Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:
«МЕТОД ТА ЗАСІБ АВТЕНТИФІКАЦІЇ КОРИСТУВАЧІВ ІЗ НУЛЬОВИМ ЗНАННЯМ»

Виконав: студент 2-го курсу, групи 1БС-20м
спеціальності 125 – Кібербезпека
(шифр і назва напряму підготовки, спеціальності)
_____ Селезньов В. І.
(прізвище та ініціали)
Керівник к. т. н., доц. каф. ЗІ
_____ Баришев Ю. В
(прізвище та ініціали)
Опонент: к. т. н., доц. каф . ОТ
_____ Войцеховська О. В.
(прізвище та ініціали)

«____» _____ 2021 р.

Допущено до захисту
Завідувач кафедри ЗІ
д. т. н., проф.
_____ Лужецький В. А.
«____» _____ 2021 р.

Вінниця ВНТУ – 2021 року

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації
Освітньо-кваліфікаційний рівень магістр
Спеціальність 125 Кібербезпека
ОПП Безпека інформаційних і комунікаційних систем

ЗАТВЕРДЖУЮ

Завідувач кафедри ЗІ, д. т. н., проф.

_____ В. А. Лужецький

_____ _____ 2021 року

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Селезньову Віталію Ігоровичу

1. Тема роботи: «Метод та засіб автентифікації користувачів із нульовим знанням» керівник роботи: Баришев Юрій Володимирович, к. т. н., доц. каф. ЗІ, затверджена наказом ректора ВНТУ 9 вересня 2021 року №207.р.
2. Строк подання студентом роботи 17 грудня 2021 р.
3. Вихідні дані до роботи:
   - довжина ключа – 256;
   - кількість раундів автентифікації – 10;
   - підхід з нульовим знанням;
   - спосіб реалізації – програмна бібліотека;
   - мова програмування Java.
4. Зміст розрахунково-пояснювальної: Вступ. Аналіз інформаційних джерел. Метод автентифікації з нульовим знанням. Алгоритми роботи засобу автентифікації з нульовим знанням. Реалізація і тестування засобу автентифікації з нульовим знанням. Економічна частина. Висновки. Перелік інформаційних джерел. Додатки.
5. Перелік ілюстративного матеріалу.

Порівняння методів автентифікації із нульовим знанням (плакат, А4). Математичний опис процесу автентифікації з нульовим знанням (плакат, А4). Протокол автентифікації з нульовим знанням (плакат, А4). Структура засобу автентифікації з нульовим знанням (плакат, А4). Алгоритм роботи засобу для

клієнтської частини (плакат, А4). Алгоритм роботи засобу для серверної частини (плакат, А4). Результати тестування (плакат, А4).

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Баришев Ю. В., к. т. н., доц. каф. ЗІ		
2	Баришев Ю. В., к. т. н., доц. каф. ЗІ		
3	Баришев Ю. В., к. т. н., доц. каф. ЗІ		
4	Баришев Ю. В., к. т. н., доц. каф. ЗІ		
5	Лесько О. Й., к. е. н., проф., зав. каф. ЕПВМ		

7. Дата видачі завдання 9 вересня 2021 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз завдання. Вступ	01.09.2021 – 04.09.2021	
2	Розробка технічного завдання	05.09.2021 – 10.09.2021	
3	Науково-технічне обґрунтування	11.09.2021 – 15.09.2021	
4	Аналіз інформаційних джерел за напрямком магістерської кваліфікаційної роботи	16.09.2021 – 04.10.2021	
5	Розробка методу автентифікації користувачів з нульовим знанням	05.10.2021 – 24.10.2021	
6	Розробка алгоритму, програмна реалізація та тестування модуля автентифікації користувачів з нульовим знанням	25.10.2021 – 17.11.2021	
7	Аналіз виконання ТЗ, висновки	18.11.2021 – 24.11.2021	
8	Оформлення пояснювальної записки	25.11.2021 – 30.11.2021	
9	Попередній захист та доопрацювання МКР	03.12.2021	
10	Перевірка магістерської роботи на наявність плагіату	03.12.2021 – 04.12.2021	
11	Представлення МКР до захисту, рецензування	05.12.2021 – 20.12.2021	
12	Захист МКР	21.12.2021 – 23.12.2021	

Студент _____ В. І. Селезньов

Керівник роботи _____ Ю. В. Баришев

АНОТАЦІЯ

УДК 004.056

Селезньов В. І. Метод та засіб автентифікації користувачів із нульовим знанням. Магістерська кваліфікаційна робота зі спеціальності 125 – кібербезпека, освітня програма – Безпека інформаційних і комунікаційних систем. Вінниця: ВНТУ, 2021. 123 с.

Укр. мовою. Бібліогр.: 39 назв; рис.: 34; табл. 12.

Магістерська кваліфікаційна робота присвячена розробці методу та засобу автентифікації користувачів з нульовим знанням. Підготовлено науково-технічне та техніко-економічне обґрунтування доцільності проведення досліджень. Здійснено аналіз сучасних методів автентифікації користувачів за результатами якого обґрунтовано вибір для розробки методу автентифікації з нульовим знанням. Розроблено метод автентифікації користувачів з нульовим знанням. Розроблено програмний засіб, що дозволяє здійснити автентифікацію користувача за розробленим методом з нульовим знанням. Виконано тестування методу та засобу автентифікації користувачів з нульовим знанням.

Графічна частина складається з 7 плакатів з демонстрацією результатів проведених досліджень та розробок.

Оцінку витрат на розробку представлено в економічному розділі.

Ключові слова: криптографічний протокол, автентифікація, доведення з нульовим знанням, криптостійкість, нульовий розголос.

ABSTRACT

Seleznov V. I. Method and means of authentication users with zero knowledge. Master's thesis in specialty 125 – cybersecurity, educational program – Security of information and communication systems. Vinnytsia: VNTU, 2021. 123 p.

In Ukrainian language. Bibliographer: 39 titles; fig.: 34; tabl.: 12.

Master's thesis is devoted to developing a method and means of users authentication with zero knowledge. The scientific and technical feasibility of the research was substantiated. Existing authentication methods were analyzed. The choice to develop a method of authentication with zero knowledge was justified. The method of authentication of users with zero knowledge is developed. Developed software that allows to authenticate user using developed authentication method with zero knowledge. The software module of user authentication with zero knowledge was tested.

The graphic part consists of 7 posters demonstrating the results of modeling and research.

In the Economic part the expenses for development are estimated.

Keywords: cryptographic protocol, authentication, proof with zero knowledge, cryptocurrency, zero publicity.

ЗМІСТ

ВСТУП

Розв’язання задачі автентифікації користувачів стало невід’ємною частиною сучасного інформаційного простору. Більшість багатокористувацьких інформаційних систем, починаючи від засобів обміну миттєвими повідомленнями, закінчуючи веб-ресурсами та операційними системами, вимагають від користувачів використання облікових записів для отримання доступу до інформації ресурсів та системи, в цілому. Кожен користувач виконує автентифікацію перед початком роботи із обліковим записом, тому виконання процесу автентифікації є надзвичайно важливим з точки зору безпеки. Від реалізація процесу автентифікації може залежати безпека не лише користувача, а й усієї системи. Не всі сучасні засоби автентифікації забезпечують достатній рівень стійкості при автентифікації користувача, тому **актуальною** задачею є розробка методу та засобу автентифікації користувачів з нульовим знанням.

Серед сучасних засобів автентифікації використання доведення з нульовим розголошенням дозволяє усунути велику кількість відомих атак на процес автентифікації за рахунок відсутності розкриття будь-якої інформації про секретні дані. Використання підходу з нульовим знанням дозволяє суттєво збільшити рівень захищеності системи, тому варто застосувати такий підхід до розробки протоколу автентифікації. Існуючі протоколи автентифікації з нульовим знанням дозволяють забезпечити достатній рівень безпеки, однак не завжди є практичними для застосування та мають ряд недоліків. Розробка модифікованого методу та засобу автентифікації з нульовим знанням дозволить підвищити рівень безпеки при автентифікації та як наслідок збільшити ефективність захищеності облікових записів користувачів.

Значний внесок у розвиток методів автентифікації користувачів з нульовим та дослідження доведення з нульовим розголошенням зробили такі вченні: Шамір А., Фіат А., Шнорр К., Гамаль Т. Голдвассер С., Мікалі С. Бен-Сассон Е., Бентов І., Рябцев М. [1, 2, 3].

Об’єктом дослідження є процес автентифікації користувачів.

Предметом є засоби та методи автентифікації користувачів.

Метою магістерської кваліфікаційної роботи є покращення захищеності автентифікації користувачів за допомогою використання підходу з нульовим знанням в якості протоколу автентифікації. Для досягнення мети необхідно:

- проаналізувати відомі засоби автентифікації;
- проаналізувати методи автентифікації з нульовим знанням;
- виконати узагальнений математичний опис процесу автентифікації;
- розробити метод автентифікації з нульовим знанням;
- розробити алгоритм роботи засобу автентифікації;
- розробити програмний модуль;
- виконати перевірку коректності роботи системи;
- обґрунтувати економічну доцільність розробки.

Наукова новизна магістерської роботи полягає в тому, що удосконалено метод автентифікації користувачів з нульовим знанням, який на відміну від відомих використовує псевдовипадковий вибір способу реалізації раунду автентифікації з нульовим знанням, що дозволяє підвищити рівень захищеності при автентифікації.

Практична цінність: метод автентифікації з нульовим знанням; програмний засіб автентифікації з нульовим знанням.

Результати магістерської роботи доповідалися на таких конференціях:

- XLIX регіональна н. т. к. професорсько-викладацького складу, співробітників та студентів університету з участю працівників науково-дослідних організацій та інженерно-технічних працівників підприємств м. Вінниці та області відбулась 18–29 травня 2020 р. [4];
- Всеукраїнська науково-практична інтернет-конференція Молодь в науці: дослідження, проблеми, перспективи (МН-2021) Вінницький національний технічний університет відбулась 1–14 травня 2021 р. [5];

За результатами магістерської кваліфікаційної роботи **опубліковано** тези доповідей у збірниках матеріалів конференцій та зареєстровано три свідоцтва авторського права на твір [6, 7, 8].

1 АНАЛІЗ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1.1 Науково-технічне обґрунтування

Автентифікація – це процедура перевірки справжності суб’єкта, яка дозволяє достовірно переконатись у тому, що суб’єкт, який пред’явив свій ідентифікатор, насправді є саме тим суб’єктом, ідентифікатор якого він використовує [4].

У сучасному світі процес автентифікації є невід’ємною складовою більшості систем. Технологія перевірки автентичності користувача забезпечує контроль доступу для систем, перевіряючи, чи облікові дані користувача, що намагається отримати доступ, збігаються з обліковими даними в базі даних авторизованих користувачів або на сервері автентифікації даних. Користувачі, як правило, ідентифікуються за ідентифікатором користувача, а автентифікація виконується, коли користувач надає облікові дані, наприклад пароль, який відповідає наданому ідентифікатору користувача. Найпростішим та найбільш відомим способом автентифікації є використання пароля в якості облікових даних, як інформації, яка повинна бути відома тільки відповідному користувачу. Після автентифікації користувач або процес, як правило, переходять до авторизації, щоб визначити, до яких об’єктів системи та з якими правами надається доступ автентичному користувачу.

Терміни автентифікації та авторизації часто використовуються як взаємозамінні. Хоча вони часто можуть бути реалізовані разом, ці дві функції відрізняються. Хоча автентифікація є процесом перевірки ідентичності зареєстрованого користувача перед тим, як дозволити доступ до захищеного ресурсу, авторизація є процесом перевірки того, що авторизований користувач отримав дозвіл на доступ до запитаних ресурсів. Процес, за допомогою якого доступ до цих ресурсів обмежується певною кількістю користувачів, називається контролем доступу. Процес автентифікації завжди проходить до процесу авторизації [9].

Після введення карантинних обмежень у 2020 році багато компаній були змушені збільшити кількість віддалених працівників, а деякі повністю перейшли у режим роботи лише віддалено. Зросла популярність використання засобів віддаленого доступу та веб ресурсів. В наслідок цього зросла кількість кібератак та рівень загрози інформаційної безпеки в усіх системах [10].

Безпека при автентифікації є надзвичайно важливою, оскільки злам модуля автентифікації дозволяє зловмиснику отримати доступ до системи від імені автентичного облікового запису та отримати доступ до конфіденційної інформації, а сучасні засоби автентифікацію не завжди дозволяють забезпечити відповідний рівень безпеки при автентифікації. Зазвичай використовуються стандартні протоколи, більшість з яких є умовно-безпечними, однак залежно від виду автентифікації та факторів, що використовуються при автентифікації, протокол може бути вразливим до певного роду атак. Тому аналіз методів автентифікації відповідних протоколів дозволяє оцінити переваги та недоліки обраних методів та, як наслідок, обрати підхід, який дозволить підвищити стійкість автентифікації, що, в свою чергу, підвищить рівень захищеності системи в цілому.

1.2 Аналіз методів автентифікації

Автентифікація користувача залежить від одного або декількох параметрів, які повинен пред’явити користувач, для успішного проходження автентифікації. Ці параметри називаються факторами автентифікації.

Фактор автентифікації являє собою деякий фрагмент даних або атрибут, який може використовуватися для автентифікації користувача, що прагне отримати доступ до системи.

Фактори автентифікації можуть бути такими [10]:

– фактор знання передбачає наявність певної відомої тільки користувачу інформації;

- фактор володіння має на увазі, що користувач володіє деяким унікальним предметом, що містить необхідну характеристику (пластикові карти, USB токени тощо);
- фактор властивості передбачає використання деякої фізичної особливості користувача, наприклад, відбиток пальця, райдужна оболонка ока тощо;
- фактор розташування передбачає перевірку місцезнаходження користувача;
- фактор часу.

Залежно від кількості сторін, що доводять свою автентичність, автентифікація може бути односторонньою (зазвичай клієнт доводить свою автентичність серверу) та двосторонньою (взаємною). Для її реалізації використовують спеціалізовані криптографічні протоколи.

Залежно від кількості факторів автентифікація може бути однофакторною та багатофакторною. Під час однофакторної автентифікації використовується один з факторів знання, володіння або властивості. Фактори розташування та часу не є достатніми для виконання однофакторної автентифікації та можуть бути використані лише як додаткові фактори [11].

Використання багатофакторної автентифікації дозволяє підвищити рівень безпеки за рахунок того, що неавторизований користувач не зможе надати всі необхідні фактори. Якщо при проходженні автентифікації хоча б один із факторів відсутній або зазначений некоректно, то система не може бути впевненою в автентичності користувача, що проходить автентифікацію, і, як наслідок, доступ до системи для користувача залишається заблокованим.

Найрозповсюдженішою формою автентифікації є використання фактору знання. З використанням фактору знання під час автентифікації користувач повинен підтвердити знання певного секрету. Як секрет найчастіше використовують пароль - секретне слово або рядок символів, що відомо лише автентичному користувачу. Більшість механізмів багатофакторної

автентифікації використовують пароль як один з факторів автентифікації. Варіації паролів включають в себе як більш довгі, сформовані з декількох слів, так і більш коротких, числових, персональних ідентифікаційних номерів, що часто використовуються для доступу до автоматичних касових апаратів [12].

Фактор володіння передбачає використання для автентифікації певного об’єкту, яким володіє тільки користувач. Основний принцип полягає в тому, що користувач може пройти автентифікацію лише з використанням перевірки необхідного об’єкту, що належить лише йому. Прикладом фактору володіння є токени, що можуть бути декількох видів [11]:

- Токени без підключення – фізичні пристрої, що не мають зв’язку з комп’ютерним пристроєм користувача. На таких пристроях зазвичай дані автентифікації відображаються на спеціалізованому екрані, а користувач самостійно копіює їх для виконання автентифікації;
- Токени з підключенням – фізичні пристрої, що підключаються до комп’ютерного пристрою користувача та автоматично передають дані автентифікації;
- Програмні токени – це тип пристрою двофакторної автентифікації, що може використовуватися для автентифікації при використання комп'ютерних сервісів. Програмні токени можуть зберігатися на будь-яких електронних пристроях та можуть бути дубльовані на декількох пристроях. Головним недоліком використання токенів є ймовірність втрати або крадіжка токену.

Фактор властивості використовує для автентифікації фізичні особливості суб’єкта, що проходить автентифікацію. Для користувача фізичними властивостями може бути відбиток пальця чи долоні, форма обличчя, голос, сітківка ока тощо. З точки зору суб’єкта використання фактору властивості спрощує автентифікацію: немає необхідності запам’ятовувати секретні фрази, зберігати додаткові програмні чи апаратні пристрої автентифікації. Однак використання фактору властивості суб’єкта потребує наявності додаткових

програмних та апаратних засобів, що дозволяють зчитувати та опрацьовувати параметри фізичної особливості суб'єкта. Відповідна система повинна бути достатньо чутливою та точною для розпізнання відповідних властивостей різних суб'єктів. Такі системи значно підвищують вартість процесу автентифікації. Недоліком використання фактору властивості є можливість підробки або крадіжки фізичних властивостей зловмисником та неможливість у деяких випадках змінити фізичні властивості суб'єкта, наприклад, якщо зловмисник підробив відбитки пальців, що використовується як фактор автентифікації, то користувачу неможливо буде змінити свої відбитки пальців для зміни доступу до системи [13].

Фактор фізичного розташування використовує перевірку мережі, в якій знаходиться користувач. У разі підключення користувача до безпечної корпоративної мережі йому надається доступ до системи, у іншому випадку в доступі за фактором фізичного розташування буде відмовлено. Для автентифікації лише фізичного розташування недостатньо, тому зазвичай такий фактор використовується в якості додаткового при багатофакторній автентифікації.

Фактор часу також використовується як додатковий фактор при багатофакторній автентифікації та передбачає перевірку параметру часу проходження автентифікації та його відповідність умовам, що відповідають системі автентифікації.

Другим фактором може бути використання QR-коду, одноразових паролів, що генеруються додатковими програмними засобами на телефоні, або перевірку на основі SMS. Таким чином, володіючи лише паролем користувача зловмиснику, не вдасться пройти автентифікацію. Однак використання телефонів не гарантує повну безпеку. Перевірку на основі SMS легко перехопити за допомогою спеціалізованих засобів. Телефони можуть бути клоновані, застосунки можуть працювати на декількох телефонах, а служби підтримки телефонів можуть мати доступ до тексту SMS. Більшість користувачів зберігають всі свої паролі у облікових записах на телефоні, що у разі втрати та

зламу мобільного пристрою надає зловмиснику повний доступ до всіх систем користувача [13].

Для автентифікації також використовують доведення з нульовим розголошенням (знанням). Доведення з нульовим знанням – метод для доведення однією стороною іншій, що твердження (зазвичай – математичне) – істинне, без розкриття будь-якої іншої інформації, окрім достовірності твердження.

У протоколах автентифікації з нульовим знанням користувач, що проходить автентифікацію (Петті) повинен довести стороні сервера (Віктору), що він володіє секретною інформацією без розкриття секрету при цьому. Тобто при виконанні процесу автентифікації відбувається нульовий розголос секрету Петті.

Доведення з нульовим розголошенням повинно мати три властивості [2]:

a) Повнота (англ. completeness): якщо твердження істинне, то Петті, що повністю слідує протоколу завжди переконає чесного Віктора.
b) Коректність (англ. soundness) захищає від прийняття хибного твердження: якщо твердження хибне, то ймовірність обману в будь-якому випадку має бути дуже низькою.
c) Нульове розголошення: якщо твердження істинне, то Віктор, який грає не за правилами, не може дізнатись нічого, окрім факту істинності.

Зокрема, після того, як Віктор переконався в істинності твердження, сам він не зможе довести його істинність третім особам.

Дві перші властивості виконуються для загальної інтерактивної системи доведення. Третя властивість призводить до нульового розголошення.

Доведення з нульовим розголошенням не є доведенням в математичному сенсі, бо існує мала ймовірність того, що нечесний Петті зможе переконати Віктора в істинності хибного твердження [2].

1.3 Аналіз протоколів автентифікації з нульовим знанням

На сьогодні класичними протоколами автентифікації з нульовим знанням є протокол Шнорра, протокол Фіата-Шаміра протокол на основі задачі про ізоморфізм графів з різними модифікаціями. Серед сучасних протоколів

доведення з нульовим розголошенням представлені протоколи zk-SNARK, zk–STARK та Bulletproofs.

Протокол Шнорра застосовує проблему дискретного логарифмування для процесу автентифікації [2]. Спочатку відкритий ключ Петті обчислюється з секретного ключа x за формулою $y = \alpha^x \bmod p$, де p – достатньо велике випадкове число; α – випадкове число великого простого порядку $q < p$. Процес автентифікації складається з декількох раундів, які, в свою чергу, складаються з трьох основних кроків:

Петті обирає випадкове число $k(k < q)$, обраховує значення $R = \alpha^k \bmod p$ та передає Віктору.

Віктор формує випадковий біт r та передає Петті.

Петті обчислює $w = k + rx \bmod p$ та відсилає Віктору. Віктор виконує перевірку співвідношення:

$$Ry^r \equiv \alpha^w \bmod p \tag{1.1}$$

У разі істини співвідношення Петті успішно проходить автентифікацію.

Протокол Фіата-Шаміра заснований на складності добування квадратного кореня за складеним модулем, але має подібну структуру до протоколу Шнорра.

Перед процесом автентифікації обирається секретний ключ, модуль та відкритий ключ, що обчислюється шляхом піднесення секретного ключа до квадрату за модулем.

Процес автентифікації розпочинається з вибору Петті випадкового числа, піднесення його до квадрату за обраним модулем та передачі результату Віктору. Віктор випадковим чином формує число 0 або 1 і передає його Петті. Петті виконує обчислення піднесення секретного ключа до степені отриманого від Віктора, виконує множення результату на випадкове число, яке вона сформувала раніше, за відповідним модулем. Результат обчислення надсилається до Віктора. Віктор виконує перевірку результату шляхом його піднесення до квадрату за

відповідним модулем та порівнянням з власним результатом, отриманим на основі відкритого ключа та випадкового числа Петті. У разі ідентичності результатів, Петті успішно проходить раунд автентифікації [14].

Подібно до протоколу Шнорра автентифікація виконується в декілька раундів, кількість яких достатня для забезпечення стійкості. Для того, щоб протокол Фіата-Шаміра коректно виконувався, сторона Петті повинна використовувати унікальні значення R для кожної раунду автентифікації.

При схемах Фіата-Шаміра та Шнорра існує два можливих шляхи, при яких ймовірність дати правильну відповідь складає 0.5. При виконанні декількох раундів протоколів ця ймовірність суттєво зменшується і обчислюється за формулою 2^{-Z}, де Z – кількість раундів [1].

Прикладом протоколу доведення з нульовим знанням є ізоморфізм графів. Нехай дана пара графів $G_0 = (U, E_0)$ та $G_1 = (U, E_1)$, де U - множина вершин графа, а E_0 і E_1 - множини ребер.

Графи G_0 та G_1 називають ізоморфними, якщо існує перестановка вершин графа, яка переводить один граф в інший. Завдання знаходження такої перестановки і пошук відповіді на питання про її існування – складна математична задача, що не має вирішення за поліноміальний час [14].

Нехай π – ізоморфізм графів G_0 та G_1. Петті володіє π та доводить це знання Віктору:

Крок 1. Петті випадковим чином обирає перестановку σ та біт $b \in \{0,1\}$, обчислює $H = (G_b)$ і передає H Віктору.

Крок 2. Віктор формує випадковий біт $b' \in \{0,1\}$ та передає Петті.

Крок 3. Петті відправляє перестановку τ до Віктора, де

$$\tau = \begin{cases} \sigma & якщо \ \ b = b' \\ \sigma\pi^{-1} & якщо \ \ b = 0, b' = 1 \\ \sigma\pi & якщо \ \ b = 1, b' = 0 \end{cases} \qquad (1.2)$$

Крок 4. Віктор приймає тоді і лише тоді, коли $H = (G_{b'})$

Розглянутий протокол має суттєвий недолік, пов'язаний із значним обсягом даних, що передаються при виконанні протоколу. Таким чином його практичне використання є недоцільним, однак він цікавий з теоретичної точки зору.

Одним із найпопулярніших протоколів з нульовим знанням на сьогодні є протокол zk-SNARK, що використовується у багатьох сучасних криптовалютах. zk-SNARK є представником неінтерактивних протоколів з нульовим знанням. На відміну від інтерактивних протоколів, у неінтерактивних протоколах Петті, що доводить знання секретного твердження, відправляє лише одне повідомлення (зазвичай математичне доведення) до Віктора, який приймає доведення або ні. Якщо розглядати з сторони конфіденційності, то дана форма є більш безпечною, оскільки відсутність взаємодії сторонами дає більшу впевненість в тому, що витоку інформації під час взаємодії не відбудеться [15].

zk-SNARK складається з трьох ефективних алгоритмів (G, P, V).

Генерування ключів G отримує на вхід секретний параметр λ та програму c і генерує два загальнодоступні ключі (англ. publicly available keys), p_k ключ для доведення (англ. proving key) та v_k ключ для верифікації (англ. verification key). Дані ключі є публічними і генеруються лише один раз для заданої програми.

Алгоритм побудови доведення P отримує на вхід ключ доведення p_k, публічний параметр x та секретний параметр w (англ. witness). Алгоритм генерує доведення π.

$$\pi = P(p_k, x, w) \tag{1.3}$$

Алгоритм верифікації V отримує на вхід ключ v_k, публічний параметер x та доведення π, після обчислення повертає 1, якщо доведення коректне, 0 – у іншому випадку.

Найбільш ефективним zk-SNARK вважається запропонований Гроссом[16], який побудований для QAP (Quadratic Arithmetic Program) та працює в білінійних групах.

Завдяки своїй структурі та особливостям алгоритму протокол є достатньо стійким до переважної більшості сучасних атак. Окрім того zk-SNARK має одні з найкращих показників швидкодії та невеликий обсяг даних, що передається при автентифікації.

Головним недоліком протоколу zk-SNARK є потреба у етапі налаштування перед виконанням автентифікації. Завдяки цьому етапу протокол вважають теоретично вразливим до атак з використанням квантових комп'ютерів [17].

На відміну від zk-SNARK, zk-STARK – прозорий протокол, тобто не вимагає попереднього налаштування і розкриття інформації третій стороні. Технологія zk-STARK дозволяє сильно прискорити процес обміну інформацією та усуває необхідність попереднього довіреного налаштування, що в попередніх протоколах ставило під загрозу конфіденційність всієї системи. Зараз нова технологія розробляється провідними фахівцями компанії StarkWare. Протокол zk-STARK використовує нову технологію інтерактивного доведення з оракулом або IOP (Interactive Oracle Proofs). Системи, засновані на інтерактивних доведеннях з оракулом, є об'єднанням систем з інтерактивними доведенням та систем з ймовірнісним доведенням [17].

Нехай, Петті хоче довести Віктору, що вона володіє деякими даними, які задовольняють якийсь функції. Тоді вона представляє ці дані у вигляді дерева Меркле і відправляє геш кореня дерева Віктору. Віктор тоді вибирає випадковим чином якусь кількість точок і просить для цих точок надіслати гілки дерева Меркле. Петті обчислює і відправляє необхідні дані. Віктор перевіряє, що ці отримані гілки відповідають тим, які належать початковому кореню дерева. Віктор також перевіряє, що в цих точках значення задовольняють деякій функції перевірки.

Зазначений трьох-кроковий алгоритм може бути перероблений в неінтерактивному доведенні, де Петті відправляє лише одне повідомлення, яке може бути перевірено будь-яким учасником.

Для прискорення процесу верифікації в zk-STARK використовується новий алгоритм швидкої верифікації для інтерактивних ймовірнісних доведень з оракулом або ж FRI (Fast Reed-Solomon Interactive Oracle Proofs of Proximity), який заснований на ідеї використання IOP і коду Ріда-Соломона.

Нехай Віктору необхідно перевірити приналежність N точок деякому поліному $f(x)$ степені D менше N. Алгоритм FRI дозволяє отримати статистичне доведення того, що більшість точок належать поліному $f(x)$. Ідея полягає у тому, щоб представити вихідну функцію як функцію двох змінних $f(x) = g(x,y) = g(x,x^k)$. Тоді поліном $g(x,y)$ матиме менший степінь $D_1 = \frac{D}{k}$. Петті повинна обчислити поліном $g(x,y)$ на множині $[1,2,...N] \times [x^k : 1 \le x \le N]$, що може бути представлено у вигляді матриці $N \times N$. Тоді на діагоналі матриці будуть значення $g(x,x^k) = f(x)$. Віктор обирає певну кількість рядків і стовпців та просить Петті надати фіксовану кількість точок для кожного рядка та стовпця, причому в кожному випадку хоча б одна точка повинна належати діагоналі матриці. Петті надає ці точки разом з обчисленими гілками дерева Меркле, щоб довести, що дані є частиною вихідних даних відповідно до протоколу IOP. Віктор перевіряє, що ці отримані гілки відповідають тим, які належать початковому кореню дерева. Далі Віктор перевіряє чи отримані від Петті відповідають трьом критеріям [18]:

a) більша частина точок стовпців належить поліномам степені менше $\frac{D}{k}$;

b) більша частина точок рядків належить поліномам степені менше $\frac{D}{k}$;

c) більша частина точок діагоналі належить поліномам степені менше $\frac{D}{k}$.

У разі відповідності Віктор переконаний у тому, що більшість точок діагоналі належать одному поліному степені D. Для зменшення розмірності матриці використовують модульну арифметику. Для збільшення ефективності алгоритму під час обчислень використовують бінарне поле Галуа [19].

Оскільки zk-STARK використовує постквантову криптографію, то прокол zk-STARK стійкий до більшості сучасних атак, в особливості до атак з використанням квантових комп'ютерів. Окрім того zk-STARK не має необхідності у попередньому налаштуванні, що усуває вразливість пов'язану із необхідністю довіри третій стороні. На відміну від zk-SNARK, zk-STARK легко масштабуються в термінах швидкості обчислень і розміру пам'яті на задачах з великою кількістю обчислень. При цьому в zk-STARK збільшився загальний час виконання протоколу та обсяг даних, що передаються під час автентифікації, але ці параметри у межах допустимих значень.

Bulletproofs – новий неінтерактивний протокол з нульовим знанням, що не потребує попереднього довіреного налаштування та використовує короткі доведення.

Bulletproofs придатний для доведення тверджень щодо фіксованих значень, таких як докази діапазонів, арифметичні схеми тощо. Протокол базується на дискретному логарифмічному припущенні та є неінтерактивним за допомогою евристики Фіата-Шаміра. Основним алгоритмом Bulletproofs є внутрішній алгоритм продукту, представлений Гротом. Алгоритм перевіряє знання двох зв'язаних векторів зобов'язання Педерсена, що відповідають вказаному внутрішньому зв'язку [20].

Bulletproofs спиратися на методику, описану в [20], що дозволяє збільшити ефективність внутрішньої взаємодії при виконанні доведення та зменшує загальну складність комунікацій до $2\log_2(n)$, де n – розмір двох векторів зобов'язань. Протокол Bulletproofs реалізує проведення доведення приналежності до короткого та агрегованого діапазону, використовуючи поліноми. Докази діапазону - це доведення того, що таємне значення (секрет)

належить до певного інтервалу. Докази діапазону не дають жодної інформації про секрет, крім того, що він належить інтервалу.

Алгоритм доведення представлено у вигляді 5 кроків:

Нехай v – значення з діапазоні $[0,n)$ і a_L – бітовий вектор, де $a_L \cdot 2^n = v$. Значення із a_L є бінарними числами з v. На основі побудованого комплементарного вектора $a_R = a_L 1^n$ із дотриманням вимоги $a_L \circ a_R = 0$, Петті надсилає Віктору A та S – сліпі зобов'язання Педерсена до a_L та a_R.

$$A = h \cdot \alpha + g \cdot a_L + \mathrm{h} \cdot a_R \in G \tag{1.4}$$
$$S = h \cdot \rho + g \cdot s_L + \mathrm{h} \cdot s_R \in G$$

Віктор надсилає перевірки y та z для виправлення A та S відповідно.

Петті формує T_1 та T_2 і надсилає до Віктора, де T_1 та T_2 – зобов'язання до коефіцієнтів t_1 поліному t побудованому на основі відомих значень з протоколу.

$$l = l(x) = a_L - z1^n + s_L x \in Z_p^n \tag{1.5}$$
$$r = r(x) = y^n \circ (a_R + z1^n + s_R x) + z^2 2^n \in Z_p^n$$
$$t = l \cdot r \in Z_p$$
$$T_i = gt_i + h\tau_i \in G, \text{ де } i \in \{1,2\}$$

Віктор перевіряє знання значення x у Петті.

Петті повертає до Віктора параметри τ, μ, t, l та r, які Віктор надалі перевіряє та виносить рішення про успішність проходження автентифікації або ні.

$$\tau_x = \tau_2 x^2 + \tau_1 x + z^2 \gamma \in Z_p \tag{1.6}$$
$$\mu = \alpha + \rho \cdot x \in Z_p$$

Розмір доведення ще більше зменшується за рахунок використання спрощеного $O(\log_n)$ доведення внутрішнього продукту.

Аргумент внутрішнього продукту у протоколі дозволяє довести знання векторів l та r, внутрішнім продуктом до яких буде t та зобов'язання $P \in G$ двох вказаних векторів. Таким чином можна замінити виконання останнього кроку протоколу надсилання параметрів τ, μ, t, l та r на відправку на перевірку лише параметрів τ, μ, t та виконання перевірки аргументу внутрішнього продукту.

Тоді замість того, щоб пересилати параметри l та r, що в свою чергу рівносильно передачі $2n$ елементів між Петті та Віктором варто застосувати передачу аргументу внутрішнього продукту, що всього застосовує лише $2\log_2 + 2$ елементи. Тоді в загальному Петті передає лише $2\log_2(n) + 4$ елементи груп та 5 елементів, що належать до Z_p, що спрощує об'єм обміну даними при автентифікації.

Представлений алгоритм на основі доведення діапазону можна застосувати для декількох значень, при цьому суттєво збільшується стійкість, при збільшенні лише об'єм пам'яті до $2\log_2(m)$ для m значень параметра v, на відміну від використання m незалежних доведення приналежності діапазону. Тоді при використанні аргументу внутрішнього продукту об'єм даних, що передаються складатиме $2\log_2(nm) + 4$ елементи груп та 5 елементів, що належать до Z_p [21].

У порівнянні із zk-SNARK та zk-STARK протокол Bulletproofs значно повільніший та потребує значного часу як на роботу на доведення Петті, так і на перевірку доведення Віктором [22, 23].

Для розглянутих протоколів автентифікації з нульовим знанням доцільно виконати порівняння. Порівняння протоколів автентифікації з нульовим знанням здійснено на основі властивостей, якими володіють протоколи. Серед переліку властивостей час виконання протоколів, потреба у початкових налаштуваннях, обсяг даних, що передається при використанні протоколу з стандартними

параметрами, стійкість до деяких атак та особливості структури. Результати порівняння протоколів автентифікації показано в таблиці 1.1.

Таблиця 1.1 – Порівняння протоколів автентифікації з нульовим знанням

№	Властивість	протокол автентифікації				
		Протокол Фіата-Шаміра	Протокол Шнорра	zk-SNARK	zk-STARK	Bulletproofs
1	Потреба у «trusted setup»	+	+	+	-	-
2	Час підтвердження «verification time»	12мс	13мс	10мс-20мс	10 мс	1 сек
3	Час доведення «proof time»	від 2 до 4 сек	від 2 до 4 сек	до 4 сек	до 2 сек	більше 10 сек
4	Розмір даних, що передаються	~135KB	~80KB	288B	45KB-200KB	1.3KB-10KB
5	Перевіряє автентичність двох сторін	-	-	-	-	+
6	Стійкість до атаки за обраним шифр-текстом	+	+	+	+	+
7	Стійкість до атаки «маскарад»	+	+	+	+	+
8	Стійкість до пост квантових атак	-	-	-	+	-
9	Побудований на основі дерев Меркла	-	-	+	+	-

В результаті дослідження та порівняння методів автентифікації з нульовим знанням з’ясовано, що високі показники захищеності мають усі розглянуті методи, однак Bulletproofs має найгірші показники швидкості, а zk-SNARK та zk-STARK побудовані на основі дерев Меркла, що доцільно для систем з подібною

структурою даних. Отже, для подальшого дослідження та моделювання протоколу автентифікації користувачів із нульовим знанням за основу варто взяти протоколи Шнорра та Фіата-Шаміра, що забезпечують оптимальні показники швидкості та безпеки автентифікації.

1.4 Практичне застосування автентифікації з нульовим знанням

Розглянуті протоколи доведення з нульовим знанням мають широке практичне застосування.

Найвідоміший протокол zk-SNARK забезпечує конфіденційність та приватність в Zcash. Zcash – криптовалюта з відритим вихідним кодом, що забезпечує вибіркову прозорість транзакцій. zk-SNARK робить перевірки більш ефективними, використовує менше даних та пам’яті під час перевірки — життєво важлива функція для блокчейну, де пам’ять та простір є дуже важливим для підтримки мережі на плаву. У 2018 році вийшло оновлення, що пришвидшило обчилення та генерацію zk-SNARK підтверджень у п’ять разів [24].

Будь-який із розглянутих протоколів з нульовим знанням можна застосувати у смарт-контрактах для платформи Ethereum. Із розглянутих прикладів для протоколів zk-SNARK, zk-STARK, протоколу Шнорра та спрощеної версії протоколу Bulletproofs уже розроблені та впроваджені відповідні програмні засоби на платформі Ethereum. Велика популярність протоколів з нульовим знанням для цієї платформи зумовлена високою конфіденційністю при виконанні транзакцій між рахунками, і як наслідок – анонімністю [25].

Окрім криптовалют існує безліч інших сфер застосування доведення з нульовим розголошенням, наприклад для аудиту, реалізації шифрування тощо. Деякі компанії вже застосували zk-SNARK у своїх проектах:

- QED-it - ізраїльська компанія, що використовує SNARK для аудиту фінансових установ [23];
- NuCypher - проект ICO, який спеціалізується на повторному шифруванні проксі [22];

– Nuggets забезпечує можливість захисту даних інтернет-магазинів [26].

Автори zk-STARK вважають корисним використовувати його в тому числі для вирішення проблеми DNA profile match (DPM), тобто для перевірки наявності ДНК людини в уже існуючих базах даних. Наприклад, для перевірки поліцією наявності ДНК кандидатів в президенти в криміналістичних базах . Таким чином, сформовані поліцією докази не вимагають розкриття інформації про бази, де зберігаються ДНК, і про ДНК кандидатів третім особам, при цьому докази підтверджені публічно. Така перевірка проходить швидше повної звірки зразка ДНК з усіма зразками бази даних. Результат перевірки: «немає збігів», «частковий збіг», «повний збіг» публікується і є загальнодоступним [19].

Наразі дві компанії Resistance і StarkWare незалежно один від одного розробляють рішення для роботи децентралізованих бірж на основі технології zk-STARK. У StarkWare Industries збираються запропонувати власне рішення для функціонування децентралізованих бірж, а в Resistance планують запустити власну біржу на основі zk-STARK [26].

Протоколи Фіата-Шаміра та Шнорра використовуються у смарт-картах. Це зумовлено потребою у малій обчислювальній потужності для обраних протоколів та відносно невеликим часом виконання обчислень.

Протоколи також використовуються для забезпечення цілісності інформації, шляхом виконання електронного цифрового підпису. Виконання цифрового підпису з використанням протоколу Шнорра мінімізує залежність обчислень, необхідних для створення підпису, від повідомлення. У цій схемі основні обчислення можуть бути зроблені під час простою процесора, що дозволяє збільшити швидкість підписання. Протокол Шнорра лежить в основі стандарту Республіки Білорусь СТБ 1176.2-99 і південнокорейських стандартів KCDSA і EC-KCDSA [27, 28].

1.5 Постановка задачі дослідження

Забезпечення безпеки при автентифікації користувачів – надзвичайно актуальна задача на сьогоднішній день, оскільки більшість сучасних веб-

ресурсів, мережевих застосунків, засобів обміну повідомленнями та систем іншого роду використовують облікові записи для користувачів, і як наслідок застосовують методи автентифікації користувачів. Відповідно існуючі засоби автентифікації не завжди дозволяють забезпечити високий рівень безпеки, оскільки є вразливими до атак різного роду та мають свої недоліки. Тому виконання досліджень у цій сфері залишиться актуальним ще довгий час.

Для того, щоб підвищити рівень захищеності під час автентифікації користувачів, необхідно розробити нову модель протоколу автентифікації опираючись на ефективні сучасні криптографічні протоколи та алгоритми.

Серед розглянутих методів автентифікації користувачів чималі перспективи демонструють саме протоколи автентифікації на основі доведення з нульовим знанням.

Переваги протоколів з нульовим знанням полягають у повній прозорості та відсутності розголошення будь-якої інформації про користувача, що виконує автентифікацію, що, в свою чергу, не дозволяє зловмиснику використати цілий ряд відомих атак, таких як атака за відомим шифр-текстом, атака «людина по середині», різного роду фішингові атаки, атак «маскарад» тощо. Тому для розробки модулі протоколу автентифікації необхідно за основу обрати протоколи з нульовим розголошенням, що показали найкращі результати під час дослідження.

На основі обраної моделі необхідно розробити модель процесу автентифікації з нульовим знанням для користувачів. Відповідну модель представити у вигляді протоколу.

Для сформованого протоколу автентифікації користувачів із нульовим знанням здійснити аналіз за допомогою BAN-логіки. Опираючись на отримані результати аналізу за допомогою BAN-логіки у разі необхідності здійснити усунення недоліків протоколу автентифікації та провести його повторний аналіз. Здійснити розробку алгоритмів сформованим протоколом автентифікації з нульовим знанням. Розроблені алгоритми реалізувати у вигляді програмного засобу, що відповідатиме таким вимогам:

- програмний засіб повинен бути представленим у програмної бібліотеки;
- забезпечити захищену автентифікацію користувача для обраного ресурсу;
- забезпечити нульове розголошення особистих даних користувача при автентифікації;
- забезпечити можливість роботи на різних операційних системах.

На основі сформованих вимог до засобу автентифікації користувачів із нульовим знанням розроблено технічне завдання.

1.6 Висновки до розділу

Отже, процес автентифікація є надзвичайно важливим, оскільки дозволяє організаціям підтримувати безпеку своїх мереж, дозволяючи лише авторизованим користувачам отримувати доступ до захищених ресурсів, які можуть включати комп'ютерні системи, мережі, бази даних, веб-сайти та інші мережеві програми або служби. В цьому розділі досліджено сучасні методи автентифікації для користувачів, розглянуто основні поняття. Розглянуто види автентифікації, їх особливості.

Виконано дослідження протоколів автентифікації з нульовим знанням, які дозволяють переконати іншу сторону у володінні секретом без його розкриття. Розглянуто алгоритми та структуру протоколів автентифікації з нульовим знанням. Виконано порівняння протоколів автентифікації з нульовим знанням за стійкістю до відомих атак та додатковими властивостями протоколів. Серед досліджених протоколів обрано оптимальні протоколи автентифікації з нульовим знанням. З'ясовано практичне застосування протоколів автентифікації з нульовим знанням.

На основі отриманих у першому розділі результатів досліджень інформаційних джерел можна перейти до моделювання протоколу автентифікації користувачів із нульовим знанням та його подальшого дослідження.

2 МЕТОД АВТЕНТИФІКАЦІЇ З НУЛЬОВИМ ЗНАННЯМ

2.1 Математичний опис процесу автентифікації

Для узагальнення сучасних методів автентифікації доцільно розглянути процес виконання автентифікації з використанням різних криптографічних перетворень та структур автентифікації.

Для формалізованого опису системи використано теоретико-множинний підхід, що дозволяє описувати множину усіх елементів, які входять до неї, та які мають певні властивості та знаходяться у певних взаємовідносинах. Дослідження процесу автентифікації користувачів як сукупності взаємопов'язаних елементів системи дозволяє представити систему автентифікації, як сукупність множин вхідних та вихідних даних, постійних та змінних параметрів системи:

$$SA =< X, Y, B, A, O > \text{, де} \tag{2.1}$$

SA – система автентифікації;

X – множина вхідних даних;

Y – множина вихідних даних;

B – множина постійних параметрів системи;

A – множина змінних параметрів системи;

O – множина виконуваних операцій.

У процесі односторонньої автентифікації користувача приймають участь користувач Петті, що проходить автентифікацію, сторона Віктора, що відповідає за перевірку автентичності Петті. Тоді для сторін Петті та Віктора множини складових системи будуть відрізнятися та належатимуть підмножинами відповідних вказаних вище множин.

$$X^{П}; X^{В} \subset X,\ Y^{П}; Y^{В} \subset Y, \tag{2.2}$$

$$B^{П}; B^{В} \subset B,\ A^{П}; A^{В} \subset A,\ O^{П}; O^{В} \subset O$$

Процес автентифікації користувача будується на основі криптографічних перетворень, що відповідають симетричній або асиметричної криптографії. Також цей процес може бути побудовано на основі доведення з нульовим розголошенням. Тоді теоретико-множинний підхід опису системи автентифікації дозволяє розглянути три системи автентифікації.

Система автентифікації на основі симетричних криптографічних перетворень представлено у вигляді сукупності множин:

$$SA_C =< X_C, Y_C, B_C, A_C, O_C > \tag{2.3}$$

Відповідні множини для системи автентифікації на основі симетричних криптографічних перетворень для сторін Петті та Віктора можуть виглядати таким чином:

Для сторони Петті:

$$X_C^{П} = \{I^B; k; r^B\},\ Y_C^{П} = \{E_C(m)\}, \tag{2.4}$$

$$B_C^{П} = \{E_C; I^B; k\},\ A_C^{П} = \{r^B, m\},\ O_C^{П} = \{O_1, O_2, ..., O_n\}$$

Для сторони Віктора:

$$X_B^{П} = \{I^B; k; E_C(m)\},\ Y_C^{П} = \{r^B; R\}, \tag{2.5}$$

$$B_C^{B} = \{D_C; I^B; k\},\ A_C^{B} = \{r^B, D_C(m); R\},\ O_C^{B} = \{O_1, O_2, ..., O_n\},$$

де:

I^B – ідентифікатор користувача;

k – секретний ключ;

r^B – певне псевдовипадкове значення;

m – повідомлення;

O_i – арифметична операція;

$E_C; D_C$ – алгоритми симетричного криптографічного шифрування;

$E_C(m); D_C(m)$ – зашифроване та розшифроване повідомлення;

R – результат автентифікації.

Процес автентифікації на основі асиметричних криптографічних перетворень застосовує особливості асиметричного шифрування, а саме використання відкритого та секретного ключів. Математичний опис автентифікації на основі асиметричного криптографічного перетворення виглядатиме таким чином:

$$SA_A =< X_A, Y_A, B_A, A_A, O_A > \tag{2.6}$$

Для сторони Петті:

$$X_A^{П} = \{I^B; kS; h(r); P^{kB}(r; I^B)\},\ Y_A^{П} = \{r^*\}, \tag{2.7}$$

$$B_A^{П} = \{P; h; I^B; kS\},\ A_A^{П} = \{r^*; h(r^*); I^{B*}\},$$

$$O_A^{П} = \{O_1, O_2, ..., O_n\}$$

Для сторони Віктора:

$$X_A^B = \{I^B; kB\},\ Y_A^B = \{h(r); P^{kB}(r; I^B); R\}, \tag{2.8}$$

$$B_A^B = \{P; h; I^B\},\ A_A^B = \{r, P^{kB}(r; I^B); h(r); r^*; I^{B*}; R\},$$

$$O_A^B = \{O_1^*, O_2^*, ..., O_n^*\},$$

де:

I^B – ідентифікатор користувача;

kS – секретний ключ;

kB – відкритий ключ;

r – певне псевдовипадкове значення;

P – алгоритми асиметричного криптографічного шифрування;

$P^{kB}(r; I^B)$ – зашифроване алгоритмом P на відкритому ключу kB значення;

$r^*; I^{B^*}$ – розшифровані алгоритмом P на секретному ключу kS значення;

h – алгоритм гешування;

$h(r); h(r^*)$ – геш псевдовипадкових значень;

O_i – арифметична операція;

R – результат автентифікації.

Автентифікації на основі доведення з нульовим розголошенням побудована на основі доведення стороною Петґі стороні Віктора, що Петґі володіє секретною інформацією без розкриття секрету при цьому. При виконанні процесу автентифікації відбувається нульовий розголос секрету Петґі. Математичний опис процесу виглядатиме таким чином:

$$SA_{НЗ} =< X_{НЗ}, Y_{НЗ}, B_{НЗ}, A_{НЗ}, O_{НЗ} > \tag{2.9}$$

Для сторони Петґі:

$$X_{НЗ}^{П} = \{I^B; \alpha; kS; r; n; y\},\ Y_{НЗ}^{П} = \{w; u\}, \tag{2.10}$$

$$B_{НЗ}^{П} = \{U; I^B; kS; n\},\ A_{НЗ}^{П} = \{r; \alpha; W; w; u; y\},$$

$$O_{НЗ}^{П} = \{O_1, O_2, ..., O_n\}$$

Для сторони Віктора:

$$X_{НЗ}^{B} = \{I^B; \alpha; n; u; y\},\ Y_{НЗ}^{B} = \{r; R\}, \tag{2.11}$$

$$B_{НЗ}^{B} = \{Ver; I^B\},\ A_{НЗ}^{B} = \{r; w; u; R; y\},$$

$$O_{НЗ}^{B} = \{O_1^*, O_2^*, ..., O_n^*\},$$

де:

I^B – ідентифікатор користувача;

kS – секретний ключ;

y – відкритий ключ;

α – велике псевдовипадкове число, що відповідає умовам необхідним для роботи методу;

n – дуже велике просте число;

r – певне псевдовипадкове значення;

$W;U$ – арифметичні обчислення, що використовують операції $O_{НЗ}$ за модулем n за участі kS та α;

$w;u$ – результати виконання операцій $W;U$;

Ver – обчислення, що використовують операції $O_{НЗ}^{B}$ за модулем n за участі u та w для підтвердження знання kS;

O_i – арифметична операція;

R – результат автентифікації.

Математичний опис процесу автентифікації на основ трьох різних підходів дозволяє узагальнити особливості кожного з підходів, що надає можливість оцінити можливості кожного з підходів для виконання безпечної автентифікації. Серед розглянутих підходів автентифікації на основі доведення з нульовим розголошенням містить множину з найменшою кількістю постійних та найбільшою кількістю змінних параметрів, що в свою чергу ускладнює процес автентифікації, але підвищує рівень захищеності. Таким чином, процес автентифікації з нульовим знанням доцільно використовувати для систем з підвищеними вимогами до рівня безпеки.

2.2 Метод автентифікації

З розглянутих у першому розділі методів автентифікації, високих результатів захисту показали методи автентифікації з нульовим знанням, що дозволяють забезпечити перевірку автентичності без передачі секрету. Для автентифікації користувача розроблено метод автентифікації з нульовим знанням на основі протоколів Фіата-Шаміра та Шнорра, що описано у [4].

Розглянутий у [4] підхід до автентифікації з нульовим розголошенням секрету поєднує для відомих протоколи автентифікації користувачів із нульовим знанням: протокол Шнорра та протокол Фіата-Шаміра. За описаним підходом два протоколи мають подібну структуру, що дозволяє уніфікувати вхідні та вихідні дані для застосування кожним із протоколів. Однак одночасне використання двох протоколів, хоч на перший погляд і дозволяє збільшити стійкість системи, однак має низку недоліків, наприклад збільшення обсягу даних, що передаються при автентифікації, збільшення часу автентифікації удвічі. Крім того, одночасне використання двох стійких протоколів автентифікації за стійкістю відповідатиме використанню одного з протоколів, але зі збільшенням кількості раундів удвічі.

Оскільки два відомих методи автентифікації з нульовим знанням базуються на різних математичних проблемах: дискретного логарифмування для протоколу Шнорра та складності добування квадратного модуля за складеним модулем, що включає два великих простих множники, які зберігаються в секреті,- для протоколу Фіата-Шаміра, то з міркувань безпеки доцільно під час автентифікації використати вибір одного із зазначених протоколів. При кожній новій автентифікації користувачів вибір протоколу для автентифікації повинен бути псевдовипадковим. Тоді у разі атаки на протокол автентифікації зловмиснику спочатку потрібно буде дізнатися, за обчисленнями якого саме з протоколів, Шнорра чи Фіата-Шаміра користувач проходить автентифікацію.

Під час процесу автентифікації користувач повинен узгодити з сервером початкові дані при автентифікації, що дозволять узгодити обраний протокол автентифікації. Для псевдовипадкового вибору протоколу використано регістр зсуву з лінійним зворотнім зв’язком РЗЛЗЗ [3]. Регістр зсуву при кожній наступній автентифікації користувача надаватиме старший біт послідовності регістру, який приймає два можливих значення, що відповідають протоколу, за яким відбуватимуться подальші обчислення. Для забезпечення синхронізації взаємодії між користувачем, що проходить автентифікацію та сервером початковий стан регістр зсуву з лінійним зворотнім зв’язком для користувача та

сервера повинен бути однаковим, тобто його необхідно узгодити між двома учасниками взаємодії.

Для здійснення встановлення початкового стану регістра зсуву з лінійним зворотнім використано додатковий програмний токен [13]. Токен надається кожному користувачу сервером та надає користувачу відповідне значення початкового стану, що передається у зашифрованому вигляді. Таким чином кожен новий процес автентифікації для користувача можливий лише за наявності токену користувача. Процес автентифікації користувача за таким методом зображено на рисунку 2.1.

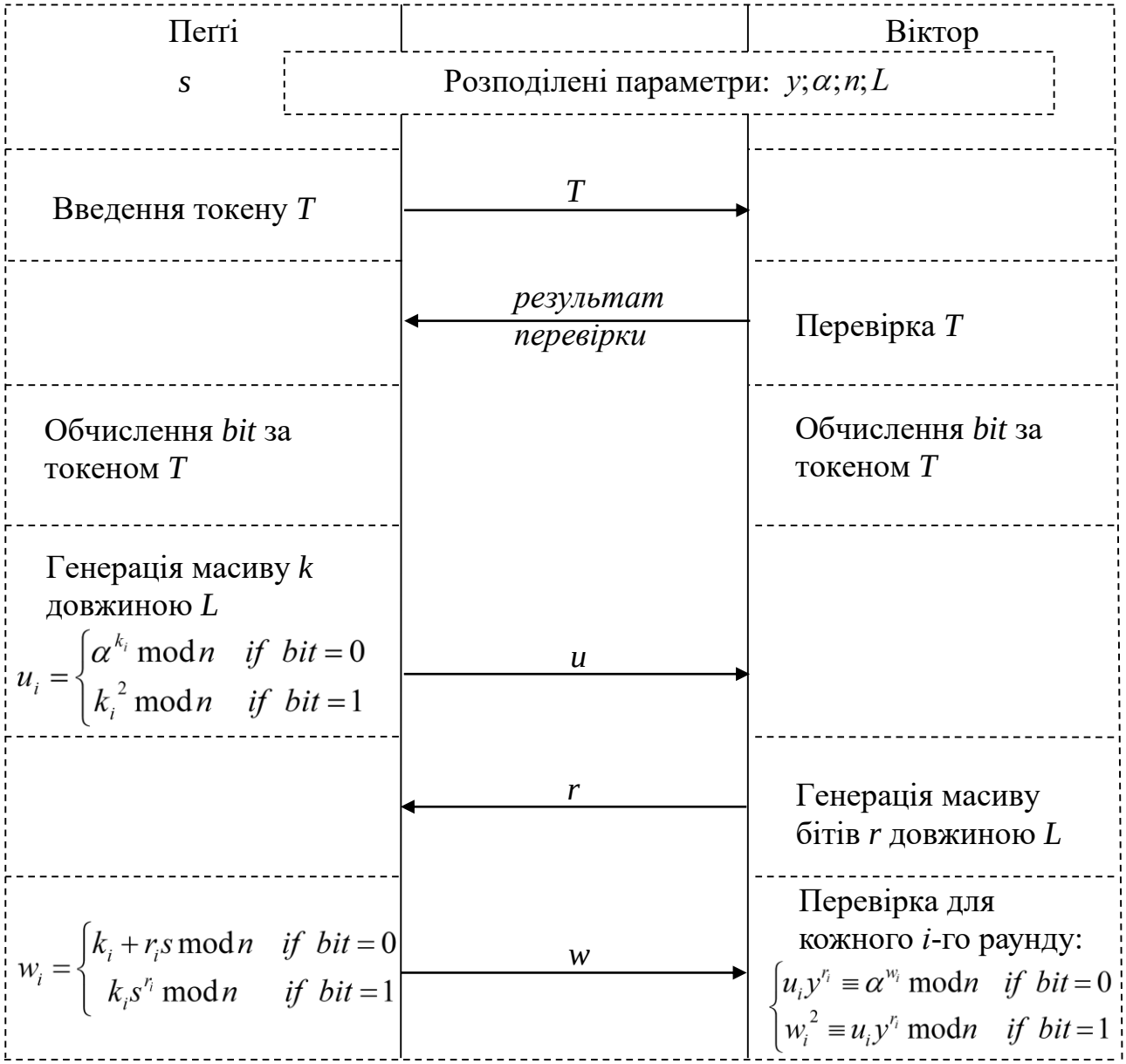


Рисунок 2.1 – Процес автентифікації користувача

Перед початком автентифікації обираються обирається модуль $n = pq$, кількість раундів автентифікації L, секретний ключ s, такий, що $1 \le s \le n-1$ та обчислюється відкритий ключ y, відомий для всіх учасників протоколу. Параметри, що відповідають одному з протоколів, а саме випадкове число, генеруються завчасно розподіляються до початку вибору протоколу автентифікації. Для початку автентифікації користувач повинен володіти токеном T.

Процес автентифікації користувача складається з декількох кроків:

Крок 1. Петті вводить токен доступу T та надсилає Віктору.

Крок 2. Віктор отримує від Петті токен доступу T. Здійснюється перевірка токену та повідомлення Петті про коректність токену. У разі некоректного токену Віктор відмовляє Петті в автентифікації.

Крок 3. Петті та Віктор обчислюють за токеном доступу T та результатом його перевірки псевдовипадкове значення *bit*.

Крок 4. Петті генерує масив псевдовипадкових чисел k довжиною L. Для кожного i-го елемента масиву k Петті обчислює значення u_i за протоколом Шнорра, якщо *bit* дорівнює нулю або за протоколом Фіата-Шаміра, якщо *bit* дорівнює одиниці. Петті надсилає масив u Віктору.

Крок 5. Віктор генерує масив псевдовипадкових бітів r довжиною L. Віктор надсилає масив r Петті.

Крок 6. Для кожного i-го елемента масиву k на основі масиву бітів r та секретного ключа s Петті обчислює значення w_i за протоколом Шнорра, якщо *bit* дорівнює нулю або за протоколом Фіата-Шаміра, якщо *bit* дорівнює одиниці. Петті надсилає масив w Віктору.

Крок 6. Для кожного i-го елемента масиву r на основі відкритого ключа y Віктор виконує перевірку значення w_i та u_i за протоколом Шнорра, якщо *bit* дорівнює нулю або за протоколом Фіата-Шаміра, якщо *bit* дорівнює одиниці. Якщо під час перевірки хоча б одне із значень w_i та u_i не відповідає необхідним умовам перевірки, то Петті отримує негативний результат автентифікації. Якщо

в результаті перевірки всі умови виконуються, то Петті успішно пройшла автентифікацію.

Використовуючи описаний метод автентифікації із псевдовипадковим вибором протоколу автентифікації з нульовим знанням ймовірність порушника успішно пройти автентифікацію зменшується удвічі, оскільки навіть якщо зловмиснику відомий метод автентифікації, йому спочатку потрібно визначити протокол за яким буде здійснено проходження автентифікації, ймовірність чого складає 2^{-1}.

2.3 Метод формування токену автентифікації

Токен користувача використовується під час автентифікації розробленим методом з нульовим знанням, як показано у підрозділі 2.2 та на рисунку 2.1. Кожен новий процес автентифікації для користувача можливий лише за наявності токену користувача.

За генерацію токенів користувачів відповідає серверна частина. Для генерації використовується шифрування та генератори псевдовипадкових чисел криптографічного блоку. Токен складається із відкритої та закритої частин. Структура токену зображена на рисунку 2.2.

Токен користувача

Відкрита частина		Закрита частина		
id	*I*	*Random*	*G*	*Rg*

Рисунок 2.2 – Структура токену користувача

Відкрита частина складається з:

- *Id* – ідентифікатор токену;
- *I* – ідентифікатор користувача.

Закрита частина складається з:

- *Random* – псевдовипадкове число;

- *G* – набори прав розмежування доступу;
- *Rg* – початковий стан регістра зсуву з лінійним зворотнім зв’язком.

Для генерації токену користувач надає серверній частині параметри *I* та *G*, які дозволяють ідентифікувати користувача, для якого генерується токен та набір прав доступу, якими користувач наділяє токен.

Після отримання параметрів від користувача розпочинається процес генерації токену сервером. Значення ідентифікатору *Id* є цілим числом, причому унікальним для кожного токену. Сервер ідентифікує токени за двома параметрами: ідентифікатором *Id* та ідентифікатором користувача *I*.

Параметр *Random* генерується з використанням генератора псевдовипадкових чисел. Це додатковий параметр, що наразі використовується лише для забезпечення унікальності закритої частини токену. Початковий стан регістра зсуву з лінійним зворотнім зв’язком *Rg* генерується з використанням генератора псевдовипадкових чисел в якості великого 32-бітного без знакового числа.

Токен користувача зберігається у вигляді об’єкту JSON, для якого елементи відкритої частини є полями з назвами, що відповідають параметрам, а закрита частина – поле з назвою *edata*. Поле *edata* включає в всі дані закритої частини, збережені у зашифрованому вигляді. Для розшифрування та зашифрування використовується додатковий ключ користувача, який зберігається у клієнта та на сервері.

У токені користувача не зберігаються змінні параметри, однак «час життя» для токенів обмежений та контролюється серверною частиною. Сервер зберігає додаткові поля для всіх токенів у базі даних токнів. До додаткових параметрів токенів відносяться:

- *ttl* – «час життя» токену;
- *status* – статус токену.

Параметр *ttl* вказує скільки разів здійснено операцію відповідним токеном. Коли значення параметру перевищує максимальне значення сервера, встановлене сервером, користувач не може більше користуватись цим токеном.

Параметр *status* вказує на те, чи використовується відповідний токен на поточний момент часу. На основі одного токену в один момент часу можна створити лише одну сесію, тому, якщо у поточний момент часу параметр *status* дорівнює одиниці, то відповідний токен використано для роботи з системою. Якщо значення статусу дорівнює нулю, то відповідний токен можна використати для автентифікації, оскільки користувач поки не створив сесію з системою на основі цього токену. Відповідно після завершення сесії для токену значення статусу змінюється на нуль.

2.4 Аналіз методу автентифікації з використанням BAN-логіки

BAN–логіка – це логіка розроблена для аналізу властивостей «знання» і «довіри» роботи криптопротоколу в цілому чи його окремих частин [29].

Аналіз методу автентифікації за допомогою BAN–логіки дозволяє з’ясувати:

- наявність у методі автентифікації надлишкових кроків, які не впливають на безпеку;
- необхідність включення у метод додаткових кроків;
- необхідність виконання шифрування;
- основний результат, якого можна досягти з використанням методу.

Основні постулати, що застосовуються у BAN–логіці:

1. *P believes X* – *P* вірить у те, що *X* – істинно;
2. *P sees X* – хто-небудь послав *P* повідомлення, що містить *X* і *P* може прочитати і повторити *X* (можливо після розшифрування);
3. *P said X* – *P* коли-небудь посилав повідомлення, що містить *X* і при цьому *P* довіряв *X*, в момент його передавання;
4. *P control X* – *P* має права на *X*;
5. #(*X*) – *X* не було використано в попередніх ітераціях протоколу;
6. $P \xleftrightarrow{K} Q$ – *P* і *Q* розділяють між собою ключ *K* і відповідно *P* і *Q* довіряють один одному;

7. $\xrightarrow{K} P$ – P має відкритий ключ K, що відповідає секретному ключу K^{-1}, який ніколи не буде розкритий іншими учасниками криптопротоколу;
8. $P \xleftrightarrow[X]{} Q$ – секретна формула X відома тільки P і Q і вони можуть використовувати її для ідентифікації один одного;
9. $\{X\}_K$ *from* P – X зашифровано на ключі K, що належить P.

Правила BAN-логіки [29]:

1. $\frac{P\,believes\,Q \xleftrightarrow{K}, P\,sees\,\{X\}_K}{P\,believes\,Q\,said\,X}$ – якщо P вірить, що Q і P розділяють між собою секретний ключ K, і P бачить повідомлення X, зашифроване на ключі K, то P вірить, що Q послав X;
2. $\frac{P believes Q control X, P believes Q believes X}{P believes X}$ – якщо P вірить, що Q має права на X і P вірить Q , який довіряє X, то P довіряє X;
3. $\frac{P believes \#(X), P believes Q said X}{P believes X}$ – якщо P вірить у те, що X до цього не використовувалося і, що Q, якому він вірить послав X, то P довіряє X.

На основі вищенаведених постулатів необхідно виконати перевірку запропонованого протоколу автентифікації для забезпечення перевірки цього методу щодо забезпечення необхідних умов, які висуваються до протоколів автентифікації.

Для представлення методу автентифікації у ідеалізованій формі вилучимо з нього кроки, під час яких дані передаються у відкритому вигляді.

Тоді першим кроком методу буде передача токену T, що містить зашифровану інформацію від Петті (P) до Віктора (V):

$$P \to V : T\,(I_B; G; E_{Kt}(Rg))$$

Для обґрунтування коректності протоколу з точки зору BAN-логіки було введено такі позначення:

$P \xleftrightarrow{Kt} V$ – секретний ключ *Kt* для токену розподіляється між Петті та Віктором перед початком автентифікації.

Rg – значення початкового стану РЗЛЗЗ.

$\{Rg\}_{Kt}$ – зашифроване на ключі *Kt* значення *X*.

$\{Rg\}_{Kt}\ from\ P$ – зашифроване на секретному ключі *Kt* секретне значення, яке Віктор отримав із токену Петті.

$V\ sees\ \{Rg\}_{Kt}\ from\ P$ – сторона V отримала зашифроване на секретному ключі секретне значення саме від сторони P і може його прочитати та розшифрувати за допомогою секретного ключа.

Наступний крок методу буде передача u, обчисленого за одним з протоколів

$$P \to V : u = U(k) \bmod n$$

$P \xleftrightarrow{n} V$ – велике число розподіляється між Петті та Віктором перед початком автентифікації.

k – псевдовипадкове число, що відомо лише Петті.

$V\ sees\ u\ from\ P$ – сторона V отримала обчислене число від P і може прочитати його.

$$P \to V : w = W(k; kS; r) \bmod n$$

kS – секретний ключ Петті.

r – псевдовипадкове число, що відомо Петті та Віктору.

$V\ sees\ w\ from\ P$ – сторона V отримала обчислене число від P і може його прочитати його.

$$V : Ver(u; w; y) \bmod n$$

$\xrightarrow{y} P$ – відкритий ключ Петті, що надається Віктору перед початком автентифікації.

Ver – операція перевірки правильності обчислень $u; w$.

$V\ believes\ Ver$ – сторона V вірить у достовірність операції Ver.

Отримано ідеалізована форма протоколу автентифікації:

$$P \to V : P \xleftrightarrow{Kt} V, V\ sees \{Rg\}_{Kt}\ from\ P, P \xleftrightarrow{n} V, \xrightarrow{y} P, \\ V\ sees\ u\ from\ P, V\ sees\ w\ from\ P \tag{2.13}$$

Таким чином, з урахування вище викладаним потулатам випливає:

$$\frac{V\ believes\ P \xleftrightarrow{Kt} V, V\ sees \{Rg\}_{Kt}\ from\ P}{V\ believes\ P\ said\ Rg} \tag{2.14}$$

$$\frac{V\ believes\ P \xleftrightarrow{y} V, V\ sees \{u\}_{y}}{V\ believes\ P\ said\ u} \quad \frac{V\ believes\ P \xleftrightarrow{r,y} V, V\ sees \{w\}_{r,y}}{V\ believes\ P\ said\ w}$$

Отже, якщо сторона V вірить, що між V та P попередньо розподілений секретний ключ Kt і V може прочитати і повторити значення $\{Rg\}_{Kt}$, що зашифровано на ключі, який належить P, то V вірить, що P надіслав значення Rg.

Якщо сторона V вірить, що між V та P попередньо розподілений відкритий ключ y, V отримує обчислені значення u та w розраховані з використанням випадкових параметрів та ключа сторони P та V може перевірити значення u та w за відкритим ключем y, то V вірить, що саме сторона P надіслала значення u та w, а тому сторона P володіє секретним ключем kS.

За результатом BAN–логіки з'ясовано, що розроблений метод автентифікації користувачів із нульовим знанням коректний, оскільки дозволяє переконати сторону V у автентичності сторони P.

Метод автентифікації не потребує включення додаткових кроків, оскільки виконує поставлену задачу з високою стійкістю, а додаткові кроки суттєво погіршать швидкість проходження автентифікації.

2.5 Розмежування прав доступу

Токен автентифікації користувача використовується не лише для узгодження протоколу автентифікації. Токен містить додаткові параметри, які надалі можна використовувати після встановлення сесії між сервером та користувачем. Для збільшення рівня безпеки доцільно у метод автентифікації додати можливість розмежування прав доступу шляхом використання різних токенів автентифікації.

Застосування токенів з різними правами доступу застосовано у сучасній платформі GitHub. Починаючи з серпня 2021 року для виконання команд git замість автентифікації за паролем на платформі запровадили обов'язкову автентифікації за ssh-ключем або токеном доступу [31, 32].

Використання токенів при автентифікації має низку переваг щодо безпеки:

- унікальність – токени є специфічними і можуть створюватися для використання або для певного пристрою;
- можливість відкликання – токени можна відкликати окремо в будь-який час без необхідності оновлення облікових даних, які не змінюються;
- обмеженість – токени можуть мати обмежені набори прав, щоб дозволити лише доступ, необхідний для випадку використання;
- випадковість – токени генеруються на основі генераторів псевдовипадкових чисел та зберігають секретні дані у зашифрованому вигляді.

Кожен користувач може сформувати токени для автентифікації. Для одного користувача може бути створено декілька токенів, з різним набором прав доступу. Завдяки використанню токенів з обмеженими правами на доступ користувачі можуть підвищити рівень безпеки при користуванні системою у різних ситуаціях. Наприклад, використання токену з обмеженими правами

дозволяє забезпечити безпеку при взаємодії з системою на сторонніх пристроях, що не належать користувачу або під час використання недовірених мереж.

Для розробленого методу автентифікації розмежування прав доступу на основі токенів користувача представлено на рисунку 2.3.

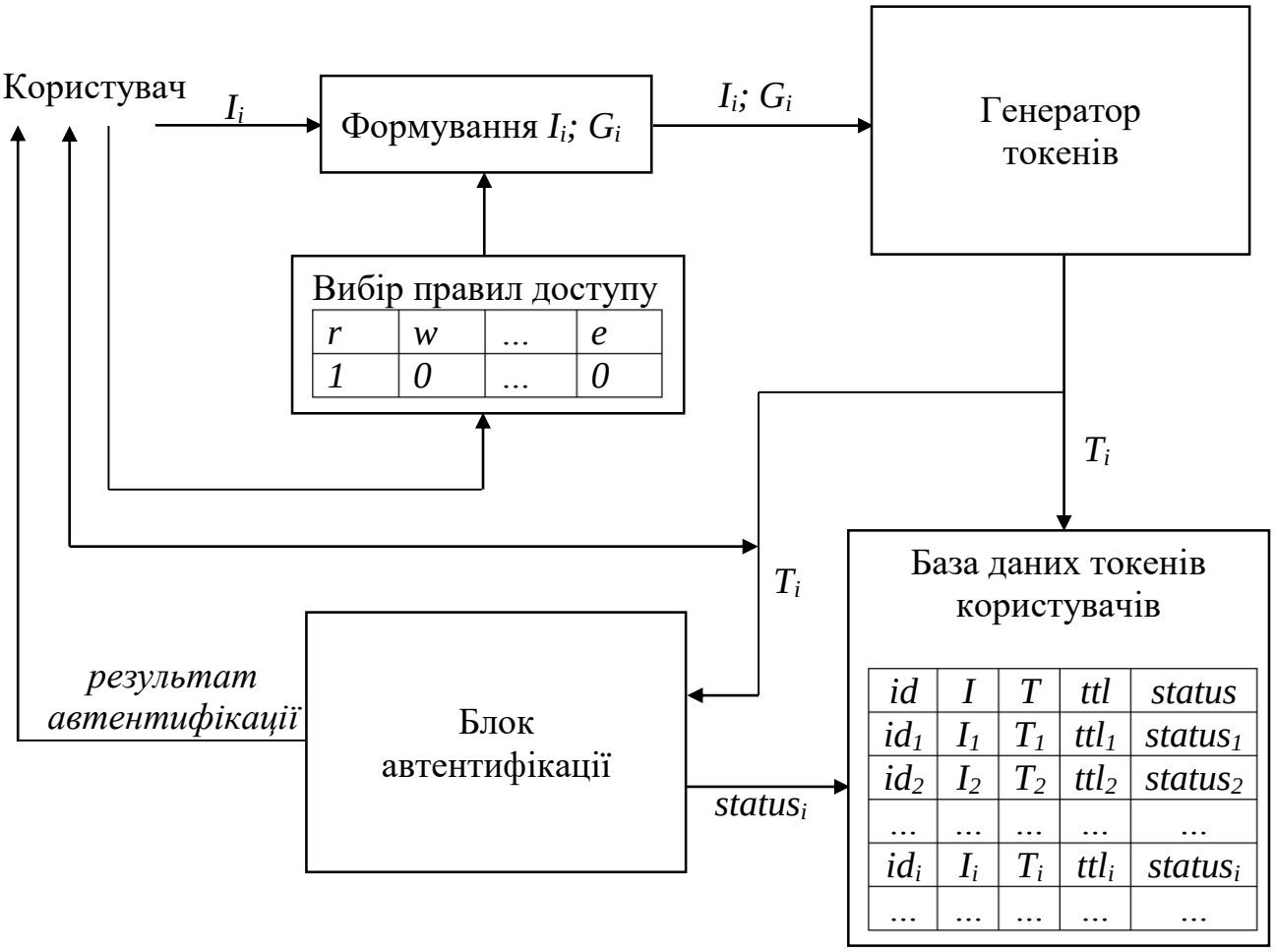


Рисунок 2.3 – Процес використання токенів при розмежуванні доступу

Для створення нового токена користувач здійснює вибір правил доступу, які надаватимуть у разі проходження автентифікації за створеним токеном. Вибір правил доступу здійснюється з множини усіх можливих наборів правил доступу. На основі обраних правила та ідентифікатора користувача генерується токен T_i та зберігається у базу даних. База токенів зберігає усі токени користувачів та додаткові параметри: час життя токену, його статус. За новим токеном користувач проходить процес автентифікації. У разі позитивного

результату автентифікації з нульовим знанням для нового токену змінюється статус у стан «активного» та оновлюється час життя токену. Надалі після автентифікації з використанням нового токену, користувачу надаватиметься лише набір прав, на основі яких було сформовано токен автентифікації.

2.6 Висновки до розділу

У цьому розділі виконано математичний опис процесу автентифікації користувачів для трьох загальновідомих методів автентифікації на основі криптографічних перетворень. Для формалізованого математичного опису системи використано теоретико-множинний підхід.

Розроблено метод автентифікації користувачів із нульовим знанням, що базується на псевдовипадковому використанні одного з відомих протоколів Шнорра та Фіата-Шаміра. Описано процес вибору протоколу автентифікації за розробленим методом з нульовим знанням. Описано процес автентифікації за розробленим методом з нульовим знанням. Детально описано особливості використання токену при автентифікації користувачів за розробленим методом.

Виконано аналіз методу автентифікації з нульовим знанням шляхом використання BAN-логіки. За результатом аналізу BAN–логіки з’ясовано, що розроблений метод автентифікації користувачів із нульовим знанням є надійним та дозволяє переконати серверну сторону у автентичності користувача. З’ясовано, що необхідна умова для використання розробленого методу автентифікації виконується.

Для розробленого методу автентифікації описано можливості використання в якості системи розмежування прав доступу. Описано відомі методи розмежування прав доступу. Описано процес розмежування прав доступу на основі токену. Для розробленого методу автентифікації здійснено розмежування прав доступу на основі токенів користувачів.

Перейдемо до опису алгоритмів роботи засобу автентифікації користувачів із нульовим знання.

3 АЛГОРИТМ РОБОТИ ЗАСОБУ АВТЕНТИФІКАЦІЇ КОРИСТУВАЧІВ

3.1 Архітектура засобу автентифікації

Засіб автентифікації користувачів із нульовим знанням повинен виконувати автентифікацію користувача розробленим методом автентифікації. Засіб автентифікації користувачів представлений у вигляді серверної та клієнтської частин.

Серверна частина засобу складається з семи основних компонентів: криптографічного блоку, блоку керування, інтерфейсу обміну даними, блоків генерування та перевірки токенів, бази даних та регістра зсуву з лінійним зворотнім зв’язком. Архітектура розробленого засобу автентифікації для серверної частини зображена на рисунку 3.1.

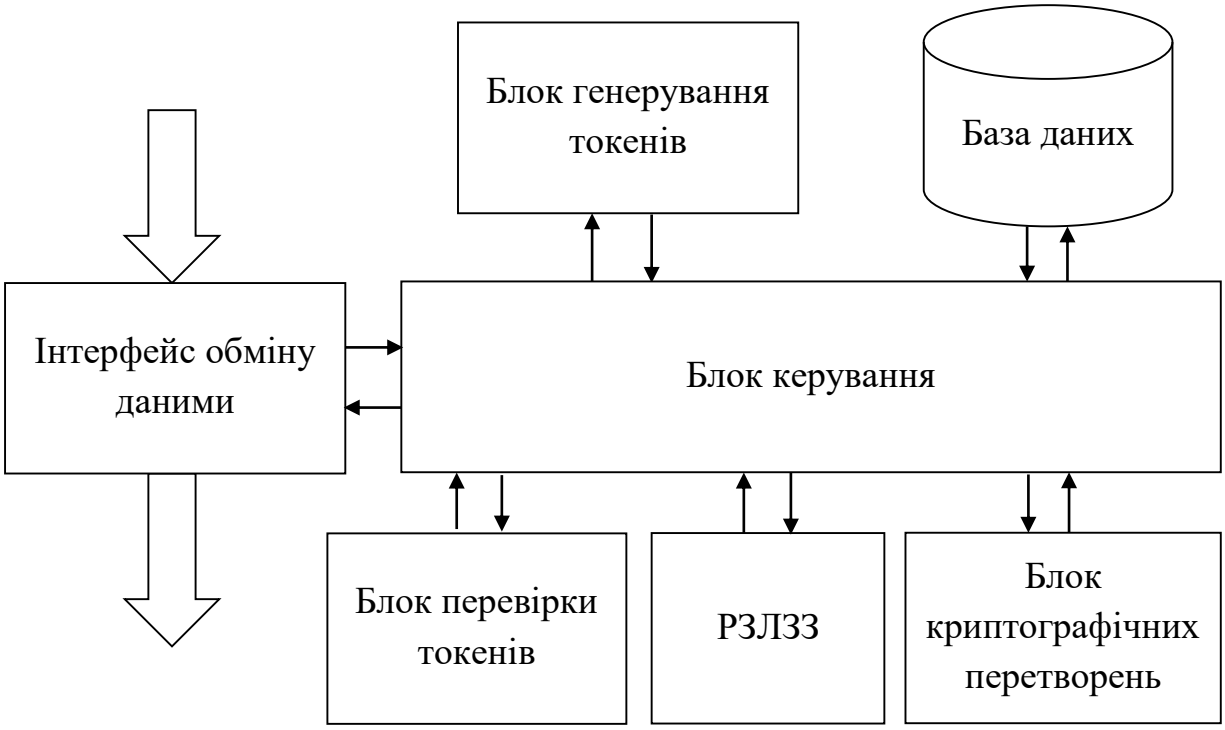


Рисунок 3.1 – Структура засобу автентифікації для серверної частини

Блок криптографічних перетворень реалізує криптографічні примітиви методу автентифікації. Для кожного кроку автентифікації обчислення виконуються криптографічним блоком над даними, отриманими від блоку

керування. Результати обчислень передаються на блок керування, який надалі виконує їх обробку та продовжує виконання процесу автентифікації. Окрім обчислень за протоколом автентифікації блок криптографічних перетворень також виконує алгоритм блокового зашифрування та розшифрування, що використовується під час генерації та перевірки токенів.

Блок керування відповідає за обробку даних, що передаються з інших блоків. Блок керування – центральний блок серверної частини, що взаємодіє з усіма іншими блоками. Блок керування формує набір команд для виконання наступного кроку у процесі автентифікації та передає відповідні команди та набір даних на потрібні блоки. Після виконання обчислень відповідно до кроку протоколу та отримання результату блок керування здійснює передачу відповіді на інтерфейс обміну даними до користувача. Блок керування також забезпечує взаємодію засобу із базою даних та формує відповідні SQL-запита до бази даних.

Блок генерування токенів виконує генерацію нових унікальних токенів користувачів. Блок керування надає блоку генерування токенів команду, ідентифікатор користувача, що прагне отримати новий токен, набір прав доступу, що будуть надані користувачу при використанні нового токену, та секретний ключ для виконання шифрування секретної частини токену. Завдяки блоку керування блок генерування токенів взаємодіє із криптографічним блоком та виконує криптографічні перетворення. Блок генерування токенів також містить генератор псевдовипадкових чисел, що використовується для забезпечення унікальності токенів. Після генерації новий токен передається до блока керування, який надалі надає зберігає його до бази даних та передає користувачу через інтерфейс обміну даними.

Інтерфейс обміну даними забезпечує передачу запитів та відповідей між клієнтом, що проходить автентифікацію, та сервером. Дані подаються на блок керування, який надалі здійснює їх обробку та виконує наступний крок для виконання.

Блок перевірки токенів з блоку керування отримує токен користувача, який необхідно перевірити. Блок перевірки токенів через блок керування взаємодіє з

базою даних, де виконує пошук токену, що відповідає наданому користувачем. Відповідні токени перевіряються за допомогою компараторів за відповідними відкритими частинами та секретній частині. Для токену виконується перевірка на можливість його подальшого використання на основі змінних параметрів токену, що отримано з бази даних. Результати перевірки надаються до блоку керування.

Регістр зсуву з лінійним зворотнім зв'язком (РЗЛЗЗ) використовується для узгодження алгоритму, за яким здійснюватиметься автентифікація користувача. РЗЛЗЗ отримує від блоку керування початковий стан регістру та кількість тактів, яку необхідно виконати. Після виконання всіх тактів значення першого біта регістра повертається до блоку керування. Надалі блок керування формує команду до блоку криптографічних перетворень в залежності від значення отриманого біту.

Клієнтська частина засобу складається з блоків керування, криптографічних перетворень, регістра зсуву з лінійним зворотнім зв'язком, інтерфейсів взаємодії з користувачем та обміну даними, що зображено на рисунку 3.2.

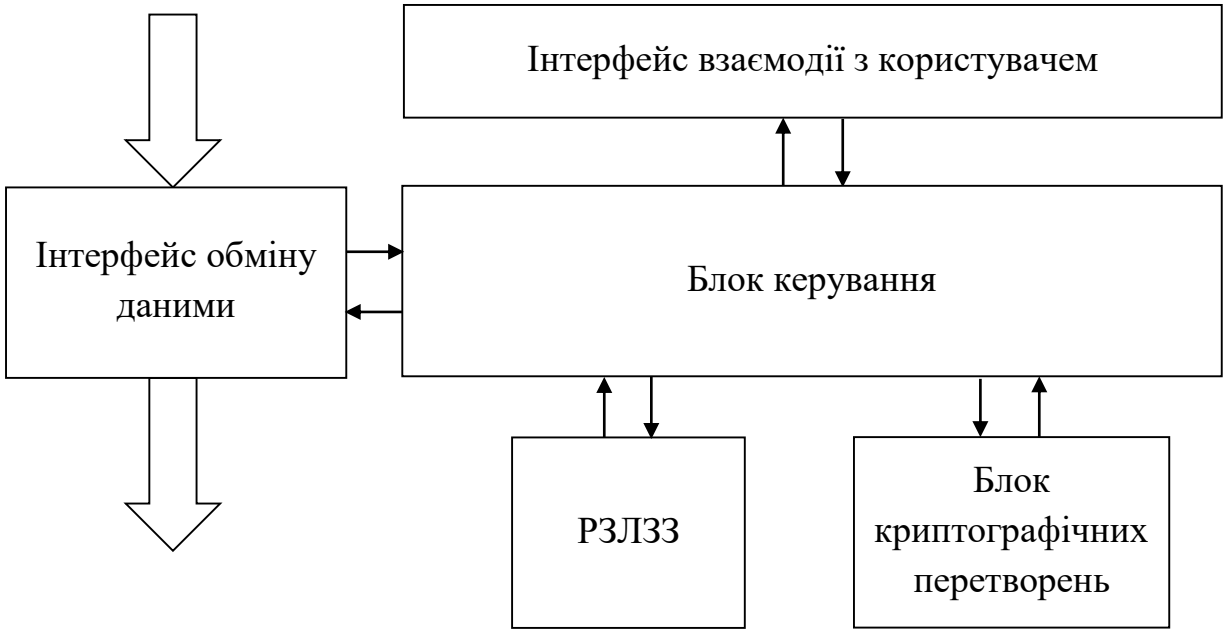


Рисунок 3.2 – Структура засобу автентифікації для клієнтської частини

Структура блоку криптографічних перетворень на клієнтській частині повністю збігається з його відповідною структурою на серверній частині. Інтерфейс обміну даними для клієнтської частини налаштований на зв’язок лише з серверною частиною. Інтерфейс обміну даними отримує відповіді від відповідного інтерфейсу серверної частини та забезпечує передачу запитів сформованих блоком керування.

Регістр зсуву з лінійним зворотним зв’язком для клієнтської частини ідентичний відповідному РЗЛЗЗ серверної частини. РЗЛЗЗ сервера та клієнта працюють одночасно на основі початкового стану регістру, що міститься у секретній частині токену користувача, що передається з блока керування.

Блок керування клієнтської частини відрізняється від відповідного блоку керування серверної частини іншим набором команд та взаємодій. Блок керування починає свою роботу лише після отримання команди та інформації з інтерфейсу взаємодії з користувачем. Блок керування для клієнтської частини розпочинає процес автентифікації користувача шляхом формування початкового запиту до серверної частини з ідентифікатором користувача та набором прав доступу або у разі наявності токену користувача. Блок керування зберігає проміжні дані, що необхідні при автентифікації користувача, а також дані сесій. Сесійні дані зберігаються у пам’яті пристрою, на якому працює засіб автентифікації користувачів.

Інтерфейс взаємодії з користувачем дозволяє користувачу зазначити дані необхідні для автентифікації, такі як ідентифікатор користувача, токен та секретний ключ або ідентифікатор та набір прав, за якими буде створено новий токен користувача. Після отримання команди з інтерфейсу користувача блок керування розпочинає виконання процесу автентифікації. Після завершення виконання блок керування повертає отримані результати до інтерфейсу користувача, який надає користувачу можливість подальшого ознайомлення та виконання подальших операцій.

3.2 Узагальнений алгоритм засобу

Узагальнений алгоритм засобу автентифікації користувачів із нульовим знанням зображено на рисунку 3.3.

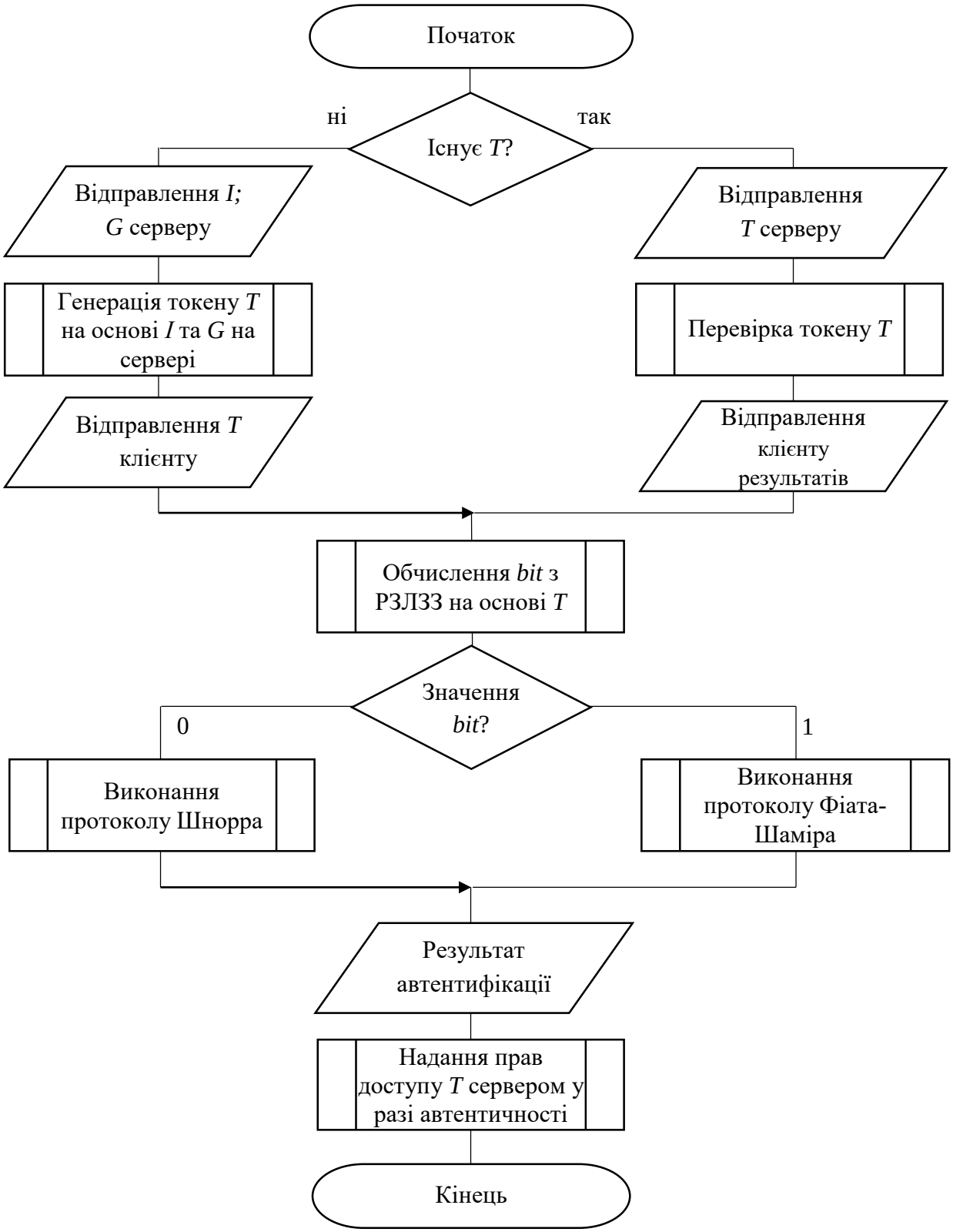


Рисунок 3.3 – Узагальнений алгоритм засобу автентифікації

Засіб автентифікації користувачів із нульовим знанням в першу чергу надає користувачу можливість проходження автентифікації з використанням токену доступу. Якщо користувач володіє токеном, то робота алгоритму засобу починається з введення токену користувача до сервера. Сервер здійснює перевірку токена та повідомляє користувача про валідність наданого токена та можливість подальшого проходження автентифікації. Якщо користувач не володіє токеном, то для проходження автентифікації йому необхідно отримати новий токен на основі ідентифікатора користувача I та набору додаткових параметрів G. Генерація токену відбувається серверною частиною засобу. Після виконання попередніх кроків клієнт та сервер володіють токеном T користувача. На основі T одночасно на клієнтській та серверній частинах встановлюється початковий стан регістрів зсуву з лінійним зворотнім зв’язком. Після роботи регістра зсуву з лінійним зворотнім зв’язком значення останнього молодшого біту *bit* використовується для вибору протоколу подальшого процесу автентифікації. У разі нульового значення здійснюється використання протоколу Шнорра, а якщо значення рівне одиниці – протоколу Фіата-Шаміра. Виконання обраного протоколу надає результат автентифікації користувачу. У разі успішного підтвердження автентичності користувачу з токеном T надаються зазначені у токені T права доступу до системи.

3.3 Алгоритми використання токену

Розроблений засіб автентифікації з нульовим знанням використовує токени користувача під час процесу автентифікації. Для роботи з токенами на серверній частині засобу застосовуються блоки генерації та перевірки токенів. Усі токени користувачів зберігаються у бази даних токеній, яка також розташована на серверній частині засобу.

Процес генерації токена розпочинається із надання клієнтом ідентифікатора користувача I, для якого необхідно створити токен та набору параметрів G до складу якого входять права доступу, що надаються після

проходження автентифікації за новоствореним токеном. Алгоритм генерації токену користувача представлено на рисунку 3.4.

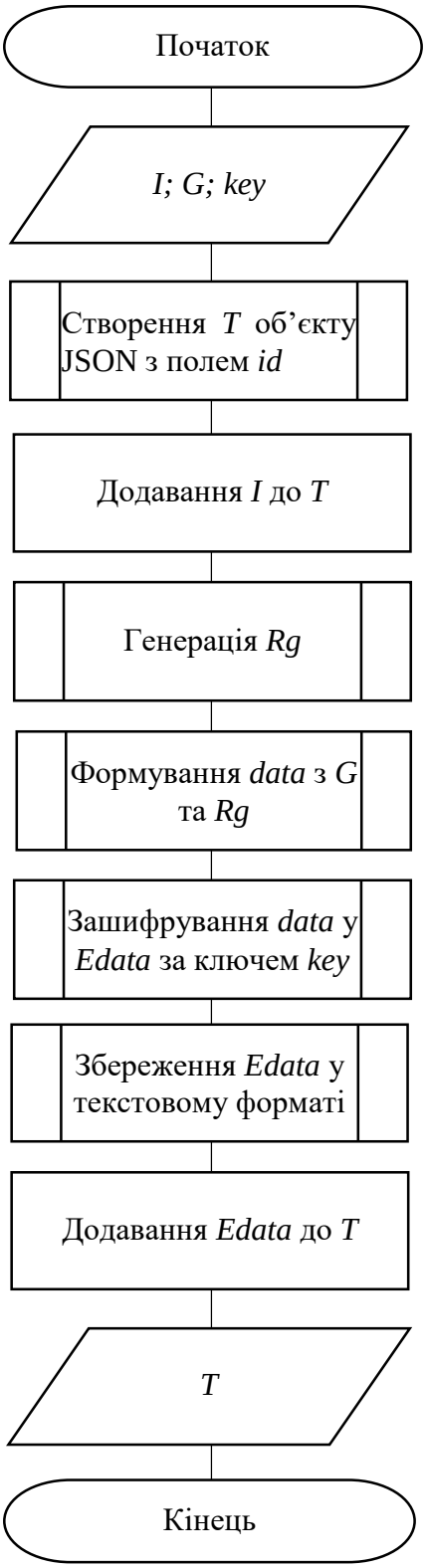


Рисунок 3.4 – Алгоритм генерації токену

Після отримання вхідних даних від клієнтської частини сервер розпочинає генерацію токену у форматі JSON. Новоствореному JSON об’єкту *T* відразу надається унікальний ідентифікатор *id*, що зберігається у вигляді поля JSON об’єкту та генерується з використанням генератора псевдовипадкових чисел. Під час генерації *T* перевіряється унікальність ідентифікатора. Після генерації до об’єкта *T* додається поле із ідентифікатором користувача *I*. Ідентифікатор токену та користувача зберігаються у відкритому вигляді та складають відкриту частину токену.

Формування закритої частини токена розпочинається із генерації початкового стану регістра зсуву з лінійним зворотнім зв’язком. З використанням генератора псевдовипадкових чисел початковий стан для РЗЛЗЗ зберігається у вигляді 32 бітного числа *Rg*. Сервер формує повідомлення *data* з додаткового параметру *G* та числа *Rg*. Повідомлення також представляється у вигляді JSON об’єкту з відповідними полями. Новостворений об’єкт *data* зашифровується з використанням секретного ключа key у повідомлення *Edata.*

В якості алгоритму шифрування обрано блоковий шифр AES-256, забезпечує високий рівень безпеки секретної частини токену користувача [33]. Для шифрування методом AES-256 використано додатковий секретний ключ *key* користувача, який розподілений між сервером та користувачем, однак не приймає безпосередньої участі у процесі автентифікації користувача. Додатковий ключ використовується лише для роботи з токенами.

Після шифрування *Edata* необхідно представити у текстовому форматі після чого *Edata* додається до об’єкта *T* в якості поля, що безпосередньо складає закриту частину токену. В результаті отримано *T* –повний токен користувача у форматі JSON. Токен *T* передається користувачу.

Після отримання від користувача токену сервер повинен виконати його перевірку для перевірки дійсності токену. Алгоритм перевірки токену зображено на рисунку 3.5.

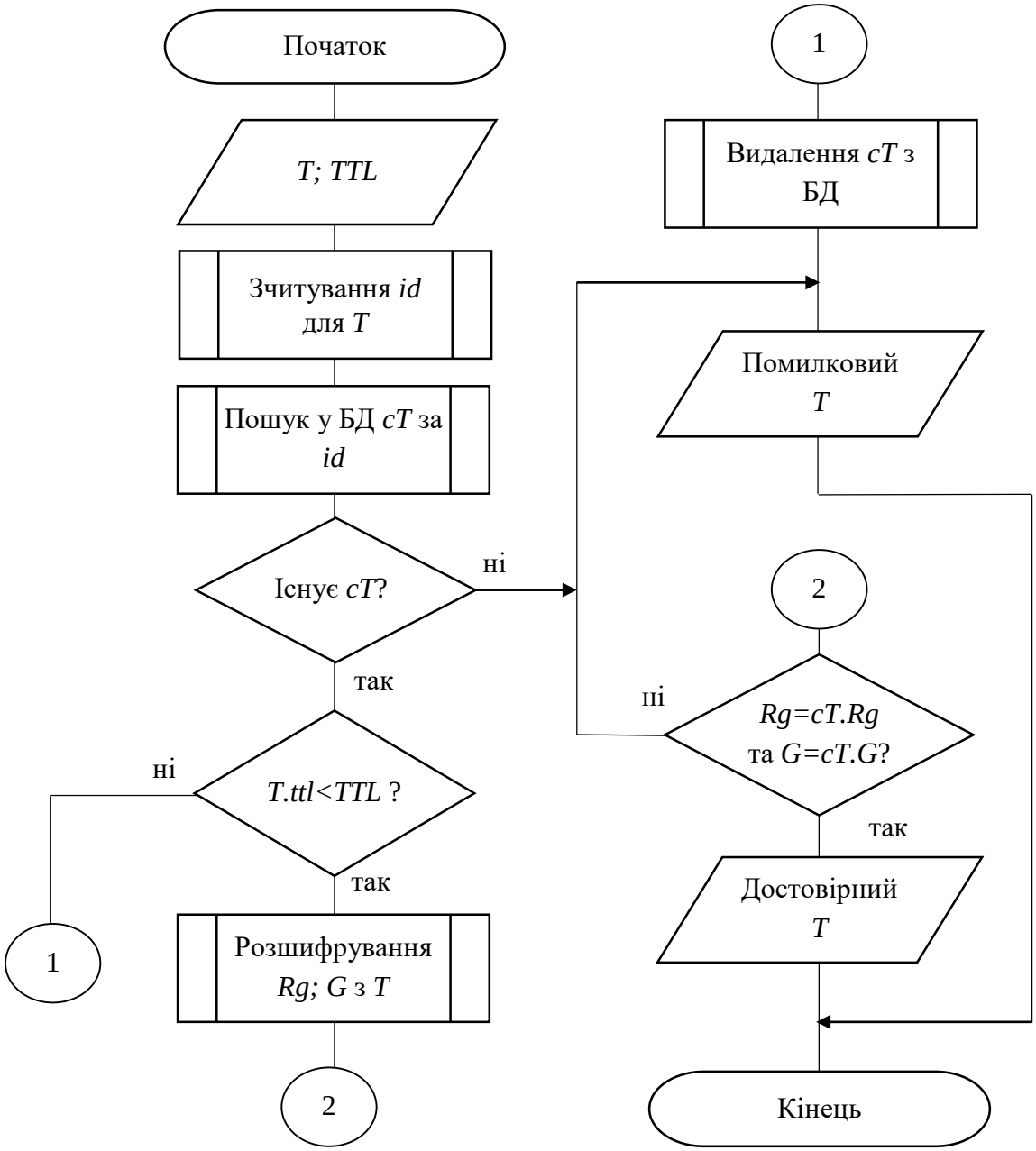


Рисунок 3.5 – Алгоритм перевірки токену

Токен користувача подається на блок перевірки токенів. Після отримання токену *T* у вигляді JSON з токену зчитується значення його ідентифікатора *id*. На серверній частині здійснюється пошук *cT* відповідного токену з бази даних токенів, що має ідентифікатор токену *T*. Якщо такого токену не існує, то користувач надав невірний токен. У іншом випадку відбувається порівняння «часу життя» *ttl* токену з максимальним значенням. Параметр *ttl* збільшується щоразу після проходження автентифікації. Якщо *ttl* більший за максимальний, то

токен більше використовувати неможна, тому він видаляється з бази даних та сервер повідомляє користувача. Інакше здійснюється перевірка закритої частини токену. Закрита частина розшифровується та порівнюється з відповідними компонентами з бази даних. У разі відповідності токен користувача вважається достовірним.

3.4 Алгоритми автентифікації користувачів

За розробленим у другому розділі методом процес автентифікації користувача виконується за одним із відомих протоколів з нульовим знанням. Вибір протоколу за яким буде виконано автентифікацію виконується псевдовипадковим чином за алгоритмом представленим на рисунку 3.6.

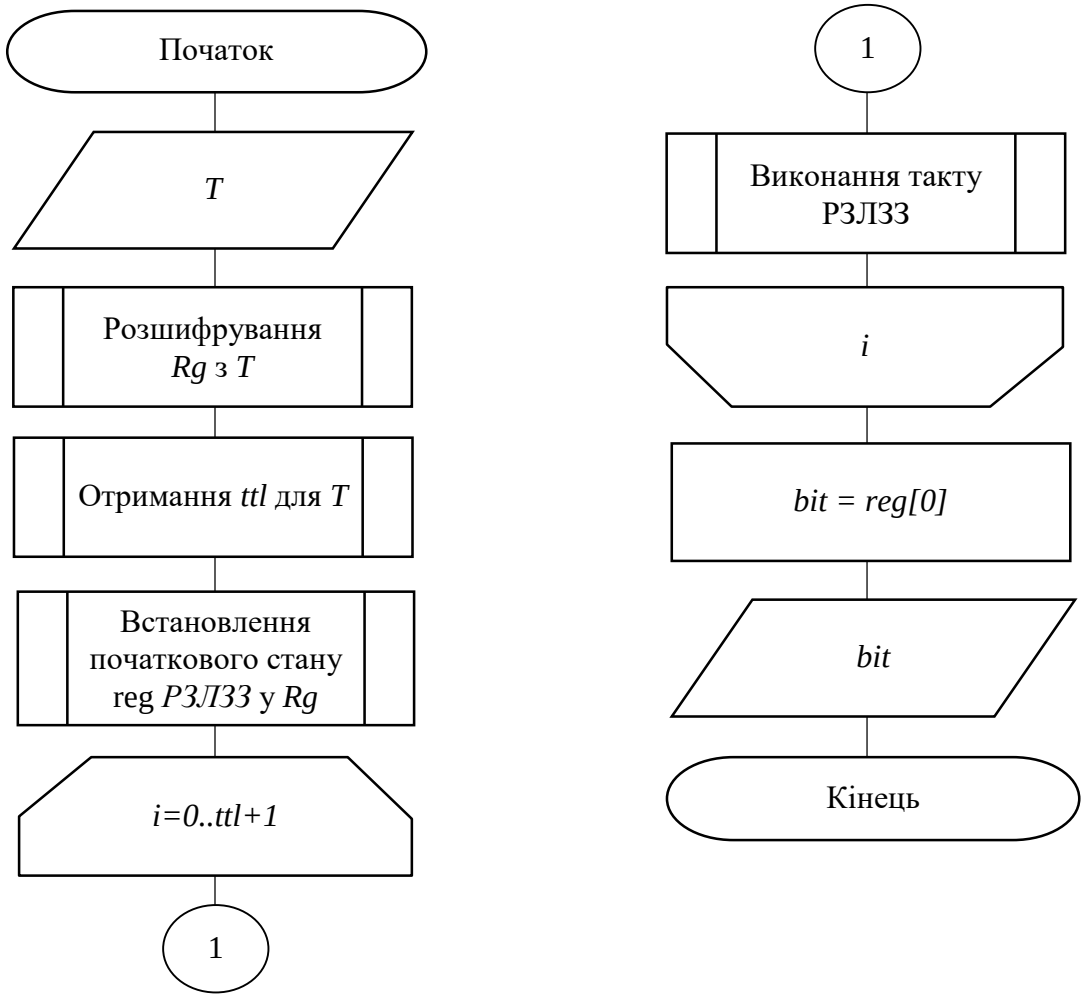


Рисунок 3.6 – Алгоритм генерації параметра вибору протоколу автентифікації

Генерація параметра вибору протоколу автентифікації виконується блоками керування та регістром зсуву з лінійним зворотнім зв’язком. В якості вхідних даних використовується *T* токен користувача, що проходить автентифікацію у форматі JSON. З токену *T* необхідна закрита частина, що розшифровується шифром AES-256. З розшифрованої закритої частини отримується значення початкового стану регістра зсуву з лінійним зворотнім зв’язком *Rg*. Для токену доступу отримується значення часу життя *ttl*. Початковий стан Rg встановлюється у регістр зсуву з лінійним зворотнім зв’язком. Протягом *ttl+1* раундів відбувається виконання такту регістра зсуву з лінійним зворотнім зв’язком. Після завершення виконання тактів з регістра зчитується молодший біт, що зберігається у змінній *bit*. Змінна *bit* повертається на блок керування.

Значення змінної *bit* – параметр вибору протоколу автентифікації з нульовим знанням. Алгоритм генерації параметра вибору протоколу автентифікації однаковий для клієнтської та серверної частин. Після генерації *bit* розпочинається виконання автентифікації користувача.

Алгоритми автентифікації з нульовим знанням для клієнтської та серверної сторін відрізняються. Автентифікація на клієнтській та серверній частинах починається з обрання параметрів (модуля *n*, кількості раундів *L*), секретного ключа *s*, відкритого ключа *y* та псевдовипадкового числа *a*. Параметри обираються до початку виконання раундів автентифікації.

Автентифікація на клієнтській стороні починається з генерації L псевдовипадкових чисел *k[i]*. Для кожного *k[i]* в залежності від значення *bit* виконується обчислення *u[i]*, як *a* в степені *k[i]* за модулем *n*, якщо *bit* приймає нульове значення, та, як *k[i]* в степені два за модулем *n*, якщо *bit* приймає значення одиниці. Отриманий набір чисел *u* надсилається до серверної частини. Клієнтська сторона отримує від сервера число *r* – набір псевдовипадкових біт.

Для кожного значення біту *r[i]* виконується розрахунок числа *w[i]*, як суми *k[i]* та добутку *r[i]* з секретним ключем *s* за модулем n або, як добутку *k[i]* та добутку секретного ключа *s* в степені *r[i]* за модулем n залежно від значення *bit*.

Розраховані значення *w* передаються до серверної сторони. Алгоритм автентифікації для клієнтської частини зображено на рисунку 3.7.

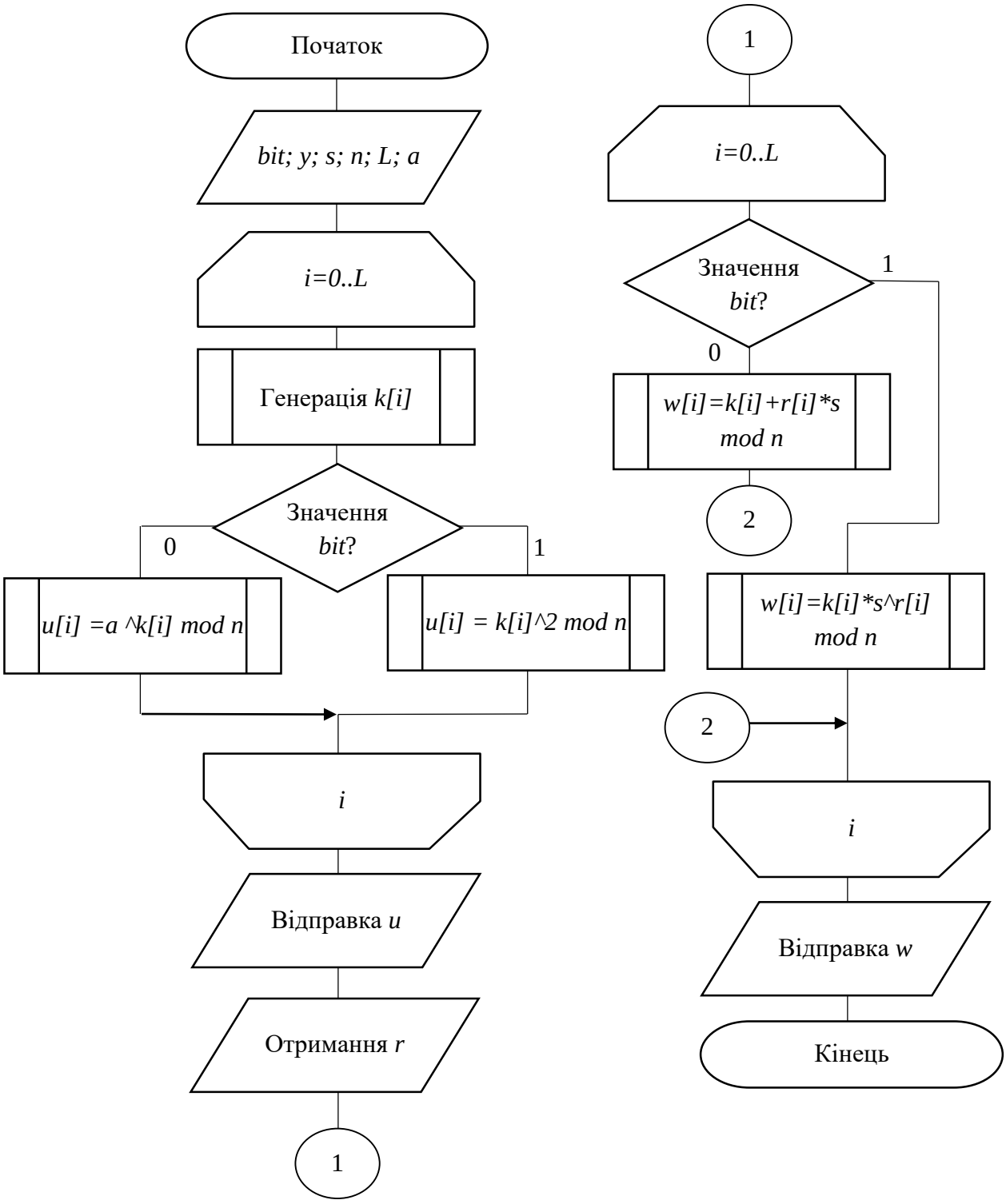


Рисунок 3.7 – Алгоритм автентифікації для клієнтської частини

На серверній частині відбувається перевірка знання клієнтом секретного ключа *s*. Алгоритм перевірки на серверній частині зображено на рисунку 3.8.

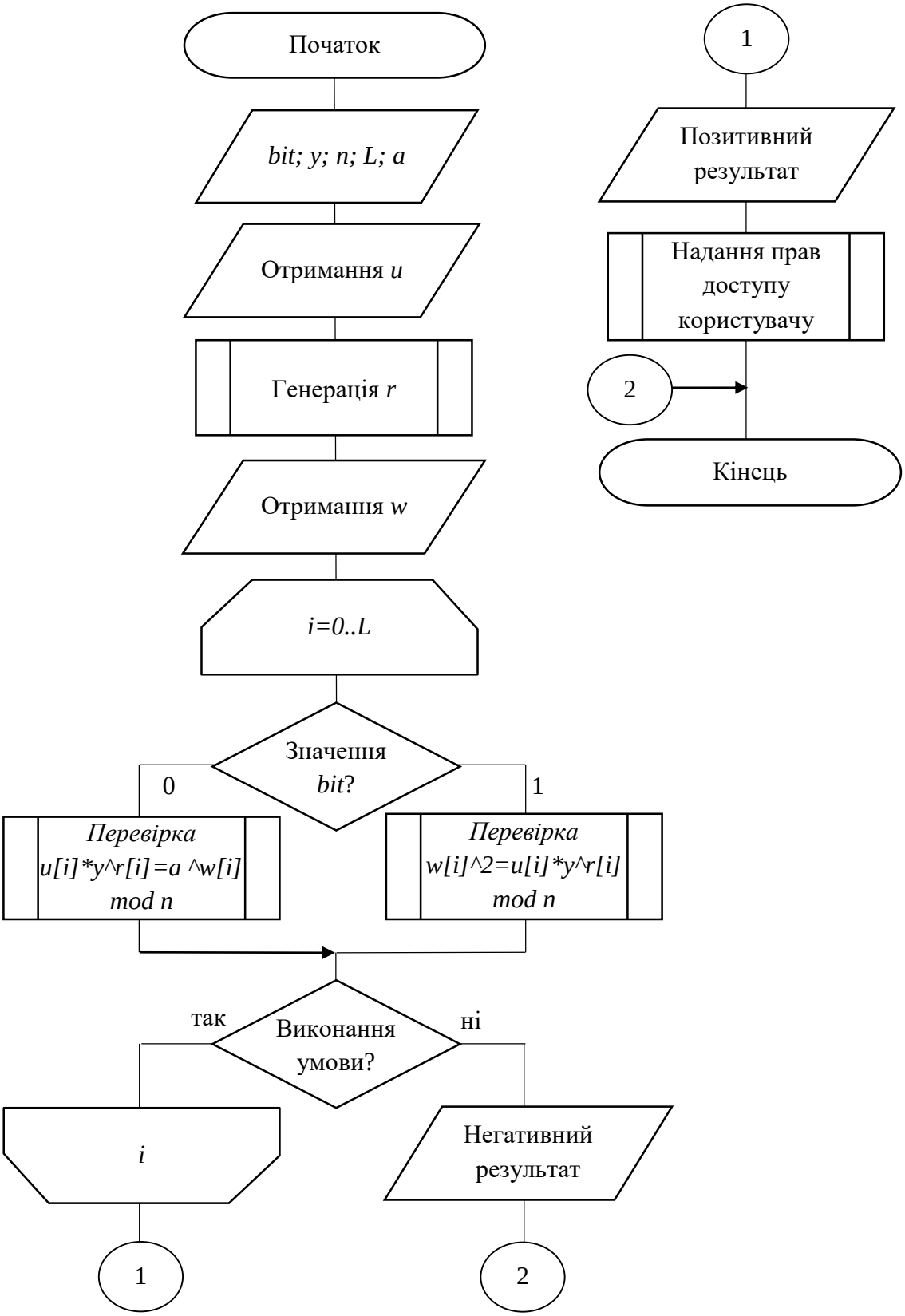


Рисунок 3.8 – Алгоритм автентифікації для серверної частини

Серверна частина отримує від клієнтської повідомлення масивом чисел *u*. Після цього за допомогою генератора псевдовипадкових чисел відбувається

генерація послідовності біт *r*, розміром у кількість раундів *L*. Послідовність біт *r* передається до клієнтської частини. Серверна частина отримує значення *w* від клієнтської частини. Розпочинається перевірка отриманих значень *u* та *w*.

Якщо параметр *bit* приймає нульове значення, то перевіряється виконання співвідношення: *u[i]* помножене на піднесене до степені *r[i]* значення *y* повинно дорівнювати піднесеному до степені *w[i]* за модулем *n* числу *a*.

Якщо параметр *bit* приймає значення одиниці, то перевіряється виконання співвідношення: *w[i]* в квадраті повинно дорівнювати добутку модулем *n* чисел *u[i]* та *y*, що попередньо піднесено до степені *r[i]*.

Перевірка одного зі співвідношень виконується протягом *L* раундів. Якщо при перевірці всіх раундів відбувається виконання умов, то було доведено автентичність клієнта після чого користувачу надається доступ до системи, що відповідає наданому ним токену. У разі першого порушення вказаних умов серверна частина повертає негативний результат автентифікації.

3.5 Висновок до розділу

Отже, у цьому розділі розроблено узагальнену архітектуру засобу автентифікації користувачів із нульовим знанням, що передбачає клієнт-серверну реалізацію засобу. На основі архітектури, сформовано узагальнений алгоритм роботи засобу автентифікації з нульовим знанням. Для засобу автентифікації розроблено алгоритми генерації та перевірки токенів користувачів. Представлено алгоритм виконання вибору протоколу автентифікації з нульовим знанням, відповідно до якого здійснюються обчислення протягом процесу автентифікації користувача. Розроблено алгоритми автентифікації користувача для серверної та клієнтської частин.

Розроблені алгоритми дозволяють перейти до їх програмної реалізації, що складатимуть засіб автентифікації користувачів із нульовим знанням.

4 ЗАСІБ АВТЕНТИФІКАЦІЇ КОРИСТУВАЧІВ ІЗ НУЛЬОВИМ ЗНАННЯМ

4.1 Обґрунтування вибору засобів розробки

Для розробки засобу автентифікації користувачів із нульовим знанням розглянуто мови програмування: Python, JavaScript, Java, Go, C # та C++. Однією з поставлених вимог до методу засобу автентифікації є можливість використання методу автентифікації з нульовим знанням на різних операційних системах. Серед розглянутих мов програмування Python, JavaScript, Java надають можливість виконання коду на багатьох платформах, однак лише мова програмування Java забезпечує високий рівень безпеки за рахунок використання власної платформи JVM. Мова Java забезпечує високий рівень швидкодії та є універсальною для розробки будь-яких застосунків [34].

Переваги та особливості Java:

- об’єктно-орієнтована мова;
- програмне забезпечення з відкритим кодом (OpenJDK);
- строгий контроль за типами даних та виключними ситуаціями;
- забезпечує автоматизований «збір сміття»;
- наявність зручних модулів Maven і Gradle для розгортання застосунків, підключення бібліотек;
- забезпечує можливість запуску застосунку на різних операційних системах завдяки Java-машині;
- велика кількість бібліотек з відкритим кодом;
- забезпечення безпеки завдяки використанню віртуалізації віртуальної машини для виконання байт коду.

Для реалізації обміну даними між серверною та клієнтською частинами засобу автентифікації обрано технологію сокетів та пакет java.net, що використовує протокол TCP для обміну інформацією. Для розробки інтерфейсу клієнтської частини засобу автентифікації обрано платформу JavaFX та бібліотеку JMetro [35, 36].

Для серверної частини застосунку обрано систему керування базами даних MySQL. Система MySQL працює з реляційними базами даних, що доцільно для багатокористувацьких застосунків. Система керування базами даних MySQL забезпечує безпеку даних та високу швидкодію, чіткий контроль цілісності даних та шифрування даних при передачі [37]. MySQL – безкоштовна система з відкритим програмним кодом. Крім того, для роботи з MySQL існує значна кількість зручних інтерфейсів взаємодії з Java [38].

4.2 Основні семантичні структури програми

Розроблений метод автентифікації користувачів із нульовим знанням реалізовано у вигляді модуля, що використовується як серверною так і клієнтською частиною засобу. Модуль автентифікації користувачів із нульовим знанням складається з п’яти основних класів Java.

4.2.1 Семантичні структури модуля автентифікації

Основою криптографічного блоку засобу є клас CryptoZK. При створенні об’єкта CryptoZK вказуються два великих простих числа p та q, добутком яких є модуль n, з використанням якого виконується більшість обчислень та кількість раундів L. Для збереження чисел великої розрядності використано стандартний Java клас BigInteger. У класі CryptoZK зберігаються лише вище зазначені параметри, а також відкриті ключі y та a, закритий ключ користувача S. Клас CryptoZK лише виконує операції над даних за допомогою методів та не здійснює збереження проміжних результатів. Метод GeneratePublicKeys виконується для генерації відкритих ключів y та a на основі закритого ключа S. Методи SetPublicKeys та GetPublicKeys надають можливість встановити або відповідно отримати значення відкритих ключів, що зберігаються у класі. Ключі передаються на ці методи у форматі JSON.

Методи, що виконують обчислення за протоколом з нульовим знанням для клієнтської частини: GenerateK, U та W. Метод GenerateK генерує масив k з L псевдовипадкових чисел та повертає його у вигляді списку об’єктів BigInteger.

Методи U та W виконують обчислення масивів u та w відповідно з використанням вхідних параметрів масиву k та параметра вибору протоколу bit для U; масиву k, параметра вибору протоколу bit та отриманого від сервера масиву біт r для W.

Для серверної частини CryptoZK використовує метод GenerateR для генерації масиву псевдовипадкових біт r та метод Verificate для виконання перевірки проходження автентифікації. Метод Verificate отримує значення масивів u, w та параметра вибору bit за якими виконує перевірку за алгоритмом представленим на рисунку 3.8. Verificate повертає значення true у разі успішного виконання перевірки та false у іншому випадку.

Клас Token описує токен користувача та надає методи для отримання та зміни токенів користувачів. Структура класу Token зображена на рисунку 4.1.

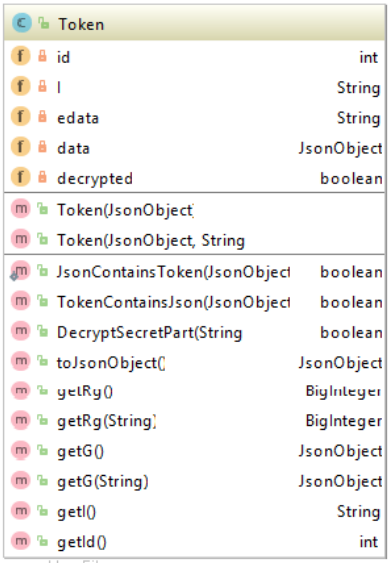


Рисунок 4.1 – Структура класу Token

Серед основних полів класу Token ідентифікатор токену id, ідентифікатор користувача-власника токену I, закрита частина токену у вигляді параметру edata. Окрім основних полів, описаних у підрозділі 2.3, у клас Token використовує об’єкт JSON data для зберігання розшифрованої закритої частини та параметра decrypted, який слідкує за зміною стану поля data. Об’єкт класу Token створюється на основі JSON об’єкту, який складається з основних полів

класу. Під час створення об’єкту Token можна зазначити секретний ключ до токену. Тоді закрита частина токену розшифрується методом DecryptSecretPart та запишеться у полі data. Методи класу Token дозволяють отримати значення полів токену, причому, якщо токен розшифрований, то для отримання полів не потрібно повторно вказувати секретний ключ, у іншому випадку метод поверне помилку. Токен можна розшифрувати під час отримання значення будь-якого поля із закритої частини за допомогою використання перевантаженого методу із ключем до токена у вигляді параметру.

Клас LFSR реалізує регістр зсуву з лінійним зворотнім зв’язком. LFSR унаслідуваний від типового класу Random. У класі три основних поля: поточний стан регістру registerState, поле characteristic – поліном для вибору розрядів, за якими буде зворотній зв’язок та поле digri – ступінь поліному. Серед методів класу nextBoolean та nextBit використовуються для для виконання одного та певної кількості тактів зсуву з лінійним зворотнім зв’язком та повернення молодшого біту регістру.

Клас Generators реалізує набір статичних методів для генерації ключів, псевдовипадкових чисел та роботи із JSON. Серед методів у класі Generators метод генерації токену без збереження та перевірки з базою даних TokenGenerator та LFSRGenerator, який використовує об’єкт LFSR для виконання вибору протоколу автентифікації. Структури класів LFSR та Generators представлені на рисунку 4.2

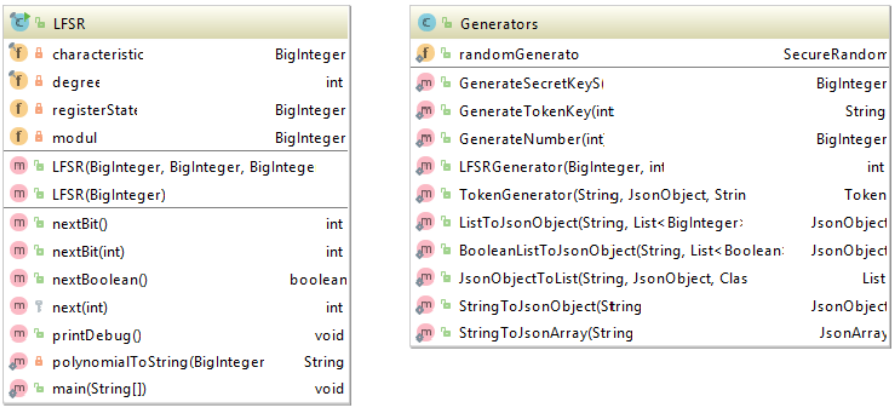


Рисунок 4.2 – Структури класів LFSR та Generators

Клас AuthenticationMethod – головна семантична структура бібліотеки. У класі відбувається зберігання всіх тимчасових даних сесії автентифікації. Поля класу: секретний ключ S, ключ токену TokenKey, параметр вибору bit, час «життя токену» ttl, об’єкти класів CryptoZK та Token, обчислені проміжні масиви великих чисел u, w та масив бітів r. Загальна UML-діаграма класів бібліотеки автентифікації зображена на рисунку 4.3.

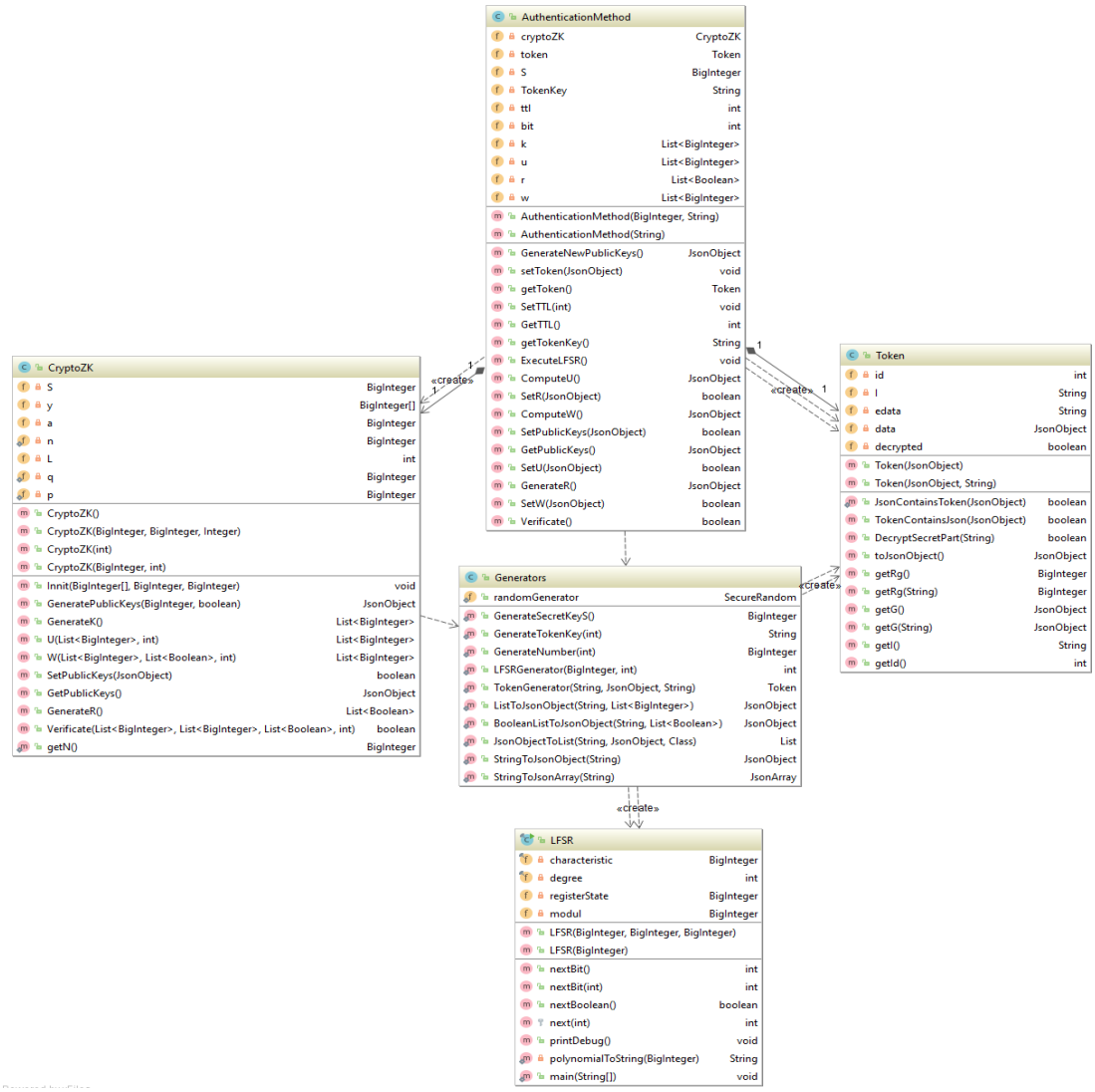


Рисунок 4.3 – UML-діаграма класів бібліотеки автентифікації

Для створення об’єкту AuthenticationMethod необхідно надати TokenKey для серверної частини та TokenKey і S для клієнта. До основних методів класу належать: GenerateNewPublicKeys, SetPublicKeys, ComputeU, SetU, ComputeW SetW, setToken, getToken, ExecuteLFSR, GenerateR, SetR, Verificate. Методи, що

виконують обрахунок та встановлення проміжних масивів при автентифікації працюють над даними у форматі JSON використовуючи відповідні методи класу CryptoZK та Generators. Додаткові методи класу здійснюють встановлення або отримання значень відповідних полів класу.

Перейдемо до розгляду семантичних структур клієнт-серверного застосунку, що використовує розроблений модуль автентифікації з нульовим знанням.

4.2.2 Семантичні структури клієнт-серверного застосунку

Клієнт-серверний застосунок здійснює збереження нотатків користувача та розмежування доступу до облікового запису користувача за допомогою токенів.

Серверна частина застосунку складається з трьох основних класів. Структури основних класів сервера зображені на рисунку 4.4.

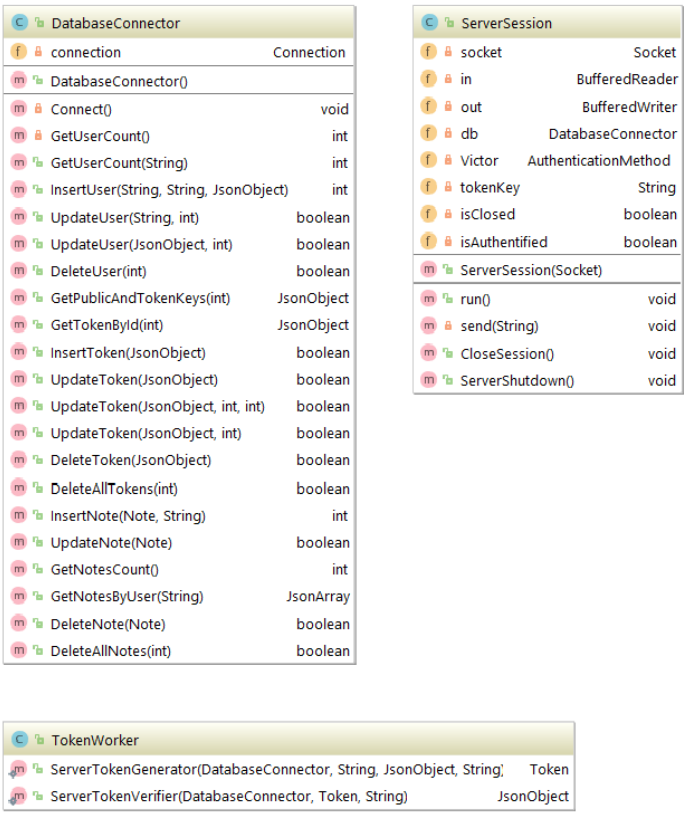


Рисунок 4.4 – Структури основних класів серверної частини застосунку

Сервер побудований на основі технології socket. У головному класі Server знаходиться зв’язний список всіх socket-сесій та відбувається додавання нових сесій у список. Для взаємодії з клієнтом використовується клас ServerSession.

Клас ServerSession успадкований від Thread та складається з полів socket, за яким відбувається підключення клієнта до сервера, потоків для передачі даних in та out, а також об’єкта модуля автентифікації AuthenticationMethod. Після підключення до сервера автоматично починає свою роботу метод run, що працює в окремому потоці та виконує обробку отриманих від клієнта команд. Для надсилання команд клієнту використовується метод send. Клас також має два методи для закриття сесії та повного зупинення сервера, що виконують попереднє повідомлення клієнтів та знищують відповідні сесії.

Клас TokenWorker виконує перевірку та генерацію токенів із взаємодією з базою даних. Клас реалізує алгоритми 3.4 та 3.5.

Клас DatabaseConnector складається лише з поля connection, що використовує JDBC драйвер для підключення до бази даних MySQL. Клас має великий перелік методів для взаємодії з трьома ключовими таблицями бази даних: таблицями користувачів, токенів та нотатків. На рисунку 4.5 зображена ER-діаграма зв’язків бази даних [38].

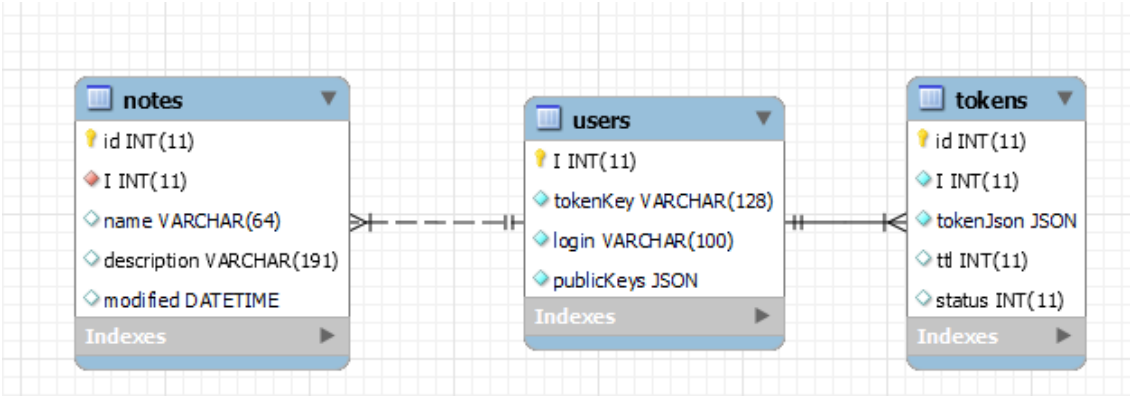


Рисунок 4.5 – ER-діаграма зв’язків бази даних

Таблиця користувачів зберігає ідентифікатор користувача, логін, ключ до токена та публічні ключі. Таблиця токенів складається з ідентифікаторів токену та користувача, повного токену, збереженого у форматі JSON та двох параметрів

часу «життя» токену та статус. Нотатки складаються з унікального ідентифікатора нотатки, назви нотатки, вмісту або опису нотатки та часу останньої модифікації. Кожна нотатка, пов’язана з одним із користувачів.

Клієнтська частина застосунку складається із двадцяти одного класу, але основними серед них є класи Core, ClientSession та Note. Структури основних класів клієнтської частини зображені на рисунку 4.6.

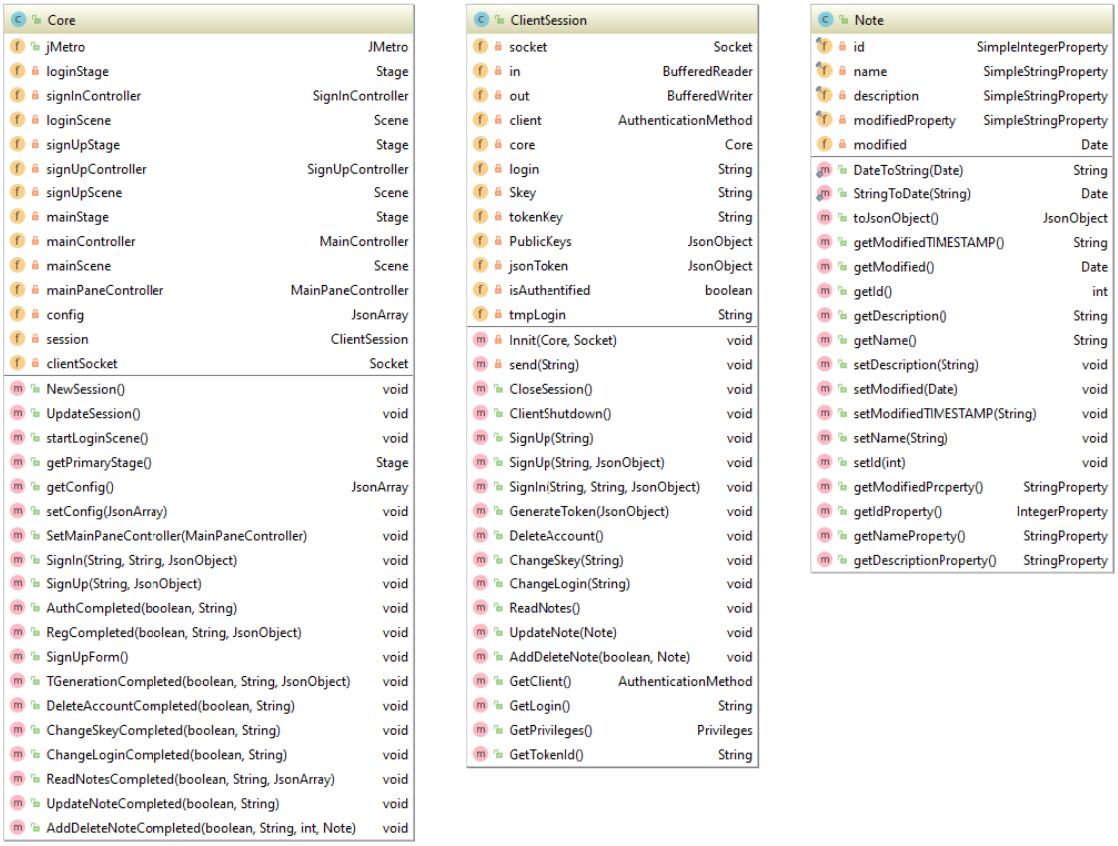


Рисунок 4.6 – Структури основних класів клієнтської частини застосунку

Клас Core виконує роль класу, що поєднує інтерфейс користувача із сесією за якою йде взаємодія із сервером. Серед полів класу об’єкт ClientSession, що взаємодіє з сервером, об’єкти Controller, для роботи з графічним інтерфейсом, config, що зчитується з файлу локально збережені для поточного клієнта облікові дані. Більшість методів класу виконують передачу команд від графічного інтерфейсу до клієнтської сесії або навпаки, отримують команди від ClientSession та передають на відповідний Controller.

Клас ClientSession клієнтський аналог класу ServerSession. На відміну від серверу до методів клієнтської частини входять команди, які користувач надсилає до сервера. Кожній команді відповідає певний метод. Наприклад, метод signIn отримує від користувача секретний ключ, ключ до токену та токен користувача та ініціює роботу AuthenticationMethod для клієнта після чого надсилає серверу токен для виконання першого кроку автентифікації.

Клас Note описує об'єкт нотатку, структура якої відповідає таблиці нотаток, зображеній на рисунку 4.6. Клас Note використовується серверною та клієнтською частинами для швидкого формування нового нотатку, його відображення у графічному інтерфейсі, зберігання у базі даних та конвертування до формату JSON для подальшої взаємодії.

Для перевірки коректності роботи та подальшого удосконалення модуля автентифікації доцільно розробити тести та провести тестування модуля автентифікації та клієнт-серверного застосунку.

4.3 Тестування засобу автентифікації

4.3.1 Блокове тестування засобу автентифікації з нульовим знанням

Для блокового тестування використано бібліотеку з відкритим вихідним кодом JUnit [39]. Для основних класів модуля автентифікації користувачів із нульовим знанням реалізовано класи з наборами тестів. Кожен тест відповідає одному або декільком методам відповідного класу модуля автентифікації. На рисунку 4.7 наведено результати виконання тестів для генератора токенів та регістра зсуву з лінійним зворотнім зв'язком.

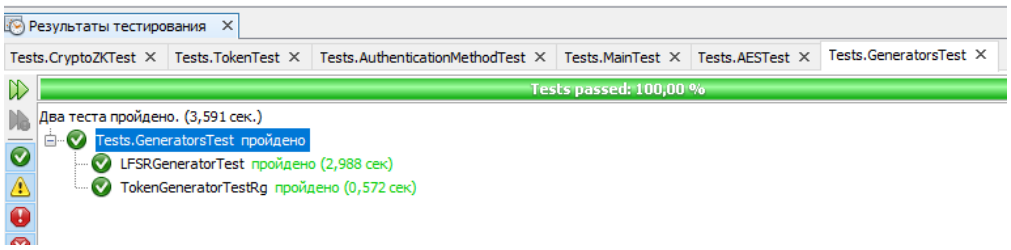


Рисунок 4.7 – Результати тестування РЗЛЗЗ та генератора токенів

Незважаючи на те, що на рисунку 4.7 зображено лише по одному тесту для об’єкту, насправді в кожному JUnit тесті відбувається декілька перевірок у циклі. Результати тестування показали коректність генерації токенів та достатній період РЗЛЗЗ.

Тестування класу CryptoZK відбувалося для функцій розрахунку проміжних параметрів на коректних та некоректних даних, а також для функції перевірки автентичності. Результати тестування класу CryptoZK зображені на рисунку 4.8.

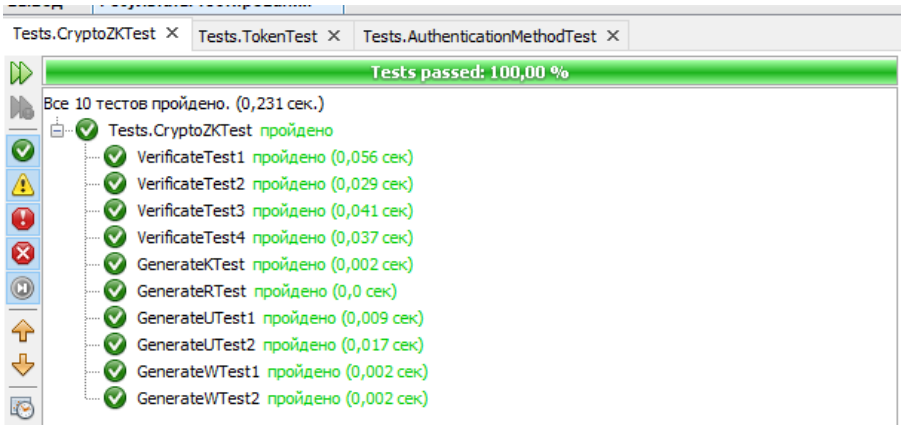


Рисунок 4.8 – Результати тестування класу CryptoZK

Для класу Token виконано тестування шифрування та розшифрування секретної частини з коректним на некоректним токеном та ключем. Перевірено роботу методів класу при розшифрованій та зашифрованій закритій частині токену. Результат блокового тестування свідчить про коректність роботи методів класу Token, що зображено на рисунку 4.9.

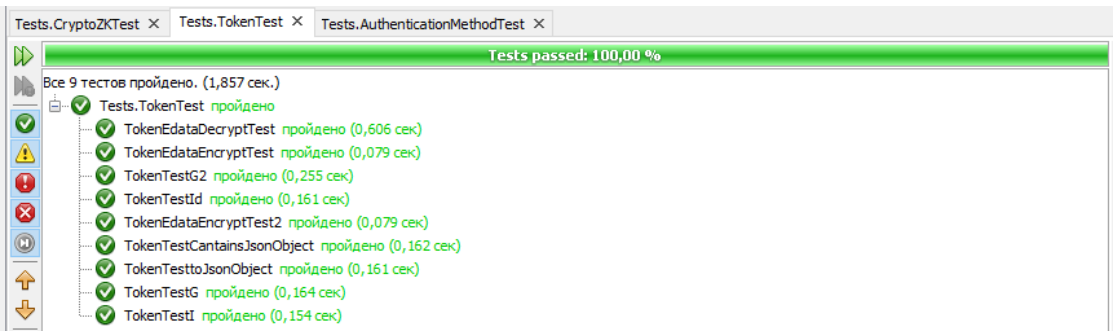


Рисунок 4.9 – Результати тестування класу Token

На рисунку 4.10 наведено результат тестування головного класу модуля автентифікації AuthenticationMethod. Тестування здійснювалось з використанням псевдовипадкових коректних та некоректних даних, кожен тест проходив декілька разів на нових даних.

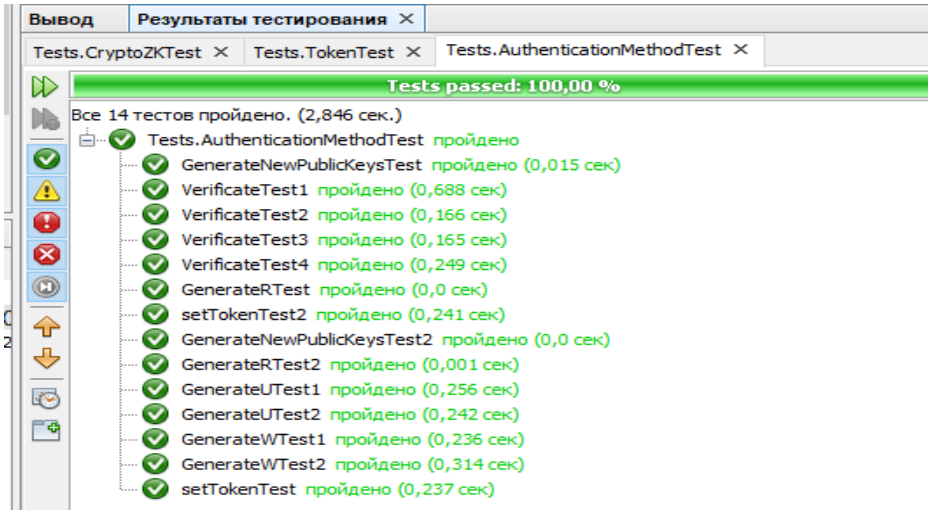


Рисунок 4.10 – Результати тестування класу AuthenticationMethod

Для перевірки коректності роботи з базою даних виконано блокове тестування класу DatabaseConnector, що входить до складу серверної частини застосунку. Кожен метод класу перевірено на обраному наборі параметрів з бази даних з використанням різних вхідних даних. Результати проходження тестування зображено на рисунку 4.11.

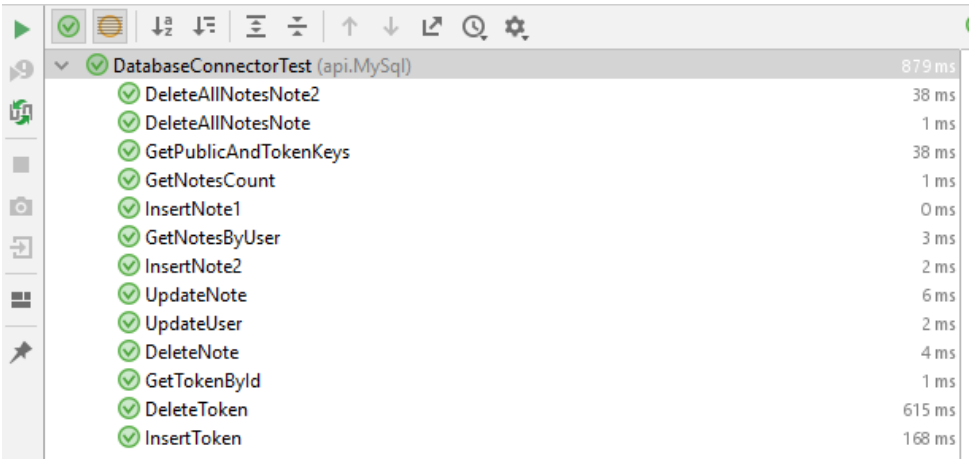


Рисунок 4.11 – Результати тестування класу DatabaseConnector

Результати блокового тестування показали коректність роботи реалізованих семантичних структур модуля автентифікації користувачів із нульовим знанням. Код розроблених класів тестування подано в додатку В.

4.3.2 Інтеграційне тестування засобу автентифікації з нульовим знанням

Для клієнт-серверного застосунок виконано інтеграційне тестування, що дозволило перевірити коректність роботи модуля автентифікації у взаємодії між клієнтською та серверною частинами, а також надало можливість перевірки розмежування доступу на основі токенів.

Для тестування на коректних даних обрано вже зареєстрованого користувача Peggy. Клієнтська частина застосунку надає можливість автоматичного введення токену та ключів користувача із config файлу. Для початку обрано токен з ідентифікатором 553353665, який надає користувачу максимальні права у системі. Результат тестування проходження автентифікації за коректними даними зображено на рисунку 4.12.

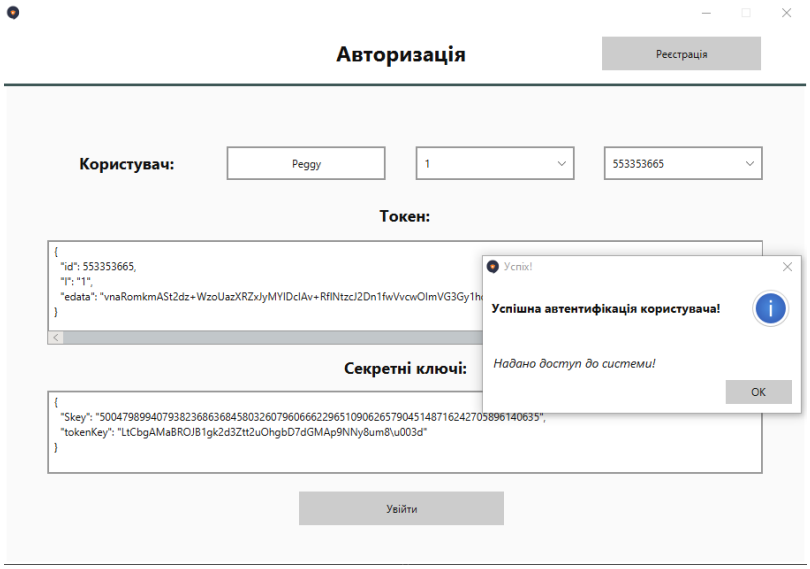
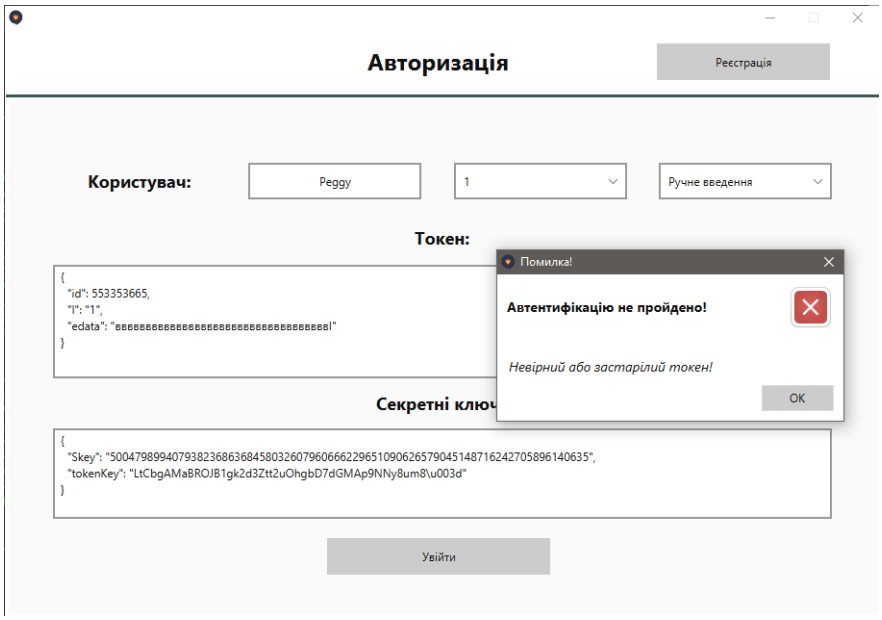


Рисунок 4.12 – Результат автентифікації за достовірними даними

У вікні авторизації обравши ручне введення даних виконано перевірку коректності проходження автентифікації з використання некоректного токену

користувача або некоректного одного з секретних ключів. Результати тестування зображені на рисунку 4.13 а) та б) відповідно.

а)



б)

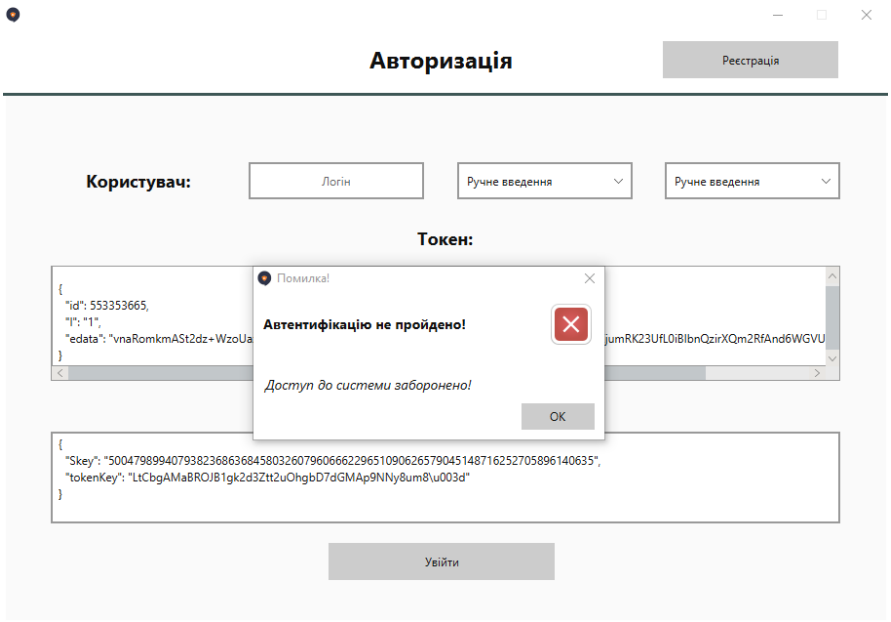


Рисунок 4.13 – Резулатат тестування автентифікації з некоректним даними (а – некоректний токен користувача, б – некоректний секретний ключ)

Результати інтеграційного тестування процесу автентифікації підтвердили коректність роботи модуля автентифікації.

Для тестування прав доступу обрано токен користувача Peggy з максимальними правами роботи у системі. Після автентифікації та авторизації у системі користувачу доступне меню, що складається з трьох пунктів: перехід на

головну сторінку, де зберігається таблиця з нотатками користувача. Пункт налаштування надає можливість змінювати дані облікового запису та пункт привілеї надає можливість користувачу дізнатися його поточні привілеї та створити новий токен. Результат тестування коректності розпізнання прав користувача з токену зображено на рисунку 4.14.

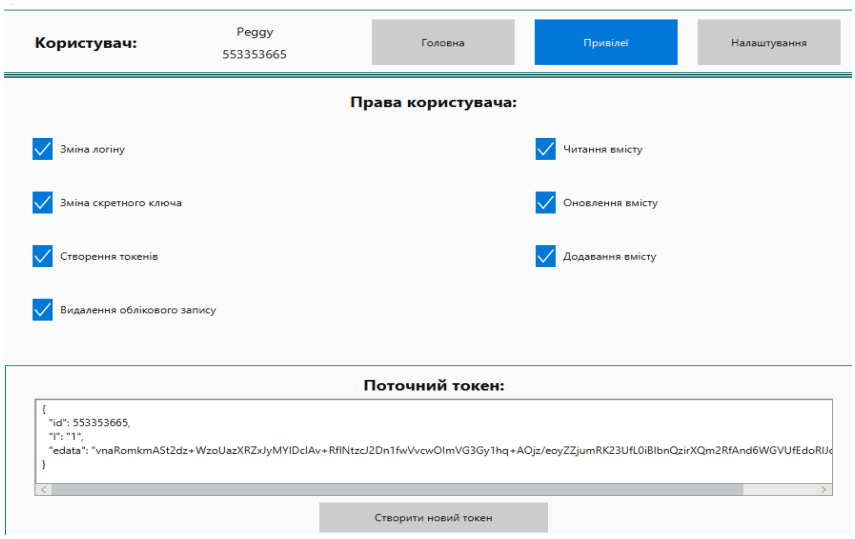


Рисунок 4.14 – Результат тестування коректності розпізнання прав користувача

Для тестування можливості модифікації на головному вікні створено новий запис у нотатках, після чого здійснено його успішну модифікацію, що зображено на рисунку 4.15.

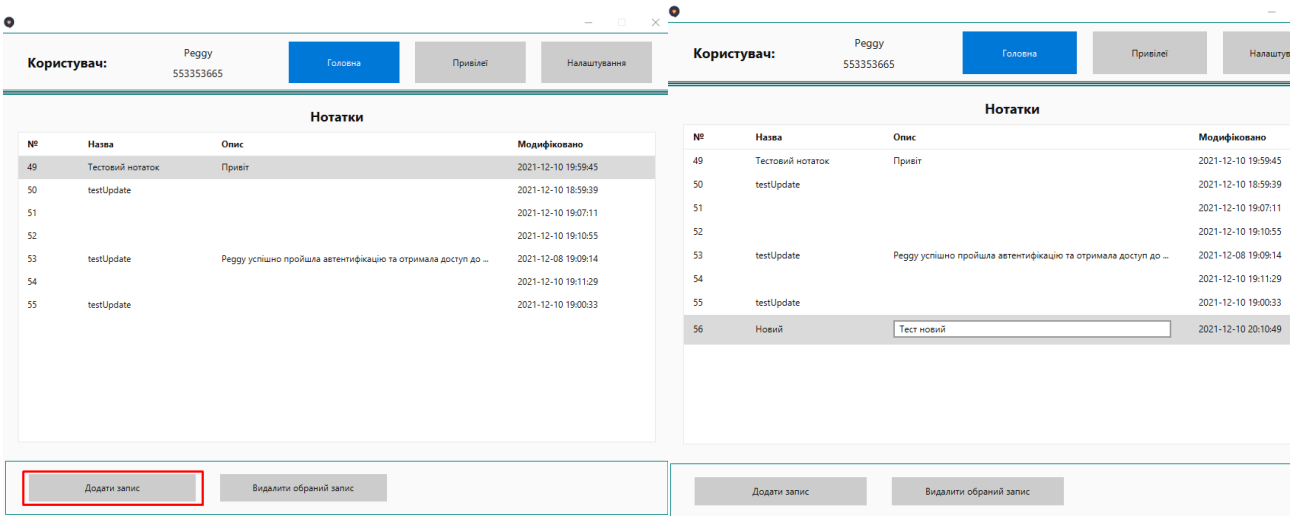


Рисунок 4.15 – Результат тестування надання прав на читання та модифікацію

Для тестування обмежених прав доступу до системи створено новий токен користувача Peggy, за яким користувачу надаються права тільки на зміну логіну та секретного ключа. Результат тестування створення токену з обмеженими правами зображено на рисунку 4.16.

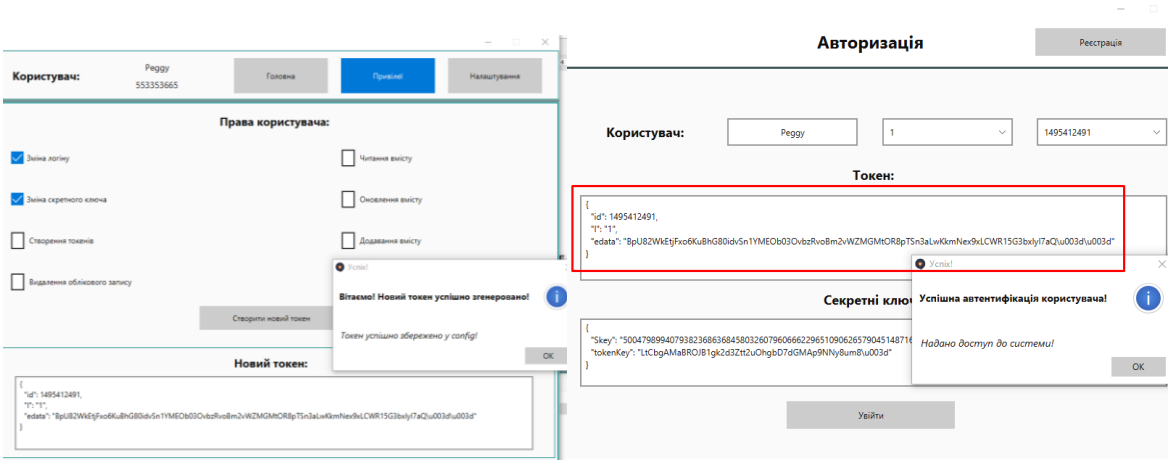


Рисунок 4.16 – Результат тестування створення нових токенів з обмеженими наборами прав доступу

В результаті повторної автентифікації вже з використанням нового токену з ідентифікатором 1495412491, що надає обмежені права доступу у системі виконано успішну повторну автентифікацію користувача. Результати тестування доступу до читання, модифікації та створення токенів за новоствореним токеном зображено на рисунку 4.17.

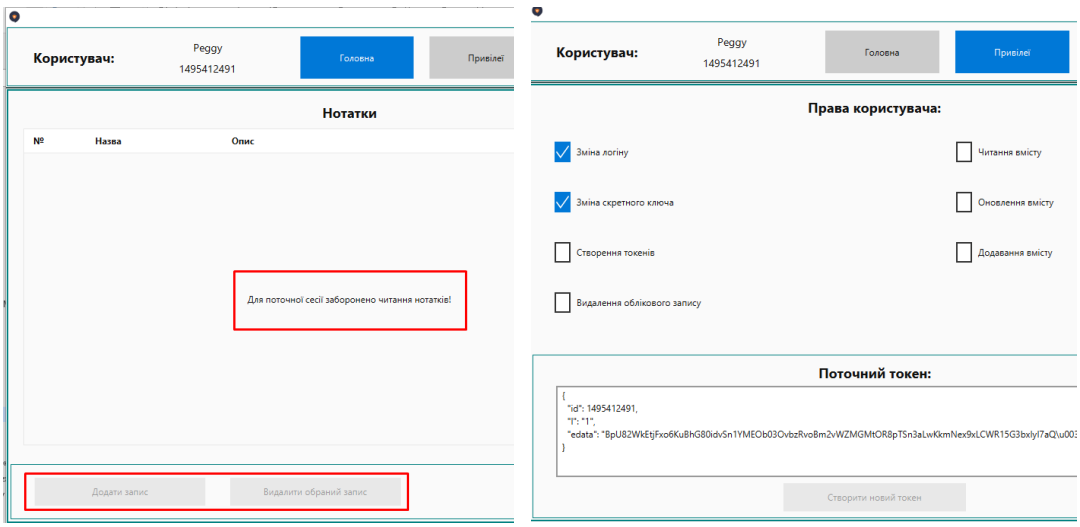


Рисунок 4.17 – Результат використання системи з обмеженими правами доступу

Для тестування коректності роботи обмежених прав виконано зміну логіну користувача Peggy на «ПеггіТеперАліса». Результат тестування зображено на рисунку 4.18.

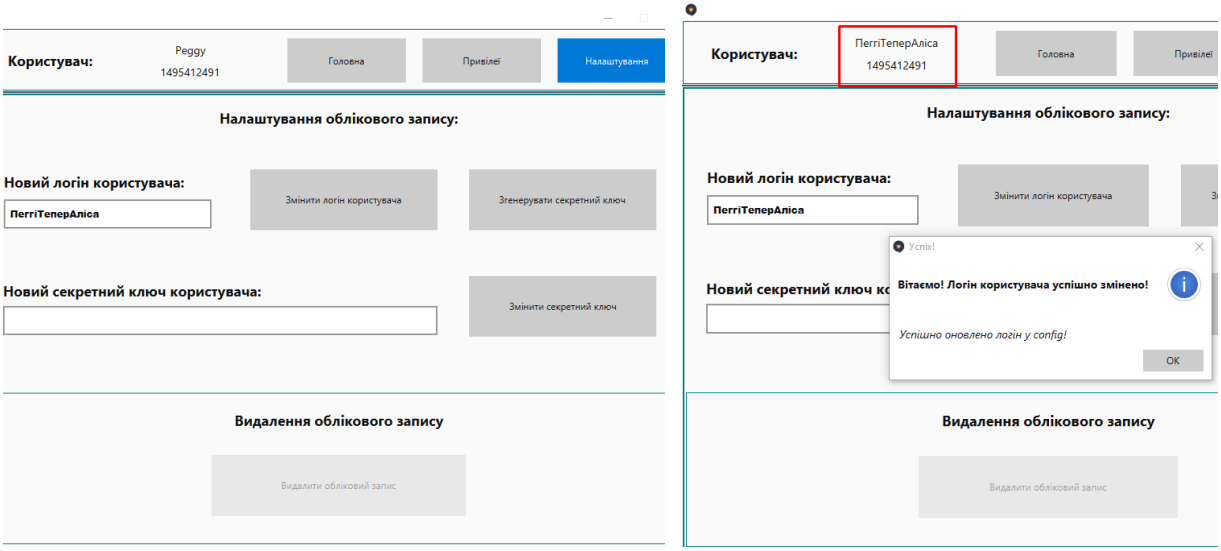


Рисунок 4.18 – Результат тестування доступних обмежених прав доступу

Результати інтеграційного тестування підтвердили коректність роботи системи.

4.4 Висновки до розділу

Отже, у цьому розділі обрано засоби розробки модуля автентифікації користувачів із нульовим знання. Виконано розробку основних семантичних структур програмного модуля, що реалізує модуль автентифікації користувачів. Виконано розробку клієнт-серверного програмного застосунку, що використовує модуль автентифікації користувачів. Здійснено блокове тестування модуля автентифікації користувачів з використання JUnit тестів та інтеграційне тестування клієнт-серверного застосунку. Результати тестування підтвердили коректність роботи модуля автентифікації. В результаті тестування перевірено коректність реалізації методу та розмежування прав доступу на основі токенів.

5 ЕКОНОМІЧНА ЧАСТИНА

Метою економічної частини магістерської кваліфікаційної роботи є обґрунтування економічної доцільності методу та засобу автентифікації користувачів із нульовим знанням. Для виконання поставленої мети необхідно:

- оцінити комерційний потенціал розробки;
- оцінити витрати на виконання та впровадження результатів наукової роботи;
- розрахувати ціну та чистий прибуток реалізації результатів розробки;
- розрахувати період окупності наукової роботи та результатів розробки.

5.1 Оцінювання комерційного потенціалу розробки (технологічний аудит розробки)

Для проведення технологічного аудиту було залучено трьох незалежних експертів: Баришев Ю. В, Куперштейн Л. М., Войтович О. П. Кожен з експертів повинен ознайомитися з запропонованою розробкою та заповнити таблицю, яка визначає рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можливу оцінку в балах. Після виконання цього, підраховується середньоарифметична сума балів та визначається який рівень комерційного потенціалу має нова розробка. Здійснюємо оцінювання комерційного потенціалу розробки за 12-ю критеріями, наведеними в додатку. Результати оцінювання комерційного потенціалу розробки наведено в таблиці 5.1.

Таблиця 5.1 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали експерта		
	Баришев Ю. В,	Куперштейн Л. М.	Войтович О. П.
	Бали, виставлені експертами:		
1	3	2	3
2	2	3	2
3	4	3	3
4	2	1	2
5	2	2	2

Продовження таблиці 5.1

Критерії	Прізвище, ініціали експерта		
	Баришев Ю. В,	Куперштейн Л. М.	Войтович О. П.
	Бали, виставлені експертами:		
6	3	3	2
7	2	3	3
8	3	4	3
9	3	1	2
10	4	4	4
11	3	3	3
12	3	4	2
Сума балів	$СБ_1$=34	$СБ_2$=37	$СБ_3$=31
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{i} СБ_i}{i} = \frac{34+37+31}{3} = 34$		

За даними таблиці 5.1 робимо висновок щодо рівня комерційного потенціалу розробки методу та засобу автентифікації з нульовим знанням. При цьому використано рекомендації, що наведено у таблиці 5.2.

Таблиця 5.2 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів	Рівень комерційного потенціалу
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-50	Високий

Отже, відповідно до результатів оцінювання експертами рівень комерційного потенціалу розробки засобу та методу автентифікації користувачів із нульовим знанням вище середнього.

Оцінювання рівня якості розробки методу та засобу автентифікації користувачів із нульовим знанням здійснюється з метою порівняльного аналізу та визначення найбільш ефективного, з технічної точки зору, варіанта інженерного рішення.

Рівень якості – кількісна характеристика міри придатності певного виду продукції для задоволення конкретного попиту на неї при порівнянні з відповідними базовими показниками за фіксованих умов споживання.

Абсолютний рівень якості розробки методу та засобу автентифікації з нульовим знанням знаходимо обчисленням обраних для вимірювання показників, не порівнюючи їх із відповідними показниками аналогічних засобів.

Для визначення рівня якості розробки обрано систему параметрів: ймовірність автентифікації порушника, швидкість автентифікації, обсяг даних, що передаються, кількість взаємодій між сторонами.

Визначаємо величину параметрів якості в балах та встановлюємо граничні значення (кращі, гірші, середні). Дані для кожного параметра представлено у таблиці 5.3.

Таблиця 5.3 – Основні параметри методу та засобу автентифікації з нульовим знанням

Параметри	Абсолютне значення параметра			Коефіцієнт вагомості параметра
	Краще +5…+4	Середнє +3	Гірше +1..+2	
ймовірність автентифікації порушника	4			0.6
швидкість автентифікації		3		0.2
обсяг даних ,що передаються		3		0.1
кількість взаємодій			2	0.1

Із врахуванням коефіцієнтів вагомості відповідних параметрів можна визначити абсолютний рівень якості інноваційного рішення за формулою

$$K_{я.а.} = \sum_{i=1}^{n} Рн_i * a_i \text{ ,} \qquad (5.1)$$

де $Рн_i$ – числове значення i-го параметра інноваційного рішення, n – кількість параметрів інноваційного рішення, що прийняті для оцінювання, a_i – коефіцієнт вагомості відповідного параметра (сума коефіцієнтів вагомості всіх параметрів повинна дорівнювати 1).

Отже, абсолютний рівень якості методу та засобу автентифікації з нульовим знанням складає 3,5 бали.

Одночасно визначаємо відносний рівень якості розробленого методу та засобу автентифікації з нульовим знанням, шляхом порівнюючи показники з абсолютними показниками якості найліпших аналогів, що представлено у таблиці 5.4.

Таблиця 5.4 – Порівняння основних параметрів засобу автентифікації з нульовим знанням та товару-конкурента

Параметри	Варіанти		Відносний показник якості	Коефіцієнт вагомості параметра
	Базовий (конкурент)	Новий		
ймовірність автентифікації порушника	2^{-20}	$2*2^{-20}$	2	0,6
швидкість автентифікації	0,25	0,2	0,8	0,2
обсяг даних ,що передаються	80КВ	82КВ	0,98	0,1
кількість взаємодій	3	5	1,6	0,1

Відносний рівень якості методу та засобу автентифікації з нульовим знанням визначаємо за формулою 5.2:

$$K_{я.в.} = \sum_{i=1}^{n} q_i * a_i \qquad (5.2)$$

За розрахунками відносний рівень якості засобу автентифікації користувачів із нульовим знанням складає 1,52. Це означає, що нова розробка якісніша на 52% відносно товару-аналога.

У найширшому розумінні конкурентоспроможність товару – це можливість його успішного продажу на певному ринку і в певний проміжок часу.

Водночас конкурентоспроможною можна вважати лише однорідну продукцію з технічними параметрами і техніко-економічними показниками, що ідентичні аналогічним показникам уже проданого товару. Для того, щоб високоякісний товар був одночасно і конкурентоспроможним, він має відповідати критеріям оцінювання споживачів конкретного ринку в конкретний період часу.

Дані для розрахунку загального показника конкурентоспроможності розробки необхідно занести до таблиці 5.5.

Таблиця 5.5 – Нормативні, технічні та економічні параметри засобу автентифікації з нульовим знанням та товару-конкурента

Параметри	Варіанти		Відносний показник якості	Коефіцієнт вагомості параметра
	Базовий (конкурент)	Новий		
ймовірність автентифікації порушника	2^{-20}	$2*2^{-20}$	2	0,6
швидкість автентифікації	0,25	0,2	0,8	0,2
обсяг даних ,що передаються	80КВ	82КВ	0,98	0,1
кількість взаємодій	3	5	1,6	0,1
Ціна за продукт, грн	20000	15000	0,75	-

Загальний показник конкурентоспроможності розробки (K) з урахуванням вищезазначених груп показників визначаємо за формулою 5.3:

$$K = \frac{I_{Т.П.}}{I_{Е.П.}} = \frac{1{,}52}{0{,}75} = 2{,}02, \tag{5.3}$$

де $I_{Т.П.}$ – індекс технічних параметрів (відносний рівень якості інноваційного рішення); $I_{Е.П.}$ – індекс економічних параметрів розрахований нижче за формулою 5.4:

$$I_{Е.П.} = \frac{РН_{EI}}{РБ_{EI}} = \frac{15000}{20000} = 0{,}75, \tag{5.4}$$

де $РН_{ЕІ}$, $РБ_{ЕІ}$ – економічні параметри (ціна придбання та споживання товару) відповідно нового та базового товарів.

Згідно розрахунків загальний показник конкурентоспроможності 2,02, що свідчить про більшу конкурентну спроможність засобу автентифікації з нульовим знанням у порівнянні з товаром-аналогом на 102%.

5.2 Прогнозування витрат на виконання науково-дослідної та конструкторсько-технологічної роботи

5.2.1 Розрахунок витрат, що стосуються виконавців розробки методу та засобу автентифікації

Команда розробки методу та засобу автентифікації користувачів із нульовим знанням складається з керівника, інженер-програміста та тестувальника.

Основна заробітна плата для розробників (дослідників) *Зо*, якщо вони працюють в наукових установах бюджетної сфери визначається за формулою:

$$З_О = \frac{M}{T_р} t \qquad (5.5)$$

де M – місячний посадовий оклад розробника;

$T_р$ – кількість робочих днів у місяці, $T_р$ = 22 дні; t – число днів роботи.

Розрахунки заробітної плати для розробників наведені в таблиці 5.6

Таблиця 5.6 – Розрахунки основної заробітної плати розробників

Працівник	Оклад М, грн.	Оплата за робочий день, грн.	Число днів роботи, t	Витрати на оплату праці, грн.
Керівник	25000	1136,36	7	7945,52
Розробник	25000	1136,36	15	17045,4
Всього:				24990,92

Основна заробітна плата робітників *Зр*, якщо вони беруть участь у виконанні даного етапу роботи і виконують роботи за робочими професіями у

випадку, коли вони працюють в наукових установах бюджетної сфери, розраховується за формулою:

$$З_Р = \sum_{i=1}^{n} t_i \cdot C_i \ , \tag{5.6}$$

де t_i – норма часу (трудомісткість) на виконання конкретної роботи, годин; n – число робіт по видах та розрядах; C_i – погодинна тарифна ставка робітника відповідного розряду, який виконує дану роботу. C_i визначається за формулою:

$$C_i = \frac{M_M * K_i}{T_p * T_{ЗМ}}, \tag{5.7}$$

де M_M – розмір мінімальної заробітної плати за місяць, грн.; в 2021 році мінімальна заробітна плата становить – 6500 грн., K_i – тарифний коефіцієнт робітника відповідного розряду, T_p = 22 дні; $T_{зм}$ – тривалість зміни, $T_{зм}$ =8 годин.

Таблиця 5.7 – Заробітна плата робітників

Найменування робіт	Трудомісткість, н-год.	Розряд роботи	Погодинна тарифна ставка	Тариф. коеф.	Величина, грн.
Розробка	8	5	56,875	1,54	455
Тестування	8	4	53.55	1,45	428,4
Впровадження	2	2	40,25	1,09	80,51
Всього					963,91

Додаткова заробітна плата $З_Д$ всіх розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховується як (10…12)% від суми основної заробітної плати всіх розробників та робітників, тобто:

$$З_Д = 0.1*(З_О + З_Р) = 0{,}1*(24990{,}92 + 963{,}91) = 2595{,}48 \ \text{(грн.)} \tag{5.8}$$

Нарахування на заробітну плату $Н_{ЗП}$ розраховується як 22% від суми основної та додаткової заробітної плати:

$$Н_{ЗП} = (З_О + З_Р + З_Д) \times \frac{\beta}{100} = (24990{,}92 + 963{,}91 + 2595{,}48) * 0{,}22 = 6281{,}06 \,(\text{грн.}) \tag{5.9}$$

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

$З_р$ – основна заробітна плата робітників, грн.;

$З_Д$ – додаткова заробітна плата розробників, грн.;

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

Розрахунок амортизаційних відрахувань виконується за такою формулою:

$$A = \frac{Ц}{t_В} \times \frac{T}{12} \tag{5.10}$$

де $Ц$ – балансова вартість обладнання, грн;

T – термін використання (T=22 дні= 0,73 місяців);

$t_В$ – корисний час використання ($t_В$ для комп'ютера становить 4 роки).

Під час виконання розробки використовувався ноутбук вартістю 40000 грн. Амортизаційні відрахування для ноутбуку представлені у таблиці 5.8.

Таблиця 5.8 - Амортизаційні відрахування

Найменування	Ціна, грн.	Корисний час використання, роки	Термін використання, міс.	Сума амортизації, грн.
Ноутбук	40000	4	0,73	608,33
Всього	608,33			

Витрати на силову електроенергію розраховуються за формулою:

$$B_Е = В \times П \times Ф \times К_П \tag{5.11}$$

де $В$ – вартість 1кВт-години електроенергії (В=4,62 грн/кВт);

$П$ – установлена потужність комп'ютеру ($П$=0,74 кВт);

$Ф$ – фактична кількість годин роботи комп'ютеру ($Ф$=22*8=176 год);

$К_П$ – коефіцієнт використання потужності ($К_П$ <1, $К_П$ = 0,8).

Відповідно до формули 5.11 витрати на силову електроенергію:

$B_Е = 4{,}62 \times 0{,}74 \times 176 \times 0{,}8 = 481{,}36$ (грн.)

Інші витрати $В_{ін}$ можна прийняти як (100-300)% від суми основної заробітної плати розробників, які виконували роботу, тобто:

$$B_{ін} = 1 \cdot (\ 24990{,}92 + 963{,}91\ ) = 25954{,}8 \text{ (грн.)} \quad (5.11)$$

Сума усіх попередніх витрат дає загальні витрати на виконання роботи. Усі витрати складають:

$$B = 24990{,}92 + 963{,}91 + 2595{,}48 + 6281{,}06 + 481{,}36 + 25954{,}8 + 608{,}33 = 61875{,}86 \text{ (грн.)}$$

Розрахунок загальної вартості наукової розробки $B_{заг}$ за формулою:

$$B_{заг} = \frac{B}{\alpha}, \quad (5.12)$$

де α – частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відносних одиницях.

$$B_{заг} = \frac{61875{,}86}{1} = 61875{,}86 \text{ (грн.)}$$

Прогнозування загальних витрат $ЗВ$ на виконання та впровадження результатів виконаної наукової роботи здійснюється за формулою:

$$ЗВ = \frac{B_{заг}}{\beta} \quad (5.13)$$

Розрахунок прогнозованих загальних витрат:

$$ЗВ = \frac{61875{,}86}{0{,}7} = 88394{,}08 \text{ (грн.)}$$

5.2.2 Розрахунок собівартості розробки методу та засобу автентифікації

Витрати на силову електроенергію розраховуються за формулою:

$$B_E = B \times П \times Ф \times K_П \quad (5.14)$$

де B – вартість 1кВт-години електроенергії (В=4,62 грн/кВт);

$П$ – установлена потужність комп'ютеру ($П$=0,74 кВт);

$Ф$ – фактична кількість годин роботи комп'ютеру ($Ф$=22*8=176 год);

$К_П$ – коефіцієнт використання потужності ($К_П$ <1, $К_П$ = 0,8).

Відповідно до формули 5.14 витрати на силову електроенергію:

$B_E = 4{,}62 \times 0{,}74 \times 176 \times 0{,}8 = 481{,}36$ (грн.)

Основна заробітна плата робітників *Зр*, якщо вони беруть участь у виконанні даного етапу роботи і виконують роботи за робочими професіями у випадку, коли вони працюють в наукових установах бюджетної сфери, розраховується за формулою:

$$З_Р = \sum_{i=1}^{n} t_i \cdot C_i \,, \tag{5.15}$$

де t_i – норма часу (трудомісткість) на виконання конкретної роботи, годин; n – число робіт по видах та розрядах; C_i – погодинна тарифна ставка робітника відповідного розряду, який виконує дану роботу. C_i визначається за формулою:

$$C_i = \frac{M_M * K_i}{T_p * T_{ЗМ}}, \tag{5.16}$$

де M_M – розмір мінімальної заробітної плати за місяць, грн.; в 2021 році мінімальна заробітна плата становить – 6500 грн., $К_i$ – тарифний коефіцієнт робітника відповідного розряду, $T_р$ = 22 дні; $T_{зм}$ – тривалість зміни, $T_{зм}$ =8 годин.

Таблиця 5.9 – Заробітна плата робітників

Найменування робіт	Трудомісткість, н-год.	Розряд роботи	Погодинна тарифна ставка	Тариф. коеф.	Величина, грн.
Розробка	8	5	56,875	1,54	455
Тестування	8	4	53.55	1,45	428,4
Впровадження	2	2	40,25	1,09	80,51
Всього					963,91

Додаткова заробітна плата $З_Д$ всіх робітників, які брали участь у виконанні даного етапу роботи, розраховується як (10…12)% від суми основної заробітної плати всіх розробників та робітників, тобто:

$$З_Д = 0.1 * (З_Р) = 0{,}1 * (963{,}91) = 96{,}39 \text{ (грн.)} \tag{5.17}$$

Нарахування на заробітну плату $H_{ЗП}$ розраховується як 22% від суми основної та додаткової заробітної плати:

$$H_{ЗП} = (З_Р + З_Д) \times \frac{\beta}{100} = (963{,}91 + 96{,}39) * 0{,}22 = 233{,}27 \text{ (грн.)} \quad (5.18)$$

де $З_р$ – основна заробітна плата робітників, грн.;

$З_Д$ – додаткова заробітна плата робітників, грн.;

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

Загальновиробничі витрати з рахунку на одиницю продукції можна розрахувати за нормативами відносно до основної заробітної плати основних робітників, які виготовляють продукцію :

$$ЗВВ = H_В * З_О, \quad (5.19)$$

Норматив загальновиробничих витрат для програмних продуктів становить 230-270%.

$ЗВВ = 2{,}7 * 963{,}91 = 2602{,}56$ (грн.)

Сума попередніх витрат утворює виробничу собівартість розробки.

$$S_В = 481{,}36 + 963{,}91 + 96{,}39 + 233{,}27 + 2602{,}56 = 4377{,}5 \text{ (грн.)} \quad (5.20)$$

5.3 Розрахунок мінімальної ціни та чистого прибутку від реалізації розробки методу та засобу автентифікації з нульовим знанням

Ціна – це грошовий вираз вартості товару (продукції, послуги). Вона завжди коливається навколо ціни виробництва (перетвореної форми вартості одиниці товару, що дорівнює сумі витрат виробництва й середнього прибутку) та відображає рівень суспільне необхідних витрат праці.

Виходячи з того, що розробки, як правило, приймаються та впроваджуються за завданням замовника, або коли результатом розробки є продукція, що підлягає державному регулюванню, то нижню межу ціни реалізації розробки можна розрахувати за формулою 5.21:

$$Ц = S_B \cdot (1 + \frac{P}{100}) \cdot (1 + \frac{\omega}{100}), \quad (5.21)$$

де S_B – виробнича собівартість інноваційного рішення, грн.;

P – норматив рентабельності узгоджений із замовником або встановлений державою, (Р=30…60%);

ω – ставка податку на додану вартість, % (з осені 2021 року $\omega = 20\%$).

$$Ц = 4185 \cdot (1 + \frac{60}{100}) \cdot (1 + \frac{20}{100}) = 8035{,}2 \text{ (грн.)} \quad (5.21)$$

Із врахуванням коефіцієнта якості ціна розробки становить 28123 грн.

Чистий прибуток від реалізації розробки можна розрахувати за формулою:

$$П = (Ц - \frac{(Ц - MP) \cdot f}{100} - S_B - \frac{q \cdot S_B}{100}) \cdot (1 - \frac{h}{100}) \cdot РП, \quad (5.22)$$

де $Ц$ – ціна розробки, грн.; MP – вартість матеріальних та інших ресурсів, що були придбані виробником для виготовлення розробки (MP=(0,1…0,2) $Ц$), грн.; f – зустрічна ставка податку на додану вартість, %; S_B – виробнича собівартість розробки, грн.; q – норматив, який визначає величину адміністративних витрат, витрат на збут та інші операційні витрати, % (рекомендовано q=5…10%); h – ставка податку на прибуток, %, $РП$ – прогнозований попит продажів.

$$П = (28123 - \frac{(28123 - 28123 \cdot 0{,}2) \cdot 14}{100} - 4377{,}5 - \frac{5 \cdot 4377{,}5}{100}) \cdot (1 - \frac{18}{100}) \cdot 2 = 33418 \text{ (грн.)}, \quad (5.23)$$

Прогнозований чистий прибуток від реалізації розробки складає 33418 грн.

5.4 Розрахунок терміну окупності коштів вкладених у наукову розробку методу та засобу автентифікації з нульовим знанням

Термін окупності вкладених у реалізацію наукового проекту інвестицій розраховано за формулою 5.24:

$$T_{ОК} = \frac{ЗВ}{П} = \frac{88394{,}08}{33418} = 2{,}65 \text{ (років)} \quad (5.24)$$

Оскільки $T_{ОК} < 3$ років, то фінансування наукової розробки методу та засобу автентифікації користувачів із нульовим знанням є доцільним.

5.5 Висновки до розділу

Отже, у цьому розділі виконано обґрунтування економічної доцільності проведення наукового дослідження та розробки методу та засобу автентифікації користувачів із нульовим знанням.

Рівень комерційного потенціалу розробки методу та засобу автентифікації користувачів із нульовим знанням вище середнього.

На основі параметрів засобу автентифікації визначено абсолютний рівень якості методу та засобу автентифікації з нульовим знанням який складає 3,5 бали.

Відносний рівень якості розробки, що складає 1,52. Це означає, що нова розробка якісніша на 52% відносно товару-аналога. Загальний показник конкурентоспроможності становить 2,02, що свідчить про більшу конкурентну спроможність засобу автентифікації з нульовим знанням у порівнянні з товаром-аналогом на 102%.

Загальні витрати, що стосуються виконавців розробки склали 88394,08 грн, а собівартість розробки – 4377,5 грн.

Розраховано мінімальну ціну та прогнозований чистий річний прибуток від реалізації розробки, які склали 28123 грн. та 33418 грн. відповідно. Термін окупності продукції вкладених інвестицій складає 2,65 років, що свідчить про доцільність фінансування розробки.

ВИСНОВКИ

Аналіз задач кібербезпеки при перевірки автентичності користувача перед отриманням доступу до роботи з інформаційними ресурсами показав необхідність використання сучасних засобів автентифікації. Завдяки проведеному дослідженню сучасних методів автентифікації користувачів обґрунтовано необхідність покращення рівня захищеності при автентифікації користувачів. Серед розглянутих методів автентифікації протоколи автентифікації з нульовим знанням дозволяють переконати іншу сторону у володінні секретом без його розкриття та забезпечують достатній рівень стійкості при автентифікації. На основі результатів порівняння сучасних протоколів автентифікації користувачів із нульовим знанням було визначено, що більшість з них забезпечують стійкість лише при великій кількості раундів та розраховані для використання лише в системах особливої структури, що породжує необхідність модифікації досліджених методів для забезпечення високого рівня захищеності при автентифікації.

Під час дослідження методів автентифікації виконано математичний опис процесу автентифікації користувачів для трьох загальновідомих методів автентифікації на основі криптографічних перетворень. Для формалізованого математичного опису системи використано теоретико-множинний підхід. Результат математичного опису підтвердив, що автентифікація на основі доведення з нульовим розголошенням забезпечує підвищений рівень захищеності, тому протоколи з нульовим знанням доцільно використовувати у інфраструктурах з обробкою критичної інформації. Тому в даній магістерській кваліфікаційній роботі для підвищення рівня захищеності запропонований новий метод автентифікації з нульовим знанням, що базується на псевдовипадковому виборі одного з протоколів Шнорра та Фіата-Шаміра у процесі автентифікації користувача. Для вибору протоколу автентифікації використовується регістр зсуву з лінійним зворотнім зв’язком, початковийй стан якого попередньо узгоджується учасниками автентифікації за рахунок використання токенів

користувачів. Розроблено структуру та метод формування токену, що також надає користувачу можливість обмеження прав доступу до системи шляхом використання токенів з різними правами доступу для підвищення рівня безпеки.

Результат проведення аналізу розробленого методу автентифікації з використанням BAN-логіки підтвердив виконання необхідних умов коректності протоколу автентифікації.

На основі проведених досліджень сформовано загальну архітектуру засобу автентифікації користувачів із нульовим знанням, який представлено у вигляді клієнт-серверного модуля. Розроблено узагальнений алгоритм модуля автентитфікації з нульовим знанням, а також алгоритми використання токенів користувачів та алгоритми процесу автентифікації користувача для серверної та клієнтської сторін.

Метод автентифікації реалізовано об’єктно-орієнтованою мовою програмування Java у вигляді програмного модуля. Для тестування та достідження коректності роботи методу автентифікації розроблено клієнт-серверний застосунок, що використовує модуль автентифікації з нульовим знанням для забезпечення безпеки при автентифікації користувачів та розмежування доступу до нотатків користувачів на основі токенів. Тестування розроблених засобів показало коректність роботи окремих складових модуля автентифікації та коректність роботи модуля автентифікації під час інтеграції у клієнт-серверний застосунок при різних ситуаціях та будь-яких вхідних даних.

Оцінювання економічної доцільності методу автентифікації з нульовим знанням показало, що дана розробка є доцільною, оскільки період окупності складає менше трьох років, що демонструє економічну перспективність отриманих результатів. Розроблений метод дозволяє вдічі підвищити захищеність при автентифікації. Розроблений метод та засіб автентифікації користувачів із нульовим знанням можна використовувати в застосунках, де є потреба до забезпечення підвищеного рівня безпеки, а токож у системах, де є потреба у розмежуванні доступу при використанні одного облікового запису різними користувачами.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Goldwasser S., Micali S., Rackoff. C. The Knowledge Complexity of Interactive Proof Systems. STOC 85. 1985. P. 291 – 304.

2. Молдовян А. А., Молдовян Д. Н., Левина А. Б. Протоколы аутентификации с нулевым разглашением секрета. СПб: Университет ИТМО, 2016. 55 с.

3. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. Триумф, 2013. 816 с.

4. Селезньов В. І., Баришев Ю. В. Протокол автентифікації з нульовим знанням. *XLIX Науково-технічна конференція підрозділів Вінницького національного технічного університету:* тези наук.-практ. конф. ВНТУ, Вінниця, 2020 р.

5. Селезньов В. І. Модуль автентифікації користувачів. *Всеукраїнська науково-практична інтернет-конференція Молодь в науці: дослідження, проблеми, перспективи (МН-2021)* URL: https://publish.vntu.edu.ua /index.php/mn/mn2021/paper/view/11169.

6. А. с. 79692 Україна. Комп'ютерна програма «Дослідження методів цифрової модуляції сигналів» / Л. М. Куперштейн, В. І. Селезньов, П. І. Дончук-Донцов № 79692; заявл. 15.05.2018; опубл. 08.06.2018.

7. А. с. 79693 Україна. Комп'ютерна програма «Дослідження методів фізичного та логічного кодування» / Л. М. Куперштейн, В. І. Селезньов, П. І. Дончук-Донцов № 79693; заявл. 15.05.2018; опубл.08.06.2018.

8. А. с. 89042 Україна. Комп'ютерна програма «Арифметика великих чисел» / А. В. Остапенко-Боженова, В. І. Селезньов; опубл. 29.05.2019.

9. Мартинова, Л. Є., Умніцин М. Ю., Назарова К. Є. Дослідження та порівняльний аналіз методів аутентифікації. *Молодий вчений*. 2016. С. 90-93.

10. Шибанов, С. В., Карпушин Д. А. Порівняльний аналіз сучасних методів аутентифікації користувача. *Математичне та програмне забезпечення систем у промисловій та соціальній сферах*. 2015р. №1. С. 33-37.

11. Автентифікація (інформаційні технології). *Велика українська енциклопедія: у 30 т.* / проф. А. М. Киридон (відп. ред.) та ін. 2016. — 592 с.

12. Баришев Ю. В., Каплун В. А. Метод автентифікації віддалених користувачів для мережевих сервісів. *Інформаційні технології та комп'ютерна інженерія*: наук.-техн. журнал. 2014. Том 30. № 2. С. 13-17.

13. Ісхаков, А. Ю., Мещеряков Р.В., Ходашінскій І.А. Двухфакторная аутентифікація на основі программного токена *Питання захисту інформації.* 2013. № 3. С. 23-28.

14. Menezes A., van Oorschot P., Vanstone S. Handbook of Applied Cryptography. CRC Press, 1996. 816 p.

15. Zk-SNARKs: Under the Hood: веб-сайт. URL: https://medium.com/@VitalikButerin/zk-snarks-under-the-hood-b33151a013f6 (дата звернення: 14.09.2021)

16. Pankova A. Succinct non-interactive arguments from quadraticarithmetic programs. Technical report, University of Tartu, Cybernetica AS, 2013.

17. Ben-Sasson E., Chiesa A., Tromer E. Succinct Non-Interactive Zero Knowledge for a von Neumann Architecture. 2015. URL: https://eprint.iacr.org/2013/879.pdf (дата звернення: 21.09.2021)

18. Ben-Sasson E., Bentov I., Horesh Y. Scalable, transparent, and post-quantum secure computational integrity. 2018. URL: https://eprint.iacr.org/2018/046.pdf (дата звернення: 21.09.2021)

19. Ben-Sasson E., Chiesa A., Spooner N. Interactive Oracle Proofs. 2016. URL: https://eprint.iacr.org/2016/116.pdf (дата звернення: 22.09.2021)

20. Bulletproofs: веб-сайт. URL: https://sikoba.com/docs/SKOR_DK _Bulletproofs_ 201905.pdf (дата звернення: 25.09.2021)

21. Adjoint: веб-сайт. URL: https://github.com/adjoint-io/bulletproofs (дата звернення: 25.09.2021)

22. Awesome zero knowledge proofs (zkp): веб-сайт. URL: https://github.com/matter-labs/awesome-zero-knowledge-proofs (дата звернення: 25.09.2021)

23. Zero-Knowledge Proofs: STARKs vs SNARKs: веб-сайт. URL: https://consensys.net/blog/blockchain-explained/zero-knowledge-proofs-starks-vs-snar ks/ (дата звернення: 25.09.2021)

24. An Analysis of Anonymity in the Zcash Cryptocurrency by Jeffrey Quesnelle: веб-сайт. URL: https://deepblue.lib.umich.edu/bitstream/handle/2027.42/143130/quesnelle-thesis.pdf (дата звернення: 26.09.2021)

25. Ethereum: веб-сайт. URL: https://ethereum.org/ru/ (дата звернення: 26.09.2021)

26. Starkware: веб-сайт. URL:https://medium.com/starkware/the-road-ahead-in-2019-8589fedfbc7a (дата звернення: 27.09.2021)

27. Запечников С. В. Криптографические протоколы и их применение в финансовой и коммерческой деятельности. М.: Горячая Линия. Телеком, 2007. 320 с.

28. Черемушкин А. В. Криптографические протоколы. Основные свойства и уязвимости. М.: Академия, 2009. 269 с.

29. Burrows M., Abadi M., Needham R. “A Logic of Authentication”. *Systems Research Center of Digital Equipment Corporation in Palo Alto*. 426: P. 44–54.

30. Баришев Ю. В., Каплун В. А., Неуйміна К. В., Дискреційна модель та метод розмежування прав доступу до розподілених інформаційних ресурсів. *Наук. Праці ВНТУ*, вип. 2, Чер 2017.

31. Token authentication requirements for Git operations: веб-сайт. URL: https://github.blog/2020-12-15-token-authentication-requirements-for-git-operations/ (дата звернення: 27.10.2021)

32. Creating a personal access token: веб-сайт. URL: https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token (дата звернення: 27.10.2021)

33. Biryukov A., Khovratovich D. Related-key Cryptanalysis of the Full AES-192 and AES-256. *Advances in Cryptology*. ASIACRYPT 2009. Vol. 5912.

34. Шилдт Г. Java 8: руководство для начинающих, 6-е изд./ Пер. с англ. М. Вильямс. 2015. 720 с.

35. Java FX: веб-сайт. URL: https://openjfx.io/ (дата звернення: 05.11.2021)

36. JMetro – Java, JavaFX: веб-сайт. URL: https://pixelduke.com/java-javafx-theme-jmetro/ (дата звернення: 05.11.2021)

37. Реляционные СУБД – сравнение MySQL и SQL сервер: веб-сайт. URL: https://www.hostinger.com.ua/rukovodstva/reljacionnye-subd-sravnenie-mysql-i-sql-server/ (дата звернення: 08.11.2021)

38. MySQL Connector / J: веб-сайт. URL: https://mvnrepository.com/artifact /mysql/mysql-connector-java (дата звернення: 15.11.2021)

39. JUnit 5: веб-сайт. URL: https://junit.org/junit5/ (дата звернення: 28.11.2021)

ДОДАТКИ

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

ЗАТВЕРДЖУЮ

Завідувач кафедри ЗІ, д. т. н., проф.

_____ В. А. Лужецький

_____ _____ 2021 року

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

на тему: «МЕТОД ТА ЗАСІБ АВТЕНТИФІКАЦІЇ КОРИСТУВАЧІВ ІЗ НУЛЬОВИМ ЗНАННЯМ»
08-20.МКР.011.00.000 ТЗ

Керівник магістерської кваліфікаційної роботи
к. т. н., доц. каф. ЗІ Ю. В. Баришев

Вінниця 2021

1 Підстави для проведення робіт

Робота проводиться на підставі наказу ректора ВНТУ від 1 вересня 2021 року № 207.

Дата початку роботи 01.09.2021 р.
Дата закінчення роботи 23.12.2021 р.

2 Мета та призначення НДР

Метою магістерської кваліфікаційної роботи є покращення захищеності автентифікації користувачів за допомогою використання підходу з нульовим знанням в якості протоколу автентифікації.

Об'єктом дослідження є процес автентифікації користувачів.

Предметом є засоби та методи автентифікації користувачів.

Актуальність теми. У сучасному світі багато систем працюють із інформацією, доступ до якої обмежений. Для забезпечення перевірки автентичності користувачів, що прагнуть отримати доступ до таких систем застосовують засоби автентифікації. З початком карантинних обмежень у 2020 році велика кількість сучасних компаній надає своїм співробітникам можливості віддаленого виконання завдань, що суттєво збільшило необхідність у забезпеченні підвищеної безпеки віддаленого підключення та процедури автентифікації, оскільки кількість зловмисних атак суттєво збільшилась.

Таким чином необхідність підвищення стійкості при автентифікації користувачів є дуже актуальною, оскільки відомі засоби не завжди надають достатній рівень безпеки для сучасних інформаційних систем. Для покращення захищеності автентифікації користувачів варто розглянути засоби автентифікації з найкращими характеристиками з точки зору безпеки та здійснити їх модифікацію.

3 Вихідні дані для проведення НДР

НДР проводиться вперше і вихідними даними для НДР є:

3.1 Селезньов В. І., Баришев Ю. В. Протокол автентифікації з нульовим знанням. *XLIX Науково-технічна конференція підрозділів Вінницького національного технічного університету:* тези наук.-практ. конф. ВНТУ, Вінниця, 2020 р.

3.2 Молдовян А. А., Молдовян Д. Н., Левина А. Б. Протоколы аутентификации с нулевым разглашением секрета. СПб: Университет ИТМО, 2016. 55 с.

3.3 Баришев Ю. В., Каплун В. А. Метод автентифікації віддалених користувачів для мережевих сервісів. Інформаційні технології та комп'ютерна інженерія: наук.-техн. журнал. 2014. Том 30. № 2. С. 13-17.

3.4 Ben-Sasson E., Bentov I., Horesh Y. Scalable, transparent, and post-quantum secure computational integrity. 2018. URL: https://eprint.iacr.org/2018/046.pdf (дата звернення: 21.09.2021)

3.5 Баришев Ю. В., Каплун В. А., Неуйміна К. В., Дискреційна модель та метод розмежування прав доступу до розподілених інформаційних ресурсів. Наук. Праці ВНТУ, вип. 2, Чер 2017.

3.6 Баришев Ю. В. Моделі псевдонедетермінованованих криптографічних перетворень. *Інформаційні технології та комп'ютерна інженерія*: П'ята міжнародна наук.-практ конф., м. Івано-Франківськ, 27-29 травня 2015 р 189-190.

4 Виконавці НДР

Студент групи 1БС-20м Селезньов Віталій Ігорович

5 Вимоги до виконання НДР

Для покращення рівня безпеки при автентифікації користувачів за рахунок модифікації процесу автентифікації користувачів із нульовим знанням необхідно розв'язати такі задачі:

- науково-досліджене обґрунтування необхідності дослідження та розробки нового методу автентифікації;
- аналіз інформаційних ресурсів щодо існуючих методів та засобів автентифікації користувачів із нульовим знанням;
- удосконалення методу автентифікації користувачів із нульовим знанням;
- розробка алгоритмів функціонування модулів програмного засобу;
- розробка програмного засобу згідно розроблених алгоритмів функціонування;
- експериментальне дослідження удосконалених методів автентифікації користувачів.

6 Вимоги до супровідної документації

Графічна і текстова документація повинна відповідати діючим стандартам України.

7 Етапи НДР

Робота з теми виконується у 8 етапів.

Зміст етапу	Початок	Закінчення	Очікувані результати	Звітна документація
Аналіз завдання. Вступ	01.09.2021	04.09.2021	Вступ	Чернетка вступу
Розробка технічного завдання	05.09.2021	10.09.2021	Технічне завдання	Проект технічного завдання
Обґрунтування доцільності досліджень, аналіз інформаційних джерел	11.09.2021	04.10.2021	Аналіз існуючих аналогів. Вибір напрямку дослідження Аналіз відомих методів. Постановка завдання	Чернетка першого розділу
Удосконалення методу автентифікації з нульовим знанням.	05.10.2021	24.10.2021	Математичний опис процесу автентифікації. Удосконалений протокол автентифікації з нульовим знанням	Чернетка другого розділу
Розробка алгоритму засобу	25.10.2021	30.10.2021	Алгоритм удосконаленого протоколу автентифікації з нульовим знанням	Чернетка третього розділу
Експериментальні дослідження	31.10.2021	20.11.2021	Програмний засіб, який реалізує розроблювані методи	Чернетка четвертого розділу
Розробка економічного розділу	21.11.2021	24.11.2021	Економічні показники дослідження	Чернетка з економічного розділу
Оформлення пояснювальної записки	25.11.2021	30.11.2021	Пояснювальна записка	Пояснювальна записка

8 Очікувані результати та порядок реалізації НДР

Передбачається розробка нових (удосконалення існуючих) методів автентифікації користувачів із нульовим знанням. Заплановане створення програмного засобу, який може бути використаний у навчальному процесі.

9 Матеріали які подаються після закінчення НДР

По завершенню роботи подається пояснювальна записка та ілюстративна частина.

10Порядок приймання НДР та її етапів

Апробація на науково-технічних конференціях та семінарах. Результати роботи будуть розглядатися на засіданні ДЕК із захисту магістерських кваліфікаційних робіт.

Попередній захист та доопрацювання МКР грудень 2021 р.

Представлення МКР до захисту 5 грудня 2021 р.

Захист МКР 21-23 грудня 2021 р.

11Вимоги до розроблення документації

Документація буде виконуватись за допомогою комп'ютерного набору у відповідності вимог ДСТУ 3008:2015 «Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання».

12Вимоги щодо технічного захисту інформації з обмеженим доступом

У зв'язку з тим, що дана робота не містить інформації, що потребує захисту у відповідності до законів України, заходи з її технічного захисту не передбачаються.

Розробив студент групи 1БС-20м _____Селезньов В. І.

Додаток Б
Код модуля автентифікації з нульовим знанням

Клас CryptoZK

```
package Auth ZK module.auth;
import com.google.gson.JsonArray;
import com.google.gson.JsonObject;
import java.math.BigInteger;
import java.util.ArrayList;
import java.util.List;
/**
 *
 * @author Vitalii
 */
public class CryptoZK {
    private BigInteger S=BigInteger.ZERO;
    private BigInteger[] y;
    private BigInteger a=BigInteger.ZERO;
    private static BigInteger n=BigInteger.ZERO;
    private int L=20;
    private static BigInteger q=new
BigInteger("671203333685202725329406691122280254749705789380462806183943715514889883237942 43");
    private static BigInteger p=new
BigInteger("95647806479275528135733781266203904794419563064407");;
    public CryptoZK()
    {
        n=q.multiply(p);
    }
    public CryptoZK(BigInteger q,BigInteger p,Integer L)
    {
        this.q=(q==null)?this.q:q;
        this.p=(p==null)?this.p:p;
        this.L=(L==null)?this.L:L;
        n=this.q.multiply(this.p);
    }
    public CryptoZK(int L)
    {
        this.L=L;
        n=q.multiply(p);
    }
    public CryptoZK(BigInteger S,int L)
    {
        this.S=S;
        this.L=L;
        n=q.multiply(p);
    }
    public void Innit(BigInteger[] y,BigInteger a,BigInteger S)
    {
        this.y=(y==null)?this.y:y;
        if(y!=null){
            this.y=new BigInteger[y.length];
            for(int i=0;i<y.length;i++)
                this.y[i]=y[i];
        }
        this.a=(a==null)?this.a:a;
        this.S=(S==null)?this.S:S;
        /*this.S=S;
        this.a=a;
        this.y=y;*/
    }

    /*                                  Peggy                                  */
    /*
        From S generate Public keys on Peggy site
    */
    public JsonObject GeneratePublicKeys(BigInteger S,boolean andInnit){
        n = q.multiply(p);
        BigInteger a = BigInteger.ZERO;

        do {
            a = new BigInteger(q.bitLength(), Generators.randomGenerator).abs().mod(q);
        } while (a.equals(BigInteger.ZERO) || a.equals(BigInteger.ONE)||
a.mod(p).equals(BigInteger.ZERO)|| a.mod(q).equals(BigInteger.ZERO)|| a.equals(n));
```

```
    BigInteger []yy=new BigInteger[2];
    yy[0]=a.modPow(S, n);
    yy[1]=S.multiply(S).mod(n);
    JsonObject json=new JsonObject();
    json.addProperty("a",a.toString());
    JsonArray yJson=new JsonArray();
    yJson.add(yy[0].toString());
    yJson.add(yy[1].toString());
    json.add("y",yJson);
    if(andInnit){
        Innit(yy,a,S);
    }
    return json;
}
/*
    Peggy step 1
*/
public List<BigInteger> GenerateK()
{
    List<BigInteger>k=new ArrayList<>();
    for(int i=0;i<L;i++)
    {
        BigInteger _k;
        do {
             k = new BigInteger(n.bitLength() + 1, Generators.randomGenerator).abs().mod(n);
        }while ( k.gcd(n).intValue() != 1);
        k.add(_k);
    }
    return k;
}
/*
    Peggy step 2
*/
public List<BigInteger> U(List<BigInteger>k,int bit)
{
    List<BigInteger>u=new ArrayList<>();
    switch(bit){
        case 0:
        {
            for(int i=0;i<L;i++)
            {
                u.add(a.modPow(k.get(i),n));
            }
            break;
        }
        case 1:
        {
            for(int i=0;i<L;i++)
            {
                u.add(k.get(i).multiply(k.get(i)).mod(n));
            }
            break;
        }
        default:
            return null;
    }

    return u;
}

/*
    Peggy step 3
*/
public List<BigInteger> W(List<BigInteger>k,List<Boolean>r,int bit)
{
    List<BigInteger>w=new ArrayList<>();
    switch(bit){
        case 0:
        {
            for(int i=0;i<L;i++)
            {
                if (r.get(i))
                    w.add(k.get(i).add(S).mod(n));
                else
                    w.add(k.get(i).mod(n));
            }
            break;
        }
        case 1:
```

```
        {
            for(int i=0;i<L;i++)
            {
                if (r.get(i))
                    w.add(k.get(i).multiply(S).mod(n));
                else
                    w.add(k.get(i).mod(n));
            }
            break;
        }
        default:
            return null;
    }
    return w;
}
/*                                  Victor                                  */
/*
   Victor SetPublicKeys form Peggy
*/
public boolean SetPublicKeys(JsonObject keys){
    n = q.multiply(p);
    try{
        JsonArray yJson=keys.get("y").getAsJsonArray();
        BigInteger []yy=new BigInteger[2];
        for(int i=0;i<yy.length;i++)
        {
            yy[i]=new BigInteger(yJson.get(i).getAsString());
        }

        Innit(yy,new BigInteger(keys.get("a").getAsString()),null);
    }catch(Exception ex0){
        return false;
    }
    return true;
}
/*
   Victor getPublicKeys form Peggy
*/
public JsonObject GetPublicKeys(){
    n = q.multiply(p);
    try{
        JsonObject keys=new JsonObject();
        keys.addProperty("a", a.toString());
        JsonArray yJson=new JsonArray();
        yJson.add(y[0].toString());
        yJson.add(y[1].toString());
        keys.add("y", yJson);
        return keys;
    }catch(Exception ex0){
        return null;
    }
}
/*
    Victor step 1
*/
public List<Boolean> GenerateR()
{
    List<Boolean>r=new ArrayList<>();
    for(int i=0;i<L;i++)
    {
        r.add(Generators.randomGenerator.nextBoolean());
    }
    return r;
}
/*
    Victor step 2
*/
public boolean Verificate(List<BigInteger>u,List<BigInteger>w,List<Boolean>r,int bit)
{
    try {
        if (w.size()!=L || u.size()!=L||r.size()!=L) return false;
        switch(bit){
            case 0:
            {
                for(int i=0;i<L;i++)
                {
                    BigInteger t1=u.get(i).multiply(r.get(i)?y[bit]:BigInteger.ONE).mod(n);
                    BigInteger t2=a.modPow(w.get(i),n);
                    if(t1.compareTo(t2)!=0) return false;
```

```
                    }
                    break;
                }
                case 1:
                {
                    for(int i=0;i<L;i++)
                    {
                        BigInteger t1=w.get(i).multiply(w.get(i)).mod(n);
                        BigInteger t2=u.get(i).multiply(r.get(i)? y[bit]:BigInteger.ONE).mod(n);
                        if(t1.compareTo(t2)!=0) return false;
                    }
                    break;
                }
                default:
                    return false;
            }
        }catch (Exception e){return false;}
        return true;
    }
    public static BigInteger getN() {
        return n;
    }
}
```

Клас Generators

```
public class Generators {
    public static SecureRandom randomGenerator=new SecureRandom(new
SecureRandom().generateSeed(32));
    public static BigInteger GenerateSecretKeyS()
    {
        BigInteger s=BigInteger.ZERO;
        s=new BigInteger(256, randomGenerator);
        return s;
    }
    public static String GenerateTokenKey(int byteslen)
    {
        byte[]ar=new byte[byteslen];
        randomGenerator.nextBytes(ar);
        return new String(Base64.getEncoder().encode(ar),StandardCharsets.UTF_8);
    }
    /**
     *
     * @param NumberofBits
     * @return big number where bitlength=NumberofBits
     */
    public static BigInteger GenerateNumber(int NumberofBits)
    {
        BigInteger s=BigInteger.ZERO;
        s=new BigInteger(NumberofBits, randomGenerator);
        return s;
    }
    public static int LFSRGenerator(BigInteger Rg,int ttl)
    {
        LFSR lfsr = new LFSR(Rg);
        return lfsr.nextBit(ttl);
    }
    /**
     *
     * @param I - user identifier
     * @param G - Token granted access
     * @param key key to encrypt/decrypt secret in/from token
     * @return token
     */
    public static Token TokenGenerator(String I,JsonObject G,String key)
    {
        JsonObject jsontoken=new JsonObject();
        jsontoken.addProperty("id", GenerateNumber(32).intValue());
        jsontoken.addProperty("I",I);
        JsonObject data=new JsonObject();;
        data.addProperty("random", GenerateNumber(32));
        data.add("G", G);
        data.addProperty("Rg",GenerateNumber(32).toString());
        jsontoken.addProperty("edata",AES.Encrypt(data.toString(), key));
        Token token = new Token(jsontoken,key);
        return token;
    }
}
```

Клас класу LFSR

```
package Auth_ZK_module.auth;
import java.math.BigInteger;
import java.util.Random;
/**
 *
 * @author Vitalii
 */
public final class LFSR extends Random {
    private final BigInteger characteristic;
    private final int degree;
    private BigInteger registerState;
    private BigInteger modul;
    /**
     *
     * @param charis polynom that used for xor operation
     * @param Rg set Rg to registerState
     * @param modul use for miniminize memory usage of registerState
     */
    public LFSR(BigInteger charis, BigInteger Rg,BigInteger modul)
    {
        this.modul=modul;
       characteristic = charis;
       degree = charis.bitLength() - 1;
       this.registerState = Rg;
    }
    /**
     *
     * @param Rg set Rg to registerState and init LFSR with inbuild settings
     */
    public LFSR(BigInteger Rg)
    {
        characteristic = new BigInteger("80200003",16);
        modul = new BigInteger("ffffffff",16);
       degree = characteristic.bitLength() - 1;
       this.registerState = Rg;
    }
    /**
     *
     * @return first int bit from registerState after 1 tact
     */
    public int nextBit()
    {
       boolean result = registerState.testBit(0);
       registerState = registerState.shiftLeft(1).mod(modul);
       if (registerState.testBit(degree))
            registerState = registerState.xor(characteristic);
       return result?1:0;
    }
    /**
     *
     * @param tacts count of tacts that LFSR will do
```

```
             * @return first int bit from registerState after 1 tact
             */
            public int nextBit(int tacts)
            {
               boolean result =false;
                for(int i=0;i<tacts+1;i++)
                {
                    result = registerState.testBit(0);
                    registerState = registerState.shiftLeft(1).mod(modul);
                    if (registerState.testBit(degree))
                        registerState = registerState.xor(characteristic);
                }
                return result?1:0;
            }
            /**
             *
             * @return first boolean bit from registerState after 1 tact
             */
            @Override
            public boolean nextBoolean()
            {
               boolean result = registerState.testBit(0);
               registerState = registerState.shiftLeft(1).mod(modul);
               if (registerState.testBit(degree))
                    registerState = registerState.xor(characteristic);
               return result;
            }
            /**
             *
             * @return first int number from registerState after bits tact`s
             */
            @Override
            protected int next(int bits)
            {
               int result = 0;
                for (int i = 0; i < bits; i++)
                    result = (result << 1) | (nextBoolean() ? 1 : 0);
                return result;
            }
        }
```

Клас Token

```
/**
 *
 * @author Vitalii
 */
public class Token {
    private int id;
    private String I;
    private String edata;
    private JsonObject data;
    private boolean decrypted=false;
    /**
     *
     * @param jsonToken JsonObject of token {"id":int,"I":string,"edata":string}
     */
    public Token(JsonObject jsonToken)
    {
```

```
    if(JsonContainsToken(jsonToken)){
        id=jsonToken.get("id").getAsInt();
        I=jsonToken.get("I").getAsString();
        edata=jsonToken.get("edata").getAsString();
    }
}
/**
 *
 * @param jsonToken JsonObject of token {"id":int,"I":string,"edata":string}
 * @param key key for decryption inline in constractor
 */
public Token(JsonObject jsonToken,String key)
{
    if(JsonContainsToken(jsonToken)){
        id=jsonToken.get("id").getAsInt();
        I=jsonToken.get("I").getAsString();
        edata=jsonToken.get("edata").getAsString();
        DecryptSecretPart(key);
    }
}
/**
 *
 * @param  jsonToken
 * @return true if JsonObject contains token properties
 */
public static boolean JsonContainsToken(JsonObject jsonToken)
{
    try{
       jsonToken.get("id").getAsInt();
       jsonToken.get("I").getAsString();
       jsonToken.get("edata").getAsString();
    }catch(Exception ex){
        return false;
    }
    return true;
}
/**
 *
 * @param  jsonToken
 * @return true if JsonObject contains token with same data
 */
public boolean TokenContainsJson(JsonObject jsonToken)
{
    try{
       if(id==jsonToken.get("id").getAsInt()
        &&I==jsonToken.get("I").getAsString()
        &&edata==jsonToken.get("edata").getAsString())
            return true;
    }catch(Exception ex){
        return false;
    }
    return false;
}
/**
 *
 * @param  key
 * @return true if in current token edata success decrypted to data
 */
public boolean DecryptSecretPart(String key)
{
    try{
        data=new Gson().fromJson(AES.Decrypt(edata, key), JsonElement.class).getAsJsonObject();
        decrypted=true;
    }catch(Exception ex){return false;}
    return true;
}
/**
 *
 * @return JsonObject of token {"id":int,"I":string,"edata":string}
 */
public JsonObject toJsonObject()
{
   JsonObject token=new JsonObject();
   token.addProperty("id", id);
   token.addProperty("I", I);
   token.addProperty("edata", edata);
   return token;
}
/**
```

```
     *
     * @return return Rg from edata if it`s decrypted, else return null
     */
    public BigInteger getRg()
    {
        try{
            if(decrypted)
            {
                return data.get("Rg").getAsBigInteger();
            }
        }catch(Exception ex){}
        return null;
    }
    /**
     *
     * @return return Rg from edata if it`s decrypted or decrypt it by key, else return null
     */
    public BigInteger getRg(String key)
    {
        try{
            if(decrypted)
            {
                return new BigInteger(data.get("Rg").getAsString());
            }else
            {
                this.DecryptSecretPart(key);
                return new BigInteger(data.get("Rg").getAsString());
            }
        }catch(Exception ex){}
        return null;
    }
   /**
     *
     * @return return G from edata if it`s decrypted, else return null
     */
    public JsonObject getG()
    {
       try{
            if(decrypted)
            {
                return data.get("G").getAsJsonObject();
            }
        }catch(Exception ex){}
        return null;
    }
    /**
     *
     * @return return G from edata if it`s decrypted or decrypt it by key, else return null
     */
    public JsonObject getG(String key)
    {
       try{
            if(decrypted)
            {
                return data.get("G").getAsJsonObject();
            }else
            {
                this.DecryptSecretPart(key);
                return data.get("G").getAsJsonObject();
            }
        }catch(Exception ex){}
        return null;
    }
    public String getI() {
        return I;
    }
    public int getId() {
        return id;
    }
}
```

Клас AuthenticationMethod

```
package Auth_ZK_module.auth;
import com.google.gson.JsonObject;
import java.math.BigInteger;
import java.util.List;
/**
 *
```

```
 * @author Vitalii
 */
public class AuthenticationMethod {
    private CryptoZK cryptoZK;
    private Token token;
    private BigInteger S;
    private String TokenKey;
    private int ttl;
    private int bit;
    //-----------Auth Params--------------
    private List<BigInteger>k;//Peggy only
    private List<BigInteger>u;
    private List<Boolean>r;
    private List<BigInteger>w;
    /**
     * For Peggy
     * @param SKey Peggy secret
     * @param Tokenkey Peggy secret key for token
     */
    public AuthenticationMethod(BigInteger SKey, String Tokenkey)
    {
        this.S=SKey;
        this.TokenKey=Tokenkey;
        cryptoZK=new CryptoZK(SKey,10);
        //stage=0;
    }
    /**
     * For Victor
     * @param Tokenkey Peggy secret key for token from BD
     */
    public AuthenticationMethod(String Tokenkey)
    {
        this.TokenKey=Tokenkey;
        cryptoZK=new CryptoZK(10);
        //stage=0;
    }
    /**
     * For Peggy
     * @return  Peggy public keys {a,y=[]} used on registration
     */
    public JsonObject GenerateNewPublicKeys()
    {
        return cryptoZK.GeneratePublicKeys(S, false);
    }
    /**
     * For Peggy  and Victor
     * @return  Peggy setToken before auth or after regist
     *          Victor setToken from DB
    */
    public void setToken(JsonObject jsonToken)
    {
        token=new Token(jsonToken,TokenKey);
    }
    public Token getToken()
    {
        return this.token;
    }
    public void SetTTL(int ttl)
    {
        this.ttl=ttl;
    }
    public int GetTTL()
    {
        return this.ttl;
    }
    public String getTokenKey()
    {
        return this.TokenKey;
    }
    /**
     * Peggy and Victor after Token check
     */
    public void ExecuteLFSR()
    {
        bit=Generators.LFSRGenerator(token.getRg(),ttl);
    }
    /**
     * Peggy Compute u and ->Victor
     */
```

```
    public JsonObject ComputeU()
    {
        k=cryptoZK.GenerateK();
        u=cryptoZK.U(k, bit);
        return Generators.ListToJsonObject("u",u);
    }
    /**
    * Peggy Set r  from Victor
    */
    public boolean SetR(JsonObject r)
    {
        try{
            this.r=Generators.JsonObjectToList("r",r,Boolean.class);
            return true;
        }catch(Exception ex){return false;}
    }
    /**
    * Peggy Compute w and ->Victor
    */
    public JsonObject ComputeW()
    {
        w=cryptoZK.W(k,r,bit);
        return Generators.ListToJsonObject("w",w);
    }

    /*              Victor                  */
    /*
       Victor SetPublicKeys form Peggy
    */
    public boolean SetPublicKeys(JsonObject keys){
        try{
            if(cryptoZK.SetPublicKeys(keys))
                return true;
        }catch(Exception ex0){

        }
        return false;
    }
    public JsonObject GetPublicKeys(){
        return cryptoZK.GetPublicKeys();
    }
    /**
    * Victor set u from Peggy
    */
    public boolean SetU(JsonObject u)
    {
        try{
            this.u=Generators.JsonObjectToList("u",u,BigInteger.class);
            return true;
        }catch(Exception ex){return false;}
    }
    /**
    * Victor generate r
    */
    public JsonObject GenerateR()
    {
        r=cryptoZK.GenerateR();
        return Generators.BooleanListToJsonObject("r",r);
    }
    /**
    * Victor set w from Peggy
    */
    public boolean SetW(JsonObject w)
    {
        try{
            this.w=Generators.JsonObjectToList("w",w,BigInteger.class);
            return true;
        }catch(Exception ex){return false;}
    }
    /**
    * Victor Verificate Peggy
    */
    public boolean Verificate()
    {
        try{
            return cryptoZK.Verificate(u, w, r, bit);
        }catch(Exception ex){return false;}
    }
}
```

Додаток В
Код модуля тестування

Тестування класу CryptoZK

```
package Tests;
import Auth_ZK_module.auth.CryptoZK;
import Auth_ZK_module.auth.Generators;
import com.google.gson.JsonObject;
import java.math.BigInteger;
import java.util.List;
import org.junit.Assert;
/**
 *
 * @author Vitalii
 */
public class CryptoZKTest {

    @org.junit.Test
    public void GenerateKTest() {
        for(int j=1;j<15;j++){
            int L=j<5?5:j;
            CryptoZK cryptoZK=new CryptoZK(L);
            List<BigInteger>k=cryptoZK.GenerateK();
            Assert.assertTrue(k.size()==L);
            for(int i=0;i<k.size();i++)
                Assert.assertTrue(k.get(i)!=null);
        }
    }
    @org.junit.Test
    public void GenerateRTest() {
        for(int j=1;j<15;j++){
            int L=j<5?5:j;
            CryptoZK cryptoZK=new CryptoZK(L);
            List<Boolean>k=cryptoZK.GenerateR();
            Assert.assertTrue(k.size()==L);
            for(int i=0;i<k.size();i++)
                Assert.assertTrue(k.get(i)!=null);
        }
    }
    @org.junit.Test
    public void GenerateUTest1() {
        for(int j=1;j<15;j++){
            int L=j<5?5:j;
            int bit=j%2;
            CryptoZK cryptoZK=new CryptoZK(L);
            cryptoZK.GeneratePublicKeys(Generators.GenerateSecretKeyS(),true);
            List<BigInteger>k=cryptoZK.GenerateK();
            List<BigInteger>u=cryptoZK.U(k, bit);
            Assert.assertTrue(k.size()==L);
            Assert.assertTrue(u.size()==L);
            for(int i=0;i<k.size();i++){
                Assert.assertTrue(u.get(i)!=null);
            }
        }
    }
    @org.junit.Test
    public void GenerateUTest2() {
        for(int j=1;j<15;j++){
            int L=j<5?5:j;
            int bit=j%2;
            CryptoZK cryptoZK=new CryptoZK(L);
            cryptoZK.GeneratePublicKeys(Generators.GenerateSecretKeyS(),true);
            List<BigInteger>k=cryptoZK.GenerateK();
            List<BigInteger>u=cryptoZK.U(k, bit);
            List<BigInteger>u2=cryptoZK.U(k, (bit+1)%2);
            Assert.assertTrue(u2.size()==u.size());
            for(int i=0;i<u.size();i++){
                Assert.assertTrue(u.get(i)!=u2.get(i));
            }
        }
    }
    @org.junit.Test
    public void GenerateWTest1() {
        for(int j=1;j<15;j++){
```

```
        int L=j<5?5:j;
        int bit=j%2;
        CryptoZK cryptoZK=new CryptoZK(L);
        List<BigInteger>k=cryptoZK.GenerateK();
        List<Boolean>r=cryptoZK.GenerateR();
        List<BigInteger>w=cryptoZK.W(k,r, bit);
        Assert.assertTrue(w.size()==L);
        for(int i=0;i<k.size();i++){
            Assert.assertTrue(w.get(i)!=null);
        }
    }
}
@org.junit.Test
public void GenerateWTest2() {
    for(int j=1;j<15;j++){
        int L=j<5?5:j;
        int bit=j%2;
        CryptoZK cryptoZK=new CryptoZK(L);
        List<BigInteger>k=cryptoZK.GenerateK();
        List<Boolean>r=cryptoZK.GenerateR();
        List<BigInteger>w=cryptoZK.W(k,r, bit);
        List<BigInteger>w2=cryptoZK.W(k,r, (bit+1)%2);
        Assert.assertTrue(w2.size()==w.size());
        Assert.assertTrue(L==w.size());
        for(int i=0;i<w.size();i++){
            Assert.assertTrue(w.get(i)!=w2.get(i));
        }
    }
}
@org.junit.Test
public void VerificateTest1() {
    for(int j=1;j<15;j++){
        int L=j<5?5:j;
        int bit=j%2;
        CryptoZK cryptoZK=new CryptoZK(L);
        cryptoZK.GeneratePublicKeys(Generators.GenerateSecretKeyS(),true);
        List<BigInteger>k=cryptoZK.GenerateK();
        List<BigInteger>u=cryptoZK.U(k, bit);
        List<Boolean>r=cryptoZK.GenerateR();
        List<BigInteger>w=cryptoZK.W(k,r, bit);
        Assert.assertTrue(cryptoZK.Verificate(u, w, r, bit));
        //Assert.assertTrue(cryptoZK.Verificate(u, w, r, bit));
    }
}
@org.junit.Test
public void VerificateTest2() {
    for(int j=1;j<15;j++){
        int L=j<10?10:j;
        int bit=j%2;
        //Peggy
        CryptoZK Peggy=new CryptoZK(L);
        BigInteger SPeggy=Generators.GenerateSecretKeyS();
        JsonObject PublicKeysPeggy =Peggy.GeneratePublicKeys(SPeggy,true);
        List<BigInteger>k=Peggy.GenerateK();
        List<BigInteger>u=Peggy.U(k, bit);
        //Victor
        CryptoZK Victor=new CryptoZK(L);
        Victor.SetPublicKeys(PublicKeysPeggy);
        List<Boolean>r=Victor.GenerateR();
        List<BigInteger>w=Peggy.W(k,r, bit);
        Assert.assertTrue(Victor.Verificate(u, w, r, bit));
    }
}
@org.junit.Test
public void VerificateTest3() {
    for(int j=1;j<15;j++){
        int L=j<10?10:j;
        int bit=j%2;
        //Peggy
        CryptoZK Peggy=new CryptoZK(L);
        BigInteger SPeggy=Generators.GenerateSecretKeyS();
        JsonObject PublicKeysPeggy =Peggy.GeneratePublicKeys(SPeggy,true);
        //Carl
        CryptoZK Carl=new CryptoZK(L);
        BigInteger SCarl=Generators.GenerateSecretKeyS();
        Carl.GeneratePublicKeys(SCarl,true);
        List<BigInteger>k=Carl.GenerateK();
        List<BigInteger>u=Carl.U(k, bit);
        //Victor
```

```
            CryptoZK Victor=new CryptoZK(L);
            Victor.SetPublicKeys(PublicKeysPeggy);
            List<Boolean>r=Victor.GenerateR();
            List<BigInteger>w=Carl.W(k,r, bit);
            Assert.assertFalse(Victor.Verificate(u, w, r, bit));
            k=Peggy.GenerateK();
            u=Peggy.U(k, bit);
            w=Peggy.W(k, r, bit);
            Assert.assertTrue(Victor.Verificate(u, w, r, bit));
        }
    }
    @org.junit.Test
    public void VerificateTest4() {
        for(int j=1;j<15;j++){
            int L=j<10?10:j;
            int bit=j%2;
            //Peggy
            CryptoZK Peggy=new CryptoZK(L);
            BigInteger SPeggy=Generators.GenerateSecretKeyS();
            JsonObject PublicKeysPeggy =Peggy.GeneratePublicKeys(SPeggy,true);
            //Carl
            CryptoZK Carl=new CryptoZK(L);
            BigInteger SCarl=Generators.GenerateSecretKeyS();
            JsonObject CarlStealPeggyPublicKeys=PublicKeysPeggy;
            Carl.Innit(null, null, SCarl);
            Carl.SetPublicKeys(CarlStealPeggyPublicKeys);
            List<BigInteger>k=Carl.GenerateK();
            List<BigInteger>u=Carl.U(k, bit);
            //Victor
            CryptoZK Victor=new CryptoZK(L);
            Victor.SetPublicKeys(PublicKeysPeggy);
            List<Boolean>r=Victor.GenerateR();
            List<BigInteger>w=Carl.W(k,r, bit);
            Assert.assertFalse(Victor.Verificate(u, w, r, bit));

            k=Peggy.GenerateK();
            u=Peggy.U(k, bit);
            w=Peggy.W(k, r, bit);
            Assert.assertTrue(Victor.Verificate(u, w, r, bit));
        }
    }
}
```

Тестування класу Generators

```
public class GeneratorsTest {
    @org.junit.Test
    public void TokenGeneratorTestRg() {
        String I="John";
        JsonObject G=new JsonObject();
        G.addProperty("r", Boolean.TRUE);
        G.addProperty("w", Boolean.FALSE);
        G.addProperty("e", Boolean.FALSE);
        String key="test this key!";
        Token token=Generators.TokenGenerator(I, G, key);
        Assert.assertNotEquals(token.getRg(), Generators.GenerateNumber(32));
    }
    @org.junit.Test
    public void  LFSRGeneratorTest()
    {
        BigInteger Rg=Generators.GenerateNumber(32);
        LFSR lfsr = new LFSR(Rg);
        long firstnumber=lfsr.nextLong();
        for(int i=0;i<BigInteger.valueOf(0x0fffff).intValue();i++){
            long next=lfsr.nextLong();
            Assert.assertNotEquals(next,firstnumber);
        }
    }
}
```

Тестування класу Token

```
/**
 *
 * @author Vitalii
 */
public class TokenTest {
    @org.junit.Test
    public void TokenEdataEncryptTest() {
```

```
        String key=Generators.GenerateSecretKeyS().toString();
        JsonObject firstdata=new JsonObject();
        JsonObject G=new JsonObject();
        G.addProperty("r", Boolean.TRUE);
        G.addProperty("w", Boolean.FALSE);
        G.addProperty("e", Boolean.FALSE);
        firstdata.add("G",G);
        firstdata.addProperty("Rg", Generators.GenerateSecretKeyS().toString());
        String edata=AES.Encrypt(firstdata.toString(),key);
        Assert.assertEquals(firstdata.toString(), data);
    }
    @org.junit.Test
    public void TokenEdataEncryptTest2() {
        String key="49547728960338614775131700184442096063613891784941023566111434564021107853469";
        JsonObject firstdata=new JsonObject();
        JsonObject G=new JsonObject();
        G.addProperty("r", Boolean.TRUE);
        G.addProperty("w", Boolean.FALSE);
        G.addProperty("e", Boolean.FALSE);
        firstdata.add("G",G);
firstdata.addProperty("Rg","290333658901565069676111254985910523586558211733396318215089513113300889
88158");        String edata=AES.Encrypt(firstdata.toString(),key);
        Assert.assertEquals(edata,
"AI2flmo9AZ4mgj5YWyk+LYueor4Ua63vLTUakZ/JrenKVkBBuDlYk8nx/6HEDkAp4IauNK+Xb/xEuGNMxKQ0n8IDKoZ15LI5EjL
OBrhLwMc3HNHtlGjK0AZ5fX0Rk7RZNIvOWVfhs+b/RtimoLTFkCYvvFYIh3Gz0FnPsjXeD3I=" +"");
    }
    @org.junit.Test
    public void TokenEdataDecryptTest() {
        String key=Generators.GenerateSecretKeyS().toString();
        JsonObject firstdata=new JsonObject();
        JsonObject G=new JsonObject();
        G.addProperty("r", Boolean.TRUE);
        G.addProperty("w", Boolean.FALSE);
        G.addProperty("e", Boolean.FALSE);
        firstdata.add("G",G);
        firstdata.addProperty("Rg", Generators.GenerateSecretKeyS().toString());
        String edata=AES.Encrypt(firstdata.toString(),key);
        String data=AES.Decrypt(edata,key);
        Assert.assertEquals(firstdata.toString(), data);
    }
    @org.junit.Test
    public void TokenTestG() {
        String I="John";
        JsonObject G=new JsonObject();
        G.addProperty("r", Boolean.TRUE);
        G.addProperty("w", Boolean.FALSE);
        G.addProperty("e", Boolean.FALSE);
        String key="test this key!";
        Token token=Generators.TokenGenerator(I, G, key);
        Assert.assertEquals(token.getG().get("r").getAsBoolean(), Boolean.TRUE);
        Assert.assertEquals(token.getG().get("w").getAsBoolean(), Boolean.FALSE);
        Assert.assertNotEquals(token.getG().get("e").getAsBoolean(), Boolean.TRUE);
    }
    @org.junit.Test
    public void TokenTestI() {
        String I="John";
        JsonObject G=new JsonObject();
        G.addProperty("r", Boolean.TRUE);
        G.addProperty("w", Boolean.FALSE);
        G.addProperty("e", Boolean.FALSE);
        String key="test this key!";
        Token token=Generators.TokenGenerator(I, G, key);
        Assert.assertEquals(token.getI(),I);
    }
    @org.junit.Test
    public void TokenTestId() {
        String I="John";
        JsonObject G=new JsonObject();
        G.addProperty("r", Boolean.TRUE);
        G.addProperty("w", Boolean.FALSE);
        G.addProperty("e", Boolean.FALSE);
        String key="test this key!";
        Token token=Generators.TokenGenerator(I, G, key);
        Assert.assertNotEquals(token.getId(),0);
    }
    @org.junit.Test
    public void TokenTesttoJsonObject() {
        String I="John";
        JsonObject G=new JsonObject();
```

```
        G.addProperty("r", Boolean.TRUE);
        G.addProperty("w", Boolean.FALSE);
        G.addProperty("e", Boolean.FALSE);
        String key="test this key!";
        Token token=Generators.TokenGenerator(I, G, key);
        Assert.assertEquals(token.toJsonObject().get("I").getAsString(),I);
    }
    @org.junit.Test
    public void TokenTestCantainsJsonObject() {
        String I="John";
        JsonObject G=new JsonObject();
        G.addProperty("r", Boolean.TRUE);
        G.addProperty("w", Boolean.FALSE);
        G.addProperty("e", Boolean.FALSE);
        String key="test this key!";
        Token token=Generators.TokenGenerator(I, G, key);

        Assert.assertEquals(token.TokenContainsJson(token.toJsonObject()),true);
    }

    @org.junit.Test
    public void TokenTestG2() {
        String I="John";
        JsonObject G=new JsonObject();
        G.addProperty("r", Boolean.TRUE);
        G.addProperty("w", Boolean.FALSE);
        G.addProperty("e", Boolean.FALSE);
        String key="test this key!";
        Token token=Generators.TokenGenerator(I, G, key);
        Token tokennew=new Token(token.toJsonObject());
        Assert.assertEquals(tokennew.getG(key).equals(G),true);
    }
}
```

Тестування класу AuthenticationMethod

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Tests;

import Auth_ZK_module.auth.AuthenticationMethod;
import Auth_ZK_module.auth.CryptoZK;
import Auth ZK module.auth.Generators;
import Auth ZK module.auth.Token;
import com.google.gson.JsonArray;
import com.google.gson.JsonObject;
import java.math.BigInteger;
import java.util.List;
import org.junit.Assert;

/**
 *
 * @author Vitalii
 */
public class AuthenticationMethodTest {
    private static BigInteger q=new
BigInteger("671203333685202725329406691122280254749705789380462806183943715514889883237942 43");
    private static BigInteger p=new
BigInteger("95647806479275528135733781266203904794419563064407");
    private static BigInteger n=q.multiply(p);
    @org.junit.Test
    public void GenerateNewPublicKeysTest() {
        for(int j=1;j<15;j++){
            int L=j<5?5:j;
            String tokenkey=Generators.GenerateTokenKey(24);
            BigInteger S=Generators.GenerateSecretKeyS();
            AuthenticationMethod am=new AuthenticationMethod(S,tokenkey);
            JsonObject keys=am.GenerateNewPublicKeys();
            String y0= new BigInteger(keys.get("a").getAsString()).modPow(S, n).toString();
            String y0real=keys.get("y").getAsJsonArray().get(0).getAsString();
            Assert.assertTrue(keys.has("a"));
            Assert.assertTrue(keys.has("y"));
            Assert.assertTrue(y0.equals(y0real));
        }
    }
```

```
@org.junit.Test
public void GenerateNewPublicKeysTest2() {
    for(int j=1;j<15;j++){
        int L=j<5?5:j;
        String tokenkey=Generators.GenerateTokenKey(24);
        BigInteger S=Generators.GenerateSecretKeyS();
        AuthenticationMethod am=new AuthenticationMethod(S,tokenkey);
        JsonObject keys=am.GenerateNewPublicKeys();
        String y1=S.multiply(S).mod(n).toString();
        String y1real=keys.get("y").getAsJsonArray().get(1).getAsString();
        Assert.assertTrue(keys.has("a"));
        Assert.assertTrue(keys.has("y"));
        Assert.assertTrue(y1.equals(y1real));
    }
}

 @org.junit.Test
public void setTokenTest() {
    for(int j=1;j<15;j++){
        int L=j<5?5:j;
        String tokenkey=Generators.GenerateTokenKey(24);
        BigInteger S=Generators.GenerateSecretKeyS();
        AuthenticationMethod am=new AuthenticationMethod(S,tokenkey);
        Token tmp=Generators.TokenGenerator("123", new JsonObject(), tokenkey);
        am.setToken(tmp.toJsonObject());
        Assert.assertEquals(am.getTokenKey(),tokenkey);
        Assert.assertEquals(am.getToken().getId(),tmp.getId());
    }
}
 @org.junit.Test
public void setTokenTest2() {
    for(int j=1;j<15;j++){
        int L=j<5?5:j;
        String tokenkey=Generators.GenerateTokenKey(24);
        BigInteger S=Generators.GenerateSecretKeyS();
        AuthenticationMethod am=new AuthenticationMethod(S,tokenkey);
        Token tmp=Generators.TokenGenerator("123", new JsonObject(), tokenkey);
        am.setToken(tmp.toJsonObject());
        Assert.assertEquals(am.getTokenKey(),tokenkey);
        Assert.assertEquals(am.getToken().getG(tokenkey),tmp.getG(tokenkey));
    }
}
@org.junit.Test
public void GenerateRTest() {
    for(int j=1;j<15;j++){
        int L=j<5?5:j;
        String tokenkey=Generators.GenerateTokenKey(24);
        BigInteger S=Generators.GenerateSecretKeyS();
        AuthenticationMethod am=new AuthenticationMethod(S,tokenkey);
        JsonObject r=am.GenerateR();
        Assert.assertTrue(r.has("r"));
        JsonArray ar=r.get("r").getAsJsonArray();
        for(int i=0;i<ar.size();i++)
            Assert.assertTrue(ar.get(i)!=null);
    }
}
@org.junit.Test
public void GenerateRTest2() {
    for(int j=1;j<15;j++){
        int L=j<5?5:j;
        String tokenkey=Generators.GenerateTokenKey(24);
        BigInteger S=Generators.GenerateSecretKeyS();
        AuthenticationMethod am=new AuthenticationMethod(S,tokenkey);
        JsonObject r=am.GenerateR();
        Assert.assertTrue(r.has("r"));
        JsonArray ar=r.get("r").getAsJsonArray();
        for(int i=0;i<ar.size();i++)
            Assert.assertTrue(!ar.get(i).isJsonNull());
    }
}

@org.junit.Test
public void GenerateUTest1() {
    for(int j=1;j<15;j++){
        int L=j<5?5:j;
        int bit=j%2;
        String tokenkey=Generators.GenerateTokenKey(24);
        BigInteger S=Generators.GenerateSecretKeyS();
        AuthenticationMethod am=new AuthenticationMethod(S,tokenkey);
```

```
        JsonObject pk=am.GenerateNewPublicKeys();
        am.SetPublicKeys(pk);
        Token token=Generators.TokenGenerator("123", new JsonObject(), tokenkey);
        am.setToken(token.toJsonObject());
        am.SetTTL(1);
        am.ExecuteLFSR();
        JsonObject u=am.ComputeU();
        JsonArray ar=u.get("u").getAsJsonArray();
        for(int i=0;i<ar.size();i++)
            Assert.assertTrue(!ar.get(i).isJsonNull());
    }
}

@org.junit.Test
public void GenerateUTest2() {
    for(int j=1;j<15;j++){
        int L=j<5?5:j;
        int bit=j%2;
        String tokenkey=Generators.GenerateTokenKey(24);
        BigInteger S=Generators.GenerateSecretKeyS();
        AuthenticationMethod am=new AuthenticationMethod(S,tokenkey);
        JsonObject pk=am.GenerateNewPublicKeys();
        am.SetPublicKeys(pk);
        Token token=Generators.TokenGenerator("123", new JsonObject(), tokenkey);
        am.setToken(token.toJsonObject());
        am.SetTTL(1);
        am.ExecuteLFSR();
        JsonObject u=am.ComputeU();
        JsonArray ar=u.get("u").getAsJsonArray();
        for(int i=0;i<ar.size();i++)
            Assert.assertTrue(!ar.get(i).isJsonNull());
    }
}
@org.junit.Test
public void GenerateWTest1() {
    for(int j=1;j<15;j++){
        int L=j<5?5:j;
        int bit=j%2;
        String tokenkey=Generators.GenerateTokenKey(24);
        BigInteger S=Generators.GenerateSecretKeyS();
        AuthenticationMethod am=new AuthenticationMethod(S,tokenkey);
        JsonObject pk=am.GenerateNewPublicKeys();
        am.SetPublicKeys(pk);
        Token token=Generators.TokenGenerator("123", new JsonObject(), tokenkey);
        am.setToken(token.toJsonObject());
        am.SetTTL(1);
        am.ExecuteLFSR();
        JsonObject u=am.ComputeU();
        JsonObject r=am.GenerateR();

        JsonObject w=am.ComputeW();
        JsonArray ar=w.get("w").getAsJsonArray();
        for(int i=0;i<ar.size();i++)
            Assert.assertTrue(!ar.get(i).getAsString().equals("0"));
    }
}
@org.junit.Test
public void GenerateWTest2() {
    for(int j=1;j<15;j++){
        int L=j<5?5:j;
        int bit=j%2;
        String tokenkey=Generators.GenerateTokenKey(24);
        BigInteger S=Generators.GenerateSecretKeyS();
        AuthenticationMethod am=new AuthenticationMethod(S,tokenkey);
        JsonObject pk=am.GenerateNewPublicKeys();
        am.SetPublicKeys(pk);
        Token token=Generators.TokenGenerator("123", new JsonObject(), tokenkey);
        am.setToken(token.toJsonObject());
        am.SetTTL(1);
        am.ExecuteLFSR();
        JsonObject u=am.ComputeU();
        try{
        AuthenticationMethod am2=new AuthenticationMethod(S,tokenkey);
        am2.SetPublicKeys(pk);
        am2.setToken(token.toJsonObject());
        am2.SetTTL(1);
        JsonObject r=am2.GenerateR();
        JsonObject w=am.ComputeW();
        JsonArray ar=w.get("w").getAsJsonArray();
```

```
            boolean f=false;
            for(int i=0;i<ar.size();i++)
                if(ar.get(i).getAsString().equals("0"))
                    f=true;
            if(f)Assert.assertTrue(true);else Assert.assertTrue(false);
            }catch(Exception e){Assert.assertTrue(true);}

        }
    }
    @org.junit.Test
    public void VerificateTest1() {
        for(int j=1;j<15;j++){
            int L=j<5?5:j;
            int bit=j%2;
            String tokenkey=Generators.GenerateTokenKey(24);
            BigInteger S=Generators.GenerateSecretKeyS();
            AuthenticationMethod am=new AuthenticationMethod(S,tokenkey);
            JsonObject pk=am.GenerateNewPublicKeys();
            am.SetPublicKeys(pk);
            Token token=Generators.TokenGenerator("123", new JsonObject(), tokenkey);
            am.setToken(token.toJsonObject());
            am.SetTTL(1);
            am.ExecuteLFSR();
            JsonObject u=am.ComputeU();
            JsonObject r=am.GenerateR();
            JsonObject w=am.ComputeW();
            Assert.assertTrue(am.Verificate());
        }
    }
    @org.junit.Test
    public void VerificateTest2() {
        for(int j=1;j<15;j++){
            int L=j<10?10:j;
            int bit=j%2;
            String tokenkey="LtCbgAMaBROJB1gk2d3Ztt2uOhgbD7dGMAp9NNy8um8=";
            BigInteger S=Generators.GenerateSecretKeyS();
            AuthenticationMethod Peggy=new AuthenticationMethod(S,tokenkey);
            JsonObject pk=Peggy.GenerateNewPublicKeys();
            Peggy.SetPublicKeys(pk);
            JsonObject
token=Generators.StringToJsonObject("{\"I\":\"1\",\"id\":553353665,\"edata\":\"vnaRomkmASt2dz+WzoUaz
XRZxJyMYIDcIAv+RflNtzcJ2Dn1fwVvcwOImVG3Gy1hq+AOjz/eoyZZjumRK23UfL0iBlbnQzirXQm2RfAnd6WGVUfEdoRIJcaM9
W6NgiMI\"}");
            Peggy.setToken(token);
            Peggy.SetTTL(1);
            Peggy.ExecuteLFSR();
            JsonObject u=Peggy.ComputeU();
            //Victor
            AuthenticationMethod Victor=new AuthenticationMethod(tokenkey);
            Victor.SetPublicKeys(pk.deepCopy());
            Victor.setToken(token.deepCopy());
            Victor.SetTTL(1);
            Peggy.ExecuteLFSR();
            Assert.assertTrue(Victor.SetU(u.deepCopy()));
            JsonObject r=Victor.GenerateR();
            Assert.assertTrue(Peggy.SetR(r.deepCopy()));
            JsonObject w =Peggy.ComputeW();
            Assert.assertTrue(Victor.SetW(w.deepCopy()));
            Assert.assertTrue(Victor.Verificate());
        }
    }
   @org.junit.Test
    public void VerificateTest3() {
        for(int j=1;j<15;j++){
            int L=j<10?10:j;
            int bit=j%2;
            //Peggy
            try{
            String tokenkey="LtCbgAMaBROJB1gk2d3Ztt2uOhgbD7dGMAp9NNy8um8=";
            BigInteger S=Generators.GenerateSecretKeyS();
            AuthenticationMethod Peggy=new AuthenticationMethod(S,tokenkey);
            JsonObject pk=Peggy.GenerateNewPublicKeys();
            Peggy.SetPublicKeys(pk);
            JsonObject
token=Generators.StringToJsonObject("{\"I\":\"1\",\"id\":553353665,\"edata\":\"vnaRomkmASt2dz+WzoUaz
XRZxJyMYIDcIAv+RflNtzcJ2Dn1fwVvcwOImVG3Gy1hq+AOjz/eoyZZjumRK23UfL0iBlbnQzirXQm2RfAnd6WGVUfEdoRIJcaM9
W6NgiMI\"}");
            Peggy.setToken(token);
            Peggy.SetTTL(1);
```

```
            Peggy.ExecuteLFSR();
            JsonObject u=Peggy.ComputeU();
            //Carl
            AuthenticationMethod Carl=new AuthenticationMethod("Carl doesn`t have");
            Carl.SetPublicKeys(pk);
            JsonObject
Carltoken=Generators.StringToJsonObject("{\"I\":\"1\",\"id\":553353665,\"edata\":\"vnaRomkmASt2dz+Wz
oUazXRZxJyMYIDcIAv+RflNtzcJ2Dn1fwVvcwOImVG3Gy1hq+AOjz/eoyZZjumRK23UfL0iBlbnQzirXQm2RfAnd6WGVUfEdoRIJ
caM9W6NgiMI\"}");
            Carl.setToken(token);
            Carl.SetTTL(1);
            Carl.ExecuteLFSR();
            JsonObject Carlu=Carl.ComputeU();
            //Victor
            AuthenticationMethod Victor=new AuthenticationMethod(tokenkey);
            Victor.SetPublicKeys(pk.deepCopy());
            Victor.setToken(token.deepCopy());
            Victor.SetTTL(1);
            Peggy.ExecuteLFSR();
            Assert.assertTrue(Victor.SetU(Carlu.deepCopy()));
            }catch (Exception ex){
                Assert.assertTrue(true);
            }
        }
    }
    @org.junit.Test
    public void VerificateTest4() {
        for(int j=1;j<15;j++){
            int L=j<10?10:j;
            int bit=j%2;
            String tokenkey="LtCbgAMaBROJB1gk2d3Ztt2uOhgbD7dGMAp9NNy8um8=";
            BigInteger S=Generators.GenerateSecretKeyS();
            AuthenticationMethod Peggy=new AuthenticationMethod(S,tokenkey);
            JsonObject pk=Peggy.GenerateNewPublicKeys();
            Peggy.SetPublicKeys(pk);
            JsonObject
token=Generators.StringToJsonObject("{\"I\":\"1\",\"id\":553353665,\"edata\":\"vnaRomkmASt2dz+WzoUaz
XRZxJyMYIDcIAv+RflNtzcJ2Dn1fwVvcwOImVG3Gy1hq+AOjz/eoyZZjumRK23UfL0iBlbnQzirXQm2RfAnd6WGVUfEdoRIJcaM9
W6NgiMI\"}");
            Peggy.setToken(token);
            Peggy.SetTTL(1);
            Peggy.ExecuteLFSR();
            JsonObject u=Peggy.ComputeU();
             //Carl have Peggy tokenkey
            AuthenticationMethod Carl=new AuthenticationMethod(tokenkey);
            Carl.SetPublicKeys(pk);
            JsonObject
Carltoken=Generators.StringToJsonObject("{\"I\":\"1\",\"id\":553353665,\"edata\":\"vnaRomkmASt2dz+Wz
oUazXRZxJyMYIDcIAv+RflNtzcJ2Dn1fwVvcwOImVG3Gy1hq+AOjz/eoyZZjumRK23UfL0iBlbnQzirXQm2RfAnd6WGVUfEdoRIJ
caM9W6NgiMI\"}");
            Carl.setToken(token);
            Carl.SetTTL(1);
            Carl.ExecuteLFSR();
            JsonObject Carlu=Carl.ComputeU();
            AuthenticationMethod Victor=new AuthenticationMethod(tokenkey);
            Victor.SetPublicKeys(pk.deepCopy());
            Victor.setToken(token.deepCopy());
            Victor.SetTTL(1);
            Peggy.ExecuteLFSR();
            Assert.assertTrue(Victor.SetU(Carlu.deepCopy()));
            JsonObject r=Victor.GenerateR();
            Assert.assertTrue(Carl.SetR(r.deepCopy()));
            Assert.assertTrue(Peggy.SetR(r.deepCopy()));
            JsonObject w =Peggy.ComputeW();
            JsonObject Carlw =Carl.ComputeW();
            Assert.assertTrue(Victor.SetW(Carlw.deepCopy()));
            Assert.assertFalse(Victor.Verificate());
            Assert.assertTrue(Victor.SetU(u));
            Assert.assertTrue(Victor.SetW(w));
            Assert.assertTrue(Victor.Verificate());
        }
    }
}
```

Додаток Г
Тестування засобу автентифікації з нульовим знанням

Тестування при коректних вхідних даних

```
Peggi secret key:11771386436330103802204419109483738750814406997405208282282373813538051366756
Paggi Token-> Viktor:{"id":1760671098,"I":"Peggi","edata":"RYXw2Ggxd2JoUcOpZUNv/QhoLPxlTk2QCFDwEm5yTRe2R2oaEz/quz46Yf5jmlXdqNQOEivCbtM8Bv
Paggi <- Viktor:{"ttl":78}
Paggi decrypted Rg from Token:988355700
Viktor decrypted Rg from Token:988355700
Paggi genereted LFSR bit:1
Viktor genereted LFSR bit:1
Paggi Public Keys:{
  "a": 541093228234554105981889224654486167851606815775521117469710832933601089289150 60,
  "y": 372543442863754438036887574123261993033537225201193548976417464694788465926571053385143445134371510671050664365412791507302455791
}
Paggi K:{
  "k": [
    61009250298965057821322277790566119596743108947287517682307615661875731485096045656793636433186869867354435521532586160363108705 23,
    20703525171298952942842346409948833889434629642207316312384001453761083535121607001395091919784790058254286078339909054282189459 69
  ]
}
Paggi u-> Viktor:{
  "u": [
    43134863188628008808454016354947347405284486922982118226817952827147067973783697446772581620844257857009019632360164248634562178 84,
    44210672924395754324909283763302675389120523963590593525810496353840004883809831384940322387906318576875157858195832373841348121 80
  ]
}
Paggi <-r Viktor:{
  "r": [
    true,
    false
  ]
}
Paggi w-> Viktor:{
  "w": [
    45620997919890020548095110195411850834955910010961640398227130296301489579210325815701089726573493152720690697502415694676116647 20,
    20703525171298952942842346409948833889434629642207316312384001453761083535121607001395091919784790058254286078339909054282189459 69
  ]
}
Viktor:Verificate(u,y,w,r,bit)=True
Paggi успішно пройшла автентифікацію!!!
```

Тестування при некоректних вхідних даних

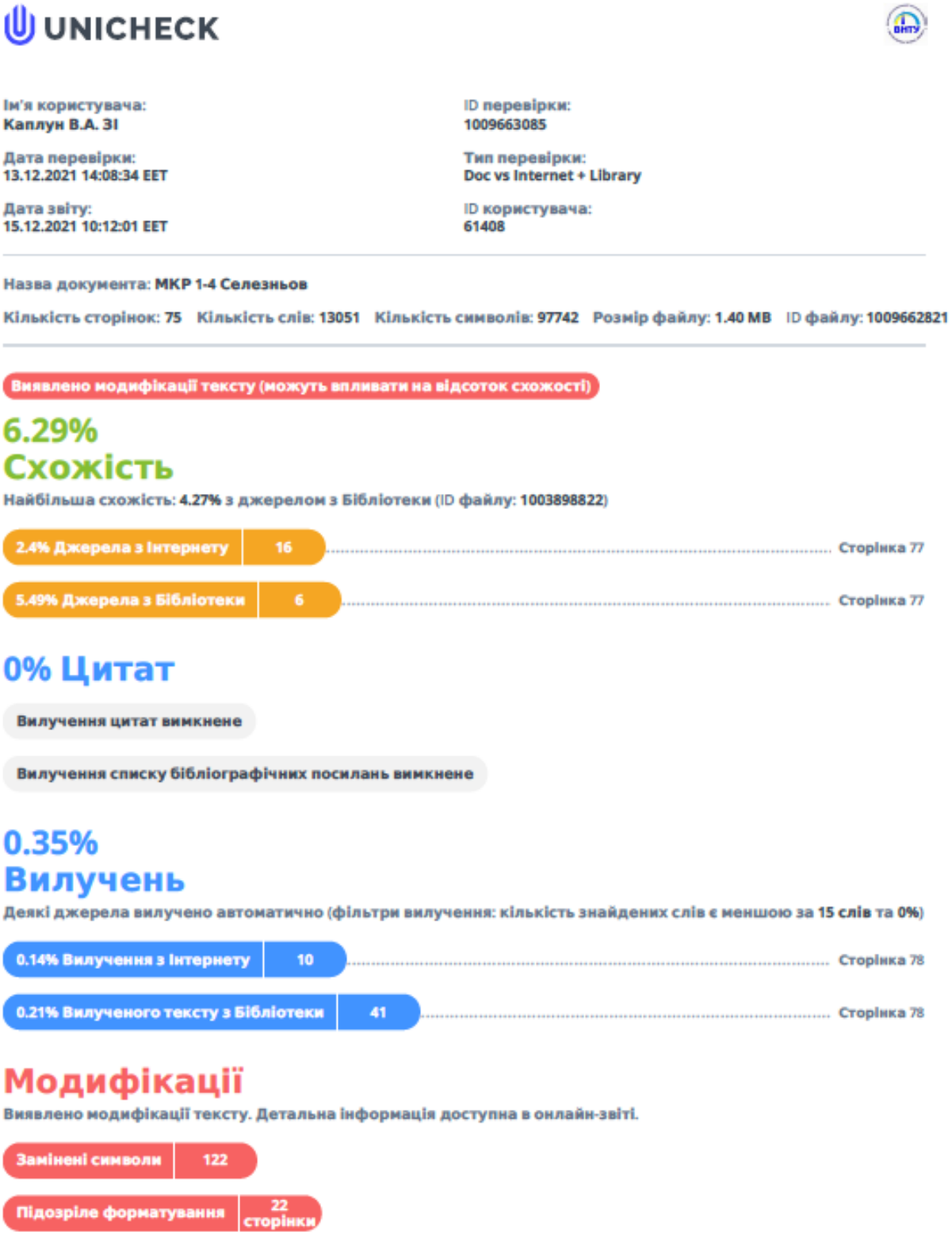
```
Peggy secret key:175288967596433170079857290579286422356564792138568727644545823731194774166 74
Carl secret key:37831220810435286752457499497598421590271487799925288869528768772507494815411
Peggy Token-> Victor:{"id":1490980280,"I":"Peggy","edata":"ASdSf2Jg9mnwHPv2WDCbElBulfLcsahNrLUpi6VwvOlp28tAoRE2m9EP4FKTsYSRjfZEdu0NjlxuLN
Peggy <- Victor:{"ttl":78}
Peggy decrypted Rg from Token:1980054452
Victor decrypted Rg from Token:1980054452
Peggy genereted LFSR bit:0
Victor genereted LFSR bit:0
Carl try bit:0
Peggy Public Keys:{
  "a": "28629139300489202561173719001805225629640871142688774087242230327607483174700032",
  "y": [
    "22761577428537868142932918495892066421021944380705554221966129985498533641020777215265920684122001399921747913106853717789074304 72",
    "38728633746891213354531804209233468890705061693992666770928277584151171072325080067649306500967682530193033216579749795780586479 77"
  ]
}
Carl stole Public Keys:{
  "a": "28629139300489202561173719001805225629640871142688774087242230327607483174700032",
  "y": [
    "22761577428537868142932918495892066421021944380705554221966129985498533641020777215265920684122001399921747913106853717789074304 72",
    "38728633746891213354531804209233468890705061693992666770928277584151171072325080067649306500967682530193033216579749795780586479 77"
  ]
}
Carl K:{
  "k": [
    14822591160117371363111013071264672910851356101136105668856304698099343929466531089198347200405084793906908665338444802916100754 9,
    12763318941513281390178517425434303436624655678399240683222966915447588539746156268202858330819274848361785852589064260417779805967
  ]
}
Carl u-> Victor:{
  "u": [
    33746240780221475118128727450195410984122663282846967035192715178503164876011218880032933332169128336084559878570440619709963 91,
    42736333140715292225385158832612791445943688964622891327373671005356289467478409266321099528289061313782481570802187799378983042 08
  ]
}
Carl <-r Victor:{
  "r": [
    true,
    false
  ]
}
Carl w-> Victor:{
  "w": [
    14822591160117371363111013071264672910851356101136106047168512803262801917521648084904050622755386478643579561821532205366558229 60,
    12763318941513281390178517425434303436624655678399240683222966915447588539746156268202858330819274848361785852589064260417779805967
  ]
}
Victor:Verificate(u,y,w,r,bit)=False
Carl Автентифікацію від імені Peggy не пройшов !!!
```

Додаток Д
Критерії оцінювання комерційного потенціалу розробки

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Критерій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює аналогам	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Критерій	0	1	2	3	4
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси. Джерела фінансування відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності більше 5-ти р	Термін реалізації ідеї менше 3-х років. Термін окупності від 3-х до 5-ти р.	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Додаток Е
Результати перевірки роботи на плагіат



UNICHECK

Ім'я користувача: Каплун В.А. ЗІ

ID перевірки: 1009663085

Дата перевірки: 13.12.2021 14:08:34 EET

Тип перевірки: Doc vs Internet + Library

Дата звіту: 15.12.2021 10:12:01 EET

ID користувача: 61408

Назва документа: МКР 1-4 Селезньов

Кількість сторінок: 75 Кількість слів: 13051 Кількість символів: 97742 Розмір файлу: 1.40 MB ID файлу: 1009662821

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

6.29%
Схожість

Найбільша схожість: 4.27% з джерелом з Бібліотеки (ID файлу: 1003898822)

2.4% Джерела з Інтернету 16 ..... Сторінка 77

5.49% Джерела з Бібліотеки 6 ..... Сторінка 77

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0.35%
Вилучень

Деякі джерела вилучено автоматично (фільтри вилучення: кількість знайдених слів є меншою за 15 слів та 0%)

0.14% Вилучення з Інтернету 10 ..... Сторінка 78

0.21% Вилученого тексту з Бібліотеки 41 ..... Сторінка 78

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи 122

Підозріле форматування 22 сторінки

Схожість

Джерела з Інтернету 16

2	https://core.ac.uk/download/pdf/323531191.pdf	0.98%
4	https://uk.wikipedia.org/wiki/%D0%94%D0%BE%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%BD%D1%8F_%D0%B7_%D0...	0.71%
7	https://topuch.ru/formuvannya-tvorchih-zdibnostej-pidlitkiv-u-pozaurochnij-robot/index3.html 13 джерел	0.54%
10	https://ua-referat.com/uploaded/vinniceka-konditerseka-fabrika/index1.html	0.17%

Джерела з Бібліотеки 6

1	**Селезньов_Р** ID файлу: **1003898822** Навчальний заклад: **Vinnytsia National Technical University**	4.27%
3	**МКР_Клешня** ID файлу: **1005753679** Навчальний заклад: **Vinnytsia National Technical University**	0.76%
5	**125БДРБрухновДА2021** ID файлу: **1008365095** Навчальний заклад: **Vinnytsia National Technical University**	0.63%
6	**192_ЛялюкАО_МКР_ПЦБ** ID файлу: **1009611173** Навчальний заклад: **Vinnytsia National Technical University**	0.55%
8	**125БДРПанасюк БВ2021** ID файлу: **1008417817** Навчальний заклад: **Vinnytsia National Technical University**	0.29%
9	**192ІванченкоСС2020МКР** ID файлу: **1003308445** Навчальний заклад: **Vinnytsia National Technical University**	0.26%

Вилучення

Вилучення 10

	http://inmad.vntu.edu.ua/graduate/avtoreferat_Ivanov.pdf 5 джерел	0.08%
	http://sambir.wunu.edu.ua/my_downloads/learning/pl/4kurs/SAKIS/POSIBN_REDAG.doc 2 джерела	0.08%
	https://en.ppt-online.org/485843	0.06%
	http://inmad.vntu.edu.ua/portal/static/CCBEB480-E136-4DBC-8AEB-D19D404EE592.pdf 2 джерела	0.06%

Вилучення по Бібліотеці акаунту 41

	Онуфрієнко_2018 ID файлу: **3680188** Навчальний заклад: **Vinnytsia National Technical University**	0.11%
	ПлБажак ID файлу: **1005788214** Навчальний заклад: **Vinnytsia National Technical University** 29 Джерело	0.1%
	192ДовгуцькаТВ2019 ID файлу: **1000790036** Навчальний заклад: **Vinnytsia National Technical University** 8 Джерело	0.09%
	073_Панкова_ВД_2021 ID файлу: **1009655297** Навчальний заклад: **Vinnytsia National Technical University**	0.09%
	БДР_Кравцев_Р ID файлу: **1004036091** Навчальний заклад: **Vinnytsia National Technical University**	0.08%
	125БДРКоломієцьДО2020 ID файлу: **1003269515** Навчальний заклад: **Vinnytsia National Technical University**	0.07%

ІЛЮСТРАТИВНА ЧАСТИНА

ПОРІВНЯННЯ МЕТОДІВ АВТЕНТИФІКАЦІЇ З НУЛЬОВИМ ЗНАННЯМ

№	Властивість	протокол автентифікації				
		Протокол Фіата-Шаміра	Протокол Шнорра	zk-SNARK	zk-STARK	Bulletproofs
1	Потреба у «trusted setup»	+	+	+	-	-
2	Час підтвердження «verification time»	12мс	13мс	10мс-20мс	10 мс	1 сек
3	Час доведення «proof time»	від 2 до 4 сек	від 2 до 4 сек	до 4 сек	до 2 сек	більше 10 сек
4	Розмір даних, що передаються	~135KB	~80KB	288B	45KB-200KB	1.3KB-10KB
5	Перевіряє автентичність двох сторін	-	-	-	-	+
6	Стійкість до атаки за обраним шифр-текстом	+	+	+	+	+
7	Стійкість до атаки «маскарад»	+	+	+	+	+
8	Стійкість до пост квантових атак	-	-	-	+	-
9	Побудований на основі дерев Меркла	-	-	+	+	-

					08–20.МКР.011.00.000 І41		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб..		Селезньов В. І.			Метод та засів автентифікації користувачів із нульовим знанням. Порівняння методів автентифікації з нульовим знанням	Літ.	Арк. 1 / Аркушів 1
Перевір.		Баришев Ю. В.					
Реценз.		Войцеховська О. В.				ВНТУ, 1БС-20м	
Н. Контр.		Баришев Ю. В.					
Затверд.		Лужецький В.А.					

МАТЕМАТИЧНИЙ ОПИС ПРОЦЕСУ АВТЕНТИФІКАЦІЇ З НУЛЬОВИМ ЗНАННЯМ

$SA_{НЗ} =< X_{НЗ}, Y_{НЗ}, B_{НЗ}, A_{НЗ}, O_{НЗ} >$, де

SA – система автентифікації;
X – множина вхідних даних;
Y – множина вихідних даних;
B – множина постійних параметрів системи;
A – множина змінних параметрів системи;
O – множина виконуваних операцій.

Для сторони клієнта:

$$X_{НЗ}^{П} = \{I^{B}; \alpha; kS; r; n; y\},\ Y_{НЗ}^{П} = \{w; u\},$$

$$B_{НЗ}^{П} = \{U; I^{B}; kS; n\},\ A_{НЗ}^{П} = \{r; \alpha; W; w; u; y\},$$

$$O_{НЗ}^{П} = \{O_1, O_2, ..., O_n\}$$

Для сторони сервера:

$$X_{НЗ}^{B} = \{I^{B}; \alpha; n; u; y\},\ Y_{НЗ}^{B} = \{r; R\},$$

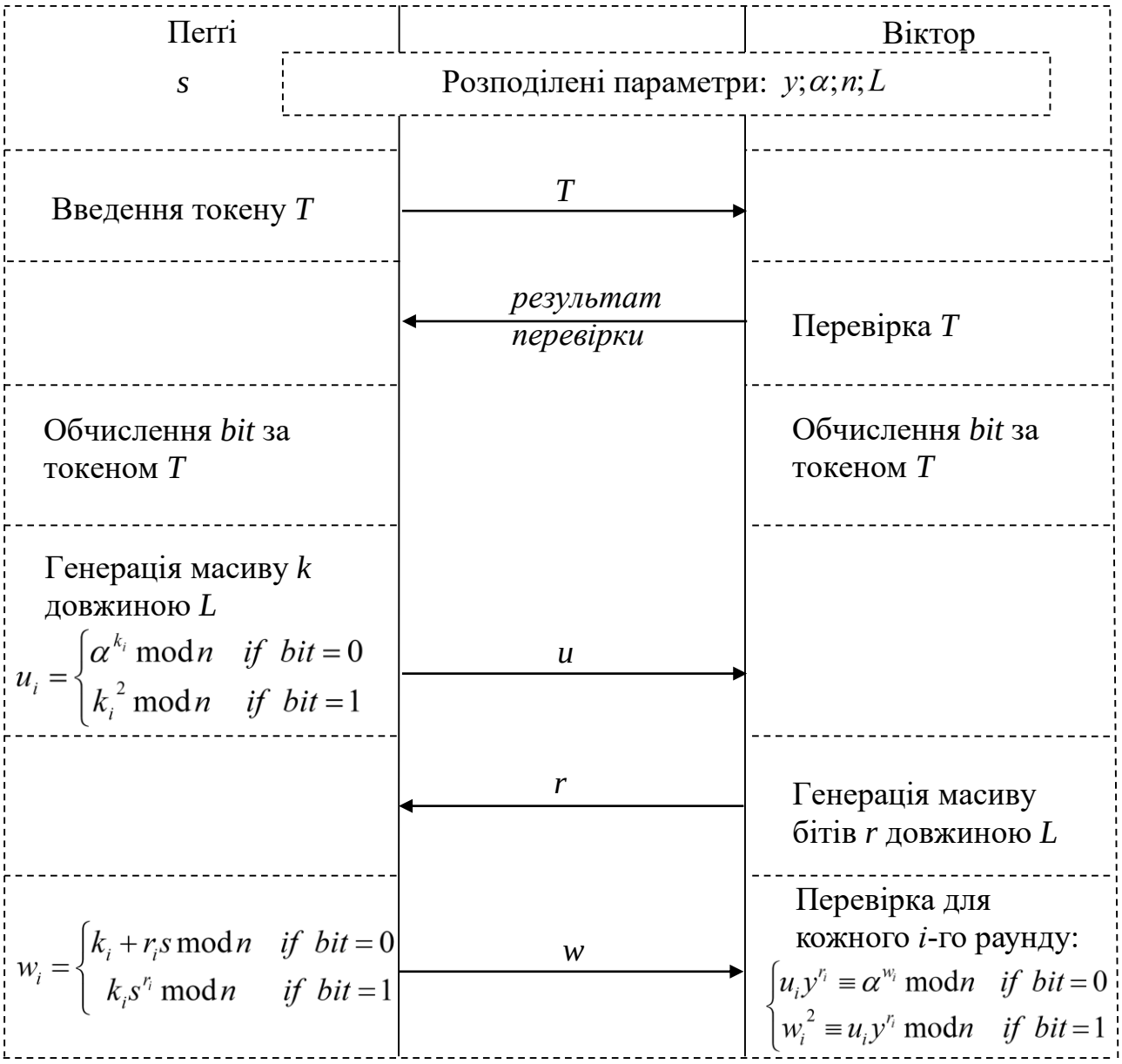
$$B_{НЗ}^{B} = \{Ver; I^{B}\},\ A_{НЗ}^{B} = \{r; w; u; R; y\},$$

$O_{НЗ}^{B} = \{O_1^*, O_2^*, ..., O_n^*\}$, де:

I^{B} – ідентифікатор користувача; kS – секретний ключ; y – відкритий ключ; α – велике псевдовипадкове число, що відповідає умовам необхідним для роботи методу; n – дуже велике просте число; r – певне псевдовипадкове значення; $W;U$ – арифметичні обрахунки, що використовують операції $O_{НЗ}$ за модулем n за участі kS та α; $w;u$ – результати виконання операцій $W;U$;
Ver – обрахунки, що використовують операції $O_{НЗ}^{B}$ за модулем n за участі u та w для підтвердження знання kS; O_i – арифметична операція; R – результат автентифікації.

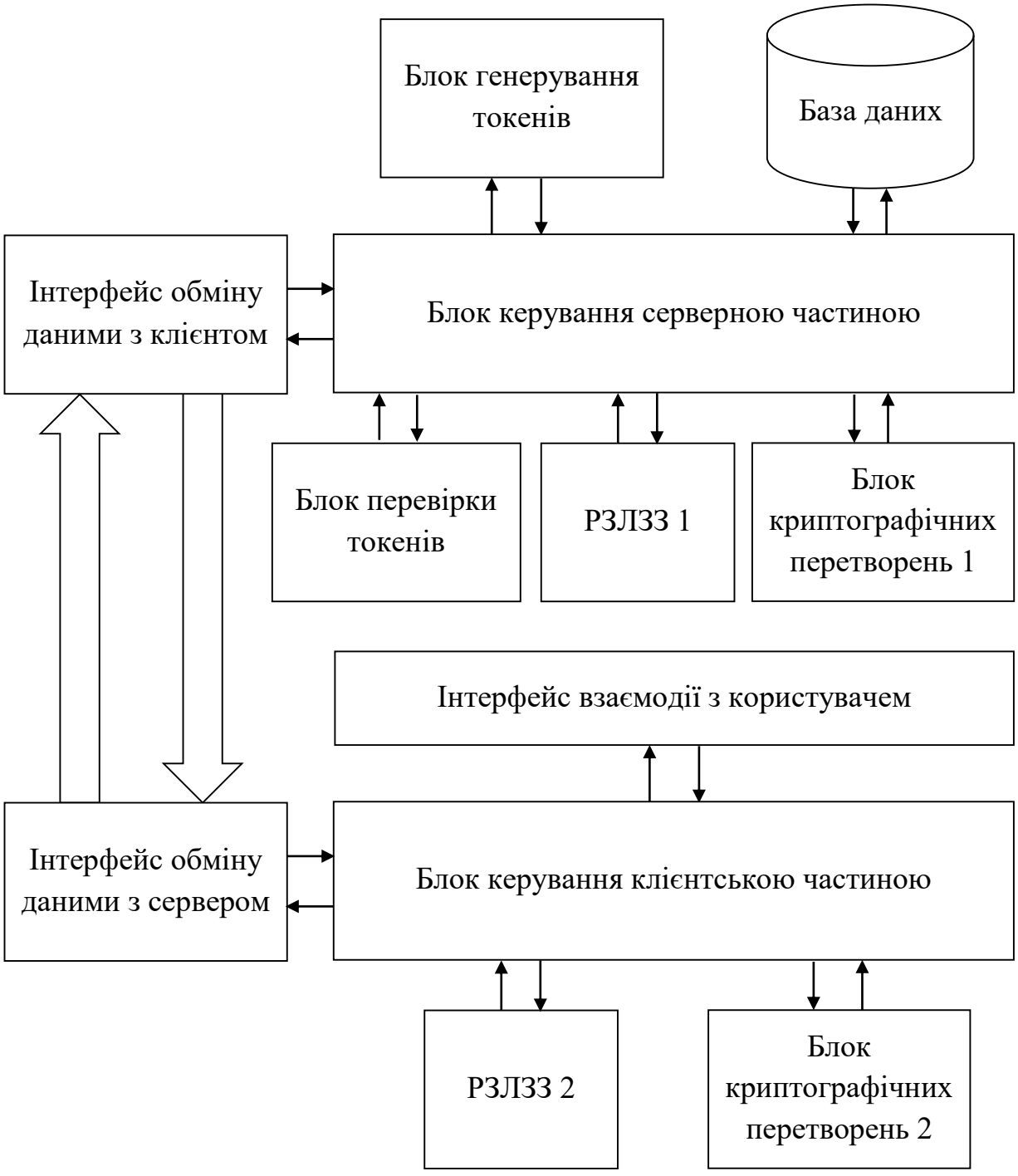
					08–20.МКР.011.00.000 І42			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб..		Селезньов В. І.			Метод та засів автентифікації користувачів із нульовим знанням. Математичний опис процесу автентифікації з нульовим знанням	Літ.	Арк.	Аркушів
Перевір.		Баришев Ю. В.					1	1
Реценз.		Войцеховська О. В.				ВНТУ, 1БС-20м		
Н. Контр.		Баришев Ю. В.						
Затверд.		Лужецький В.А.						

ПРОТОКОЛ АВТЕНТИФІКАЦІЇ З НУЛЬОВИМ ЗНАННЯМ



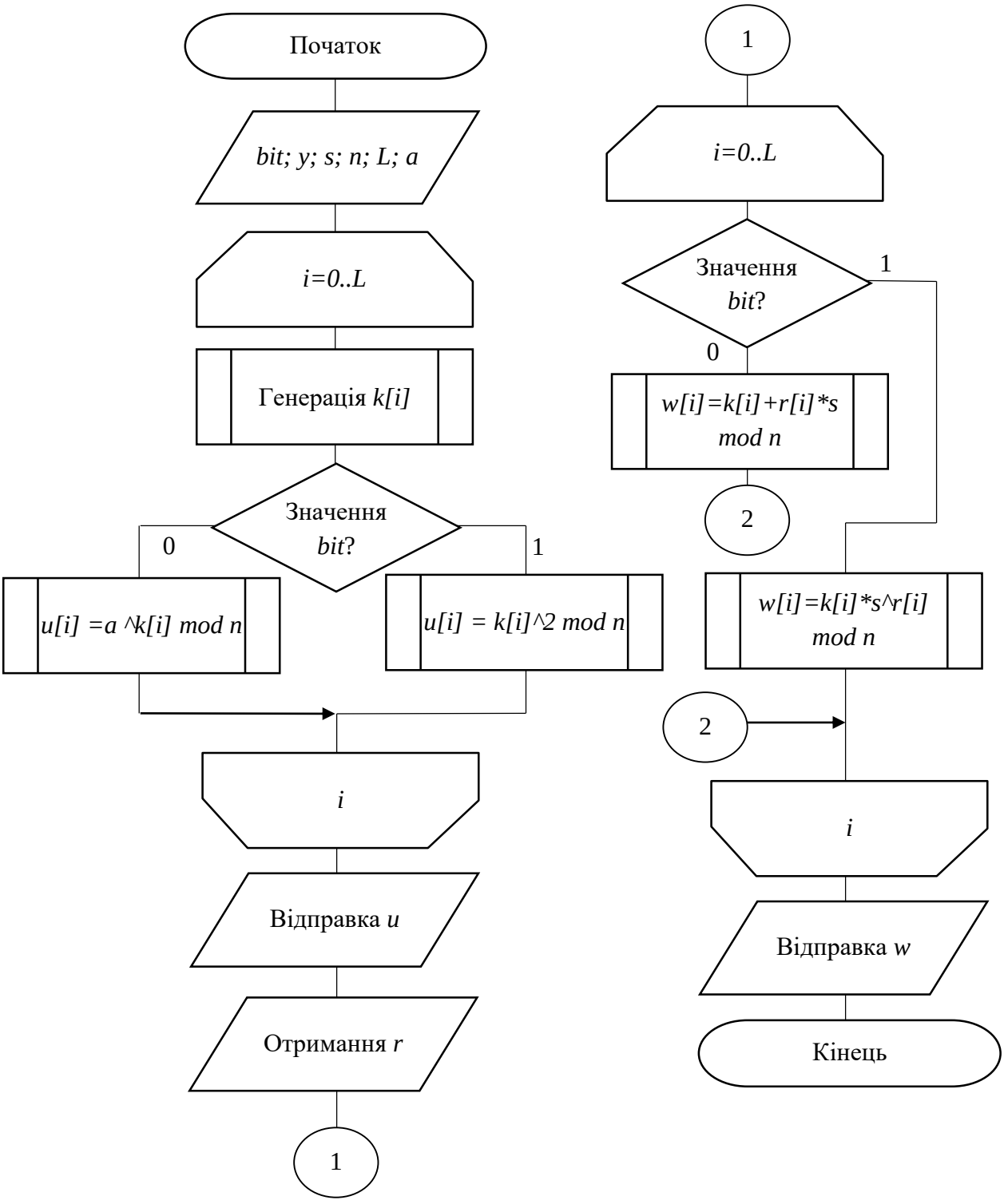
					08–20.МКР.011.00.000 ІЧЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб..		Селезньов В. І.			Метод та засів автентифікації користувачів із нульовим знанням. Протокол автентифікації з нульовим знанням	Літ.	Арк.	Аркушів
Перевір.		Баришев Ю. В.					1	1
Реценз.		Войцеховська О. В.				ВНТУ, 1БС-20м		
Н. Контр.		Баришев Ю. В.						
Затверд.		Лужецький В.А.						

СТРУКТУРА ЗАСОБУ АВТЕНТИФІКАЦІЇ З НУЛЬОВИМ ЗНАННЯМ



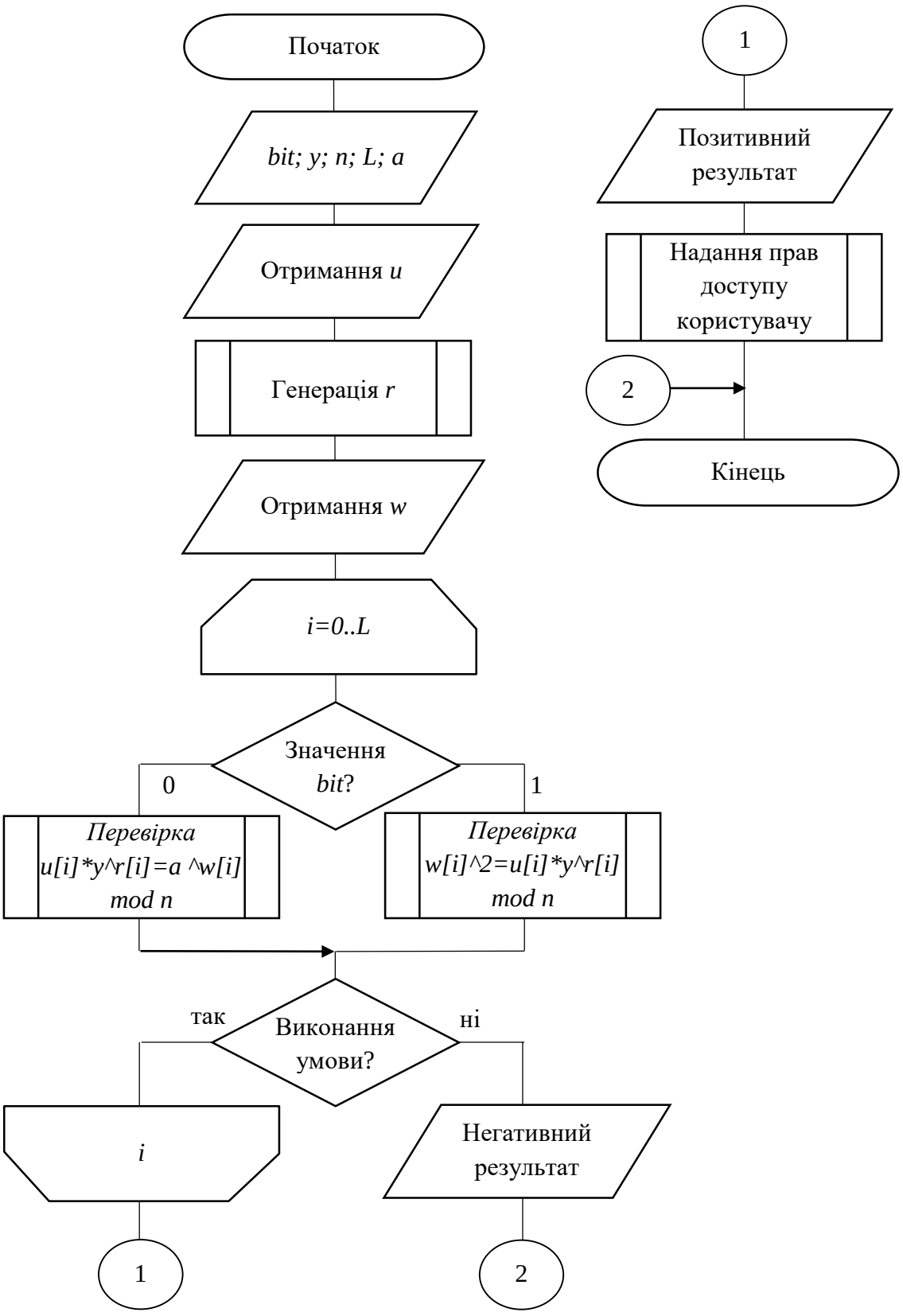
					08–20.МКР.011.00.000 І4		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб..		Селезньов В. І.			Метод та засів автентифікації користувачів із нульовим знанням. Структура засобу автентифікації з нульовим знанням	Літ.	Арк. 1 / Аркушів 1
Перевір.		Баришев Ю. В.					
Реценз.		Войцеховська О. В.				ВНТУ, 1БС-20м	
Н. Контр.		Баришев Ю. В.					
Затверд.		Лужецький В.А.					

АЛГОРИТМ РОБОТИ ЗАСОБУ ДЛЯ КЛІЄНТСЬКОЇ ЧАСТИНИ



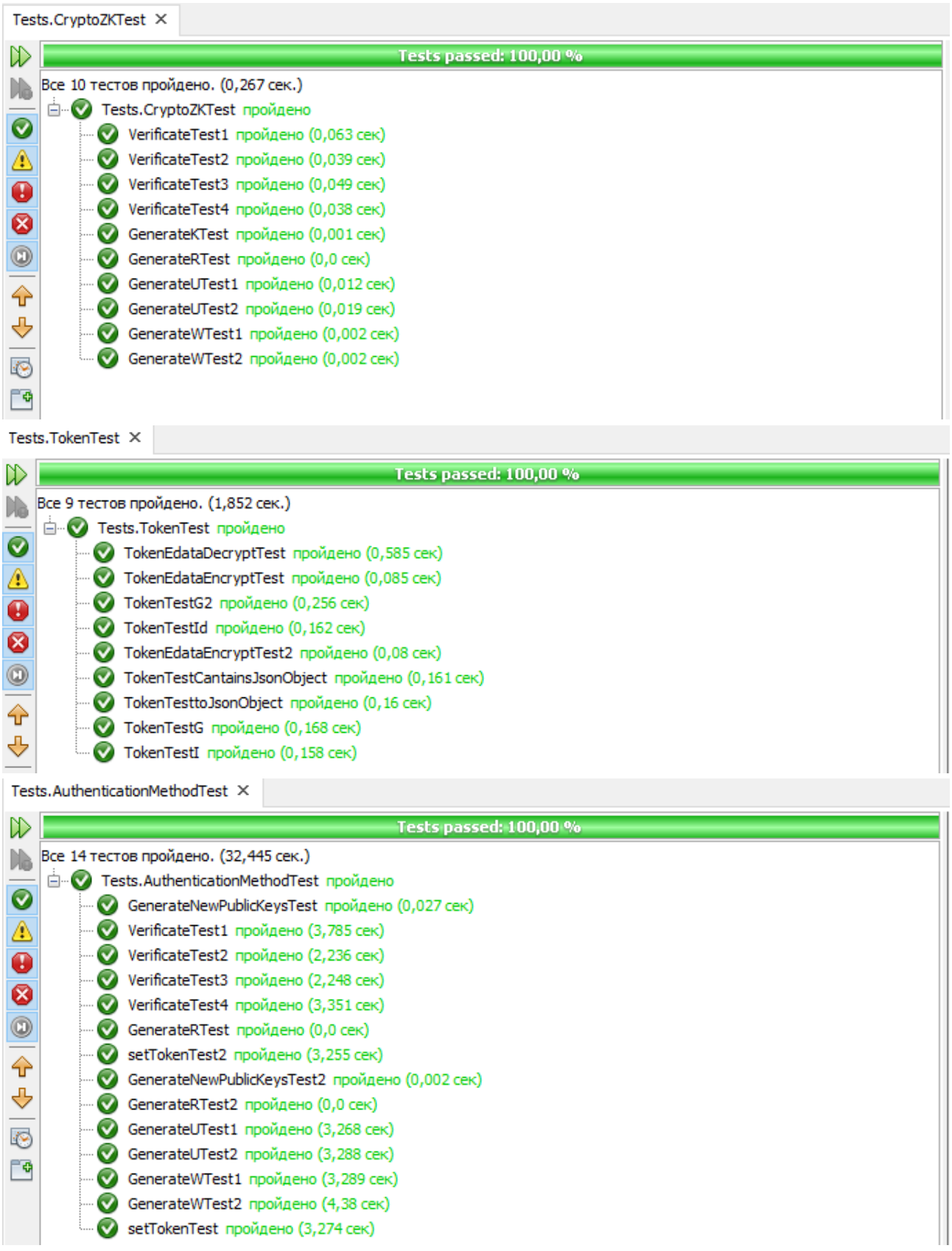
					08–20.МКР.011.00.000 І45		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб..		Селезньов В. І.			Метод та засів автентифікації користувачів із нульовим знанням. Алгоритм роботи засобу для клієнтської частини	Літ.	Арк. 1 / Аркушів 1
Перевір.		Баришев Ю. В.					
Реценз.		Войцеховська О. В.				ВНТУ, 1БС-20м	
Н. Контр.		Баришев Ю. В.					
Затверд.		Лужецький В.А.					

АЛГОРИТМ РОБОТИ ЗАСОБУ ДЛЯ СЕРВЕРНОЇ ЧАСТИНИ



					08–20.МКР.011.00.000 І46			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб..		Селезньов В. І.			Метод та засів автентифікації користувачів із нульовим знанням. Алгоритм роботи засобу для серверної частини	Літ.	Арк.	Аркушів
Перевір.		Баришев Ю. В.					1	1
Реценз.		Войцеховська О. В.				ВНТУ, 1БС-20м		
Н. Контр.		Баришев Ю. В.						
Затверд.		Лужецький В.А.						

РЕЗУЛЬТАТИ ТЕСТУВАННЯ



					08–20.МКР.011.00.000 І47		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб..		Селезньов В. І.			Метод та засів автентифікації користувачів із нульовим знанням. Результати тестування	Літ.	Арк. 1 / Аркушів 1
Перевір.		Баришев Ю. В.					
Реценз.		Войцеховська О. В.				ВНТУ, 1БС-20м	
Н. Контр.		Баришев Ю. В.					
Затверд.		Лужецький В.А.					