаблоном. Серед найпоширеніших: відбитки пальців, розпізнавання обличчя, райдужної оболонки, динаміка набору тексту, ритм ходьби. Технологія реалізується з використанням сенсорів, модулів машинного навчання та локальних/хмарних баз шаблонів. Біометрична автентифікація є зручним та інтуїтивно зрозумілим методом, який підвищує рівень безпеки за рахунок прив'язки до унікальних фізичних або поведінкових властивостей особи [5].

Незважаючи на високу унікальність біометричних ознак, ризики таких систем полягають у неможливості зміни біометричних ключів після їх витоку, складнощах точного розпізнавання в умовах варіативного середовища (освітлення, шум, пошкодження), а також загрозі підміни шаблонів через злам серверної інфраструктури. Біометрія зазвичай використовується як додатковий фактор у багатофакторних рішеннях, посилюючи основну автентифікацію.

Методи автентифікації, що базуються на застосуванні криптографії з відкритим ключем, забезпечують високий рівень захисту завдяки використанню цифрових сертифікатів (зокрема стандарту X.509) та інфраструктури відкритих ключів (PKI). Суть методу полягає в перевірці автентичності користувача на основі валідного сертифіката, підписаного довіреним центром сертифікації, у поєднанні з криптографічною перевіркою цифрового підпису або аутентифікованого з’єднання.

Ключовими перевагами сертифікатної автентифікації є висока стійкість до підробки, можливість використання апаратних засобів зберігання, а також гнучкість у політиках анулювання (revocation). Проте функціонування таких систем вимагає складної інфраструктури підтримки — генерації, розповсюдження, відстеження статусу та оновлення сертифікатів. Крім того, вразливість клієнтських пристроїв до компрометації приватного ключа нівелює переваги криптографічного захисту, якщо не реалізовані жорсткі механізми контролю доступу до ключових контейнерів.

Одноразові паролі (OTP) реалізують механізм автентифікації, при якому кожне використання пароля є унікальним, і такий пароль втрачає чинність одразу після застосування або по завершенню визначеного інтервалу часу. Основними реалізаціями є алгоритми HOTP (HMAC-Based One-Time Password) та TOTP (Time-Based One-Time Password), стандартизовані відповідно в RFC 4226 та RFC 6238. В основі механізмів лежить використання криптографічного HMAC-алгоритму з симетричним секретним ключем, що зберігається на стороні клієнта та сервера.

OTP забезпечує значно вищу стійкість до атак повторного відтворення (replay attack), компрометації статичних паролів та перехоплення в процесі транспортування, особливо за умови додаткового шифрування каналу передачі. Проте захищеність методу прямо залежить від способу доставки пароля (SMS, мобільний додаток, push-сповіщення) та реалізації захисту кінцевих пристроїв. Завдяки своїй унікальності та обмеженому терміну дії, OTP ефективно протидіють ризикам, пов'язаним із витоками статичних паролів, і є ключовим компонентом більшості систем багатофакторної автентифікації [6].

У відповідь на підвищення складності кіберзагроз, зокрема мультивекторних атак, які поєднують соціотехнічні та технічні підходи, сучасна практика автентифікації передбачає використання комбінованих та адаптивних методів. У першому випадку реалізується одночасна перевірка декількох незалежних факторів (наприклад, знання + володіння), що забезпечує значне зниження ймовірності успішної атаки навіть у разі компрометації одного з факторів.

Адаптивні методи, у свою чергу, реалізують концепцію risk-based authentication, при якій механізм автентифікації автоматично змінюється залежно від поведінкового та контекстуального аналізу: IP-адреса, географія, час доступу, тип пристрою, рівень аномалії.

1.3 Автентифікація за допомогою одноразових паролів

Автентифікаційні механізми, що базуються на одноразових паролях (One-Time Password, OTP), належать до категорії динамічних методів перевірки ідентичності, у межах яких кожна автентифікаційна сесія супроводжується генеруванням унікального символьного маркера, призначеного для одноразового використання. На відміну від статичних автентифікаційних схем, які покладаються на незмінні облікові дані, OTP-технології забезпечують значно вищий рівень стійкості до повторного використання автентифікаційної інформації, перехоплення в мережі або атак типу «людина посередині».

Узагальнена модель OTP-автентифікації передбачає двокомпонентну структуру: механізм генерації одноразового пароля на стороні клієнта (у вигляді програмного або апаратного токена) та механізм його верифікації на стороні сервера. Ідентифікаційний процес супроводжується введенням користувачем OTP-коду, що був сформований в межах синхронізованого алгоритму, параметризованого на основі спільного секретного ключа. Ця архітектура дозволяє уникнути зберігання статичних паролів на сервері та значно знижує ризики, пов'язані з витоками баз даних облікових записів [7].

Згідно із загальноприйнятою термінологією, OTP реалізується на основі двох основних підходів: генерація на основі лічильника (HOTP) та генерація на основі поточного часу (TOTP). Обидві моделі спираються на HMAC-конструкцію, що реалізує криптографічне хешування з використанням симетричного ключа, однак різняться за логікою визначення моменту активації чергового OTP-коду. У HOTP застосовується лічильник викликів, що інкрементується при кожному запиті, тоді як у TOTP використовується дискретизована шкала часу, яка поділяється на інтервали фіксованої тривалості (зазвичай 30 секунд), що забезпечує часову обмеженість дії OTP.

На відміну від класичних паролів, які можуть бути об’єктом автоматизованих атак перебору, OTP коди є криптографічно обмеженими як у часовому, так і в числовому просторі. За умов правильної реалізації криптографічного алгоритму, значення одноразового пароля в конкретний момент часу або після певної кількості запитів неможливо передбачити без знання секретного ключа. Крім того, реалізація OTP не вимагає збереження стану попередньої автентифікації, що дозволяє зменшити поверхню атаки на серверному боці.

Принципова особливість OTP-механізмів полягає у гнучкості способів доставки автентифікаційного маркера до користувача. Серед реалізованих варіантів: SMS-OTP, push-нотифікації у мобільних додатках генератори в офлайн-режимі, а також десктопні програмні інтерфейси. Вибір конкретного каналу має критичне значення для рівня захищеності системи, оскільки

транспортний рівень є найбільш уразливою ланкою у багатьох практичних реалізаціях.

У системах підвищеної чутливості до несанкціонованого доступу — зокрема, в електронному банкінгу, фінансових транзакціях, медичних інформаційних системах, електронному урядуванні — OTP-механізми посідають ключове місце в забезпеченні автентифікації. Наприклад, у банківських системах OTP застосовується як підтвердження не лише автентичності входу користувача, а й транзакційної цілісності, виступаючи одночасно як метод ідентифікації та верифікації конкретної дії. У хмарних платформах OTP інтегрується у системи єдиного входу (Single Sign-On, SSO), забезпечуючи багатофакторну автентифікацію в поєднанні з протоколами OAuth 2.0 або SAML.

Застосування OTP у корпоративному секторі часто відбувається на базі централізованих систем управління ідентичністю (IDM/IAM), де одноразовий пароль генерується на основі інтегрованих політик безпеки з урахуванням ризикового профілю користувача. У таких випадках OTP не є ізольованим методом, а виступає компонентом адаптивної системи автентифікації, яка динамічно змінює політику перевірки залежно від контексту: наприклад, вимога введення OTP лише при виході за межі корпоративної мережі або доступі до критичних ресурсів.

Технологічно OTP добре поєднується з мобільними пристроями, що дозволяє розгортати масштабовані рішення без необхідності використання додаткового обладнання. Програмні генератори OTP зокрема, Google Authenticator, FreeOTP, Authy, OTPClient функціонують у автономному режимі, зберігаючи синхронізацію з сервером виключно на основі часу або лічильника, без потреби у постійному мережевому з’єднанні. Це дозволяє реалізувати високозахищені офлайн-сценарії автентифікації, забезпечуючи зручність використання та стійкість до мережевих атак [8].

Проте незалежно від технічної реалізації, одноразові паролі не є самодостатнім захистом від складних цілеспрямованих атак, особливо у випадку,

коли застосовуються в однорівневій схемі без додаткових факторів. Наприклад, уразливості транспортного шару, такі як: перехоплення OTP через зламаний SS7 у мережах GSM, атаки типу SIM-swapping, фішинг із вбудованим MITM, а також зловживання push-нотифікаціями (push bombing) можуть призвести до обходу OTP-автентифікації навіть без компрометації облікових даних користувача.

У зв'язку з цим, у сучасних архітектурах безпеки OTP рідко використовується як єдиний метод перевірки. Натомість він інтегрується в багатофакторні моделі автентифікації, що передбачають комбінацію OTP з іншими факторами. Це дозволяє значно посилити загальну безпеку системи, створюючи багатошаровий захист від різних типів кіберзагроз [9].

1.4 Багатофакторні системи автентифікації

У контексті зростаючої складності атак на автентифікаційні механізми, багатофакторна автентифікація (Multi-Factor Authentication, MFA) виступає як базовий елемент посилення стійкості цифрових систем до компрометації облікових записів. Концептуально багатофакторний підхід базується на одночасному використанні щонайменше двох автентифікаційних факторів, що належать до різних категорій: знання (що відоме користувачеві), володіння (що належить користувачеві), притаманність (що характеризує користувача). Застосування MFA значно підвищує рівень безпеки, вимагаючи від користувача надати кілька доказів своєї ідентичності з різних категорій, що ускладнює несанкціонований доступ навіть у випадку компрометації одного з факторів [4]. Реалізація таких систем дозволяє істотно зменшити ризик несанкціонованого доступу навіть у разі часткової компрометації автентифікаційних даних.

Порівняно з однофакторними схемами, багатофакторні рішення формують додаткові бар'єри на шляху атак, орієнтованих на соціальну інженерію, фішинг, автоматизований підбір паролів або крадіжку пристроїв. Зокрема, одноразовий пароль, який може бути перехоплений або відновлений шляхом маніпуляцій з SIM-карткою, не дозволяє отримати доступ до системи без наявності первинного

статичного пароля або біометричного зчитувача, якщо така перевірка передбачена у схемі автентифікації.

Типова схема MFA включає комбінацію одного основного фактора (зазвичай — статичний пароль або цифровий сертифікат) з другим фактором динамічного типу — одноразовим паролем, push-нотифікацією, USB-ключем або біометричним зчитуванням. При цьому багатофакторність може реалізовуватись як у послідовному режимі (каскадна перевірка), так і паралельно (з використанням SSO-інтерфейсів або сторонніх провайдерів ідентичності). Багатофакторну використовують у різних галузях, приклади застосувань наведені у (таблиці 1.1).

Таблиця 1.1

Приклади використання багатофакторної автентифікації

Галузь	**Сценарій**	**Фактори MFA**	**Особливості**
Банківська справа	Інтернет-банкінг	Пароль + SMS або push	Стандарт для захисту рахунків
Охорона здоров’я	Доступ до ЕМС	Пароль + біометрія	Швидкий і захищений доступ
Держсектор	Урядові портали	Смарт-карта + PIN	Висока довіра до апаратних ключів
Корпорації	VPN-доступ	Пароль + OTP/U2F	Інтеграція з AD, централізований контроль
Хмарні сервіси	Доступ до Google/MS	Пароль + push/ключ	Підтримка безпарольних схем
Е-комерція	Оплата	Пароль + OTP/біометрія	Адаптивна MFA для великих сум
Освіта	Вхід до LMS	Пароль + моб. код	Простота та масштабованість

У сучасних корпоративних і державних системах MFA часто є обов’язковим компонентом політики безпеки, особливо в контексті дотримання нормативних вимог. Крім того, MFA інтегрується у більшість фреймворків Zero

Trust, де жоден користувач або пристрій не вважається «довіреним» без повторної багатофакторної перевірки при кожному сеансі доступу.

Удосконаленою формою класичного MFA є адаптивна багатофакторна автентифікація, що застосовує контекстуальні змінні для динамічного формування вимог до перевірки користувача. У таких системах ступінь автентифікаційної жорсткості змінюється залежно від ризикового профілю сесії, що розраховується на основі історичних даних, геолокації, часу входу, профілю пристрою, а також результатів поведінкового аналізу. е дозволяє оптимізувати взаємодію з користувачем, зменшуючи кількість кроків для низькоризикових дій, одночасно посилюючи захист у випадках виявлення аномалій або підвищеного ризику доступу [10].

Ключовим трендом у розвитку MFA є інтеграція з системами поведінкової біометрії, що використовують такі ознаки, як динаміка набору тексту, траєкторія миші, шаблон навігації, ритм використання інтерфейсу. Ці параметри, що не потребують активної участі користувача після початкової автентифікації, дозволяють реалізувати концепцію безперервної автентифікації, постійно верифікуючи ідентичність протягом усього сеансу використання системи [11].

Іншою перспективною реалізацією MFA є так звані безпарольні системи, де жоден із факторів не базується на знанні пароля. У таких моделях автентифікація може бути реалізована виключно на основі комбінації факторів володіння (наприклад, криптографічний ключ у модулі TPM пристрою) та притаманності (розпізнавання обличчя, біометрія голосу). Стандарти FIDO2/WebAuthn підтримують подібні схеми і вже інтегруються у браузери, мобільні ОС та хмарні сервіси. У межах таких підходів значно знижується навантаження на користувача, зменшується поверхня атаки, пов’язана із зберіганням та введенням паролів, однак виникають нові виклики щодо безпечного управління біометричними шаблонами та апаратними ідентифікаторами.

Незалежно від архітектури, впровадження багатофакторної автентифікації передбачає комплексну інтеграцію з існуючими інформаційними системами, що

вимагає дотримання сумісності з існуючими протоколами автентифікації (RADIUS, LDAP, SAML, OAuth), підтримки централізованого управління користувачами, а також журналювання та моніторингу автентифікаційних подій. Окрім технічної реалізації, критичне значення має формування адекватної політики доступу, яка враховує специфіку робочих процесів, рівні ризику, категорії користувачів і чутливість даних, до яких здійснюється доступ.

Таким чином, багатофакторна автентифікація є не просто надбудовою над базовими методами перевірки ідентичності, а самостійним механізмом управління ризиками у середовищах з високими вимогами до захищеності. Проте навіть найбільш комплексні MFA-рішення не є цілковито захищеними від вразливостей, що зумовлює потребу в критичному аналізі їхньої ефективності, а також глибокому розумінні характерних недоліків методів автентифікації.

1.5 Характерні недоліки сучасних методів автентифікації

Незважаючи на значну еволюцію методів автентифікації у напрямі підвищення їх криптографічної стійкості, зручності інтеграції та адаптивності до контексту користувацької поведінки, жоден із застосовуваних на практиці підходів не є універсально ефективним у протистоянні цілеспрямованим атакам або компрометації середовища автентифікації. Сукупність технічних, організаційних та людських чинників зумовлює збереження високої ймовірності успішного обходу механізмів перевірки ідентичності навіть у багатофакторних або контекстно-чутливих системах. У цьому підрозділі проаналізовано найбільш характерні недоліки, притаманні сучасним автентифікаційним рішенням, із позиції їх експлуатаційних властивостей, типових вразливостей та обмежень масштабування.

Одним із найбільш поширених і водночас складних для нейтралізації типів загроз є соціотехнічні атаки, які не спрямовані безпосередньо на технічні компоненти автентифікації, а орієнтовані на маніпуляцію діями користувача. Типовими прикладами є фішинг, вішинг, атаки через соціальні мережі, а також

сценарії маскування під служби технічної підтримки. Навіть за наявності технічних засобів захисту, людський фактор залишається одним із найслабших місць, оскільки зловмисники експлуатують психологічні особливості жертв, а не недоліки системи [12]. Це свідчить про те, що жоден технічний механізм не гарантує безпеку без належного супроводу політик цифрової гігієни та навчання персоналу.

Особливо уразливими до фішингових атак залишаються реалізації MFA з push-нотифікаціями, де відсутній контекст дії (наприклад, точна адреса запиту або транзакція), а користувач діє рефлекторно. У разі багаторазового отримання запитів на підтвердження автентифікації може спрацьовувати феномен «підтвердження за інерцією», що на практиці використовується в атаках типу push bombing.

Більшість сучасних методів автентифікації передбачають активну участь кінцевого пристрою користувача, який виступає джерелом генерації автентифікаційного токена, зчитування біометричних даних або каналом прийому одноразових паролів. У випадках, коли пристрій скомпрометований (шкідливе ПЗ, root-доступ, фішинговий додаток), ефективність навіть найсучасніших алгоритмів автентифікації знижується до нуля, оскільки будь-які дані можуть бути перехоплені ще до їх криптографічного захисту або підміни.

Крім того, мобільні платформи мають обмеження у рівні захисту середовища виконання, що ускладнює реалізацію надійної ізоляції між додатками. Деякі мобільні ОС допускають доступ до сповіщень або інтерфейсів введення інших додатків, що створює потенційну можливість зчитування OTP або імітації автентифікаційного екрана. Таким чином, без інтеграції з модулями захищеного виконання (TEE, Secure Enclave) або апаратними криптомодулями (TPM, YubiKey) — автентифікація залишається вразливою до локального зламу пристрою.

Незважаючи на позиціонування біометричних технологій як високонадійних і зручних для користувача механізмів автентифікації, практичне їх впровадження супроводжується низкою суттєвих технічних обмежень. По-

перше, точність і стабільність роботи біометричних систем безпосередньо залежать від якості сенсорного обладнання та умов експлуатації. Варіативність зовнішніх факторів — освітлення, вологість, механічні пошкодження, природні фізіологічні зміни — може призводити як до хибнопозитивних (false acceptance), так і до хибнонегативних (false rejection) результатів.

По-друге, у разі витоку біометричних шаблонів, їх заміна на відміну від паролів або токенів є неможливою, що створює критичну проблему незворотності компрометації. Наявність відкритих або незахищених баз біометричних даних у поєднанні з методами їх підробки робить ці методи вразливими до атак спуфінгу, що підкреслює необхідність їх застосування виключно як один із факторів у багатофакторних схемах [13]. З цієї причини біометрія має використовуватися виключно в комбінації з іншими факторами автентифікації та за умов наявності захищеної інфраструктури обробки даних (наприклад, локальне зберігання шаблонів у TEE).

Одноразові паролі у формі HOTP або TOTP, незважаючи на їхню популярність і доведену криптографічну ефективність, мають низку організаційних та операційних обмежень. Передусім, функціонування таких систем вимагає наявності на обох сторонах (сервер і клієнт) спільного секретного ключа, що формує залежність від якісної синхронізації і безпечного зберігання ключового матеріалу. У випадку з багатокористувацькими системами це ускладнює масштабування і розгортання, оскільки потребує індивідуальної генерації та інсталяції ключів для кожного користувача, з подальшою логістикою оновлення у разі втрати чи компрометації.

Суттєвим обмеженням для ряду сучасних автентифікаційних рішень є залежність від зовнішніх або неконтрольованих каналів передачі автентифікаційної інформації. Наприклад, OTP через SMS — хоч і широко розповсюджене рішення, — є технічно застарілим і схильним до атак через вразливості SS7, перенаправлення повідомлень (SMS forwarding), SIM swap або спуфінг базових станцій. Вразливість каналів доставки, таких як SMS, створює значні ризики для безпеки, оскільки зловмисники можуть перехоплювати

одноразові паролі, обходячи навіть багатофакторну автентифікацію [14]. Навіть push-нотифікації через мобільні додатки можуть бути заблоковані або затримані, що призводить до порушення користувацького досвіду і зниження надійності автентифікації.

У системах, що інтегрують сторонніх провайдерів (наприклад, Firebase, Twilio, Auth0), виникає додаткова загроза зниження контролю над ланцюгом довіри. Компрометація API-ключів, затримки в доставці або несанкціоноване втручання у процеси аутентифікації з боку третіх осіб можуть призвести до зниження рівня безпеки навіть у добре спроєктованій архітектурі.

Висновки за розділом 1

У результаті системного аналізу сучасних підходів до автентифікації користувачів в інформаційних системах встановлено, що формування ефективної автентифікаційної політики є критичним елементом загальної архітектури інформаційної безпеки, здатним істотно впливати на стійкість до зовнішніх та внутрішніх загроз. Визначення і класифікація систем автентифікації на основі кількості факторів, архітектурних моделей, способу ініціації та поведінкових характеристик дозволяє формалізувати підходи до побудови стійких до атак автентифікаційних контурів.

Серед основних методів автентифікації, що застосовуються у практиці забезпечення захисту інформації, виділено паролі, токени, цифрові сертифікати, біометричні рішення та OTP-технології. Автентифікація на основі одноразових паролів, розглянута як окремий клас методів, характеризується високим ступенем динамічності та здатністю обмежувати часову актуальність токена, що знижує ймовірність успішного використання перехоплених даних. Проте ізольоване використання навіть криптографічно стійких OTP-механізмів не забезпечує гарантованого захисту за відсутності багатофакторного підходу.

Багатофакторна автентифікація, що ґрунтується на комбінуванні факторів різної природи, має найбільшу ефективність у контексті протидії як технічним,

так і соціотехнічним загрозам. Її впровадження супроводжується розширенням функціональності автентифікаційних систем за рахунок контекстно-адаптивних механізмів, поведінкових профілів і безперервного моніторингу, що потребує комплексної інтеграції із засобами керування доступом, журналюванням подій та ризик-орієнтованою аналітикою.

Разом з тим виявлено, що навіть найрозвинутіші системи автентифікації залишаються вразливими до низки об’єктивних недоліків: залежності від стану кінцевого пристрою, транспортної інфраструктури, складності управління ключами, незворотності біометричних ознак та вразливостей людського фактора.

РОЗДІЛ 2
ДОСЛІДЖЕННЯ АВТЕНТИФІКАЦІЇ КОРИСТУВАЧІВ НА ОСНОВІ ОДНОРАЗОВИХ ПАРОЛІВ

2.1 Принципи роботи систем на основі одноразових паролів

Системи автентифікації, що реалізують перевірку ідентичності користувача на основі одноразових паролів (One-Time Password, OTP), формують окремий клас криптографічно підтриманих механізмів доступу, орієнтованих на забезпечення динамічної валідації автентичності шляхом унеможливлення повторного використання автентифікаційного маркера. Принципова ідея OTP-систем полягає у тому, що кожен пароль, згенерований в рамках певного автентифікаційного сеансу, є валідним виключно протягом заздалегідь визначеного періоду часу або в межах конкретної взаємодії, що істотно підвищує рівень безпеки порівняно зі статичними паролями, оскільки навіть у разі перехоплення OTP він швидко стає недійсним і не може бути використаний зловмисником повторно [15]. Це практично усуває ефективність атак типу replay та значно обмежує часовий горизонт потенційної компрометації.

У загальному випадку функціонування OTP-системи описується як взаємодія між генератором одноразового коду (який може бути апаратним або програмним) та сервером автентифікації, що виконує верифікацію введеного користувачем коду. Генерація коду здійснюється згідно із заздалегідь визначеним алгоритмом, який приймає на вхід симетричний секретний ключ та синхронізуючий параметр — лічильник або часову мітку. Отриманий цифровий маркер обробляється криптографічною функцією HMAC (Hash-based Message Authentication Code), після чого результат піддається усіченню до визначеного формату — зазвичай 6 або 8 десяткових цифр.

Системна модель OTP-автентифікації включає такі логічні компоненти:

- генератор OTP-кодів, який реалізується на стороні користувача у вигляді мобільного додатку, апаратного токена або інтегрованої функції в операційній системі;
- верифікатор OTP, що функціонує на сервері й містить алгоритм обчислення очікуваного OTP-коду на основі тих самих параметрів, які використовуються генератором;
- механізм синхронізації, який відповідає за узгодження параметрів генерації, зокрема лічильника (у випадку HOTP) або часу (у випадку TOTP);
- канал введення OTP, через який користувач передає згенерований код до системи — найчастіше це веб-форма, інтерфейс командного рядка, мобільний додаток або спеціалізований API.

Важливою особливістю OTP-систем є їх криптографічна односпрямованість, що означає неможливість відновлення вхідного параметра або секретного ключа зі згенерованого OTP-коду. Це досягається шляхом використання геш-функцій зі стійкістю до колізій і підбору, таких як SHA-1, SHA-256 або SHA-512 у складі HMAC-конструкції. Рівень захищеності системи при цьому залежить не лише від обраного алгоритму, але й від коректності реалізації механізму синхронізації, безпечності зберігання ключа та надійності середовища виконання генератора.

З огляду на критичну роль узгодженості параметрів генерації в загальній структурі OTP-систем, особливої уваги потребує механізм синхронізації між генератором та верифікатором, зокрема в контексті часових відхилень або зсувів лічильника, що можуть виникати внаслідок несподіваних збоїв, мережевих затримок або багаторазових хибних спроб автентифікації. У разі використання лічильникової моделі (як у HOTP), сервер, як правило, реалізує алгоритм з допуском певного вікна відхилення (наприклад, 5–10 значень), що дозволяє компенсувати часткову розсинхронізацію без втрати можливості ідентифікації. Водночас у часових моделях (TOTP) необхідно враховувати ймовірність розбіжностей між системним часом клієнтського пристрою та серверу. Стандарт RFC 6238 рекомендує використовувати "вікно толерантності" для TOTP, що

дозволяє приймати OTP, згенеровані протягом декількох кроків часу до або після поточного, для усунення проблем, пов'язаних з неточностями синхронізації годинників [8].

У типових реалізаціях сервер обчислює OTP-коди для низки можливих значень параметра синхронізації — лічильника або часу — і порівнює їх з кодом, наданим користувачем. У разі виявлення збігу в межах допустимого вікна здійснюється автентифікація, а верифікатор фіксує використане значення з метою запобігання повторному прийняттю ідентичного OTP у подальшому. У синхронізованих моделях лічильник після вдалого сеансу збільшується до нового значення, яке автоматично ігнорує всі попередні. Таким чином реалізується гарантія одноразовості OTP, що є ключовою перевагою цієї категорії механізмів автентифікації.

Варто окремо зазначити, що системи, побудовані на основі одноразових паролів, хоча й формально спираються на принципи симетричної криптографії, відрізняються від класичних схем автентифікації за паролем унікальним підходом до управління ключовими матеріалами. Зокрема, секретний ключ, що використовується для генерації OTP, зазвичай є сталим для одного користувача протягом всього життєвого циклу токена, однак сам автентифікаційний код змінюється щоразу. Цей підхід мінімізує залежність безпеки системи від секретності каналу передачі OTP, оскільки навіть повне перехоплення коду не дозволяє використати його повторно. Водночас це створює додаткові вимоги до захисту самого секретного ключа, що зберігається на клієнтському пристрої або в апаратному модулі.

Технічні аспекти реалізації генераторів OTP вимагають особливої уваги до питань захисту середовища виконання. У разі використання мобільних додатків, які реалізують алгоритми TOTP або HOTP, критичним є захист не лише самого додатку, а й середовища операційної системи, доступу до сховищ з ключовими параметрами, а також контроль за можливими втручаннями в роботу генератора з боку стороннього програмного забезпечення. У цьому контексті особливої ваги набуває використання апаратних модулів захисту, зокрема Trusted Execution

Environment (TEE) або Secure Element, які забезпечують апаратну ізоляцію критичних обчислень та зберігання секретних ключів, що підвищує стійкість системи до програмних атак на клієнтському пристрої [16].

Із боку серверу, ефективна обробка OTP-запитів передбачає впровадження не лише алгоритмічної верифікації, а й додаткових контекстних перевірок, таких як аналіз частоти запитів, джерела з'єднання, географічного розташування користувача, характеру змін у поведінковій моделі тощо. Сучасні серверні рішення здатні використовувати машинне навчання для формування профілів звичайної активності користувачів і виявлення аномалій, що можуть свідчити про спробу компрометації навіть при правильному OTP-коді.

Таким чином, ефективність систем автентифікації на основі одноразових паролів базується не лише на криптографічній надійності використовуваних алгоритмів, але й на комплексності підходу до архітектури рішень, що охоплює всі етапи життєвого циклу OTP — від генерації й передачі до верифікації та знищення. Визначальним чинником стає ступінь стійкості системи до зовнішнього впливу, зокрема спроб атак через соціальну інженерію, підміну середовища виконання, доступ до ключового матеріалу або маніпуляцію контекстом автентифікації.

2.2 Алгоритм Time-based One-time Password

Алгоритм Time-based One-time Password (TOTP) являє собою криптографічний механізм, який забезпечує формування одноразових паролів, значення яких є дійсним лише протягом обмеженого проміжку часу. Цей алгоритм, формалізований у стандарті RFC 6238, виник як розвиток HOTP, з урахуванням необхідності усунення недоліків, пов'язаних із відсутністю часової прив'язки у генерації кодів автентифікації. TOTP вирішує проблему, властиву HOTP, яка полягає в тому, що коди OTP залишаються дійсними до моменту їх використання, що робить їх вразливими до атак повторного відтворення, якщо лічильник не синхронізований. Завдяки інтеграції часової компоненти, TOTP

забезпечує автоматичну недійсність коду після короткого проміжку часу, підвищуючи безпеку автентифікації [8]. Такий підхід, орієнтований на тимчасову залежність від поточного значення системного часу, дозволяє значно знизити ймовірність повторного використання перехопленого одноразового пароля. Загальна схема роботи алгоритму зображена на (рисунку 2.1).

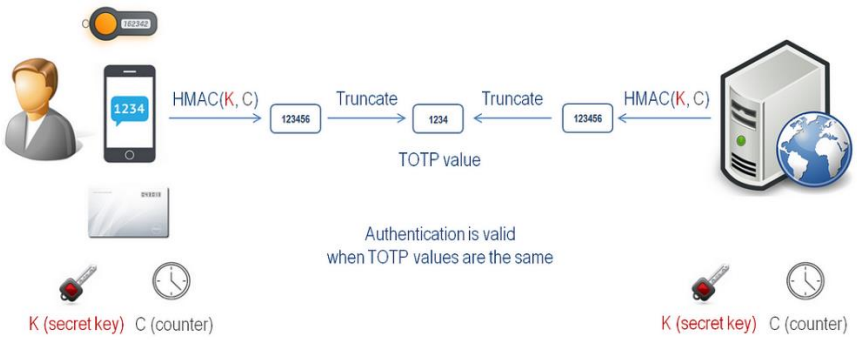

Рисунок 2.1. – Схема роботи алгоритму TOTP

Основою функціонування алгоритму TOTP є симетричний криптографічний примітив — геш-функція HMAC (Hash-based Message Authentication Code), яка забезпечує математичну односторонність та криптостійкість обчислень. У загальному вигляді, обчислення TOTP-значення може бути представлено (виразом 2.1).

$$TOTP(K) = Truncate(HMAC\_Hash(K, T)) \quad (2.1)$$

де K — секретний криптографічний ключ, що є спільним для клієнта та сервера;

T — значення лічильника часу, отриманого на основі епохального часу Unix, а функція;

Truncate — функція, яка виконує операцію усічення результату гешування до фіксованої кількості десяткових цифр, зазвичай шістьох.

Величина T обчислюється відповідно до (формули 2.2).

$$T = floor((Current\ Unix\ Time - T_0) / X) \quad (2.2)$$

де T_0— фіксована початкова епоха, здебільшого встановлюється як 0;

X — інтервал часу (у секундах), протягом якого згенероване значення OTP вважається дійсним. Стандартним значенням параметра X у реалізаціях є 30 секунд. Таким чином, часовий інтервал визначає «вікно» допустимого використання конкретного OTP без потреби в явному збереженні стану між викликами функції.

Процес усічення, що застосовується до результату функції HMAC, має вирішальне значення для приведення отриманого бінарного хешу до форми короткого числового коду, зручного для введення користувачем. Згідно з рекомендаціями, виведене значення HMAC піддається динамічному усіченню (dynamic truncation), яке передбачає вибір 4 байтів починаючи з позиції, визначеної останніми 4 бітами хешу згідно (формули 2.3).

$$Offset = HMAC(K, T)[19] \ \&\ 0x0F \quad (2.3)$$

Після чого обрані 4 байти інтерпретуються як беззнакове ціле 32-розрядне число. Отримане значення приводиться до потрібної кількості десяткових знаків за допомогою операції взяття залишку від ділення на 10^d, де d — кількість цифр одноразового пароля. Відбувається це відповідно до (формули 2.4).

$$OTP = (Binary\ Code) \bmod 10^d \quad (2.4)$$

Результатом є числове значення, зручне для практичного використання, що одночасно зберігає властивості непередбачуваності та односторонності, притаманні криптографічним геш-функціям.

До особливостей реалізації TOTP-алгоритму слід віднести вимогу до високої точності синхронізації системного часу між клієнтським і серверним середовищем. Зокрема, навіть незначні відхилення годинника можуть призводити до ситуацій, у яких сформовані OTP виявляються недійсними з точки

зору сервера автентифікації. У зв’язку з цим у типовій реалізації передбачається вікно толерантності (time-step window), що допускає перевірку декількох сусідніх часових інтервалів — як поточного, так і одного-двох попередніх або наступних. Такий підхід дозволяє компенсувати похибки синхронізації до межі визначеного інтервалу X, не погіршуючи при цьому загальну безпеку процесу. Ця можливість компенсувати невеликі розбіжності в часі між клієнтом і сервером є критично важливою для зручності використання, оскільки мінімізує кількість відмов через незначні зсуви годинника, при цьому зберігаючи високий рівень безпеки за рахунок обмеженого вікна валідності [17].

З практичного погляду, алгоритм TOTP отримав широке розповсюдження завдяки своїй простоті імплементації та відсутності необхідності в збереженні стану між сесіями автентифікації, що робить його придатним для використання у великомасштабних системах із високими вимогами до продуктивності. Його реалізації підтримуються більшістю сучасних рішень двофакторної автентифікації, включно з апаратними токенами, мобільними застосунками та вбудованими механізмами корпоративних платформ.

Не менш важливою характеристикою є те, що криптографічна стійкість алгоритму TOTP цілком базується на секретності ключа KK та складності підбору значення OTP у припущенні відсутності доступу до цього ключа. Таким чином, безпека системи, що використовує TOTP, прямо залежить від якості генерації, зберігання та обміну секретним ключем, що визначає потребу в його належному захисті за допомогою апаратних модулів безпеки (HSM), шифрування при зберіганні, а також процедур керування життєвим циклом ключів.

У той же час, незважаючи на переваги тимчасової залежності, TOTP залишається вразливим до певних класів атак, зокрема до атак повторного використання у випадках перехоплення OTP у межах допустимого часової вікна. Тому розгортання TOTP має супроводжуватися застосуванням додаткових захисних механізмів, таких як захищений канал передавання (TLS), контроль частоти введення, а також облік попередніх значень для виявлення аномалій.

Дослідження показують, що, незважаючи на високу стійкість, TOTP не є панацеєю від усіх типів атак, зокрема, він може бути скомпрометований через фішинг, коли користувач вводить OTP на підробленому сайті. Це підкреслює необхідність комплексного підходу до безпеки, де TOTP є частиною більш широкої стратегії захисту [18].

Таким чином, алгоритм TOTP є важливим представником класу криптографічних механізмів автентифікації на основі одноразових паролів, що поєднує високу криптостійкість, ефективність реалізації та практичну зручність використання. У контексті розвитку більш адаптивних методів захисту інформаційних систем, що включають часову динаміку, він утворює логічну основу для побудови складніших схем автентифікації, зокрема із залученням елементів контекстної обробки або машинного навчання.

2.3 Алгоритм HMAC-based One-time Password

У процесі побудови систем автентифікації, які мають гарантувати високий рівень стійкості до атак типу «повторного використання паролів» та перехоплення, важливою конструкцією виявляється клас одноразових паролів, що ґрунтуються на геш-функціях із ключем. Серед них, HMAC-based One-Time Password (HOTP) алгоритм є базовим криптографічним механізмом, на основі якого згодом було розроблено похідні, зокрема алгоритм TOTP. Таким чином, HOTP варто розглядати як фундаментальну реалізацію симетричної генерації паролів із використанням лічильника як синхронізувального чинника.

Протокол HOTP був стандартизований у межах специфікації RFC 4226, яка визначає обчислювальну модель, правила усічення, формат пароля, а також допустимі криптографічні хеш-функції**.** Ця специфікація деталізує, як генерується OTP, використовуючи спільний секретний ключ і лічильник, що інкрементується з кожною автентифікаційною подією. Логіка роботи алгоритму у системі автентифікації наведена на (рисунку 2.2).

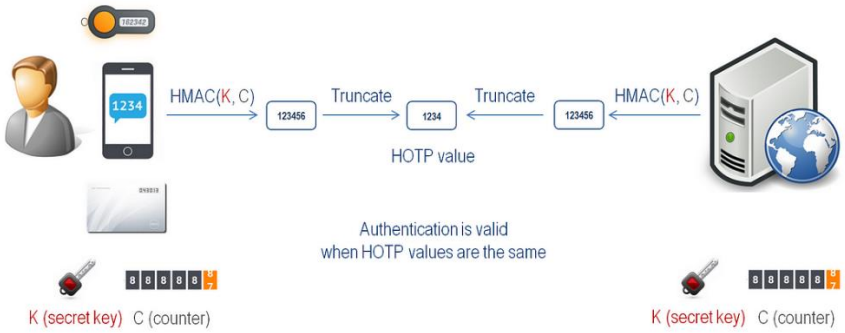

Рисунок 2.2. – Схема роботи алгоритму TOTP

Унікальність кожного згенерованого пароля та його неможливість повторного використання забезпечує лічильник, що є ключовим для протидії атакам повторного відтворення [7]. В основі алгоритму лежить обчислення HMAC-коду на основі спільного секретного ключа та значення лічильника. Саме лічильник, що зберігає поточний стан сесії або транзакційної взаємодії, виступає в ролі індексу генерації наступного OTP, забезпечуючи послідовну зміну значень паролів у кожному новому циклі автентифікації.

Формально, HOTP-значення обчислюється за (формулою 2.5).

$$HOTP(K, C) = Truncate(HMAC\text{-}SHA\text{-}1(K, C)) \bmod 10^{d} \quad (2.5)$$

де K — секретний симетричний ключ, що є спільним між клієнтом і сервером;

C — лічильник, що інкрементується під час кожного запиту OTP;

HMAC-SHA-1(K, C) — результат застосування HMAC-функції на основі SHA-1 до пари ключ–лічильник;

Truncate(...) — функція динамічного усічення, яка забезпечує приведення гешу до цілого числа обмеженої довжини;

d — кількість десяткових знаків (типово 6 або 8), що визначає формат OTP.

Функція HMAC-SHA-1(K, C) виконує класичну побудову HMAC відповідно до (визначення 2.6).

$$HMAC(K,C) = SHA-1\Big((K' \oplus opad)||\, SHA-1\big((K' \oplus \backslash ipad)||C\big)\Big) \quad (2.6)$$

де K' — ключ, доповнений до довжини блоку SHA-1 (64 байти);

opad, ipad — фіксовані значення зовнішньої та внутрішньої маски відповідно (0x5c та 0x36).

Окремо слід відзначити механізм динамічного усічення Truncate(...), який реалізується відповідно до наступної процедури:

З останнього байта результату HMAC вилучається 4-ох бітний offset (молодші біти останнього байта).

На основі offset обираються 4 послідовні байти в геші.

Обрані байти об'єднуються в 31-бітове ціле число, у якому старший біт примусово обнулюється для запобігання знаковому перетворенню.

Цей процес формалізується (виразом 2.7).

$$\begin{aligned} &Offset = HMAC(K, C)[19] \,\&\, 0x0F \\ &BinaryCode = (HMAC(K, C)[Offset] \,\&\, 0x7F) << 24 \\ &\qquad\qquad | \,(HMAC(K, C)[Offset + 1] \,\&\, 0xFF) << 16 \\ &\qquad\qquad | \,(HMAC(K, C)[Offset + 2] \,\&\, 0xFF) << 8 \\ &\qquad\qquad | \,(HMAC(K, C)[Offset + 3] \,\&\, 0xFF) \end{aligned} \quad (2.7)$$

Фінальний OTP обчислюється з використанням операції взяття залишку, який визначається (формулою 2.8).

$$OTP = BinaryCode \bmod 10^{d} \quad (2.8)$$

Ця операція забезпечує, що значення одноразового пароля буде належати множині десяткових чисел заданої довжини, чим задовольняється вимога до компактності та зручності використання в інтерфейсах введення. Водночас такий

формат не створює додаткових складностей у контексті користувацької взаємодії, особливо в обмежених мобільних або вбудованих середовищах.

На відміну від алгоритму TOTP, який вимагає точного хронометричного узгодження між клієнтом і сервером, HOTP передбачає синхронізацію через лічильник, що зростає лише при виконанні операцій, пов’язаних із генерацією пароля. Це дозволяє йому працювати без залежності від синхронізації часу, що є перевагою в умовах нестабільного зв'язку або відсутності доступу до точних часових сервісів [20]. Такий підхід знижує залежність від часових відхилень, однак створює іншу проблему — необхідність обробки розсинхронізації лічильників між сторонами. У типових реалізаціях ця задача вирішується шляхом введення допустимого вікна значень лічильника на сервері, що сканує обмежену кількість майбутніх станів у межах визначеного вікна, наприклад, від 5 до 10 кроків.

Криптографічна стійкість HOTP безпосередньо пов’язана з характеристиками використовуваної геш-функції, а також з ентропією ключа K. Хоча початковий стандарт обмежувався HMAC-SHA-1, сучасні реалізації допускають використання HMAC на основі SHA-256 або SHA-512. Проте такі відступи від специфікації RFC 4226 потребують ретельної перевірки сумісності між реалізаціями клієнта та сервера. Для систем, орієнтованих на високі вимоги до безпеки, рекомендується поступовий перехід до більш стійких геш-функцій у рамках розширеної специфікації, з урахуванням зворотної сумісності.

Таким чином, алгоритм HOTP реалізує класичний приклад лічильникової автентифікації із симетричним шифруванням, який, завдяки простоті імплементації та відсутності залежності від годинника, набув широкого поширення в апаратних токенах і мобільних додатках. У подальшому ця модель розширюється до повноцінних архітектур багатофакторної автентифікації, в яких OTP є одним із компонентів складнішої логіки перевірки достовірності користувача.

Попри архітектурну простоту та достатню гнучкість у впровадженні, практичне застосування алгоритму HOTP потребує врахування кількох

критичних аспектів, пов’язаних із гарантуванням захисту від несанкціонованого використання або компрометації. Насамперед ідеться про захист секретного ключа K, який, будучи спільним між двома сторонами автентифікації, в умовах компрометації призводить до повної втрати цілісності механізму. Тому збереження ключа в апаратному криптопровайдері, токені з Trusted Execution Environment (TEE), або в межах засобів Hardware Security Module (HSM), є технічно обґрунтованим підходом для систем підвищеної критичності. Належний захист секретного ключа є життєво важливим, оскільки будь-яка його компрометація повністю нівелює безпеку HOTP, дозволяючи зловмиснику генерувати дійсні одноразові паролі. Це вимагає впровадження надійних механізмів управління ключами та їх зберігання в захищених апаратних сховищах [19].

Крім того, важливою особливістю реалізації HOTP є контроль за повторним використанням паролів, що особливо актуально у випадках, коли користувач зберігає OTP у зовнішніх джерелах, або коли виникає затримка передачі між сторонами. Для запобігання атакам типу replay-attack у таких ситуаціях сервер має фіксувати останнє прийняте значення лічильника та автоматично зростати його лише у разі успішного проходження автентифікації. Для цього запроваджується контроль допустимого вікна перевірки window, в межах якого відбувається порівняння OTP на відповідність одному з майбутніх значень. Типове значення вікна — 5 або 10 — обирається з урахуванням ризику скомпрометованого входу проти ризику хибної відмови в автентифікації.

Одним із вагомих недоліків HOTP у контексті сучасних мобільних та веб-орієнтованих середовищ є потенційна десинхронізація лічильника, що може виникнути, наприклад, у разі декількох спроб генерації OTP на клієнті без відповідного запиту до сервера. Застосування двобічної синхронізації вимагає додаткової логіки: з боку клієнта — збереження стану лічильника, з боку сервера — буферного пошуку OTP у межах вікна та підтвердження актуального значення. Ускладнення архітектури, спричинене необхідністю врахування цих факторів, у ряді випадків призводить до переваги варіантів на основі

синхронізації за часом (TOTP), однак у пристроях, позбавлених стабільного доступу до точного часу, HOTP залишається єдино можливим механізмом.

У реалізаціях, що відповідають специфікації RFC 4226, рекомендовано додержуватись таких параметрів:

- Довжина секретного ключа K — не менше 128 біт.
- Формат OTP — шістцифровий (6 десяткових цифр), тобто d = 6.
- Хеш-функція — HMAC на основі SHA-1.
- Максимальна кількість OTP, що перевіряються сервером у вікні — до 10.

Узагальнюючи, можна відобразити алгоритм HOTP у вигляді наступної послідовності операцій:

1) Отримання поточного значення лічильника C.
2) Генерація HMAC-коду з використанням ключа K та C.
3) Застосування динамічного усічення (dynamic truncation) для перетворення результату HMAC до 31-бітного числа.
4) Застосування операції взяття модуля для отримання десяткового OTP.
5) Передача OTP на сервер для перевірки в межах допустимого вікна.

Алгоритм HMAC-based One-Time Password залишається високонадійним рішенням для задач одноразової автентифікації в системах, де забезпечено надійне збереження секретного ключа та відсутня критична залежність від часових сервісів.

2.4 Архітектура систем автентифікації з використанням одноразових паролів

Функціонування механізмів автентифікації, що базуються на принципі використання одноразових паролів, неможливо забезпечити без впровадження чітко визначеної, структурно впорядкованої архітектури, яка інтегрує в собі компоненти генерації, верифікації, синхронізації та захисту критичних

параметрів. Архітектура таких систем, що використовують OTP, має бути комплексною, охоплюючи не лише криптографічні аспекти генерації паролів, але й питання інтеграції з різними програмними та апаратними платформами, що підкреслює її складність та багатокомпонентність [20]. Вона має враховувати не лише логіку криптографічного перетворення, як-от у випадку з HOTP або TOTP, але і прикладні аспекти інтеграції з програмними й апаратними платформами, зокрема мобільними клієнтами, серверними середовищами, засобами багатофакторної автентифікації та контролю доступу.

У типових реалізаціях система автентифікації на основі одноразових паролів включає мінімум чотири взаємопов'язані функціональні модулі:

- Модуль генерації одноразового пароля, реалізований на стороні клієнта (наприклад, у мобільному застосунку або апаратному токені), виконує локальне обчислення OTP згідно з визначеним алгоритмом (TOTP або HOTP), використовуючи захищений секретний ключ і часову мітку або лічильник. Цей модуль є критично вразливим до атак у випадку витоку ключа або підміни середовища виконання, що передбачає необхідність апаратної або криптографічної ізоляції обчислень.
- Сервер автентифікації, що функціонує як точка прийому та перевірки OTP, забезпечує логіку верифікації шляхом обчислення еталонного OTP із використанням відповідного секретного ключа, асоційованого з конкретним користувачем. У випадку з TOTP алгоритмом сервер додатково враховує допустиме часове вікно дійсності, а для HOTP — зберігає лічильник і забезпечує синхронізацію після кожної успішної автентифікації.
- Сховище облікових записів та ключів, відповідальне за безпечне збереження асоціацій між користувачами і їхніми відповідними секретними ключами, а також атрибутами автентифікації (як-от обмеження на вікно перевірки, параметри алгоритму, часові зони тощо). Реалізація цього компоненту потребує суворого контролю доступу та криптографічного захисту на рівні бази даних.

- Модуль інтеграції з прикладною інфраструктурою, який опосередковує зв'язок автентифікаційного сервера з іншими інформаційними системами, що потребують перевірки ідентичності користувача. Цей компонент, як правило, реалізується через API, зокрема REST або SOAP, та має підтримувати стандарти автентифікації й авторизації, зокрема OAuth 2.0, OpenID Connect або SAML 2.0.

Сукупна архітектура може бути реалізована в централізованій або розподіленій формі. У централізованому варіанті всі ключові компоненти розміщуються у межах єдиного домену довіри, що спрощує адміністрування, але водночас створює єдиний вузол відмови. Розподілені архітектури, натомість, дозволяють географічне масштабування і розподіл навантаження, однак потребують складнішої координації та засобів забезпечення узгодженості ключової інформації між вузлами.

Реальні приклади впровадження архітектур OTP-систем можна спостерігати в рамках широковживаних сервісів двофакторної автентифікації. Так, Google Authenticator, Microsoft Authenticator, FreeOTP та аналогічні мобільні застосунки, що використовують TOTP, реалізують генерацію OTP локально, без потреби підключення до мережі, що відповідає принципам офлайн-автентифікації. З іншого боку, системи типу DUO Security або Authy застосовують підхід, що поєднує OTP з push-повідомленнями, тим самим розширюючи архітектуру автентифікації до багатокомпонентної системи перевірки з додатковими каналами сповіщення та зворотного зв’язку.

У межах корпоративного середовища інтеграція OTP-систем із Active Directory, LDAP, Radius або іншими службами каталогів передбачає впровадження проміжних шлюзів (gateway), які одночасно виконують автентифікацію за допомогою OTP та передають атрибути ідентифікації на подальшу авторизацію у прикладних системах.

У контексті забезпечення масштабованості архітектур OTP-систем у розподілених середовищах критичним є питання узгодження таймінгу між клієнтськими пристроями та серверною інфраструктурою, особливо при

реалізації TOTP-алгоритму. Похибка синхронізації, навіть у межах кількох секунд, може призвести до відмови в автентифікації, що зумовлює необхідність застосування механізмів допустимого вікна часу (time window) з можливістю його адаптивного налаштування. Для ефективної роботи TOTP-систем у розподілених архітектурах критично важливо забезпечити точну синхронізацію часу між усіма компонентами, або ж використовувати механізми "вікна толерантності" для компенсації можливих відхилень, що є ключовим для підтримки надійності та доступності сервісу автентифікації [21, 22]. Серверна логіка при цьому реалізується так, щоб перевірка OTP охоплювала не лише поточну часову мітку, але й обмежену кількість попередніх та наступних інтервалів.

З метою підвищення відмовостійкості й доступності сервісу автентифікації, архітектура систем OTP часто розгортається у вигляді кластеризованих серверних вузлів із горизонтальним масштабуванням. У такому випадку критичні елементи, зокрема сховище секретних ключів, повинні синхронізуватись через заздалегідь визначений механізм реплікації або централізоване сховище типу Vault, що підтримує апаратне шифрування та контроль доступу на основі політик. Це дозволяє не лише уникати дублювання конфігурацій, але й запобігати розсинхронізації між екземплярами серверів під навантаженням.

У разі реалізації OTP-систем у хмарних середовищах (наприклад, AWS, Azure або GCP), до архітектури додаються елементи безперервного моніторингу, автоматизованого розгортання, масштабування на вимогу (auto-scaling), а також використання керованих сервісів для зберігання облікових даних (наприклад, AWS Secrets Manager). Розгортання OTP-систем у хмарних середовищах вимагає особливої уваги до безпеки API-інтерфейсів та ізоляції обчислювальних ресурсів, оскільки компрометація одного компонента може вплинути на всю систему. Інтеграція з інструментами моніторингу та SIEM-системами стає ключовою для виявлення аномальної активності та оперативної реакції на інциденти безпеки. В таких випадках основна увага приділяється не лише

коректності криптографічних обчислень, а й ізоляції середовищ виконання, захисту API-інтерфейсів і виявленню аномальної активності через інтеграцію з системами SIEM [23, 24].

Ще одним важливим аспектом архітектури є процес початкового розгортання і ініціалізації ключа. Ініціалізація передбачає формування унікального секретного ключа, його прив'язку до конкретного користувача, а також захищене передання цього ключа на клієнтський пристрій. У практиці застосовуються різні методи розповсюдження ключа: сканування QR-коду, передача через захищений канал зв’язку або генерація безпосередньо на пристрої з подальшим імпортом публічного ідентифікатора до сервера.

У зв’язку з цим критично важливо передбачити механізм повторного завантаження ключа у разі втрати пристрою. Надмірна складність або відсутність такого механізму часто призводить до неможливості відновлення доступу до облікового запису, що суперечить принципам доступності та зручності користування. У корпоративних середовищах ця проблема частково вирішується через централізовані засоби керування пристроями (MDM), що дозволяють здійснювати ротацію ключів, скидання автентифікаційного стану або повторну ініціалізацію токенів без безпосереднього втручання користувача.

Нарешті, у процесі проєктування архітектури сучасної системи автентифікації на основі одноразових паролів доцільним є включення компонентів аналітики та адаптивної перевірки — зокрема, поведінкових шаблонів користувача, часових аномалій, аналізу геолокацій та пристроїв доступу. Хоча такі функції виходять за межі базової архітектури OTP-системи, вони нерозривно пов’язані з ефективністю виявлення зловмисної активності та загальною стійкістю системи до атак типу "man-in-the-middle" або фішингу.

Загальна архітектура системи автентифікації на основі одноразових паролів повинна забезпечувати баланс між криптографічною стійкістю, продуктивністю, надійністю та зручністю інтеграції з існуючими IT-інфраструктурами. Незважаючи на зовнішню простоту механізму одноразового пароля, практична реалізація його архітектури вимагає дотримання складного

комплексу умов, пов’язаних із безпекою зберігання ключів, точністю синхронізації, захистом каналів передачі даних та управлінням життєвим циклом автентифікаційних токенів.

2.5 Відомі вразливості систем автентифікації на основі одноразових паролів

Незважаючи на загальновизнану ефективність одноразових паролів як механізму зниження ризику компрометації облікових даних, практична реалізація систем OTP супроводжується низкою технічних, процедурних і криптографічних вразливостей. Значна частина цих вразливостей зумовлена або недосконалістю інтеграції механізмів OTP у загальну інфраструктуру автентифікації, або помилками конфігурації, а також особливостями поведінки користувачів у контексті соціальної інженерії. Оцінка таких вразливостей має ґрунтуватися на ретельному аналізі моделей загроз, в яких атакувальник потенційно має можливість перехоплювати, відновлювати або передбачати OTP-паролі у межах обмеженого часового вікна, або маніпулювати каналами передачі автентифікаційних повідомлень.

Однією з критичних точок у системах, що використовують алгоритм TOTP, залишається проблема так званого перехоплення в реальному часі. У випадках, коли OTP надсилається через незахищені або частково захищені канали, зокрема через SMS або електронну пошту, ймовірність перехоплення пароля суттєво зростає. Застосування протоколів безпеки на транспортному рівні (TLS) не гарантує захисту від атак на рівні кінцевих пристроїв, таких як інфіковані мобільні клієнти або браузери з розширеннями, що збирають дані. Відомими прикладами таких атак є застосування шкідливого ПЗ типу banking trojans, що інфікує мобільні телефони, перехоплює SMS з OTP і пересилає їх на сервери зловмисника до завершення сеансу автентифікації.

Іншою поширеною вразливістю є можливість синхронізованого фішингового нападу, коли атакувальник створює підроблену сторінку

автентифікації, що миттєво передає введений користувачем OTP на легітимний сервер у реальному часі, тобто до завершення терміну його дії. Такий тип атаки, відомий як "real-time phishing" або "adversary-in-the-middle" (AiTM) фішинг, дозволяє обійти навіть надійні OTP-системи, оскільки зловмисник діє як проксі між користувачем і сервером, перехоплюючи і миттєво використовуючи дійсний OTP. Це підкреслює, що лише OTP недостатньо для повного захисту від складних фішингових схем [25]. Подібні атаки вже неодноразово демонстрували свою ефективність проти платформ, які використовують TOTP- або HOTP-алгоритми без додаткового захисту у вигляді прив’язки до конкретного пристрою або контексту запиту.

Значну небезпеку становить атака повторного використання, що можлива у системах, де одноразові паролі, згенеровані за HOTP, приймаються упродовж декількох наступних кроків через неправильно налаштоване або надто велике вікно зсуву лічильника. Якщо автентифікаційний сервер допускає використання OTP з кількома кроками наперед без інвалідації вже використаних значень, атакувальник може скористатися записаним або перехопленим OTP-кодом пізніше, коли сервер знову розпізнає його як дійсний. Таке явище є наслідком некоректного керування станом лічильника або відсутності механізму відмітки використаних OTP-кодів.

Крім того, до числа уразливостей можна віднести невдалу реалізацію генерації секретного ключа. У низці відомих інцидентів використовувалися криптографічно нестійкі або передбачувані генератори псевдовипадкових чисел для створення OTP-секретів, що уможливлювало брутфорс або генерацію еквівалентних токенів зловмисником. Додаткові ризики виникають у випадку повторного використання одного і того ж секретного ключа для декількох облікових записів, що створює передумови для масових компрометацій при розкритті одного токена.

У реальному середовищі були зафіксовані випадки використання технік атак на бічні канали для отримання OTP, зокрема у випадках використання апаратних токенів. Наприклад, пристрої, які виводять OTP на дисплей, можуть

бути піддані відеоспостереженню або зчитуванню даних через електромагнітний витік. Хоча такі атаки мають низький рівень імовірності у масовому застосуванні, вони залишаються релевантними для об'єктів критичної інфраструктури та високозахищених систем.

Особливу увагу слід приділяти ситуаціям, коли уразливості виникають не безпосередньо в OTP-механізмі, а в його взаємодії з іншими компонентами системи автентифікації. Наприклад, при інтеграції OTP з LDAP або SSO-рішеннями можуть виникати проблеми з некоректною обробкою помилок автентифікації, що дозволяє зловмиснику здійснити enumeration-атаку або обійти частину процедур аутентифікації через помилки логіки.

Неможливо залишити поза увагою конфігураційні помилки, що виникають при розгортанні OTP-рішень у корпоративному середовищі, зокрема, недостатнє обмеження кількості спроб введення одноразового пароля. У випадках, коли сервер автентифікації не впроваджує обмеження на частоту або кількість неправильних запитів, атакувальник може здійснити так звану brute force-атаку навіть у межах відносно короткого періоду життя OTP. При цьому, через обмежений розмір OTP (наприклад, шість цифр), загальна кількість можливих комбінацій становить 10^6, що з використанням автоматизованих скриптів і за відсутності захисних механізмів типу rate limiting, стає досяжним обсягом пошуку.

Вразливості, пов’язані з людським фактором, продовжують залишатися одним із найсерйозніших ризиків. Типовими сценаріями експлуатації слабкостей поведінки користувача є атаки з використанням соціальної інженерії, при яких жертва добровільно передає OTP-код, вважаючи, що взаємодіє із легітимною службою підтримки або корпоративним адміністратором. Навіть якщо технічна реалізація системи OTP є бездоганною і відповідає криптографічним стандартам, успіх атаки соціальної інженерії повністю нівелює ці технічні переваги. Це підкреслює, що навчання користувачів основам кібербезпеки є не менш важливим, ніж технологічні заходи захисту [1].

Окрему категорію становлять атаки на цілісність пристроїв-генераторів OTP, зокрема у випадках, коли OTP-генератором виступає мобільний додаток, встановлений на загальнодоступному або компрометованому пристрої. Із врахуванням того, що більшість сучасних мобільних клієнтів для OTP використовують загальнодоступні бібліотеки з відкритим кодом, потенційне впровадження шкідливого коду через модифіковані версії додатків, встановлені зі сторонніх джерел, може призводити до повного компрометування автентифікаційного процесу. Аналогічним чином загрозу становлять експлойти, що використовують вразливості в операційних системах мобільних пристроїв для отримання доступу до області пам’яті, де зберігається секретний ключ OTP.

Серед відомих практичних інцидентів варто згадати атаки на платформи дистанційного банкінгу, де OTP-паролі, що надсилалися через SMS, перехоплювалися шкідливими додатками типу Android.Bankosy або Anubis, які не лише перехоплювали повідомлення, а й ініціювали фонові транзакції через захоплений браузер або API мобільного банкінгу. Такі атаки демонструють, що розміщення OTP у зовнішніх каналах зв’язку з користувачем створює додаткову площину атаки, незалежну від стійкості криптографічного алгоритму.

Крім того, особливу увагу привертають ситуації, коли системи OTP не здійснюють криптографічну перевірку контексту запиту, внаслідок чого стає можливою атака із повторним використанням коду в іншому сеансі або для іншої операції. Відсутність прив'язки OTP до конкретної транзакції або операції є значною вразливістю, оскільки дозволяє зловмиснику використати перехоплений код для несанкціонованих дій. Для посилення безпеки необхідно, щоб OTP був "підписаний" контекстом запиту, що запобігатиме його використанню в інших сценаріях [26, 27]. Уразливість полягає у відсутності зв’язку між OTP-кодом і конкретною дією користувача, наприклад — авторизацією транзакції певного розміру або типу. Це уможливлює використання підставленого або підробленого запиту, на який легітимний користувач генерує OTP, що потім застосовується зловмисником до своєї власної, зміненої операції.

Ще одним фактором, що сприяє потенційній уразливості, є відсутність механізму захищеного резервного копіювання та ротації секретів, які використовуються в алгоритмах HOTP і TOTP. У випадку втрати або компрометації секретного ключа (наприклад, через доступ до резервної копії пристрою, що не була зашифрована), система втрачає здатність забезпечувати однозначність автентифікації. У деяких реалізаціях, зокрема в системах з обмеженим бюджетом або в застарілих корпоративних середовищах, відсутні політики регулярної зміни ключів, що суперечить загальноприйнятим принципам криптографічної гігієни [28].

Варто також розглядати вразливості, які проявляються у багатофакторних схемах, де OTP застосовується як другий фактор. У таких випадках, якщо перший фактор (наприклад, пароль) є скомпрометованим, а другий — OTP — уразливий до згаданих вище атак, загальна ефективність автентифікаційного процесу знижується до рівня найслабшого компонента [29]. Непоодинокими є випадки, коли OTP-другий фактор реалізується формально, без додаткового контексту, що дозволяє атакувальнику, який має контроль над першою фазою входу, ефективно експлуатувати систему.

Проблеми виникають також унаслідок невдалого впровадження міжплатформних або мультидоменних рішень, коли сервер автентифікації довіряє результатам генерації OTP, отриманим із зовнішніх джерел або неавторитетних суб’єктів. При відсутності механізмів перевірки сертифікованих компонентів або каналів обміну ключами з постачальниками ідентифікації, виникає ризик реалізації атак на основі моделі «людина посередині», які можуть бути майже непомітними для кінцевого користувача [30, 31].

Таким чином, відомі вразливості, характерні для OTP-систем, охоплюють як криптографічні аспекти, пов’язані із генерацією, синхронізацією та валідацією OTP-кодів, так і широке коло питань, що стосуються процедурного забезпечення, взаємодії між компонентами та поведінки користувачів. Актуальність аналізу цих загроз особливо зростає у світлі постійного підвищення рівня автоматизації атак, їхньої орієнтованості на конкретні

реалізації та глибокої інтеграції у цифрові платформи, що передбачає необхідність подальшого розвитку захисних механізмів і удосконалення архітектурних підходів у відповідних системах автентифікації.

Висновки за розділом 2

У межах другого розділу було здійснено комплексний аналіз концептуальних, алгоритмічних і архітектурних засад побудови систем автентифікації, що функціонують на основі механізмів одноразових паролів, зокрема на основі HMAC та часових маркерів. Послідовне дослідження відповідних алгоритмів дозволило визначити відмінності у підходах до генерації паролів, способів синхронізації між клієнтськими і серверними компонентами, а також межі ефективності кожного з методів у контексті реального використання в інформаційних системах із підвищеними вимогами до стійкості автентифікації.

Зосередження уваги на структурних характеристиках архітектур OTP-систем, включно з особливостями їх взаємодії з іншими елементами безпекової інфраструктури, дало змогу сформувати цілісне уявлення про роль одноразових паролів у загальному механізмі ідентифікації та контролю доступу. При цьому встановлено, що, попри значні переваги у протидії повторному використанню автентифікаційних даних та мінімізації ризику компрометації у разі перехоплення, системи цього класу мають обмеження, зумовлені як технічними, так і організаційними чинниками.

У межах розділу також проведено критичний огляд актуальних вразливостей, властивих OTP-системам у їх сучасних реалізаціях. У результаті було встановлено, що, незважаючи на застосування криптографічно стійких алгоритмів, ефективність автентифікаційного процесу значною мірою залежить від належного управління ключами, контекстної прив’язки OTP до дій користувача, а також від стійкості кінцевих пристроїв і рівня їх захисту від стороннього втручання. Виявлено, що велика частина загроз виникає не через

уразливість самих алгоритмів, а через похибки у впровадженні, недоліки в експлуатаційній моделі або зловживання довірою з боку користувачів.

Загалом отримані результати окреслюють як переваги, так і обмеження одноразових паролів у ролі ключового елемента автентифікації, що дозволяє сформулювати вимоги до удосконалення як самих механізмів генерації та перевірки OTP, так і супутніх процедур забезпечення безпеки.

РОЗДІЛ 3
РОЗРОБКА СИСТЕМИ АВТЕНТИФІКАЦІЇ НА ОСНОВІ ОДНОРАЗОВИХ ПАРОЛІВ

3.1 Вибір інструментальних засобів

У процесі реалізації системи автентифікації користувачів на основі одноразових паролів особливу увагу було зосереджено на виборі інструментальних засобів, здатних забезпечити належний рівень безпеки, масштабованість та модульність розроблюваної програмної архітектури. Вибір конкретного стеку технологій був обумовлений як функціональними вимогами до системи, так і необхідністю забезпечення зручної подальшої підтримки, розширення та інтеграції з іншими інформаційними сервісами.

Основою для побудови серверної частини системи було обрано фреймворк Flask, який реалізується мовою програмування Python. Застосування даного фреймворку обґрунтовано його легкістю у розгортанні, підтримкою модульної структури, а також широким спектром бібліотек для роботи з криптографічними функціями, протоколами автентифікації та взаємодії з базами даних. Серед основних переваг Flask, що стали вирішальними у контексті цієї розробки, слід виокремити мінімалістичний підхід до побудови маршрутизації запитів, високу швидкість обробки даних у поєднанні з можливістю налаштування кожного етапу взаємодії між клієнтом і сервером, а також легку інтеграцію сторонніх модулів.

Вибір операційної системи Ubuntu як основного середовища виконання системи був зумовлений стабільністю даного дистрибутива Linux, його широким використанням у виробничих середовищах, а також підтримкою великої кількості інструментів для забезпечення інформаційної безпеки. Ubuntu дозволяє ефективно працювати з такими компонентами, як менеджери пакетів (apt), віртуальні середовища Python (venv), служби автоматизації (systemd), а також

надає гнучкий інструментарій для адміністрування прав доступу, конфігурації мережі та логування подій.

Для генерації та перевірки одноразових паролів у системі було обрано бібліотеку PyOTP, яка підтримує реалізацію двох основних алгоритмів — HOTP та TOTP. Зазначена бібліотека є відкритим і підтримуваним рішенням, яке відповідає специфікаціям RFC 4226 та RFC 6238, і дозволяє гнучко керувати параметрами генерації OTP, зокрема інтервалом часу для TOTP, кількістю кроків для HOTP, використанням власного секретного ключа тощо. Реалізація перевірки одноразових кодів здійснюється без потреби зберігати коди в базі даних, що істотно знижує ризики компрометації інформації.

В якості системи керування базами даних обрано SQLite, що забезпечує достатній рівень надійності та продуктивності для автономного розгортання та тестування системи в умовах обмеженого навантаження. Незважаючи на те, що SQLite не є оптимальним вибором для великомасштабних продакшн-рішень, її використання на етапі розробки і тестування дозволяє скоротити час налаштування, уникнути конфігураційної складності серверних рішень на зразок PostgreSQL або MySQL, а також зосередитись на реалізації функціональних складових системи автентифікації. У подальшому перехід до масштабованої СКБД не становитиме суттєвих труднощів, оскільки структура моделі даних була з самого початку проєктована з урахуванням ORM-підходу на базі SQLAlchemy.

Для реалізації інтерфейсу взаємодії з користувачем було застосовано шаблонізатор Jinja2, який дозволяє створювати динамічні HTML-сторінки з логікою відображення стану автентифікації, повідомлень та інструкцій для користувача. Крім того, в межах клієнтської частини використовувалися CSS-фреймворк Bootstrap та бібліотека FontAwesome, що дозволили реалізувати адаптивний і візуально привабливий інтерфейс із підтримкою інтерактивних елементів управління. На (рисунку 3.1) наведено фрагмент інтерфейсу сторінки входу, який демонструє логіку вибору способів отримання одноразового пароля, а також повідомлення про стан сесії автентифікації.

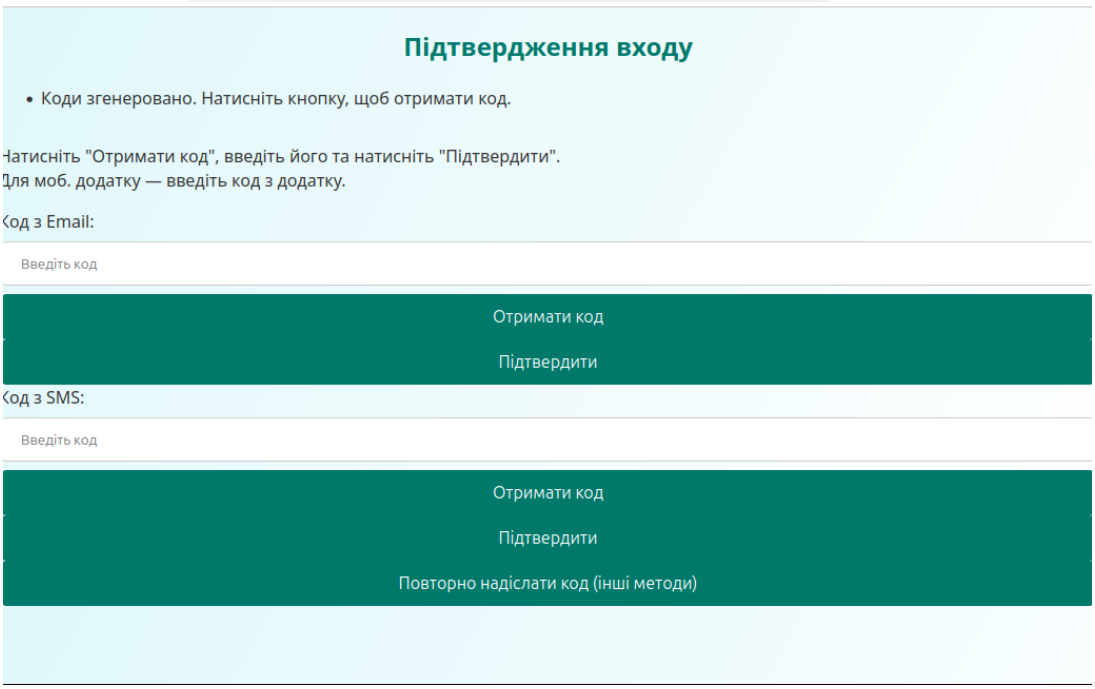

Рисунок 3.1. – Інтерфейс програми

Особливу увагу було приділено вибору інструментів для надсилання OTP-кодів через різні канали доставки. Для реалізації надсилання кодів на електронну пошту використано бібліотеку smtplib разом із сервером Postfix, налаштованим на локальну доставку пошти. Такий підхід дозволяє уникнути використання сторонніх сервісів і забезпечує контроль над передачею повідомлень, у тому числі можливість логування всіх транзакцій SMTP. Надсилання кодів через SMS реалізовано шляхом інтеграції з API національного мобільного оператора, що дозволяє досягти необхідного рівня оперативності, надійності та відповідності вимогам локального законодавства щодо зберігання персональних даних.

На завершення варто зауважити, що обраний набір інструментальних засобів забезпечив можливість побудови системи автентифікації, яка відповідає сучасним вимогам до захисту даних, гнучкості розширення функціоналу, простоти супроводу та візуальної інтеграції у вже наявну інфраструктуру сервісів.

3.2 Структура і логіка програмної реалазації

З урахуванням попередньо обґрунтованого вибору інструментальних засобів, програмна реалізація системи автентифікації на основі одноразових паролів була здійснена з дотриманням принципів модульності, масштабованості та безпечного зберігання й обробки конфіденційних даних користувачів. Реалізована архітектура побудована за схемою клієнт-сервер, у межах якої функціональні компоненти чітко розмежовані як за рівнями доступу, так і за відповідальністю за окремі процеси: генерацію, доставку, перевірку та валідацію одноразових паролів, збереження подій автентифікації, а також взаємодію з алгоритмами оцінки ризику.

Під час проектування було визначено, що для покращення безпеки та ефективності системи, використовувати на вибір два способи доставки повідомлень з одноразовим паролем. Відповідно для цього з’явилась необхідність створення вікна для реєстрації користувачів. Отже, для того, щоб програма коректно працювала, була забезпечена ідентифікація користувача за іменем(логіном). Загальна схема автентифікації користувача наведена на (рисунку 3.2).

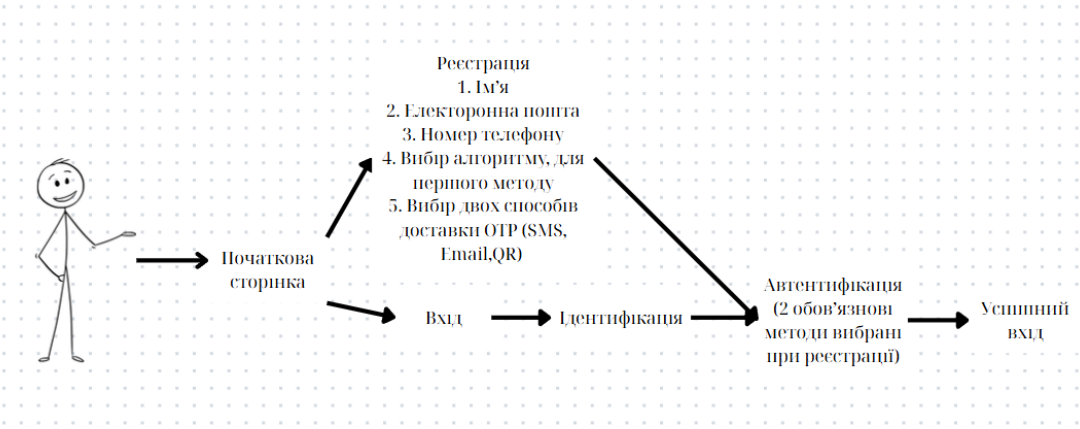

Рисунок 3.2. – Загальна схема роботи системи авентифікації користувачів

Проєкт організовано відповідно до концепції багаторівневої структури Flask-додатку, де центральним елементом є ініціалізаційний модуль __init__.py, відповідальний за конфігурацію Flask-додатку, реєстрацію маршрутів, ініціалізацію бази даних через SQLAlchemy та запуск розширень безпеки (наприклад, Flask-Limiter для контролю частоти запитів). Модулі логіки, представлені окремими файлами у відповідних директоріях, забезпечують сувору ізоляцію відповідальностей. Зокрема, взаємодія з базою даних — у models.py, а функції з доставки повідомлень — у модулі notifications.py. Усі взаємозв'язки між компонентами реалізовано із дотриманням принципів інверсії залежностей та мінімізації міжмодульних зв'язків, що дозволяє забезпечити легкість подальшого масштабування або заміни окремих компонентів.

На (рисунку 3.3) наведено загальну структуру розробленого застосунку, яка відображає логічне розмежування функціональних зон відповідальності. Візуалізована структура ілюструє, яким чином забезпечується інкапсуляція функціональності.

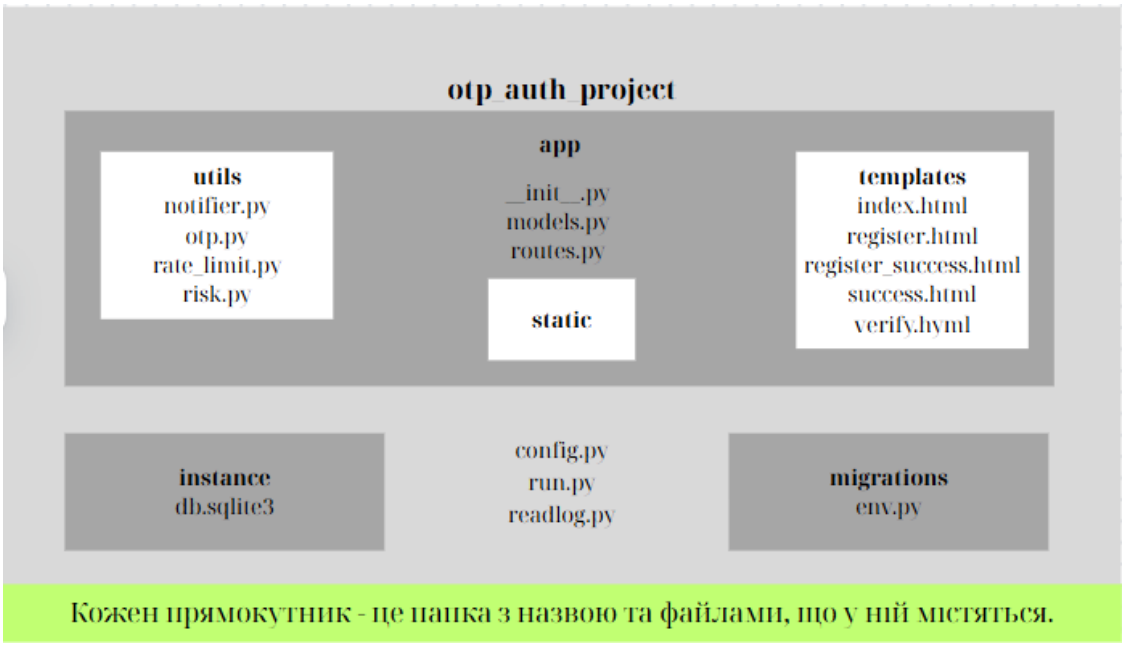

Рисунок 3.3. – Структура проєкту системи автентифікації користувачів

Функціональна логіка автентифікації побудована на умовному поділі етапів автентифікації на чотири фази: первинна ініціація сесії користувача,

генерація OTP відповідно до обраного методу (електронна пошта, SMS або QR), доставка одноразового пароля, перевірка коду користувачем, після чого здійснюється логування результату. Всі транзакції супроводжуються записом до таблиці auth_events бази даних, де фіксуються атрибути часу, IP-адреса, рівень ризику, статус автентифікації, використані методи тощо.

Використання об'єктно-реляційного відображення дозволяє реалізувати обробку даних бази на рівні об’єктів, уникаючи ручного формування SQL-запитів, що, у свою чергу, підвищує рівень захищеності від ін’єкцій та спрощує підтримку коду. Кожен об’єкт, що представляє сутність чітко описаний відповідною моделлю, де визначені як поля, так і правила їх валідації та взаємозв’язку.

Варто акцентувати увагу на механізмах перевірки OTP, що реалізовані з урахуванням підтримки як TOTP, так і HOTP протоколів, залежно від типу доставки. Зокрема, функція generate_totp(secret) формує код за допомогою бібліотеки pyotp із застосуванням системного часу як аргументу генерації, тоді як generate_hotp(secret, counter) використовує лічильник, збережений у базі даних, і автоматично інкрементує його після кожного успішного запиту. Обидва підходи інтегровані через єдину абстрактну обгортку, що дозволяє динамічно обирати відповідний алгоритм залежно від параметрів користувача.

Крім того, реалізація сторінки логіну передбачає адаптивний інтерфейс, що змінюється залежно від стадії процесу автентифікації. Після введення адреси електронної пошти або номера телефону, система динамічно підвантажує відповідні кнопки доставки OTP, дозволяючи користувачу обрати будь-які два із трьох запропонованих способів підтвердження особи. У випадку успішної верифікації першого коду, система автоматично блокує повторну доставку за цим каналом, надаючи користувачу можливість скористатись альтернативними каналами, що запобігає як зловживанням, так і дублюванню.

На (рисунку 3.4) наведено фрагмент інтерфейсу користувача у момент очікування підтвердження другого фактора, де реалізовано інтерактивне візуальне відображення статусу кожного з підтверджень.

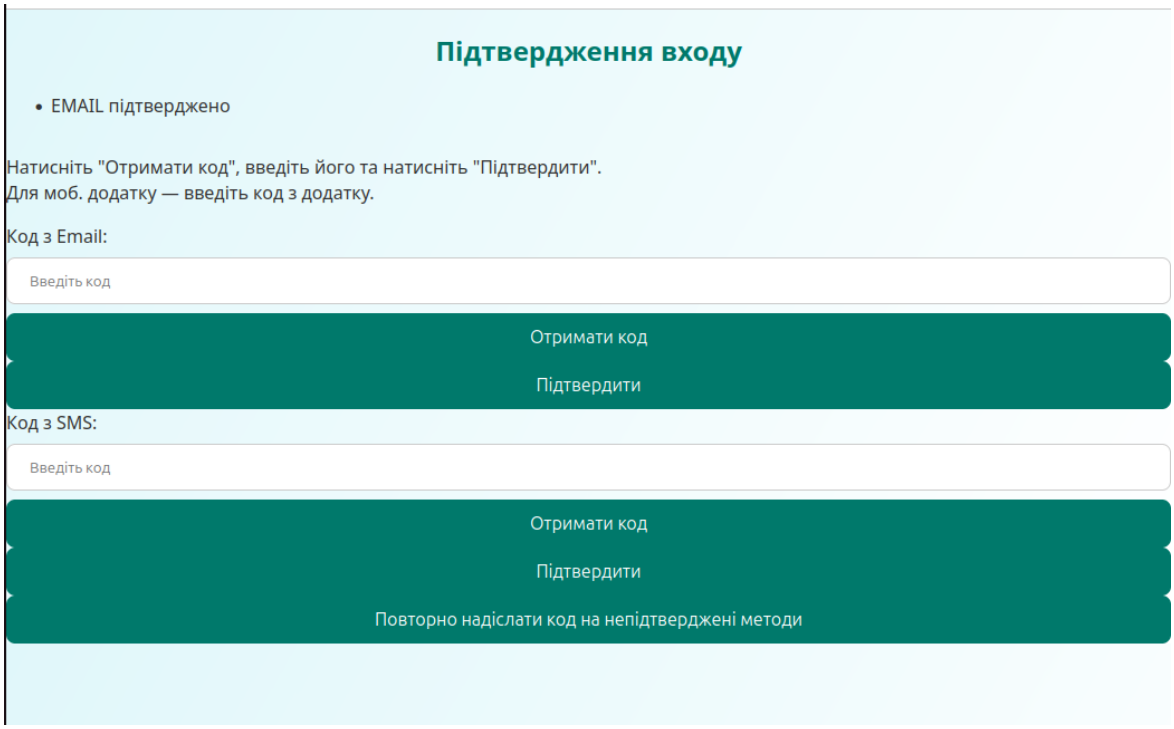

Рисунок 3.4. – Інтерфейс користувача на етапі верифікації OTP, з одним підтвердженим методом автентифікації

Контроль кількості спроб авторизації здійснюється за допомогою механізму Flask-Limiter, конфігурованого на обмеження кількості запитів за IP-адресою та обліковим записом користувача. У випадку перевищення порогового значення запитів система автоматично блокує відповідний маршрут і реєструє подію в журналі інцидентів.

Функціонал розсилки реалізовано в окремому модулі notifications.py, де сконцентровані функції для відправлення електронних листів та SMS-повідомлень. Надсилання OTP через електронну пошту базується на стандартному SMTP-з'єднанні, тоді як відправка через SMS реалізована шляхом взаємодії з API мобільного оператора Київстар. Це дозволяє зменшити залежність від сторонніх сервісів та забезпечити інтеграцію із локальними телеком-провайдерами.

Файл risk.py, який буде детально проаналізовано у наступному підрозділі, реалізує механізми оцінки рівня ризику на основі сукупності параметрів сеансу автентифікації, включно з геолокацією, часом, типом пристрою, історією

верифікацій та частотою запитів. Результати оцінювання використовуються для прийняття рішень щодо допустимості входу або необхідності додаткової верифікації.

Інтерфейсна частина реалізована засобами шаблонів Jinja2 та представлена у директорії templates, що містить сторінки index.html, register.html, verify.html, а також окремі елементи інтерфейсу, які забезпечують взаємодію користувача із системою в зручний та інтуїтивно зрозумілий спосіб. Зовнішній вигляд сторінок підтримується файлами з директорії static, що включає таблиці стилів CSS, JavaScript-компоненти та ілюстрації.

Сценарій запуску серверної частини визначено у файлі run.py, який забезпечує ініціалізацію застосунку на основі конфігураційного модуля config.py та запускає сервер розробки або продакшн-середовище, залежно від змінних середовища.

Наявність розділеної архітектури, чітко ізольованих модулів та підтримки шаблонного підходу в реалізації інтерфейсу значно спрощує масштабування системи, введення нових каналів доставки OTP, реалізацію додаткових рівнів перевірки.

3.3 Програмна реалізація модулю генерації та перевірки OTP

У межах розробленої системи автентифікації користувачів модуль генерації та перевірки одноразових паролів виконує роль ядра процесу безпечного підтвердження особи, забезпечуючи криптографічну стійкість та гарантовану часову актуальність автентифікаційного маркера. Усі функції цього модуля реалізовано відповідно до специфікацій RFC 4226 (HOTP) та RFC 6238 (TOTP), що забезпечує сумісність із популярними програмними аутентифікаторами, зокрема Google Authenticator, FreeOTP, Microsoft Authenticator тощо.

Програмну реалізацію здійснено у межах файлів otp.py та routes.py, розташованого в межах логічного блоку app/, що відповідає за реалізацію

прикладної логіки. Лістинг даних файлів, можна переглянути у Додатку А. Сам модуль побудовано з урахуванням принципів модульності та повторного використання коду, що уможливлює масштабування функціоналу без втрати узгодженості з рештою компонентів системи.

Генерація одноразового пароля здійснюється через використання бібліотеки pyotp, яка реалізує основні алгоритми OTP відповідно до вказаних стандартів. Параметри ініціалізації генератора включають секретний ключ, що генерується під час реєстрації користувача. Формування секрету здійснюється із використанням криптографічно безпечного генератора випадкових байтів, що зберігається у базі даних у вигляді закодованого рядка.

У випадку з TOTP алгоритмом, під час генерації коду використовується поточний часовий інтервал, тривалість якого визначено фіксованим параметром (наприклад, 30 секунд). При перевірці OTP-секвенції, введеної користувачем, система проводить синхронізований аналіз актуального часового вікна, а також суміжних вікон для компенсації можливої розсинхронізації. Результати перевірки зберігаються в лог-файлі.

У загальному вигляді, структура логіки генерації OTP включає такі етапи: ініціалізація генератора на основі секретного ключа, генерація коду відповідно до обраного алгоритму (TOTP або HOTP), збереження події генерації у базі даних для забезпечення трасування, надсилання коду користувачу через один або кілька обраних каналів доставки (електронна пошта, SMS, QR).

Перевірка коду передбачає аналогічну ініціалізацію генератора з використанням збереженого секрету. Під час верифікації у випадку TOTP система генерує діапазон допустимих кодів на підставі часових вікон і виконує порівняння введеного значення. Для HOTP, що базується на лічильнику, додатково реалізовано механізм поступового збільшення індексу генерації, що синхронізується між сервером і клієнтом у разі підтвердження валідного коду. У випадку кількох невдалих спроб верифікації система переходить до режиму блокування тимчасового доступу відповідно до параметрів, визначених у конфігурації. Ці механізми буде детально розглянуто у наступному підрозділі.

На (рисунку 3.5) подано узагальнену схему логіки генерації та перевірки одноразових кодів у рамках функціонування серверної частини системи автентифікації.

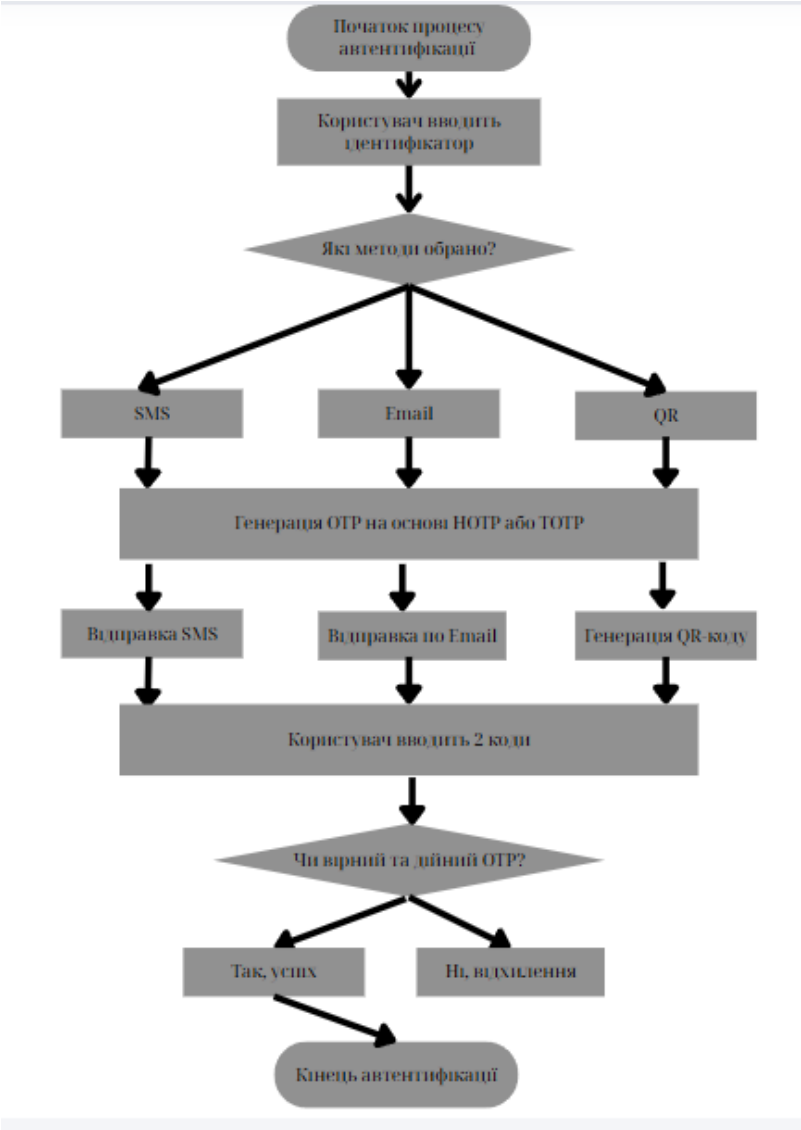

Рисунок 3.5. – Схема логіки роботи модуля генерації та перевірки OTP

У загальному вигляді, структура логіки генерації OTP включає такі етапи: ініціалізація генератора на основі секретного ключа, генерація коду відповідно до обраного алгоритму (TOTP або HOTP), збереження події генерації у базі даних для забезпечення трасування, надсилання коду користувачу через один або кілька обраних каналів доставки (електронна пошта, SMS, QR).

Перевірка коду передбачає аналогічну ініціалізацію генератора з використанням збереженого секрету. Під час верифікації у випадку TOTP система генерує діапазон допустимих кодів на підставі часових вікон і виконує порівняння введеного значення. Для HOTP, що базується на лічильнику, додатково реалізовано механізм поступового збільшення індексу генерації, що синхронізується між сервером і клієнтом у разі підтвердження валідного коду. У випадку кількох невдалих спроб верифікації система переходить до режиму блокування тимчасового доступу відповідно до параметрів, визначених у конфігурації. Ці механізми буде детально розглянуто у наступному підрозділі.

Важливою особливістю реалізації є підтримка трьох незалежних каналів доставки OTP – електронної пошти, SMS та QR-коду – кожен з яких реалізовано як окрема функціональна гілка у процесі генерації. Такий підхід забезпечує гнучкість багатофакторної автентифікації та дозволяє користувачу обирати найзручніший або найменш ризикований спосіб підтвердження. На (рисунку 3.6) наведено фрагмент клієнтського інтерфейсу, що демонструє можливість вибору методу доставки одноразового коду.

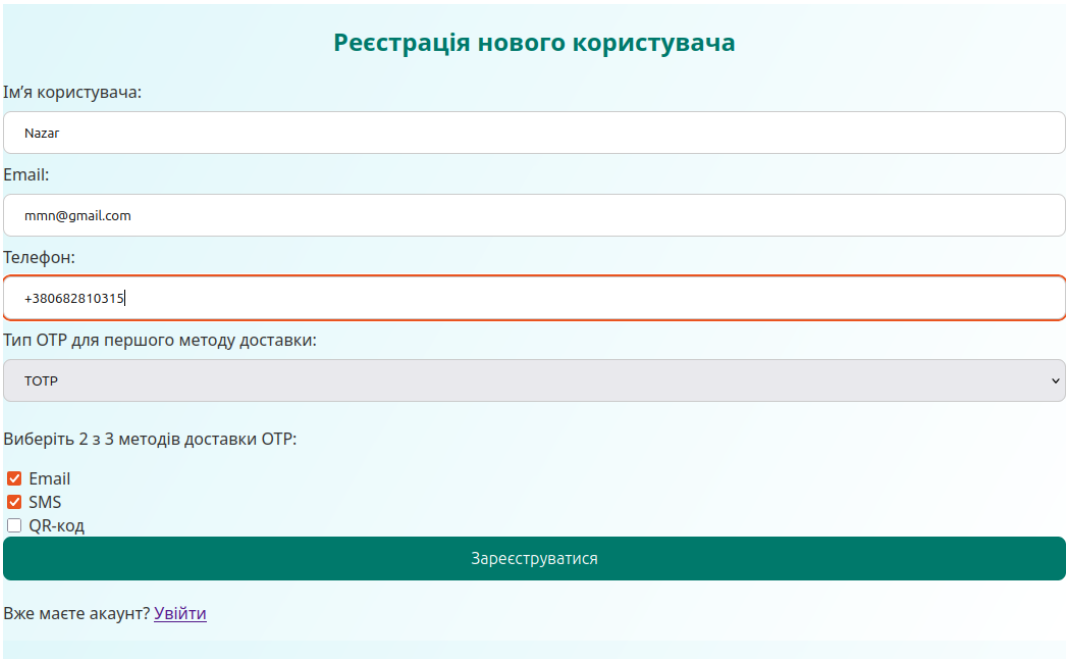

Рисунок 3.6. – Відображення доступних способів доставки OTP-коду на інтерфейсі користувача

Для QR-коду реалізовано механізм генерації токену у форматі otpauth URI з подальшим його кодуванням у графічне зображення з використанням бібліотеки qrcode. Згенероване зображення інтегрується у шаблон register.html, що забезпечує зручне сканування мобільним застосунком без потреби ручного введення секретного ключа. Візуальне представлення реалізованого функціоналу наведено на (рисунку 3.7).

Рисунок 3.7. – Відображення QR-коду з одноразовим ключем під час реєстрації користувача

У межах логіки забезпечення безперервності автентифікаційного процесу реалізовано контроль таймінгу дії коду. Для алгоритму TOTP часове вікно визначається на рівні генератора та становить 30 секунд. Перевірка виконується в межах допустимого діапазону ±1 вікно, що дозволяє компенсувати невеликі відхилення системного часу між клієнтом і сервером. Цей параметр конфігурується через файл config.py, а фактичний інтервал може бути адаптований у процесі використання.

Усі наведені реалізації модулю OTP є повністю інтегрованими у загальну архітектуру системи, їх виклики відбуваються через маршрути, описані у файлі

routes.py. У функції обробки запитів логіка генерації коду, перевірки та обробки помилок відокремлена, що забезпечує гнучкість у реалізації механізмів адаптивного реагування на потенційні загрози.

3.4 Алгоритм адаптивного захисту

У контексті сучасних систем автентифікації, зокрема таких, що базуються на механізмах одноразових паролів, впровадження адаптивного підходу до оцінювання достовірності кожного сеансу входу набуває ключового значення. Такий підхід дозволяє автоматично враховувати змінні контексту користувача, що дає змогу істотно підвищити ефективність виявлення аномальної активності без прямого втручання адміністратора. Запропонована в рамках цієї роботи система адаптивного захисту реалізує механізм оцінювання ризику входу на основі набору поведінкових, мережевих та часових характеристик, що динамічно обчислюються для кожної спроби автентифікації.

Реалізований алгоритм побудовано на основі балової моделі, у якій кожному фактору, що впливає на ризикованість сесії, призначається певна вага у вигляді умовного числового коефіцієнта. Сукупна оцінка формується через агрегацію цих коефіцієнтів, після чого результат зіставляється з пороговим значенням, яке позначає високий рівень ризику.

У розробленій системі передбачено сім основних критеріїв оцінювання, серед яких:

- Тип пристрою: система реєструє специфіку клієнтського середовища на рівні браузера або клієнтської бібліотеки. Відомі, звичні системи (наприклад, Windows, Linux) мають знижений ризиковий коефіцієнт.
- Тип мережі: визначення, чи належить IP-адреса до внутрішньої (локальної) або загальнодоступної мережі. Останні класифікуються як більш ризикові через потенційно ненадійні канали зв’язку.

- Аномалії географічного характеру: перевіряється, чи спостерігалася дана IP-адреса в останніх сесіях користувача. Виявлення нової адреси підвищує ризик.
- Кореляція пристрою та IP: поєднання нового пристрою з новою IP-адресою розцінюється як потенційно компрометований доступ.
- Час доби: реєстрація сесії в нетиповий для користувача період (наприклад, пізньої ночі або рано вранці) спричиняє незначне збільшення оцінки ризику.
- Відхилення від середнього часу входу: за умови накопичення достатньої кількості сесій, формується профіль звичного часу активності користувача. Відхилення більш ніж на дві години інтерпретується як поведінкова аномалія.
- Аномальна частота спроб: якщо за останню годину кількість входів перевищує певне емпіричне значення, це сигналізує про потенційну автоматизовану атаку.

Результати розрахунку ризику в реальному часі дозволяють формувати три рівні відповідей системи: низький ризик, середній ризик, високий ризик з блокуванням сесії. Одночасно з цим цих три рівні відповідають певному числу балів, які отримуються, як сукупність факторів. На (рисунку 3.7) наведений файл логування у якому для кожного з входів визначений рівень ризику.

```
=== Логи автентифікації ===
/home/nazar0101/otp_auth_project/readlog.py:15: LegacyAPIWarning: The Query.get() method is considered legacy as of the 1.x serie
s of SQLAlchemy and becomes a legacy construct in 2.0. The method is now available as Session.get() (deprecated since: 2.0) (Back
ground on SQLAlchemy 2.0 at: https://sqlalche.me/e/b8d9)
  user = User.query.get(log.user_id)
[2025-06-03 20:07:37] Користувач: Nazar33, Успішно: True, IP: 127.0.0.1, Метод: qr,email, Ризик: 3.0
[2025-05-30 12:43:22] Користувач: Nazar33, Успішно: False, IP: 127.0.0.1, Метод: qr, Ризик: 3.0
[2025-05-30 12:43:05] Користувач: Nazar33, Успішно: False, IP: 127.0.0.1, Метод: qr, Ризик: 2.0
[2025-05-30 12:42:52] Користувач: Nazar33, Успішно: False, IP: 127.0.0.1, Метод: qr, Ризик: 2.0
[2025-05-30 12:41:45] Користувач: Nazar33, Успішно: False, IP: 127.0.0.1, Метод: qr, Ризик: 3.0
[2025-05-30 12:41:33] Користувач: Nazar33, Успішно: False, IP: 127.0.0.1, Метод: qr, Ризик: 3.0
[2025-05-30 12:41:17] Користувач: Nazar33, Успішно: False, IP: 127.0.0.1, Метод: qr, Ризик: 8.0
[2025-05-30 12:37:42] Користувач: Nazar22, Успішно: True, IP: 127.0.0.1, Метод: sms,email, Ризик: 7.0
[2025-05-30 02:50:32] Користувач: Nazar, Успішно: False, IP: 127.0.0.1, Метод: email, Ризик: 6.0
[2025-05-30 02:50:31] Користувач: Nazar, Успішно: False, IP: 127.0.0.1, Метод: email, Ризик: 5.0
```

Рисунок 3.8. – Логи автентифікації з визначеним рівнем ризику

З метою додаткового захисту від атак типу brute force або сплесків автоматизованих запитів (зокрема скриптових атак), до архітектури захисного модуля було включено механізм контролю частоти спроб автентифікації. Реалізований алгоритм, що враховує часові вікна, веде облік сесій на основі ідентифікатора користувача або IP-адреси, і при перевищенні певного порогу — миттєво активує обмеження. Відповідний захист системи зображений на (рисунку 3.9).

Рисунок 3.9. – Обмежений доступ через велику частоту спроб автентифікації

3.5 Тестування та оцінка ефективності системи автентифікації користувачів на основі одноразових паролів

Реалізація повнофункціональної системи автентифікації на основі одноразових паролів з підтримкою трьох незалежних каналів доставки (електронна пошта, SMS та QR-код), а також з урахуванням механізмів перевірки надійності автентифікаційної сесії, потребує всебічного тестування в умовах, максимально наближених до реальних сценаріїв використання. Основна мета цього етапу полягала у перевірці коректності реалізації всіх модулів системи, включаючи генерацію коду, його своєчасну доставку, перевірку верифікаційних даних, обробку помилок та управління сеансами взаємодії з користувачем.

На початковому етапі було проведено функціональне тестування основних сценаріїв використання. Кожен метод доставки одноразового пароля перевірявся окремо на предмет своєчасності та надійності надсилання коду, його валідації на сервері та подальшого переходу до вікна підтвердження автентифікації.

Скріншот прикладу успішного надсилання OTP-коду на електронну пошту наведено на (рисунку 3.10). Він демонструє, що система коректно ініціює формування повідомлення, надсилає його через налаштований SMTP-сервер, та інформує користувача про успішне завершення операції.

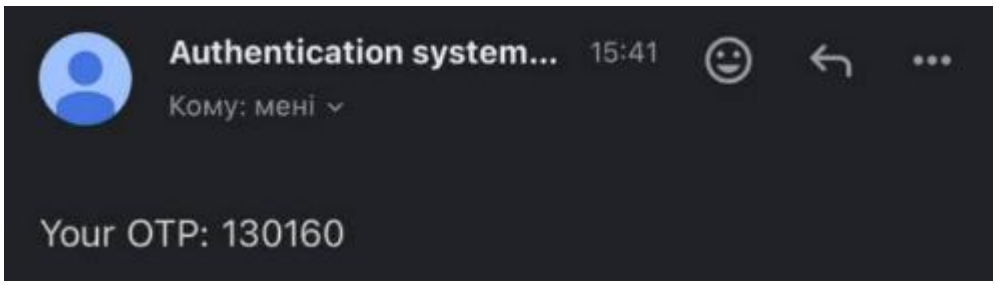

Рисунок 3.10. – Надіслане повідомлення з одноразовим кодом на електронну пошту

Надсилання одноразового пароля через SMS реалізовано через відповідне API-інтерфейс стороннього провайдера мобільного зв'язку. У процесі тестування було перевірено, що система автоматично формує запит, передає його на сервер оператора із дотриманням вимог до формату, отримує відповідь про статус доставки, та відображає результат користувачеві. Результат успішної доставки через канал SMS відображено на (рисунку 3.11).

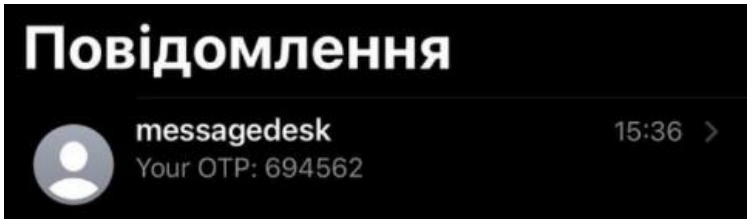

Рисунок 3.11. – Надіслане смс-повідомлення з одноразовим кодом

Варіант використання QR-коду, який інтегрується у веб-інтерфейс і підлягає зчитуванню мобільним застосунком з підтримкою TOTP, також було перевірено. На (рисунку 3.12) наведено приклад відображення OTP-коду у застосунку Google Authenticator на мобільному пристрої та користувач вводить його у відповідному полі підтвердження.

Рисунок 3.12. – Одноразовий код у додатку Google Authenticator

Загальна логіка автентифікації передбачає підтвердження принаймні двох з трьох каналів доставки. При успішному проходженні цього етапу система автоматично перенаправляє користувача до вікна завершення автентифікації. На (рисунку 3.13) наведено приклад вигляду інтерфейсу у разі успішного підтвердження двох методів — електронної пошти та QR-коду. Система повідомляє про успішний вхід та надає можливість перейти на початкову сторінку.

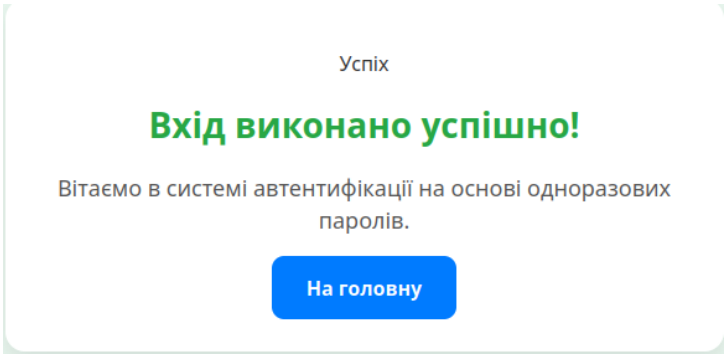

Рисунок 3.13. – Вікно успішного входу в систему автентифікації на основі одноразових паролів

Крім перевірки успішних сценаріїв, окрему увагу було приділено випадкам відхилення введених паролів — система правильно обробляє некоректні OTP, інформує користувача про помилку та забезпечує можливість повторного надсилання коду через інший доступний метод, якщо попередній уже підтверджено. Усі перевірки реалізовано відповідно до логіки, згрупованої у окремому модулі файлу routes.py, який наведений у Додатку А, де для кожного способу передбачено окрему кнопку надсилання та незалежне підтвердження.

Важливою складовою тестування є виявлення переваг і недоліків розробленої системи у контексті загальноприйнятих практик та існуючих продуктів, оскільки існує багато цікавих та популярних рішень, які є досить схожими з розробленою системою. Аналіз дозволяє оцінити конкурентоспроможність реалізованої системи автентифікації паролів, виявити суттєві переваги розробки з точки зору безпеки.

З метою об’єктивної оцінки ефективності розробленої системи було проведено порівняння її функціональних можливостей із популярними існуючими рішеннями: Duo Security, Microsoft Authenticator та Authy. Для оцінювання використано низку критеріїв, таких як кількість автентифікаційних каналів, наявність адаптивної оцінки ризику, підтримка журналювання, захист від атак типу brute-force тощо. Кожен критерій оцінювався за 5-бальною шкалою (0 – відсутність функції, 5 – повна підтримка). Результати порівняння та визначення ефективності наведено в (таблиці 1.2).

Таблиця 1.2

Оцінка ефективності системи автентифікації на основі одноразових паролів за 5-ти бальною шкалою

Критерій	**Duo Security**	**Microsoft Authenticator**	**Authy**	**Розроблена система автентифікації**
Тип OTP	4	4	3	5
Кількість каналів автентифікації	3	3	2	5
Адаптивна оцінка ризику	5	3	0	5
Логування подій	5	3	0	5
Інтерфейс користувача	5	5	5	4

продовження таблиці 1.1

Можливість резерву/синхронізації	3	2	5	3
Середній час автентифікації	4	5	5	3
Захист від brute-force атак	5	5	3	5
Гнучкість вибору способу OTP	1	1	1	5
Сумісність з іншими системами	5	5	3	3
Сумарна оцінка	40	31	27	43

Порівняльний аналіз показує, що розроблена система демонструє високий рівень функціональності за більшістю обраних критеріїв. Вона перевищує інші рішення завдяки підтримці трьох незалежних каналів доставки OTP із можливістю вибору будь-яких двох, адаптивній оцінці ризику на основі часу, IP та контексту доступу, а також повному логуванню дій користувачів.

Хоча час автентифікації в розробленій системі дещо більший через додаткові етапи перевірки, це компенсується гнучкістю налаштувань і вищим рівнем безпеки. Отже, запропонована система не лише не поступається, а й перевершує багато популярних рішень за інтегральними показниками ефективності, забезпечуючи як надійний захист, так і зручність для кінцевого користувача.

Висновки до розділу 3

У даному розділі було здійснено розгляд розробки функціонально завершеної системи автентифікації користувачів з використанням механізму одноразових паролів, що охоплює як вибір програмно-апаратного стеку, так і побудову архітектури, реалізацію окремих модулів, а також впровадження процедур динамічного реагування на потенційно аномальну поведінку. В основу

технічної реалізації покладено стек технологій на базі мови Python та фреймворку Flask, що забезпечує гнучкість модульної побудови та масштабованість подальшої інтеграції з іншими інформаційними системами. Обґрунтований вибір інструментальних засобів дозволив досягти необхідного балансу між продуктивністю, безпекою та простотою супроводу реалізованого рішення.

Структура системи побудована з дотриманням принципів розмежування відповідальності між модулями, що забезпечило прозору логіку керування потоками автентифікаційної сесії, починаючи з ініціалізації взаємодії користувача та завершуючи перевіркою валідності введених OTP-кодів. Програмна реалізація підтримує одразу декілька механізмів доставки одноразових паролів, зокрема через електронну пошту, SMS-повідомлення та QR-коди для TOTP-генераторів, з можливістю підтвердження двох із трьох каналів. Така конфігурація підвищує стійкість системи до часткової відмови окремих векторів комунікації та підвищує загальний рівень захисту.

Інтеграція адаптивного захисту в архітектуру системи дозволила реалізувати механізми оцінювання ризику входу на основі контекстних атрибутів, що формуються динамічно відповідно до поточних параметрів взаємодії користувача. Виявлення аномалій та обмеження надмірної кількості спроб входу реалізовано через логіку тимчасового блокування, яка активується при перевищенні допустимого порогу помилок, а також через моніторинг нетипових дій під час автентифікації. Результати тестування підтвердили, що система здатна у режимі реального часу адаптуватися до змін поведінки користувачів, зберігаючи при цьому високі показники доступності та стабільності функціонування.

ВИСНОВКИ

У рамках кваліфікаційної роботи було:

1. Досліджено сучасні методи та системи автентифікації користувачів, зокрема їхню класифікацію, принципи дії та характерні вразливості.
2. Проаналізовано алгоритми генерації одноразових паролів (OTP), зокрема TOTP та HOTP.
3. Розроблено прототип системи автентифікації на основі одноразових паролів.
4. Реалізовано адаптивну оцінку ризиків при автентифікації у систему автентифікації на основі одноразових паролів.

У результаті виконання кваліфікаційної роботи було досягнуто поставленої мети — розроблено систему автентифікації користувачів на основі одноразових паролів з додатковим захистом, орієнтованим на протидію сучасним загрозам.

Реалізована система дозволяє здійснювати автентифікацію з високим рівнем достовірності, враховуючи не лише OTP-коди, а й ризик-фактори, сформовані на основі контекстної інформації. Поєднання механізмів багатоканального OTP і адаптивної моделі оцінки ризику забезпечило високий рівень захищеності від атак типу «повторне відтворення», компрометації каналів доставки паролів, міжсесійних втручань.

Експериментальні результати свідчать про стабільну роботу системи в реальних умовах і демонструють її стійкість до атак, правильність генерації і перевірки одноразових кодів, а також здатність знижувати ризик несанкціонованого доступу завдяки динамічній оцінці поведінкових шаблонів.

Таким чином, результати роботи свідчать про ефективність запропонованої моделі автентифікації та демонструють її практичну доцільність для подальшого впровадження у системи з підвищеними вимогами до інформаційної безпеки. Виконане дослідження заклало основу для розширення

можливостей застосування одноразових паролів у поєднанні з контекстною оцінкою ризику, що є актуальним напрямом розвитку сучасних багатофакторних систем автентифікації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. National Institute of Standards and Technology. (2017). Digital identity guidelines: Authentication and lifecycle management (NIST Special Publication 800-63B). U.S. Department of Commerce. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63b.pdf

2. Chaker, A., Djahel, S., & Talhi, C. (2019, August). A survey on adaptive authentication [Conference presentation]. ResearchGate. https://www.researchgate.net/publication/335768541_A_Survey_on_Adaptive_Authentication

3. Sakhnini, A. A. (n.d.). Behavioral biometrics for continuous authentication. Journal of Biosensors and Bioelectronics Research, 1(3), 2-4. https://onlinescientificresearch.com/articles/behavioral-biometrics-for-continuous-authentication.pdf

4. National Institute of Standards and Technology. (2020). Digital identity guidelines (NIST SP 800-63-3). U.S. Department of Commerce. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63-3.pdf

5. International Organization for Standardization. (2022). Information technology — Vocabulary — Part 37: Biometrics. International Organization for Standardization *(*ISO Standart No 2382-37:2022). ISO/IEC 2382-37:2022 - Information technology — Vocabulary — Part 37: Biometrics

6. Logan M. Merritt E. Carlsson R. (2010). Erlang and OTP in Action. Manning Publications Co., United States.

7. Internet Engineering Task Force. (2005). HOTP: An HMAC-based one-time password algorithm (RFC 4226). https://www.rfc-editor.org/rfc/rfc4226

8. Internet Engineering Task Force. (2011). TOTP: Time-based one-time password algorithm (RFC 6238). https://www.rfc-editor.org/rfc/rfc6238

9. Karovaliya, M., Karedia, S., Oza, S., & Kalbande, D. R. (2015). Enhanced Security for ATM Machine with OTP and Facial Recognition Features. In Procedia

Computer Science: Vol. 45. (pp. 390–396). Elsevier. https://doi.org/10.1016/j.procs.2015.03.166

10. International Organization for Standardization. (2024). Information technology — Security techniques — A framework for identity management — Part 1: Terms and concepts. (ISO Standart No 24760-1:2024). https://online.budstandart.com/ua/catalog/doc-page.html?id_doc=113008

11. Schneier, B. (2004). Secrets and lies: Digital security in a networked world. John Wiley & Sons.

12. Wayman, J. L., Jain, A. K., Maltoni, D., & Maio, D. (Eds.). (2005). *Biometric systems:* Technology, design and performance evaluation. Springer Science & Business Media.

13. Karia, A. R., Patankar, A. B., & Tawde, P. (2014). SMS-Based One Time Password Vulnerabilities and Safeguarding OTP Over Network. In International Journal of Engineering Research & Technology (IJERT): Vol. 3, Issue 5. (pp. 1396–1398). IJERT. sms-based-one-time-password-vulnerabilities-and-safeguarding-otp-over-network-IJERTV3IS051538-libre.pdf

14. Stallings, W. (2021). Cryptography and network security: Principles and practice (8th ed.). Pearson.

15. Adamu, H., Mohammed, A. D., Adepoju, S. A., & Aderiike, A. O. (2022). A Three-Step One-Time Password, Textual and Recall-Based Graphical Password for an Online Authentication. In 2022 IEEE Nigeria 4th International Conference on Disruptive Technologies for Sustainable Development (NIGERCON) (pp. 1–5). IEEE. https://doi.org/10.1109/NIGERCON54645.2022.9803122

16. Erdem, E., & Sandıkkaya, M. T. (2019). OTPaaS—One Time Password as a Service. IEEE Transactions on Information Forensics and Security, 14(3), 743–756. https://doi.org/10.1109/TIFS.2018.2866025

17. Sharma, D., & Saini, P. (2013). Enhancing security using time-based one-time password (TOTP) in client server authentication system. International Journal of Computer Applications, 73(10), 28–32. https://www.ijcaonline.org/archives/volume73/number10/12586-8800

18. Ding, Z., & Wang, D. (2025). HTOTP: Honey Time-Based One-Time Passwords. In IEEE Transactions on Information Forensics and Security: Vol. 20 (pp. 4438–4453). IEEE. https://doi.org/10.1109/TIFS.2025.3550075

19. Nithyashree, G., & Kumar, R. A. (2015). Enhanced user authentication using hybrid cryptography with one time password generation. International Journal of Computer Applications, 110*(15),* 1–4. https://www.ijcaonline.org/archives/volume110/number15/19324-1065

20. Mwelema, D., & Matela, M. (2025). Design and Development of an online Fund Transfer System with Data encryption standard and One-Time-Password. In International Journal of Advances in Scientific Research and Engineering (ijasre): Vol. 11, Issue 3. (p. 38). IJASRE. ijasre_34524-libre.pdf

21. International Organization for Standardization. (2020). Information security — Entity authentication — Assurance framework (ISO Standart No 29115:2013). https://groups.oasis-open.org/higherlogic/ws/public/document?document_id=44751

22. Pfleeger, C. P., & Pfleeger, S. L. (2015). Security in computing (5th ed.). Pearson.

23. Aleksandr Ometov et al. “Multi-factor authentication: A survey”. In: Cryptography 2.1 (2018), p. 1.

24. Andrea Visconti and Federico Gorla. “Explo iting an HMAC-SHA-1 optimization to speed up PBKDF2”. In: IEEE Transactions on Dependable and Secure Computing 17.4 (2018), pp. 775–781.

25. Охота Д.Б. (2011) Технології комп’ютерної безпеки. Монографія. МЕГУ, Рівне. - 97 с

26. Suratose Tritilanunt, Napat Thanyamanorot, and Nattawut Ritdecha. “A secure authentication protocol using HOTP on USB storage devices”. In: 2014 International Conference on Information Science, Electronics and Electrical Engineering. Vol. 3. IEEE. 2014, 1908–1912.

27. Christophe Kiennert, Samia Bouzefrane, and Pascal Thoniel. “Authentication systems”. In: Digital identity management. Elsevier, 2015, pp. 95–135.

28. Marc Briceno et al. Advanced authentication techniques and applications. US Patent 10,270,748. Apr. 2019.

29. Yang, Z., Jin, C., Ning, J., Li, Z., Dinh, A., & Zhou, J. (2021). Group Time-based One-time Passwords and its Application to Efficient Privacy-Preserving Proof of Location. In Proceedings of the 37th Annual Computer Security Applications Conference (ACSAC '21) (pp. 497–512). ACM. Group Time-based One-time Passwords and its Application to Efficient Privacy-Preserving Proof of Location | Proceedings of the 37th Annual Computer Security Applications Conference

30. Dhamija, D., & Dhamija, A. (2022). A Secure and Reliable Architecture for User Authentication Through OTP in Mobile Payment System. In M. Dua, A. K. Jain, A. Yadav, N. Kumar, & P. Siarry (Eds.), Proceedings of the International Conference on Paradigms of Communication, Computing and Data Sciences. Algorithms for Intelligent Systems (pp. 95–109). Springer.

31. Ausheva, N. M., Melnyk, Y. V., Otrokh, S. I., & Mordas, I. S. (2023). System of Two-Factor Authentication of the User of the Corporate Environment Using a QR Code. ЗВ’ЯЗОК, 5, 38–40. Перегляд Система двофакторної автентифікації користувача корпоративного середовища з використанням QR-коду

ДОДАТОК А

Фрагмент коду програмної реалізації файлу routes.py:

```
from flask import Blueprint, render_template, request, redirect, url_for, session, flash
from .models import db, User, AuthLog
from .utils.otp import generate_otp_secret
from .utils.notifier import send_email, send_sms
from .utils.rate_limit import is_rate_limited, register_attempt
from .utils.risk import evaluate_risk
import pyotp
import qrcode
import io
import base64
import time


bp = Blueprint('main', __name__)

@bp.route('/', methods=['GET', 'POST'])
def index():
    return render_template('index.html')

@bp.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        username = request.form.get('username')
        email = request.form.get('email')
        phone = request.form.get('phone')
        otp_type = request.form.get('otp_type', 'TOTP')
        methods = request.form.getlist('methods')

        if len(methods) < 2:
            flash('Оберіть два методи отримання OTP', 'warning')
            return redirect(url_for('main.register'))


        preferred_methods = ','.join(methods)
        otp_secret = generate_otp_secret()
        hotp_secret = generate_otp_secret()
        totp_secret = generate_otp_secret()


        existing_user = User.query.filter_by(username=username).first()
        if existing_user:
```

Продовження додатку А

```
            flash('Користувач із таким іменем вже існує', 'danger')
            return redirect(url_for('main.register'))

        user = User(
            username=username,
            email=email,
            phone=phone,
            otp_secret=otp_secret,
            otp_type=otp_type,
            preferred_methods=preferred_methods,
            hotp_secret=hotp_secret,
            totp_secret=totp_secret,
            hotp_counter=0
        )
        db.session.add(user)
        db.session.commit()

        qr_code_data = None
        if 'qr' in methods:
            qr_uri = pyotp.TOTP(totp_secret).provisioning_uri(name=username,
issuer_name="OTP Auth System")
            qr_img = qrcode.make(qr_uri)
            buffered = io.BytesIO()
            qr_img.save(buffered, format="PNG")
            qr_code_data = base64.b64encode(buffered.getvalue()).decode()

        return render_template('register_success.html', username=username,
qr_code=qr_code_data)

    return render_template('register.html')

@bp.route('/verify', methods=['GET', 'POST'])
def verify():
    user_id = session.get('user_id')
    selected_methods = session.get('selected_methods')
    verified_methods = session.get('verified_methods', [])

    if request.method == 'POST':
        if 'username' in request.form:
            username = request.form.get('username').strip()
            user = User.query.filter_by(username=username).first()
```

Продовження додатку А

```
    if not user:
        flash("Користувача не знайдено", 'danger')
        return redirect(url_for('main.verify'))

    identifier = request.remote_addr or username
    if is_rate_limited(identifier):
        flash('Забагато спроб. Повторіть пізніше.', 'danger')
        return redirect(url_for('main.index'))

    register_attempt(identifier)
    session['user_id'] = user.id
    session['selected_methods'] = user.preferred_methods.split(',')
    session['verified_methods'] = []

    now = time.time()
        if user.otp_type == 'HOTP':
        hotp = pyotp.HOTP(user.hotp_secret)
        totp = pyotp.TOTP(user.totp_secret, interval=30)

        session['email_otp'] = hotp.at(user.hotp_counter)
        session['sms_otp'] = totp.now()
        session['qr_otp'] = totp.now()
    else:
        totp = pyotp.TOTP(user.totp_secret, interval=30)
        now_code = totp.now()
        session['email_otp'] = now_code
        session['sms_otp'] = now_code
        session['qr_otp'] = now_code

    session['otp_time'] = now
    session['resend_last'] = 0
    flash("Коди згенеровано. Натисніть кнопку, щоб отримати код.", 'info')
    return redirect(url_for('main.verify'))

user = User.query.get(user_id)
selected_methods = session.get('selected_methods', [])
verified_methods = session.get('verified_methods', [])
otp_time = session.get('otp_time', 0)

# Надсилання коду
if 'send_method' in request.form:
    method = request.form.get('send_method')
```

Продовження додатку А

```
            try:
                if method == 'email' and 'email' in selected_methods and method not in
verified_methods:
                    send_email(user.email, "OTP for authentication", f"Your OTP:
{session.get('email_otp')}")
                    flash("Код надіслано на Email", 'info')
                elif method == 'sms' and 'sms' in selected_methods and method not in
verified_methods:
                    send_sms(user.phone, f"Your OTP: {session.get('sms_otp')}")
                    flash("Код надіслано на SMS", 'info')
                elif method == 'qr':
                    flash("Скануйте QR або введіть код з додатку", 'info')
            except Exception as e:
                flash(f"Помилка надсилання: {e}", 'danger')
            return redirect(url_for('main.verify'))
        # Підтвердження методу
        if 'confirm_method' in request.form:
            method = request.form.get('confirm_method')
            input_code = request.form.get(f"{method}_otp", '').strip()
            session_code = session.get(f"{method}_otp")

            is_valid = False
            expired = time.time() - otp_time > 120  # 2 хвилини
            if expired:
                flash("Час дії OTP вичерпано. Отримайте новий код.", 'danger')
                return redirect(url_for('main.verify'))
            if input_code:
                if method == 'qr':
                    totp = pyotp.TOTP(user.totp_secret, interval=30)
                    is_valid = totp.verify(input_code)
                else:
                    is_valid = input_code == session_code

            if is_valid:
                if method not in verified_methods:
                    verified_methods.append(method)
                    flash(f"{method.upper()} підтверджено", 'success')
                else:
                    flash(f"{method.upper()} вже підтверджено", 'info')
            else:
                flash(f"Невірний код з {method.upper()}", 'danger')
                db.session.add(AuthLog(
```

Продовження додатку А

```
                user_id=user.id,
                success=False,
                ip_address=request.remote_addr,
                risk_score=evaluate_risk(user, request.remote_addr,
request.headers.get('User-Agent', '')),
                method_used=method
            ))
            db.session.commit()

        session['verified_methods'] = verified_methods

        if len(verified_methods) >= 2:
            if user.otp_type == 'HOTP':
                user.hotp_counter += 1

            db.session.add(AuthLog(
                user_id=user.id,
                success=True,
                ip_address=request.remote_addr,
                risk_score=evaluate_risk(user, request.remote_addr,
request.headers.get('User-Agent', '')),
                method_used=','.join(verified_methods)
            ))
            db.session.commit()
            flash("Вхід успішний", 'success')
            session.pop('user_id', None)
            session.pop('selected_methods', None)
            session.pop('verified_methods', None)
            session.pop('otp_time', None)
            session.pop('resend_last', None)
            return redirect(url_for('main.success'))

        return redirect(url_for('main.verify'))

    # Повторне надсилання на непідтверджені
    if 'resend_other' in request.form:
        now = time.time()
        last_sent = session.get('resend_last', 0)
        if now - last_sent < 30:
            flash("Почекайте перед повторним надсиланням кодів.", 'warning')
```

Продовження додатку А

```
            return redirect(url_for('main.verify'))

        try:
            if user.otp_type == 'HOTP':
                hotp = pyotp.HOTP(user.hotp_secret)
                totp = pyotp.TOTP(user.totp_secret, interval=30)

                if 'email' in selected_methods and 'email' not in verified_methods:
                    session['email_otp'] = hotp.at(user.hotp_counter)
                    send_email(user.email, "OTP for authentication", f"Your OTP:
{session['email_otp']}")
                if 'sms' in selected_methods and 'sms' not in verified_methods:
                    session['sms_otp'] = totp.now()
                    send_sms(user.phone, f"Ваш код: {session['sms_otp']}")
            else:
                totp = pyotp.TOTP(user.totp_secret, interval=30)
                new_code = totp.now()
                if 'email' in selected_methods and 'email' not in verified_methods:
                    session['email_otp'] = new_code
                    send_email(user.email, "OTP for authentication", f"Your OTP:
{new_code}")
                if 'sms' in selected_methods and 'sms' not in verified_methods:
                    session['sms_otp'] = new_code
                    send_sms(user.phone, f"Ваш код: {new_code}")

            session['otp_time'] = now
            session['resend_last'] = now
            flash("Коди повторно надіслані на непідтверджені методи", 'info')
        except Exception as e:
            flash(f"Помилка повторного надсилання: {e}", 'danger')

        return redirect(url_for('main.verify'))

    return render_template(
        'verify.html',
        step='otp' if user_id else 'username',
        selected_methods=selected_methods,
        verified_methods=session.get('verified_methods', [])
    )
@bp.route('/success')
def success():
    return render_template('success.html')
```