МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»

Освітньо-професійна програма: «Безпека комп'ютерних систем і мереж»

Група: 4КБ-02

Дипломний проект

здобувача освіти денної форми навчання
КБ.02.11.000.ДП

ЛИПОВОГО
ОЛЕКСАНДРА СЕРГІЙОВИЧА

м. Одеса
2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **123 «Комп'ютерна інженерія»**

Освітньо-професійна програма: «**Безпека комп'ютерних систем і мереж»**

Група: **4КБ-02**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту на тему:

Розробка програмної моделі генерування та валідації надійних паролів

Проектний матеріал складається з пояснювальної записки на 77 сторінках та графічного (презентаційного) матеріалу на 15 аркушах (слайдах)

Дипломник _____ (Липовий О.С.)

Керівник _____ (Скорняков В.С.)

Консультанти:

з економічного розділу _____ (Канський М.Ю.)

з розділу охорони праці та техніки безпеки _____ (Чорновол Н.І.)

з нормоконтролю _____ (Петрашова В.І.)

старший консультант _____ (Кривченко Ю.В.)

До захисту допущений

Голова циклової комісії _____ (Кривченко Ю.В.)

Завідувач відділення _____ (Краснокутська К.Г.)

Захист «20» червня 2025 р. Протокол ЕК № 1

Оцінка ЕК 4/добре/ / 805.

Секретар ЕК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення *комп'ютерних систем* Комісія *КТ та ПІ*

Спеціальність *123 «Комп'ютерна інженерія»*

Освітньо-професійна програма *«Безпека комп'ютерних систем і мереж»*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР *Беркань І.В.*

"19" 05 2025 р.

ЗАВДАННЯ

на дипломний проект

Здобувачеві освіти *Липового Олександра Сергійовича*

(прізвище, ім'я, по батькові)

1. Тема проекту *Розробка програмної моделі генерування та валідації надійних паролів*

затверджена наказом по коледжу від "14" 11 2024 р. № 246

2. Термін здачі закінченого проекту 16.06.25.

3. Вихідні данні до проекту *1. Передбачити довжину генерованих паролів від 5 до 50 символів; - генерація на основі слова або словосполучення користувача; 2. Передбачити режими розмиття генерованого паролю за допомогою застосування цифр, спеціальних символів та заглавних літер; 3. Передбачити використання надійного генератору випадкових чисел; 4. Передбачити хешування для валідації паролів; 5. Реалізувати окремі модулі для генерування та валідації паролів; 6. Розробку застосунку виконати мовою C#*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)

Аналітичний огляд засобів генерування надійних паролів; Аналіз криптографічних засобів захисту паролів; Складання математичної моделі генерування та валідації надійних паролів; Розробка структури та алгоритмів роботи програмного застосунку для генерування та валідації надійних паролів; Реалізація програмної моделі та інтерфейсу застосунку; Тестування програмної моделі генерування та валідації надійних паролів

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

Багатофакторна автентифікація і взаємодія різних методів захисту; Алгоритм генерації паролів із застосуванням трансформації символів та перевірки ентропії; Схема роботи алгоритму BBS; Графічне представлення обробки даних SHA-256; Структурна схема застосунку для генерування та валідації надійних паролів; БСА роботи системи генерації паролів; БСА валідації паролю; Скріншоти роботи модулів генерування та валідації паролів

6. Консультанти по проекту, із зазначенням розділів проекту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Основний розділ	*Скорняков В.С.*		
Економічний розділ	*Канський М.Ю.*		
Розділ охорони праці	*Чорновол Н.І.*		
Нормоконтроль	*Петрашова В.І.*		
Старший консультант	*Кривченко Ю.В.*		

7. Дата видачі завдання 15.05.24

Керівник *Скорняков В.С.* _____
(підпис)

Завдання прийняв до виконання *Липовий О.С.* _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проекту	Термін виконання етапів дипломного проекту (роботи)	Відмітка про виконання
1	*Вступ. Постановка задачі проектування*	15.05-25	
2	*Аналіз технічного завдання та пошук літератури*	17.05.25	
3	*Аналітичний огляд засобів генерування надійних паролів*	19.05.25	
4	*Аналіз криптографічних засобів захисту паролів*	20.05.25	
5	*Складання математичної моделі генерування паролів*	27.05.25	
	Розробка структури та алгоритмів роботи застосунку		
6	*Реалізація програмної моделі та інтерфейсу застосунку*	30.05.25	
7	*Розробка БСА та діаграми класів застосунку*	1.06.25.	
8	*Реалізація візуального інтерфейсу застосунку*	2.06.25	
9	*Написання коду програми мовою C#*	3.06.25	
10	*Тестування модулів генерування та валідації паролів*	10.06.25	
11	*Випробування застосунку та аналіз результатів*	12.06.25	
12	*Виконання економічних розрахунків*	13.06.25	
13	*Розробка питань з охорони праці та техніки безпеки*	14.06.25	
14	*Підготовка мультимедійної презентації проекту*	15.06.25	

Дипломник _____
(підпис)

Керівник _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			КБ 02. 11 000. 00 ДП	Дипломний проект		
А4			КБ 02. 11 000. 00 ДП ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	КБ 02. 11 000. 00 ДП			
Розробив		Липовий О.С.		16.06	Розробка програмної моделі генерування та валідації надійних паролів	Літ.	Аркуш	Аркушів
Перевірив		Скорняков В.С.		16.06		Н Д П	4	77
Н. контр.		Петрашова В.І.		16.06		ВСП "ОТФК ОНТУ" гр.4КБ-02		
Затверд.		Кривченко Ю.В.		16.06				

ЗМІСТ

Вступ……...……..6
1 Основний розділ…..…….…….....8
1.1 Аналітичний огляд засобів генерування надійних паролів.…….....….....8
1.1.1 Методи і засоби автентифікації користувача.....8
1.1.2 Методи генерації паролів.....11
1.1.3 Застосування генераторів випадкових чисел.....13
1.1.4 Застосування вихорю Мерсенна для генерації випадкових чисел....15
1.1.5 Застосування алгоритму BBS.....17
1.2 Аналіз криптографічних засобів захисту паролів …….....18
1.2.1 Застосування алгоритму AES.....19
1.2.2 Застосування хеш-функцій.....20
1.2.3 Застосування алгоритму хешування SHA-256.....22
1.2.4 Застосування алгоритму хешування SHA-512.....23
1.2.5 Аналіз криптографічної стійкості систем захисту паролів.....25
1.2.6 Запобігання криптоаналізу.....27
1.3 Складання математичної моделі генерування та валідації надійних паролів..... 29
1.3.1 Створення математичної моделі автентифікації користувачів.....29
1.3.2 Створення математичної моделі генерування надійних паролів.....31
1.3.3 Адаптація математичної моделі алгоритму SHA-256 для валідації паролів.....34
1.4 Розробка структури та алгоритмів роботи програмного застосунку для генерування та валідації надійних паролів.....36
1.4.1 Розробка БСА генерування паролів.....39
1.4.2 Розробка БСА валідації паролів.....42
1.5 Реалізація програмної моделі та інтерфейсу застосунку.....45

1.5.1 Розробка модуля генерування паролів.....45
1.5.2 Розробка модуля валідації паролів.....48
1.6 Тестування програмної моделі генерування та валідації надійних паролів.....51
1.6.1 Тестування модуля генерування паролів.....51
1.6.2 Модуль валідації паролів.....53
2 Економічний розділ……..…….……56
2.1 Резюме.....56
2.2 Визначення трудомісткості розробки програмного забезпечення.....56
2.3 Розрахунок ціни програмного продукту.....59
3 Розділ охорони праці та техніки безпеки…….……..……61
3.1 Аналіз небезпечних і шкідливих факторів, що впливають на користувача ПК.....61
3.2 Гігієнічні вимоги до виробничого середовища.....62
3.2.1 Вимоги до приміщення.....62
3.2.2 Освітлення.....62
3.2.3 Шум.....63
3.3 Вимоги до організації робочого місця працівника.....63
3.4 Мікроклімат.....64
3.5 Електробезпека.....64
3.6 Пожежна безпека.....65
Висновки……..……66
Перелік використаних інформаційних джерел…….……...….67
Додаток А. Фрагменти коду мовою C# модулів генерування та валідації паролів.....68
Додаток Б. Слайди мультимедійної презентації.....71

ВСТУП

У сучасних умовах кібербезпеки слабкий пароль стає однією з найсерйозніших вразливостей систем захисту інформації. Проблематика використання надійних паролів набуває особливої актуальності через зростання числа атак, що використовують методи перебору та словникові атаки. Отже, завдання дипломної роботи полягає в розробці програмної моделі, яка забезпечує генерацію та валідацію складних паролів із високою ентропією, що є незамінною умовою для запобігання несанкціонованому доступу.

Основний напрям роботи включає аналіз сучасних підходів до формування паролів, розробку математичної моделі для генерації випадкових комбінацій із застосуванням таблиць заміни символів та визначення критеріїв оцінки їхньої криптографічної надійності. Передбачається дослідити різні алгоритмічні рішення для підвищення випадковості вихідних даних, що дозволить визначити оптимальні параметри генерації, такі як мінімальна довжина, використання літер різного регістру, цифр та спеціальних символів. Ретельне моделювання процесу дозволить не тільки оцінити ентропію, а й забезпечити стійкість до спеціалізованих криптоаналітичних атак. Особлива увага буде приділена питанням інтеграції додаткових захисних механізмів, таких як використання "солі" для ускладнення атаки за допомогою попередньо розрахованих таблиць (rainbow tables). Планується застосування об'єктно-орієнтованого підходу до розробки програмної моделі з поділом функціональності на окремі компоненти, що відповідають за генерацію пароля, його перевірку на складність та подальше хешування. Для практичної реалізації рішення буде обрано мову програмування C# у середовищі Visual Studio із застосуванням стандартних бібліотек криптографії .NET. Виконання цієї роботи має на меті розробку програмної моделі, що об'єднує математичне моделювання генерації паролів, алгоритмічний підхід до підвищення їхньої ентропії та застосування сучасних криптографічних методів для хешування. Плануються напрями експериментальної перевірки, що дозволять не лише визначити оптимальні параметри для генерації надійних паролів, а й знайти ефективні засоби захисту, що є основою для створення більш безпечних інформаційних систем.

1 ОСНОВНИЙ РОЗДІЛ

1.1 Аналітичний огляд засобів генерування надійних паролів

Надійність паролів є критичним чинником забезпечення інформаційної безпеки сучасних систем. З огляду на постійне зростання кількості кібератак, що базуються на переборі, словникових атаках та використанні попередньо обчислених таблиць (rainbow tables), ефективний захист аутентифікації користувачів набуває надзвичайної актуальності. У цьому підрозділі проведено аналітичний огляд існуючих засобів та методів, які застосовуються для посилення безпеки паролів.

Огляд зосереджується на аналізі сучасних принципів формування паролів, вимог до їх складності та алгоритмів їхнього захисту. Розглядаються як традиційні засоби, наприклад, впровадження політик складності пароля та використання стандартних методів хешування, так і інноваційні підходи, що базуються на математичному моделюванні процесу генерації та змінах символів із застосуванням таблиць заміни. Особлива увага приділяється оцінці ефективності кожного з методів у протидії конкретним видам атак, що дозволяє виявити переваги та недоліки існуючих рішень. Далі буде проведено детальний аналіз окремих аспектів безпеки паролів, зокрема, методів генерації, критеріїв оцінки ентропії, алгоритмів хешування та застосування додаткових заходів захисту, таких як використання "солі". Аналіз спрямований на виявлення оптимальних шляхів підвищення стійкості паролів, що стане базою для розробки ефективних криптографічних механізмів у подальшій частині роботи.

1.1.1 Методи і засоби автентифікації користувача

Процес автентифікації користувача є ключовою ланкою забезпечення інформаційної безпеки, оскільки саме від нього залежить, чи зможе особа отримати доступ до захищених ресурсів системи. Автентифікація реалізується через перевірку унікальних характеристик, які можуть бути засновані на знаннях користувача, фізичних об’єктах, що знаходяться у його розпорядженні, або ж на його біометричних ознаках. Цей підхід дозволяє звести до мінімуму ризик несанкціонованого доступу, враховуючи, що кожен із методів має власні сильні та

слабкі сторони.

Принцип перевірки того, що користувач знає, найчастіше реалізується через введення секретної інформації, наприклад, пароля чи PIN-коду. Надійність такого методу безпосередньо залежить від складності секретного коду, адже низька ентропія або використання легковгадуваних комбінацій роблять систему вразливою до атак методом перебору та соціальної інженерії. З іншого боку, автентифікація, що базується на тому, що користувач має певний фізичний об’єкт (наприклад, смарт-карту або токен), забезпечує додатковий рівень захисту; оскільки навіть при розкритті секрету зловмиснику необхідно мати доступ до фізичного засобу, що знижує шанс компрометації даних. Однак і цей метод не позбавлений недоліків – втрата чи викрадення пристрою може призвести до небажаного доступу, тому застосування такого підходу вимагає впровадження супутніх засобів контролю, наприклад, блокування при численних неуспішних спробах аутентифікації.

За принципом ідентифікації користувача на основі його унікальних физичних чи поведінкових характеристик застосовують біометричні технології. Вони включають сканування відбитків пальців, розпізнавання обличчя, аналіз голосу, геометрію кисті та навіть особливості набору тексту. Ці технології забезпечують високий рівень точності, оскільки фізіологічні ознаки майже неможливо скопіювати або фальсифікувати. Проте впровадження біометричних рішень часто пов'язане з високими витратами на обладнання та обмеженнями, що випливають із можливих технічних збоїв або змін у зовнішньому вигляді користувача, що може вплинути на точність розпізнавання.

Сучасні тенденції у сфері автентифікації спрямовані на інтеграцію кількох методів для створення багатофакторних систем (MFA). Такі рішення поєднують два або більше способи підтвердження особистості: наприклад, пароль у поєднанні з фізичним пристроєм або біометричними даними. Подібний підхід дозволяє суттєво підвищити загальну безпеку системи, оскільки зловмиснику буде значно складніше одночасно обійти різні рівні захисту. До того ж, використання цифрових сертифікатів, що базуються на криптографії з відкритими і закритими ключами, стає популярним рішенням, оскільки воно забезпечує високий рівень довіри і

конфіденційності в електронному середовищі. Табл. 1.1 узагальнює ключові параметри для традиційної парольної автентифікації, фізичних засобів (смарт-карт, токенів), біометричних технологій, цифрових сертифікатів та систем одноразових паролів (OTP). У цій таблиці враховано такі аспекти, як надійність, вразливість до атак, складність впровадження та можливості інтеграції з існуючими системами.

Таблиця 1.1. Порівняльна характеристика методів автентифікації користувача

Метод автентифікації	**Основні характеристики**	**Переваги**	**Недоліки**
Парольна автентифікація	Використання знань користувача (паролі, PIN-коди). Надійність залежить від складності секретної інформації та її конфіденційності.	Простота впровадження та низька вартість.	Схильність до соціальної інженерії, атаки перебором, недостатня стійкість при використанні слабких паролів.
Фізичні засоби	Аутентифікація через наявність у користувача фізичних носіїв (смарт-карти, токени).	Додатковий фізичний бар'єр, що ускладнює несанкціонований доступ при компрометації секретної інформації.	Ризик втрати/крадіжки пристрою, необхідність апаратного забезпечення, можливість підміни при фізичному доступі до пристрою.
Біометричні дані	Використання унікальних фізичних або поведінкових ознак користувача (відбитки пальців, розпізнавання обличчя, голос).	Висока точність завдяки унікальності біометричних даних, неможливість точного копіювання.	Висока вартість обладнання, потенційні помилки розпізнавання, питання конфіденційності та зберігання особистих даних.
Цифрові сертифікати	Використання засобів криптографії з відкритими та закритими ключами для підтвердження автентичності користувача.	Забезпечення високої конфіденційності і цілісності даних, складність фальсифікації за допомогою криптографічних алгоритмів.	Складність управління сертифікатами, високі витрати на впровадження та обслуговування системи.
Одноразові паролі (OTP)	Генерація динамічних паролів, які використовуються лише один раз для кожного сеансу автентифікації.	Захищеність від повторного використання викрадених даних, підвищення безпеки за рахунок змінності пароля в реальному часі.	Необхідність синхронізації між пристроями користувача та системою, додаткові витрати на реалізацію алгоритмів генерації OTP.

Рис.1.1 демонструє загальну логіку роботи системи, де комбінація секретної інформації, фізичних носіїв та біометричних даних створює комплексний бар'єр для потенційного зловмисника.

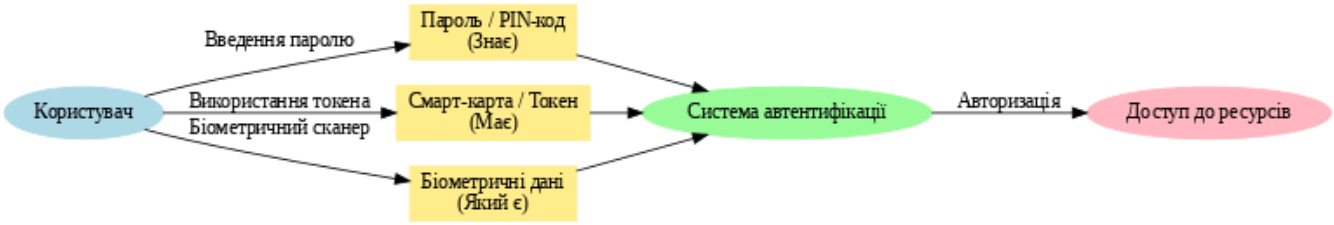


Рисунок 1.1. Багатофакторна автентифікація і взаємодія різних методів захисту

Сучасні засоби автентифікації базуються на використанні багатьох підходів, що забезпечують перевірку як знань, так і фізичних або біометричних характеристик користувача. Інтеграція різних методів автентифікації дозволяє зменшити слабкі ланки кожного окремого підходу, забезпечуючи більш високий рівень захисту систем.

1.1.2 Методи генерації паролів

Ефективний захист облікових записів безпосередньо залежить від якості використовуваних паролів. Надійність пароля формується за рахунок його складності, що визначається як комбінація різноманітних символів, змінності довжини та непередбачуваності структури. У сучасних інформаційних системах існують численні підходи до генерації паролів, які орієнтовані на досягнення високої ентропії та стійкості до атак зловмисників. Для побудови алгоритмів генерації використовують як прості випадкові методи, так і методи, засновані на математичному моделюванні процесу формування парольних комбінацій із застосуванням таблиць заміни символів та генераторів псевдовипадкових чисел.

Одним із основних критеріїв для створення надійного пароля є достатня довжина, зазвичай не менше 8–10 символів, а оптимально – ще більше, що дозволяє значно збільшити кількість можливих комбінацій. Крім того, пароль має містити символи різних груп: літери у верхньому і нижньому регістрах, цифри та спеціальні знаки. Ці вимоги забезпечують складність структури пароля, що унеможливлює його легке відгадування або підбір за допомогою стандартних словникових атак.

Сучасні методи генерації паролів включають як повністю випадкові

алгоритми, так і алгоритми, що інтегрують елементи асоціацій та трансформації. Метод повністю випадкового набору символів використовує криптографічно стійкий генератор псевдовипадкових чисел, який забезпечує високий рівень випадковості, що є важливим для запобігання атак типу brute force. Однак виключно випадковий пароль часто важко запам’ятати, що створює необхідність у розробці методів, що поєднують складність із зручністю для користувача. В цьому контексті доцільним є використання так званих методів асоціацій, при яких користувач може ввести запам’ятовуване ключове слово, що потім трансформується за допомогою математичних операцій і таблиць заміни символів у складну комбінацію, зберігаючи при цьому високий рівень криптографічної стійкості.

Окрім вищезгаданих підходів, існують методи, які ґрунтуються на трансформації базових текстових даних – наприклад, використання алгоритмів, що аналізують структуру або навіть геометрію символів, і заздалегідь визначених шаблонів для заміни окремих елементів. Такий алгоритмічний підхід дозволяє не лише додатково "підсилити" пароль за рахунок випадковості, а й забезпечити його унікальність для конкретного ресурсу, що є важливим для запобігання масштабної компрометації даних при зламі одного облікового запису. Для ілюстрації можна навести умовну схему (рис. 1.2), яка демонструє послідовність дій алгоритму генерації паролю: від початкового введення базових параметрів (довжини, набору символів, ключового слова) до застосування математичних трансформацій та замін символів із певною таблицею, що забезпечує вимірювану ентропію результуючого пароля. Така схема дає можливість визначити оптимальні кроки для впровадження алгоритму як у вигляді окремої служби генерації, так і як компонент інтегрованої системи захисту. Однак, незважаючи на численні переваги сучасних методів генерації паролів, важливо враховувати, що будь-який алгоритм може бути вразливим, якщо основні параметри (довжина, різноманітність символів) не будуть дотримані належним чином. Тому в розробці системи рекомендується поєднувати різні методики і проводити тестування створюваних паролів за критеріями ентропії та складності, а також впроваджувати механізми періодичного оновлення алгоритмів генерації відповідно до сучасних стандартів кібербезпеки.

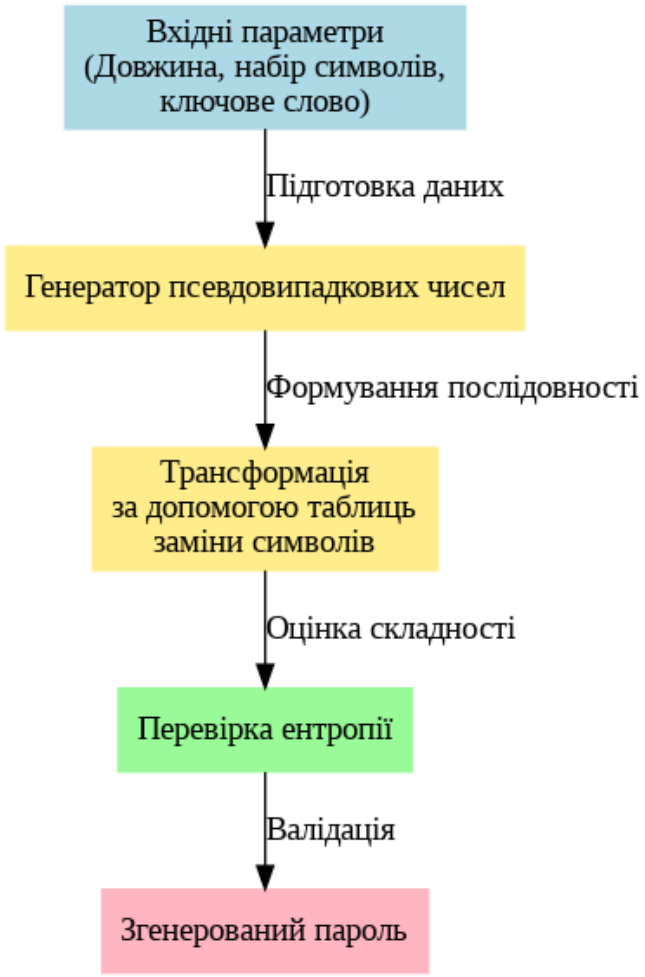


Рисунок 1.2. Алгоритм генерації паролів із застосуванням трансформації символів та перевірки ентропії

Таким чином, методи генерації паролів охоплюють як криптографічно стійкі випадкові алгоритми, так і методи трансформації та асоціацій, що дозволяють забезпечити високий рівень захисту при збереженні зручності для користувача. Ретельний аналіз можливих підходів і оптимізація параметрів алгоритмів є обов'язковою умовою для створення системи, здатної ефективно протистояти сучасним кіберзагрозам.

1.1.3 Застосування генераторів випадкових чисел

Генератори випадкових чисел є базовим компонентом у створенні систем безпеки, зокрема при формуванні паролів, які повинні мати високу непередбачуваність. Випадкова величина характеризується тим, що її значення не можна заздалегідь точно припустити, що забезпечує високу інформаційну ентропію (1.1).

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log P(x_i) \quad (1.1)$$

де X – випадкова величина; xi – можливі значення; pi – ймовірність події.

З математичної точки зору, випадковість може бути представлена як дискретна чи неперервна величина. Для дискретних змінних існує певний набір можливих значень, кожне з яких має свою ймовірність, сума яких дорівнює одному, а математичне очікування визначається через сумування добутків кожного значення на відповідну ймовірність (1.2).

$$E(X) = \sum_{i=1}^{\infty} p_i x_i \quad (1.2)$$

де X – випадкова величина; xi – можливі значення; pi – ймовірності події.

Натомість неперервні випадкові величини приймають усі можливі значення на заданому інтервалі, і їхній розподіл описується функцією густини, що дозволяє отримати ймовірності через інтегрування по інтервалам (1.3).

Для неперервної випадкової величини математичне сподівання описується за формулою (1.5).

$$E(X) = \int_{-\infty}^{\infty} x f(x)\, dx \quad (1.3)$$

де X – випадкова величина; $f(x)$ – густина розподілу.

У комп'ютерних системах справжня випадковість досягти неможливо, оскільки всі числові значення генеруються за допомогою алгоритмічних процесів. Тому більшість сучасних систем використовують псевдовипадкові генератори чисел, які, базуючись на початковому значенні (seed), створюють послідовність чисел, що має характерні ознаки випадковості, проте залишається детермінованою. Для завдань, що вимагають високого рівня безпеки, таких як генерація паролів, використовуються криптографічно стійкі генератори псевдовипадкових чисел (CSPRNG). Ці алгоритми розроблені таким чином, щоб їхні вихідні дані було надзвичайно складно передбачити навіть при знанні методів генерації, що значно підвищує стійкість системи до атак методом перебору.

Ключовим завданням при застосуванні генераторів випадкових чисел є забезпечення рівномірного розподілу випадкових значень, що гарантує однакову ймовірність виникнення будь-якого числа з заданого набору. Ідеальний випадковий генератор забезпечує, що кожне генероване число є незалежним від попередніх, що

є критично важливим для мінімізації ризиків компрометації паролів. На практиці, оскільки використовуються алгоритмічні методи, необхідно застосовувати спеціальні статистичні тести для перевірки якості отриманих послідовностей. Такі тести, як метод Монте-Карло або оцінка Колмогоровської складності, дозволяють визначити ступінь наближення до ідеальної випадковості та оцінити рівень ентропії вихідних даних (1.4).

$$K_f(s) = \min\{|p| : f(p) = s\} \quad (1.4)$$

де s – рядок з даними; f – машина Тюрінга.

У контексті розробки систем генерації паролів використання CSPRNG дозволяє не лише автоматизувати процес створення складних паролів, а й гарантувати, що результуючі ключові дані матимуть достатній рівень непередбачуваності для протидії сучасним методам криптоаналізу. Таким чином, впровадження генераторів випадкових чисел є фундаментальним чинником, що сприяє підвищенню стійкості систем автентифікації, забезпечуючи надійний захист від атак, здатних використати алгоритмічну передбачуваність традиційних методів генерації.

1.1.4 Застосування вихорю Мерсенна для генерації випадкових чисел

Вихор Мерсенна є одним із найпопулярніших алгоритмів для генерації псевдвипадкових чисел, що відзначається надзвичайно великим періодом повторення – $2^{19937}-1$. Завдяки такому періоду алгоритм забезпечує високий рівень непередбачуваності послідовностей, що генеруються, що критично важливо для криптографічних застосувань. Цей алгоритм здатен виробляти 32-бітні значення, кожне з яких формується незалежно від попередніх, що мінімізує кореляцію між послідовними елементами і сприяє рівномірному розподілу результатів у заданому інтервалі.

Формально, натуральне число Мерсенна (M_n) можна визначити за формулою

$$M_n = 2^n - 1, \quad (1.5)$$

де n – натуральне число.

Рис. 1.3 ілюструє спрощену схему роботи вихорю Мерсенна. Алгоритм базується на використанні внутрішнього регістру, складеного з 624 елементів, який

оновлюється за допомогою лінійного зворотного зв'язку і спеціального процесу «загартування». Ці перетворення забезпечують додаткове перемішування бітових векторів, що дозволяє наблизити генеровану послідовність до ідеального рівномірного розподілу.

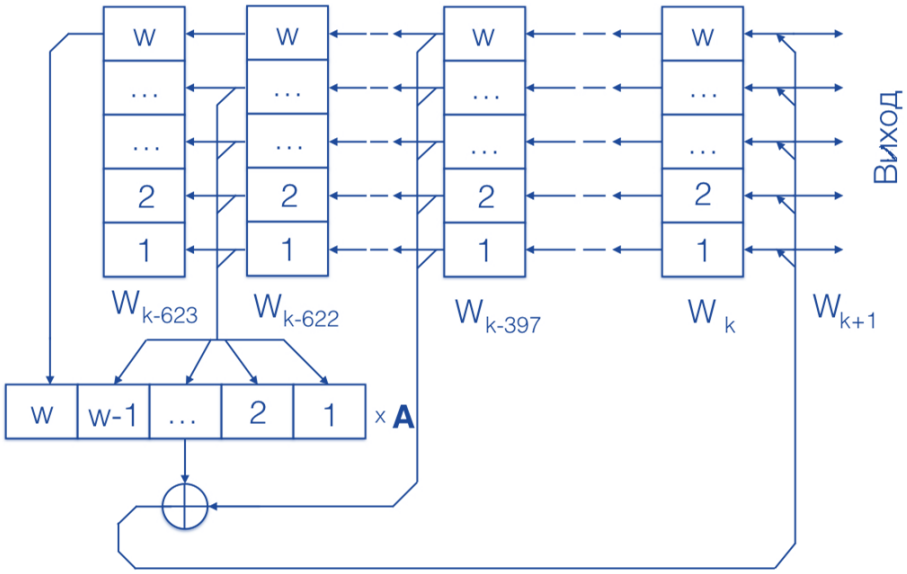


Рисунок 1.3. Схема вихорю Мерсенна

Процес ініціалізації вихорю Мерсенна починається з подачі початкового значення – seed, завдяки якому заповнюється увесь регістр. Кожні 624 вихідних числа здійснюється перемішування внутрішнього стану, що гарантує оновлення генератора і подовження його періоду. Найголовнішим аспектом є те, що, незважаючи на незалежність кожного згенерованого числа, існує потенційний ризик: якщо зловмиснику вдасться отримати 624 послідовних значення, він може відновити внутрішній стан алгоритму і, таким чином, точно прогнозувати подальші числа. Саме тому для застосування цього алгоритму у критично важливих криптографічних системах рекомендовано поєднувати його з додатковими алгоритмами хешування, що забезпечують додаткову криптостійкість.

У сучасній практиці вихор Мерсенна реалізовано за допомогою стандартних бібліотек, таких як у просторі імен std, де алгоритм mt19937 є класичним прикладом його застосування. Початкове значення для генератора формується за допомогою об'єкта seed_seq, який, у свою чергу, ініціалізується з використанням даних, отриманих з пристроїв, що забезпечують справжню випадковість (random_device). Цей підхід дозволяє задовольнити вимоги до високої ентропії для початкового

заповнення регістру і, відповідно, підвищує ефективність генерування випадкових послідовностей.

Отже, застосування вихорю Мерсенна дозволяє створювати псевдовипадкові послідовності з високою якістю випадковості, що є незамінним для формування складних паролів та інших криптографічних ключів. Незважаючи на потенційні вразливості при розкритті внутрішнього стану генератора, комплексне використання алгоритму разом із супутніми методами шифрування дозволяє забезпечити належний рівень захисту інформації в сучасних системах безпеки.

1.1.5 Застосування алгоритму BBS

Алгоритм BBS (Блум-Блум-Шуба) є класичним методом генерації псевдовипадкових чисел, який відзначається високою криптографічною стійкістю. У цьому алгоритмі основну роль відіграє факторизація двох великих простих чисел, що забезпечує подолання атаки методом відновлення внутрішнього стану генератора. На відміну від багатьох інших генераторів, таких як вихор Мерсенна, алгоритм BBS працює повільніше, але використовує математично обґрунтований підхід, який значно ускладнює спроби передбачення наступних чисел у послідовності.

Основна ідея алгоритму полягає у використанні наступного рекурентного співвідношення:

$$x_{n+1} = x_n^2 \, mod \, (p * q) \tag{1.6}$$

де p і q – великі прості числа, які підбираються таким чином, що їхній добуток створює стабільну основу для криптографічного захисту. Початкове значення x_0 (seed) вибирається так, щоб бути взаємно простим із M, що гарантує належне розподілення вихідних чисел. Після кожного обчислення x_{n+1} із отриманого числа вибирається певний біт (наприклад, найменш значущий або біт парності), який включається у фінальну послідовність псевдовипадкових чисел.

Схема роботи алгоритму BBS ілюструється на рис. 1.4. На першому етапі відбувається генерація двох великих простих чисел, які множаться для отримання модуля M. Далі, з використанням початкового значення, алгоритм ітеративно обчислює послідовність чисел за формулою $x_{n+1} = x_n^2 \bmod M$. Витягнення окремих

бітів із кожного числа забезпечує формування вихідного вектору, що характеризується високою непередбачуваністю та рівномірністю розподілу.

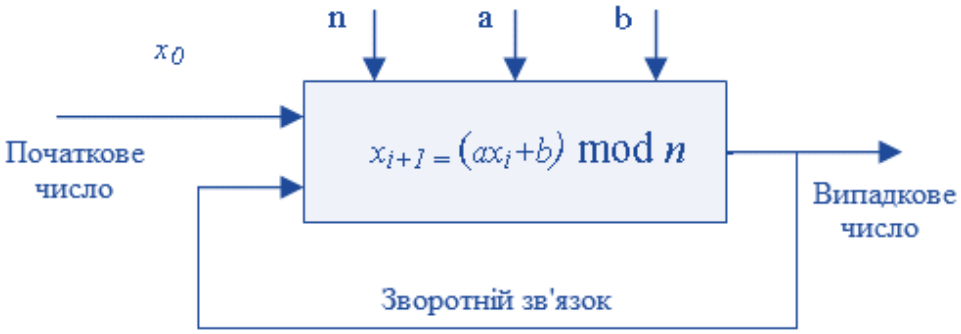


Рисунок 1.4. Схема роботи алгоритму BBS

Незважаючи на свою повільну продуктивність порівняно з іншими методами генерації випадкових чисел, алгоритм BBS виступає як ефективне рішення для застосувань, де критично важлива криптографічна стійкість. Його використання для генерації вектору ініціалізації особливо актуальне в тих системах, де повторне шифрування однакових даних має забезпечувати різні ключі, що значно підвищує загальний рівень захисту інформації.

Незважаючи на певні компроміси в швидкості роботи, застосування алгоритму BBS дозволяє отримувати послідовності випадкових чисел з високою ентропією, що є невід'ємною складовою сучасних криптографічних систем.

1.2 Аналіз криптографічних засобів захисту паролів

Захист паролів є критичним компонентом сучасних систем безпеки, адже від цього залежить збереження конфіденційності та цілісності даних користувачів. У цьому розділі розглядаються криптографічні засоби, що використовуються для захисту паролів, їх принципи функціонування, а також переваги та вразливості кожного з методів. Аналіз дозволяє окреслити основні підходи до формування хешів, застосування сольових значень та використання сучасних алгоритмів, що забезпечують високий рівень непередбачуваності даних.

Ключовою задачею криптографічного захисту паролів є перетворення вхідної інформації у незворотну форму, яка ефективно протидіє спробам вилучення початкових даних навіть при компрометації бази даних. Для цього використовують алгоритми, що забезпечують рівномірний розподіл вихідних значень і гарантують

високу ентропію. Окрім класичних методів хешування, таких як MD5 або SHA-1, сучасні рішення все частіше опираються на більш стійкі алгоритми, наприклад, SHA-256, bcrypt, Argon2 тощо. Ці методи поєднують ефективність обчислень із високою криптографічною стійкістю, що є необхідним для протидії як словниковим атакам, так і атакам методом перебору.

1.2.1 Застосування алгоритму AES

Алгоритм AES є одним із найпопулярніших симетричних блокових шифрів, який широко використовується в системах захисту інформації завдяки своїй високій стійкості до криптоаналізу та ефективності обчислень (рис.1.5). Використовуючи фіксовану довжину блоку даних та можливість застосування ключів різної довжини (128, 192 або 256 біт), AES забезпечує значний рівень захисту, що задовольняє сучасні вимоги до криптографічних стандартів.

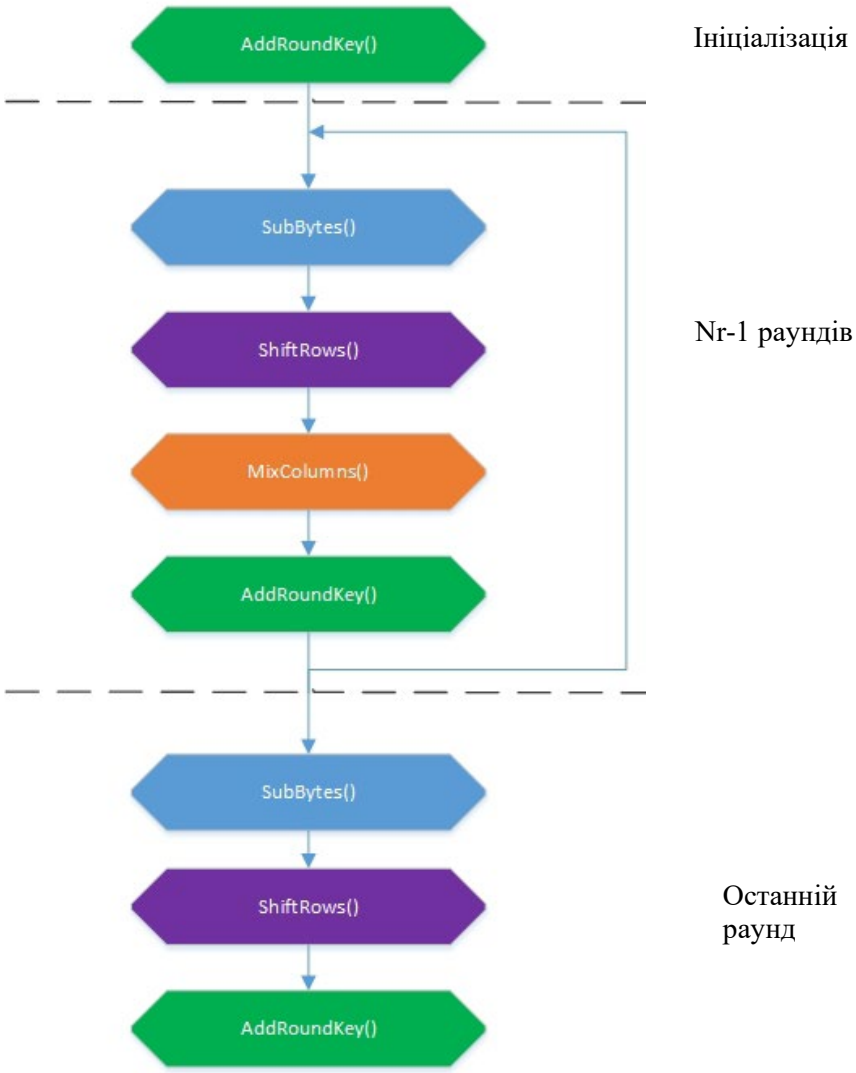


Рисунок 1.5. Реалізація алгоритму шифрування AES

У контексті захисту паролів алгоритм AES застосовується переважно для шифрування конфіденційних даних або для створення безпечного сховища, де інформація зберігається у зашифрованому вигляді. Незалежно від того, чи мова йде про шифрування сховища даних із паролями, або про їх додатковий захист у процесі автентифікації, AES демонструє високу продуктивність і надійність завдяки своїм математичним властивостям. Важливим елементом при застосуванні цього алгоритму є правильне управління ключами розшифрування, що гарантує, що доступ до даних матимуть лише авторизовані користувачі.

При впровадженні алгоритму AES особливу увагу приділяють вибору режиму роботи шифрування (таких як CBC, GCM, або інші), що впливають на безпечність обробки даних. Використання ініціалізаційних векторів високої ентропії дозволяє кожному блоку зашифрованих даних бути унікальним навіть при використанні одного й того ж ключа, що суттєво ускладнює спроби криптоаналізу. Крім того, при розробці комплексних систем захисту паролів часто комбінують AES із іншими криптографічними методами, наприклад, із застосуванням хеш-функцій та сольових значень, що створює багаторівневу схему безпеки.

Завдяки своїй універсальності, алгоритм AES інтегрується у численні програмні рішення та апаратні платформи, що підтверджено численними стандартами та сертифікаціями, виданими міжнародними організаціями. Його здатність до швидкого шифрування з мінімальним навантаженням на ресурси системи робить AES невід’ємним інструментом як у комерційних, так і в урядових інформаційних системах. Таким чином, застосування алгоритму AES у захисті паролів дозволяє досягти необхідного балансу між високою криптографічною стійкістю та ефективністю роботи системи, створюючи надійний фундамент для сучасних методів захисту конфіденційних даних.

1.2.2 Застосування хеш-функцій

Хеш-функції є невід'ємною частиною криптографічних систем, що забезпечують цілісність і конфіденційність даних, зокрема при захисті паролів. Вони представляють собою математичні алгоритми, які на вході приймають інформаційний потік будь-якого обсягу та перетворюють його у стислий майстер-

хеш фіксованої довжини. Незалежно від того, чи йдеться про великі файли чи невеликі текстові повідомлення, хеш-функція завжди повертає результат однакової довжини, що ілюструє принцип, зображений на рис. 1.6.

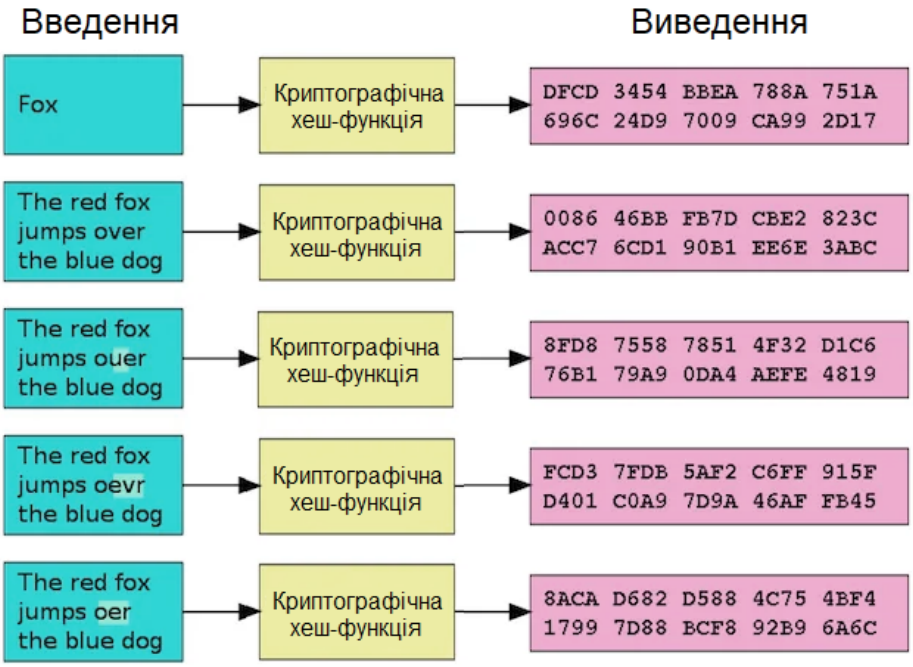


Рисунок 1.6. Принцип хешування за допомогою хеш-функцій

Швидкість обчислення хеш-функцій значно перевищує швидкість симетричних шифрувальних алгоритмів, тому вони активно використовуються для перевірки цілісності даних і верифікації автентичності інформації. Основною властивістю хеш-функції є її односторонність: перетворення вихідних даних у хеш відбувається за алгоритмічними правилами, але відновити початкове значення з отриманого хешу практично неможливо через втрату інформації, що виникає при стисканні повідомлення. Така необоротність гарантує, що навіть у випадку компрометації збережених хешів зловмиснику буде важко отримати конкретні значення паролів або інших критичних даних.

Застосування хеш-функцій в системах захисту паролів має свої переваги: з одного боку, зашифровані хеш-коди можуть зберігатися як безпечний «відбиток» оригінальних даних, а з іншого – вони слугують засобом перевірки цілісності інформації, коли будь-яка зміна у файлі або рядку даних призводить до корінної зміни хешу. Незважаючи на те, що хешування звільняє систему від потреби зберігати вихідне повідомлення, цей же факт може стати недоліком – первинні дані

неможливо відновити за наявного хешу, тому правильне управління і зберіганням таких даних залишається критичним завданням.

Широке застосування хеш-функцій обумовлено їх ефективністю і універсальністю, що робить їх незамінними для створення контрольних сум, електронних підписів та безпечного зберігання паролів. Проте, зловмисники можуть спробувати замінити оригінальний файл його хешованою копією, тому сучасні системи захисту використовують комбінації методів для забезпечення комплексного та надійного рівня безпеки. Таким чином, хеш-функції виступають ключовим інструментом для криптографічного захисту, забезпечуючи оперативність обробки даних і високий рівень стійкості до несанкціонованих змін.

1.2.3 Застосування алгоритму хешування SHA-256

Алгоритм SHA-256 (Secure Hash Algorithm 256-bit) є представником родини криптографічних хеш-функцій SHA-2, що широко використовується в системах захисту інформації завдяки своїй високій стійкості до криптоаналізу та здатності створювати односторонні хеш-коди фіксованої довжини. Згідно зі стандартом, алгоритм приймає вхідні дані довільної довжини і перетворює їх у 256-бітовий хеш, який представлений у вигляді 64-символьного шестнадцяткового рядка. Завдяки цьому навіть незначна зміна у вхідному повідомленні призводить до кардинально різного результату, що робить SHA-256 надзвичайно ефективним засобом для перевірки цілісності даних та зберігання паролів.

Внутрішня структура SHA-256 (рис.1.7) базується на послідовній обробці даних блоками фіксованої довжини, при цьому кожен блок проходить через серію раундів, де застосовуються операції зсуву, додавання, побітових логічних функцій та модульних арифметичних операцій. Такий підхід забезпечує значну дифузію вихідних даних, що гарантує стійкість алгоритму до колізійних атак, де два різні вхідні значення можуть давати однаковий хеш. Ця односторонність та високий рівень ентропії роблять SHA-256 ідеальним варіантом для завдань, де необхідно забезпечити незворотність перетворення інформації.

Застосування SHA-256 особливо актуальне у сферах, де критично важлива безпека збереження даних, таких як шифрування паролів, створення цифрових

підписів, перевірка контрольних сум файлів та багатьох інших протоколах захисту інформації, зокрема у криптовалютних системах. Наприклад, для зберігання паролів широко використовується комбінація Salt + SHA-256, що дозволяє унеможливити використання попередньо обчислених таблиць підбору (rainbow tables) та підвищує загальний рівень безпеки збережених даних.

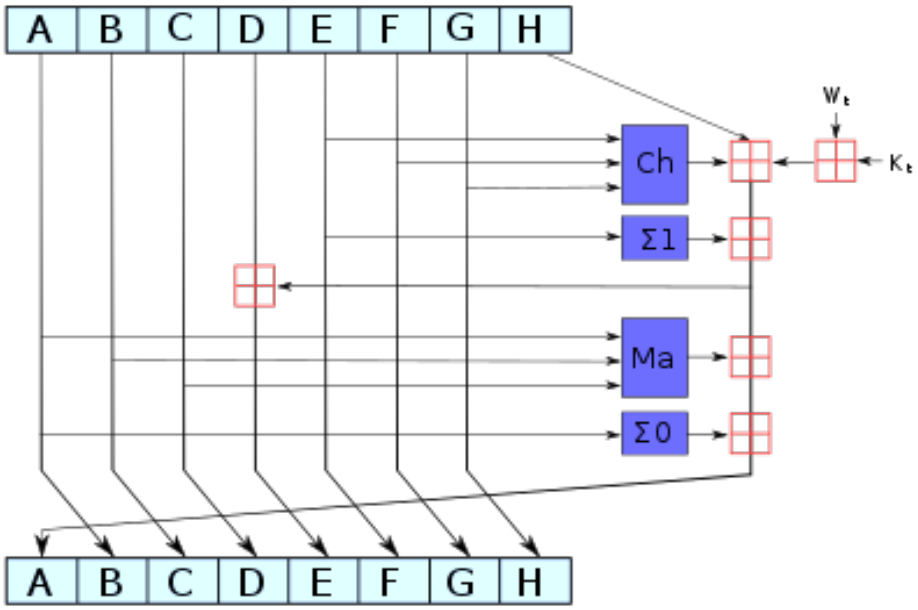


Рисунок 1.7. Графічне представлення одної ітерації обробки блоку даних

Крім того, застосування SHA-256 дозволяє забезпечити високошвидкісне обчислення хешу без значного навантаження на обчислювальні ресурси, що робить алгоритм придатним як для онлайн-систем, що потребують оперативної автентифікації, так і для офлайн-рішень. Завдяки своїй ефективності та підтвердженим стандартам, SHA-256 став одним із ключових інструментів у сучасних криптографічних засобах, сприяючи надійному захисту інформації у широкому спектрі застосувань. Використання алгоритму SHA-256 дозволяє досягти необхідного рівня криптографічного захисту завдяки поєднанню високої швидкості обчислення, односторонньості та стійкості до сучасних криптоатак, що робить його фундаментальним компонентом безпечних систем зберігання та верифікації інформації.

1.2.4 Застосування алгоритму хешування SHA-512

Алгоритм SHA-512 належить до сімейства SHA-2 і використовується для створення хеш-кодів з високим ступенем криптографічної стійкості. Він приймає

вхідні дані будь-якого розміру та обчислює 512-бітовий хеш, який зазвичай подається у вигляді 128-символьного шістнадцяткового рядка. Завдяки довжині вихідного повідомлення, SHA-512 забезпечує значно вищий рівень безпеки порівняно з алгоритмами з коротшими хешами, адже пошук колізій (тобто, знаходження двох різних вхідних даних, що дають однаковий хеш) стає значно складнішим.

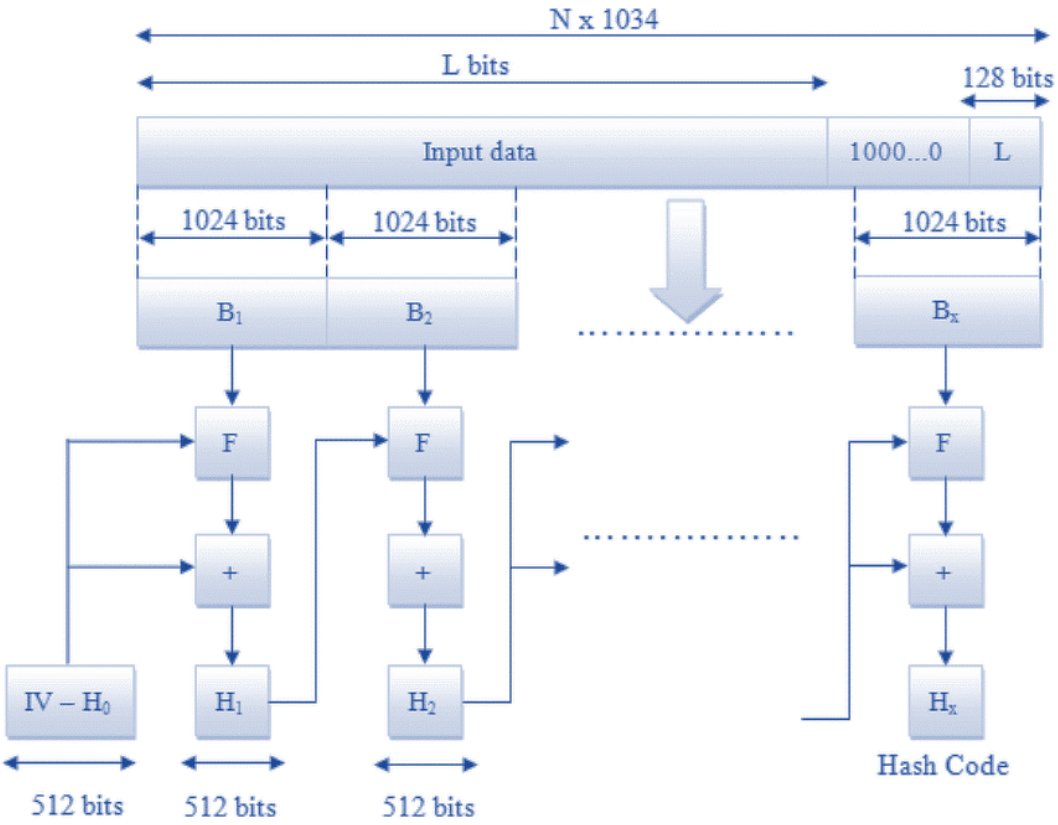


Рисунок 1.8. Структура алгоритму SHA-512

Внутрішня структура алгоритму (рис.1.8) базується на обробці даних блоками фіксованого розміру та включає велике число побітових операцій, таких як циклічні зсуви, логічні функції та модульне додавання. Ці операції забезпечують високу дифузію інформації, завдяки чому навіть незначна зміна у вхідному повідомленні призводить до кардинально відмінного хешу. Таке «ефект лавини» є ключовою вимогою для забезпечення односторонності хешування, що гарантує неможливість відновлення початкових даних на основі хешу.

Практичне застосування SHA-512 знаходить своє відображення в багатьох сферах криптографії: від перевірки цілісності файлів і електронних підписів до безпечного зберігання паролів. Поєднання алгоритму з додатковими механізмами,

наприклад, використанням солей (salt), дозволяє значно підвищити стійкість систем зберігання паролів до атак типу «rainbow tables». Це стає особливо актуальним у сучасних системах, де компрометація баз даних може призвести до масштабного витоку інформації.

Незважаючи на дещо більший обчислювальний ресурс для генерації хешу порівняно з менш потужними алгоритмами, переваги SHA-512 в плані криптографічної безпеки роблять його незамінним інструментом для випадків, коли необхідна максимальна захищеність даних. Тому застосування SHA-512 рекомендовано у тих середовищах, де питання безпеки має першорядне значення, а додаткові витрати на обчислення прийнятні заради отримання високостійкого криптографічного захисту.

Обидва алгоритми SHA-256 і SHA-512 мають свою нішу застосування: SHA-256 широко використовується в багатьох стандартних протоколах безпеки, включаючи цифрові підписи, перевірку цілісності файлів та зберігання паролів, тоді як SHA-512 часто вибирають, коли потрібно забезпечити додатковий рівень захисту за рахунок більшої довжини хешу. Вибір конкретного алгоритму зазвичай залежить від вимог до безпеки системи і характеристик апаратних ресурсів, які використовуються у конкретному застосуванні. Таким чином, хоча SHA-512 забезпечує вищу криптографічну стійкість завдяки збільшеному розміру вихідного хешу, SHA-256 залишається популярним завдяки ефективності та меншому навантаженню на ресурси. Обидва алгоритми, як невід'ємні представники сімейства SHA-2, відіграють ключову роль у забезпеченні безпеки даних, і їх вибір визначається компромісом між рівнем захисту та продуктивністю конкретної системи.

1.2.5 Аналіз криптографічної стійкості систем захисту паролів

Криптографічна стійкість систем захисту паролів визначається складністю та витратами часу і ресурсів, необхідних для відновлення початкового (відкритого) тексту з його зашифрованої або хешованої форми. Сучасні криптографічні стандарти, наприклад, із використанням 256-бітових ключів, вважаються достатньо безпечними, однак безперервне зростання обчислювальних потужностей вимагає

регулярного перегляду і вдосконалення алгоритмів забезпечення захисту.

Основні вимоги до ключових елементів криптосистем, що гарантують їхню стійкість, включають:

- Випадковість ключа: Ключі повинні генеруватися із використанням високоякісних генераторів випадкових чисел, що забезпечує їхню непередбачуваність;
- Конфіденційність передачі: Передача ключових даних здійснюється через захищені канали, що мінімізує ризик їх компрометації;
- Відповідність розміру: Розмір ключа повинен бути не меншим за довжину повідомлення, яке шифрується;
- Одноразовість використання: Принцип одноразового застосування ключа робить неможливим повторне використання одного і того ж значення, що запобігає ряду криптографічних атак;
- Відсутність інформації про ключ у вихідному повідомленні: Відкритий текст не має містити даних, які могли б дати змогу відновити інформацію про ключ.

Навіть при потенційно необмежених обчислювальних ресурсах зловмисників дотримання цих принципів суттєво підвищує захищеність систем. Одним із прикладів абсолютної криптографічної стійкості є шифр одноразового блокноту (шифр Вернама), який, за умови використання випадкового ключа, такого ж розміру, як і повідомлення, гарантує абсолютну незламність за умови коректної експлуатації:

$$k = m = c \in \{0,1\}^n \qquad (1.7)$$

Проте практичне впровадження цього методу стикається з проблемами генерації, зберігання та безпечної передачі ключів. Багаторазове використання одного й того ж ключа також створює вразливості. Наприклад, якщо застосовувати той самий ключ для двох різних повідомлень, то операція виключного або (XOR) дозволяє отримати:

$$(m_1 \oplus k) \oplus (m_2 \oplus k) = m_1 \oplus m_2 \qquad (1.8)$$

де m_1 і m_2 – відкриті тексти, k – один і той самий ключ. Це призводить до того, що при перехопленні зашифрованих даних зловмисник може отримати інформацію

про відносини між повідомленнями.

Принцип Керкгоффса, сформульований ще в ранніх працях з військової криптографії, визначає, що стійкість шифру не повинна залежати від секретності алгоритму, а лише від секретності ключа. Відкритість коду алгоритмів дозволяє криптоаналітикам вчасно виявляти потенційні вразливості та вдосконалювати методи захисту. Аналіз криптографічної стійкості систем захисту паролів підкреслює, що навіть найсучасніші алгоритми можуть бути зламані, якщо не забезпечити належне формування і обробку ключів. Висока стійкість системи формується завдяки комплексному підходу: використанню випадкових ключів, дотриманню принципів одноразового застосування, захищеній передачі даних та постійним оновленням алгоритмів відповідно до сучасних стандартів безпеки.

1.2.6 Запобігання криптоаналізу

Запобігання криптоаналізу є одним із ключових завдань у розробці криптографічних систем, спрямованих на захист паролів та іншої конфіденційної інформації. Основною метою є зробити процес відновлення початкового тексту настільки складним, що навіть за наявності значних обчислювальних ресурсів зловмисника це практично неможливо здійснити протягом прийнятного проміжку часу. Одним із фундаментальних принципів у протидії криптоаналізу є відсутність залежності безпеки системи від секретності алгоритму. Згідно з принципом Керкгоффа, безпека повинна ґрунтуватися виключно на секретності ключа, а не на приховуванні конструкції алгоритму. Це дозволяє експертам із криптографії та криптоаналізу спільно аналізувати алгоритм, виявляти можливі вразливості та своєчасно вдосконалювати систему.

Ключовими заходами запобігання криптоаналізу є:

- Випадковість і якість ключів: Використання високоякісних генераторів випадкових чисел для створення ключів гарантує їхню непередбачуваність. Чим більше ентропії міститься в ключі, тим важче зловмиснику здійснити успішну атаку методом перебору чи інших способів аналізу;
- Застосування сольових значень: Додавання унікальних сольових значень до паролів перед хешуванням значно ускладнює побудову словникових атак і

попередньої обчислення ймовірних відповідностей (rainbow tables). Такий підхід дозволяє навіть у разі використання однакових паролів отримувати різні хеш-коди, що запобігає їхньому масовому підбору;

- Контроль послідовності обчислень: Алгоритмічні методи, що реалізують принцип ефекту лавини, гарантують, що невелика зміна у вхідних даних призводить до кардинально різного результату. Це значно ускладнює аналіз структури шифрованого повідомлення і робить атакувальнику набагато важче встановити зв'язок між вхідним даним та його шифротекстом;
- Реалізація та апаратна безпека: Крім математичних властивостей алгоритмів, величезну роль відіграє правильне їхнє впровадження. Системи повинні бути оптимізовані так, щоб уникнути витоків інформації через побічні канали, такі як витік електромагнітного випромінювання або аналіз часу виконання операцій (side-channel attacks). Використання захищених програмних середовищ і апаратних модулів, таких як TPM (Trusted Platform Module), додатково знижує ризики експлуатації таких вразливостей;
- Регулярне оновлення і аудит: Зважаючи на постійне зростання обчислювальних потужностей і появу нових методів криптоаналізу, необхідно періодично проводити аудит системи та оновлювати алгоритми й протоколи, що використовуються. Публічний розгляд алгоритмів у спільноті експертів сприяє своєчасному виявленню вразливостей та їх оперативному усуненню.

Для практичного запобігання криптоаналізу важливо застосовувати комплексний підхід, що поєднує математичні аспекти криптографічного захисту з технічними заходами безпеки. Наприклад, використання комбінацій симетричних алгоритмів шифрування (наприклад, AES), хеш-функцій (SHA-256, SHA-512) і алгоритмів управління ключами разом із заходами фізичної безпеки дає змогу створити систему, стійку до широкого спектру атак. Таким чином, запобігання криптоаналізу вимагає не лише розробки надійних математичних алгоритмів, але й уваги до деталей реалізації, управління ключами та дотримання принципів відкритості і постійного вдосконалення.

1.3 Складання математичної моделі генерування та валідації надійних паролів

Математична модель, яка розглядається в цьому розділі, має враховувати низку ключових параметрів, що впливають на безпеку пароля. Серед них — довжина пароля, різноманітність використаних символів (великі та малі літери, цифри, спеціальні символи), а також методи генерації, які забезпечують високий рівень випадковості. Для цього використовуються алгоритмічні підходи, базовані на генераторах псевдовипадкових чисел, трансформаціях із застосуванням таблиць заміни, а також асоціативних методах, що сприяють збільшенню ентропії. Математичний аналіз цих підходів дозволяє побудувати залежності між обраними параметрами та стійкістю згенерованих паролів до криптографічних атак.

Особлива увага приділяється як теоретичним, так і практичним аспектам моделювання. З одного боку, модель включає аналіз розподілу ймовірностей, який допомагає формалізувати процес генерації випадкових послідовностей, а з іншого — враховує оптимізацію алгоритмів для забезпечення високої продуктивності та зручності використання. Такий комплексний підхід дозволяє визначити оптимальні умови для генерування паролів, що відповідають вимогам сучасних стандартів безпеки, і розробити ефективні методи їхньої валідації.

1.3.1 Створення математичної моделі автентифікації користувачів

Для систематизації методів автентифікації та розробки нових підходів до її вдосконалення доцільно використати теоретико-множинний підхід, який дозволяє представити процес автентифікації як композицію окремих складових елементів. Така модель допомагає чітко визначити, з яких компонентів складається система, як ці компоненти взаємодіють між собою та які результати можливі в результаті цієї взаємодії.

У загальному вигляді систему автентифікації можна описати наступною трійкою:

$$\text{UserAuthentication} = \{\text{AuthenticationFactors}, \text{AuthenticationResults}, \text{retrieveFactor}(\cdot), \text{verifyFactor}(\cdot)\} \quad (1.9)$$

Тут AuthenticationFactors являють собою множину даних, які вводяться для підтвердження особистості користувача. Згідно з попереднім аналізом, ці дані можуть базуватись на знаннях, що належать користувачу (наприклад, паролі або PIN-коди), біометричних характеристиках або ж апаратних засобах, що є у розпорядженні людини. Формально, це можна виразити як:

$$\text{AuthenticationFactors} = \{\text{Knowledge}, \text{Biomentric Feature}, \text{Device}\} \quad (1.10)$$

Невід'ємною частиною моделі є також AuthenticationResults — множина можливих наслідків автентифікації. Зазвичай, система повертає лише два статуси: успішну автентифікацію, яка позначається як true, або невдалу спробу, що позначається як false:

$$\text{AuthenticationResults} = \{\text{true}, \text{false}\} \quad (1.11)$$

Додаткові функції, такі як retrieveFactor(·) та verifyFactor(·), виконують ключову роль у моделюванні, забезпечуючи отримання необхідних автентифікаційних даних із заданих факторів та їх перевірку відповідно. Ці функції інтегрують всі складові системи в єдиний процес, що дозволяє формалізувати процедуру автентифікації.

Для ілюстрації, розглянемо конкретний приклад — парольну автентифікацію. У цьому випадку множина факторів складається з паролів, що є теоретично нескінченною комбінацією символів з визначеного алфавіту $\mathcal{A}$ (наприклад, набір символів, доступних на стандартній клавіатурі). Таким чином, модель для парольної автентифікації може бути представлена наступним чином:

$$\text{PasswordAuthentication} = \{\text{Passwords}, \{\text{true}, \text{false}\}, \text{passwordGeneration}(\cdot), \text{verifyPassword}(\cdot)\} \quad (1.12)$$

Оскільки властивості пароля визначають не лише його стійкість до злому (breakingComplexity), але й зручність користування (rememberComplexity), слід враховувати компроміс між цими параметрами. З одного боку, більш складний пароль, який важче зламати, може бути складним для запам'ятовування, а з іншого — занадто короткий ключ не забезпечить необхідного рівня безпеки. Цю співвідношення можна виразити умовно як:

$$\begin{aligned} &\text{rememberComplexity} \to \min; \\ &\text{breakingComplexity} \to \max. \end{aligned} \quad (1.13)$$

При цьому існують дві можливі постановки задачі: або максимізувати стійкість пароля при фіксованій складності запам'ятовування, або мінімізувати складність запам'ятовування, гарантувавши певний мінімальний рівень захищеності. Для практичного застосування більш доцільно обирати підхід, який встановлює нижню межу стійкості, що дозволяє спростити завдання користувача у запам'ятовуванні пароля, але одночасно гарантує мінімально необхідний рівень захисту:

$$\begin{aligned} &\text{rememberComplexity} \le \text{const}; \\ &\text{breakingComplexity} \to \max. \end{aligned} \quad (1.14)$$

Розробка математичної моделі процесу автентифікації дозволяє більш чітко визначити взаємозв'язок між факторами, що використовуються для авторизації користувача, та результатами цього процесу. Такий формалізм сприяє розробці алгоритмів для генерації надійних паролів, де оптимізується компроміс між криптографічною стійкістю та зручністю використання. Модель є основою для подальшого аналізу і впровадження інноваційних рішень у сфері захисту інформації, що забезпечує високий рівень безпеки сучасних автентифікаційних систем.

1.3.2 Створення математичної моделі генерування надійних паролів

У цьому підрозділі пропонується математична модель генерації надійних паролів, яка поєднує криптографічно стійкий генератор випадкових чисел на базі алгоритму BBS (Блум-Блум-Шуба) із можливістю налаштування параметрів пароля. Метою моделі є забезпечення високої ентропії згенерованих паролів за рахунок використання непередбачуваної послідовності чисел, а також надання користувачу вибору щодо довжини пароля та включення додаткових типів символів.

Нехай параметром довжини пароля є L, де L належить до цілочисельного інтервалу від 5 до 50: $L \in 5,6,\dots,50$. Також визначимо базову множину символів $\mathcal{A}\text{bs}$, що містить нижній регістр латинських літер, наприклад: $\mathcal{A}\text{bs} = \text{a}, \text{b}, \text{c}, \dots, \text{z}$. Додатково користувач може надати параметри, що визначають включення інших

категорій символів:

- $\delta_d \in 0,1$ – прапорець для включення цифр $\mathcal{A}_{\text{dgt}=0,1,\ldots,9}$;
- $\delta_u \in 0,1$– прапорець для включення великих літер $\mathcal{A}_{\text{upr}=A,B,\ldots,Z}$;
- $\delta_s \in 0,1$ – прапорець для включення спеціальних символів $\mathcal{A}_{\text{se}}$, наприклад, !, @, #, , %,' &,∗.

Таким чином, кінцева множина допустимих символів, з якої формується пароль, визначається як: $\mathcal{A} = \mathcal{A}\text{bs} \cup \delta_{\text{d}} \cdot \mathcal{A}dgt \cup \delta_u \cdot \mathcal{A}\text{upr} \cup \delta_{\text{s}} \cdot \mathcal{A}\text{se}$.

Генерацію пароля можна формалізувати як функцію $f: 5,6, \ldots ,50 \times 0,1^3 \times \Omega \rightarrow \mathcal{A}^{\mathcal{L}}$, де Ω — множина можливих початкових значень seed, а $\mathcal{A}^{\mathcal{L}}$ — множина усіх можливих рядків довжини L, що складаються з символів множини $\mathcal{A}$.

Функція $f(L, \delta_d, \delta_u, \delta_s, \text{seed}) = p$ генерує пароль p шляхом випадкового вибору L символів із множини $\mathcal{A}$. Для отримання справді випадкової послідовності використовується криптографічно стійкий генератор випадкових чисел, реалізований на основі алгоритму BBS. Алгоритм BBS забезпечує генерацію чисел із високою якістю випадковості, де кожне наступне число утворюється за допомогою операції зведення в квадрат за модулем добутку двох великих простих чисел: $x_{n+1} = x_n^2 \backslash mod M, \quad M = p \cdot q,$ при чому M обирається таким чином, що гарантує тривалий період генерації і мінімізує можливість передбачення послідовності.

Процес генерації надійного пароля складається з наступних етапів:

1. Налаштування параметрів: Користувач задає бажану довжину пароля $L \in [5,50]$ та визначає, які додаткові характеристики пароля мають бути забезпечені (включення цифр, великих літер, спеціальних символів);
2. Формування алфавіту: На основі встановлених прапорців формується допустима множина символів $\mathcal{A}$;
3. Ініціалізація генератора: Використовується криптографічно стійкий генератор випадкових чисел BBS, ініціалізований початковим значенням seed;
4. Генерація пароля: За допомогою BBS генерується послідовність випадкових чисел. Кожне число інтерпретується як індекс символу з множини $\mathcal{A}$ і використовується для послідовного формування пароля p довжиною L.

Для кожного запропонованого методу вводиться параметр — ймовірність заміни відповідного символу або вставлення додаткового знаку. Процес розмивання організовано наступним чином:

1. Заміна символів на спеціальні: Користувач задає ймовірність заміни кожної букви телефону на спеціальний символ. Для кожної літери генерується псевдовипадкове число za допомогою криптографічно стійкого генератора (BBS). Якщо згенероване значення потрапляє в заданий інтервал, літера замінюється на один із символів, що були обрані як можливий варіант. Варіанти заміни визначаються згідно з попереднім опитуванням користувачів, і для кожного набору символів формується свій проміжок частоти. Нижче наведено приклад табличного подання варіантів заміни (табл.1.2).

Таблиця 1.2. Приклад подання варіантів заміни символів

Буква	**Можливі варіанти заміни**
A	@, 4, (L, /)
B	8, I3, 3
C	<, (
H	#, /-/,]-[, }{
I	1, !
K	1<,]<, <
M	/v, //\
O	0, (), []
P	*, o, /*, /o
Q	() , (), 0
S	5, $
W	\/\/, vv, \x/, uu
X	><, }{,][,)(

2. Зміна регістра: Для кожної літери, що є символом алфавіту, використовується аналогічний алгоритм із встановленням вхідного параметра — ймовірності зміни регістра. Якщо генероване випадкове число відповідає встановленій умові, символ перетворюється з верхнього в нижній або навпаки;
3. Вставлення цифр: Метод полягає у випадковому додаванні цифри посередині пароля. На кожний можливий розділ визначається інтервал, і за умови, що згенероване число потрапляє у цей інтервал, у відповідну позицію додається

один із цифрових символів (від 0 до 9).

Оскільки ключовим елементом у цій моделі є якісний генератор випадкових чисел, для розподілу випадкових значень використовується не тільки алгоритм BBS, що забезпечує високу криптографічну стійкість, а й удосконалений лінійний конгруентний генератор (як описано в попередніх підрозділах). Завдяки цьому поєднанню отримано рівномірне розподілення числових значень у довільних діапазонах, що надалі використовується для визначення режимів модифікації.

На виході системи генеруються варіанти "розмитих" ключових слів, кожен з яких проходить наступну перевірку:

- Перевірка довжини: пароль має містити задану кількість символів;
- Наявність спеціальних символів: перевіряється, чи присутні символи з визначених категорій;
- Включення цифр: визначається наявність хоча б однієї цифри;
- Зміна регістра: відбувається стеження за використанням як верхнього, так і нижнього регістру.

Паролі, що не задовольняють мінімальним вимогам, відсіюються, а інші представляються користувачу як кінцевий результат генерації.

Таким чином, математична модель генерування надійних паролів інтегрує кілька методів розмивання початкового тексту, що дозволяє досягти високої складності (breakingComplexity) при збереженні прийнятної зручності запам’ятовування (rememberComplexity). Використання надійного генератора на базі BBS, разом із налаштованими режимами модифікації — заміною символів, зміною регістра та вставкою цифр — забезпечує створення паролів, стійких до сучасних методів криптоаналізу, і дозволяє налаштовувати процес в залежності від потреб користувача.

1.3.3 Адаптація математичної моделі алгоритму SHA-256 для валідації паролів

Для забезпечення високої безпеки зберігання та перевірки паролів широко використовується одностороння хеш-функція SHA-256, яка перетворює вхідний пароль у фіксований 256-бітовий хеш. Адаптація математичної моделі алгоритму

SHA-256 до процесу валідації паролів полягає у формалізації цього перетворення та побудові механізму перевірки автентичності введеного паролю на основі порівняння отриманого хешу із збереженим значенням.

Нехай P — множина усіх можливих паролів, що користувач може ввести, а $\mathcal{H}: P \to 0{,}1^{256}$— функція, яка відображає пароль $p \in P$ у хеш h за допомогою алгоритму SHA-256. Тобто ми записуємо:

$$h = \mathcal{H}(p) = \mathrm{SHA256}(p) \qquad (1.15)$$

Завдяки властивостям SHA-256, навіть незначна зміна у вхідному паролі призводить до кардинально іншої вихідної послідовності бітів (ефект лавини), що забезпечує його односторонність — відновити *p* із *h* практично неможливо.

Щоб посилити захист і унеможливити використання попередньо обчислених таблиць (rainbow tables), до процесу додатково впроваджується соль. Нехай *s* — унікальне значення, генероване для кожного паролю. Тоді функція хешування набуває вигляду:

$$h = \mathcal{H}(s, |, p) = \mathrm{SHA256}(s, |, p) \qquad (1.16)$$

де | означає конкатенацію. Збереження як хешу *h*, так і використаної солі *s* для кожного користувача забезпечує більш глибокий рівень захисту.

Процес валідації пароля можна представити як функцію перевірки:

verifyPassword(p, s, h_stored) =

- *true, якщо SHA256(s || p) == h_stored*
- *false, якщо SHA256(s || p) != h_stored*

де:

- p — введений користувачем пароль;
- s — збережена сіль, що була використана для первинного хешування;
- h_{stored} — збережений хеш, отриманий на етапі реєстрації або попередньої генерації.

Таким чином, даний алгоритм порівнює хеш нового введення з заздалегідь збереженим значенням. Якщо результати співпадають, система приймає пароль як автентичний.

Адаптована модель включає наступні основні етапи:

1. Вхідні дані: Користувач вводить пароль p. Для кожного користувача зберігається пара (s, h_{stored});

2. Хешування: Обчислюється $h = \mathcal{H}(s, |, p)$ за допомогою алгоритму SHA-256;

3. Порівняння: Отриманий хеш h порівнюється з h_{stored}. Якщо $h = h_{\text{stored}}$, аутентифікація проходить успішно.

Математична модель адаптації забезпечує ефективність валідації, оскільки алгоритм SHA-256 гарантує односторонність та високий рівень криптографічної стійкості, а використання солі значно ускладнює потенційний криптоаналіз. Таким чином, система перевірки паролів може бути математично описана як відображення «пароль + сіль» у 256-бітовий простір, де співпадіння хешів є єдиним критерієм успішної автентифікації. Цей підхід забезпечує не лише перевірку цілісності та автентичності введених даних, а й явний захист від атак, спрямованих на відновлення оригінального пароля із збереженого хешу. Комплексна математична модель валідації, заснована на SHA-256, є фундаментальним компонентом сучасних систем захисту паролів, дозволяючи забезпечити високий рівень безпеки навіть при зростаючих обчислювальних можливостях потенційних зловмисників.

1.4 Розробка структури та алгоритмів роботи програмного застосунку для генерування та валідації надійних паролів

Пропонований програмний засіб спрямований на збільшення криптографічної стійкості паролів через інтеграцію модульного підходу, що включає як генерацію, так і валідацію кінцевих результатів. Структура застосунку (рис. 1.9) побудована таким чином, що забезпечує гнучке налаштування процесу «розмивання» початкового ключового слова та його подальшу перевірку за заданими критеріями.

Основні компоненти системи:

- Інтерфейс користувача. За допомогою графічного інтерфейсу користувач вводить початкове ключове слово та встановлює параметри генерації. Серед параметрів – вибір методів розмивання (наприклад, заміна літер на спеціальні символи, зміна регістра окремих символів та вставлення цифр у середину слова) та значення ймовірності для кожного з методів. Це дозволяє кожному

користувачу адаптувати процес генерації відповідно до власних потреб та переваг;

- Блок керування. Цей модуль координує роботу всіх інших блоків, формує відповідні команди та забезпечує інтеграцію даних між різними підсистемами. Він відповідає за послідовність виконання операцій, що починається з отримання параметрів від користувача і закінчується передачою відфільтрованих, надійних варіантів пароля на екран;
- Блок заміни символів. Для підвищення складності парольної фрази застосовується метод заміни окремих літер на спеціальні символи. Коли користувач задає певний відсоток ймовірності заміни, кожна літера перевіряється за допомогою генератора псевдовипадкових чисел. Якщо згенероване число входить у заданий діапазон, відповідний символ замінюється на один з варіантів, що було відфільтровано за результатами попереднього опитування (табл. 1.2). Цей підхід дозволяє отримати релевантні для користувача варіанти заміни, які візуально схожі на оригінальні символи;
- Блок заміни регістра. Модуль здійснює зміну регістра літер за користувацьким параметром. Для кожного символу, який належить до алфавіту, генерується випадкове число. Якщо воно потрапляє в інтервал, визначений користувачем, регістр даної літери змінюється (з великого в малий або навпаки). Це дозволяє додатково ускладнити пароль, зберігаючи при цьому можливість його запам’ятовування;
- Блок інтеграції випадкових чисел. Додатковою мірою підвищення стійкості пароля є вставлення випадкових цифр у випадкові позиції всередині ключового слова. За допомогою цього блоку, знову використовуючи методи псевдовипадкового генерування чисел, визначається місце для додавання цифри. Цей механізм значно ускладнює атаки за допомогою словників, адже він порушує звичну послідовність символів;
- Блок генерації псевдовипадкових чисел. Основою для роботи методів розмивання є надійний генератор випадкових чисел. Для цього застосовується

алгоритм, який забезпечує рівномірне розподілення чисел у заданому діапазоні (криптографічно стійкий генератор на основі BBS). Це гарантує, що випадковість вибору для кожного методу не залежить від послідовності попередніх операцій і забезпечує високий рівень криптографічного захисту;

- Блок перевірки та валідації. Важливим етапом є оцінка згенерованих паролів, що включає перевірку їх відповідності мінімальним вимогам: правильна довжина, наявність спеціальних символів, цифр, використання літер обох регістрів тощо. Крім цього, для додаткової безпеки застосовується модуль валідації хешу (на базі алгоритму SHA-256), який перевіряє, що кінцевий пароль має достатній рівень криптографічної стійкості. Надто слабкі варіанти відкидаються, а ті, що відповідають рекомендованим стандартам, виводяться на екран для подальшого використання.

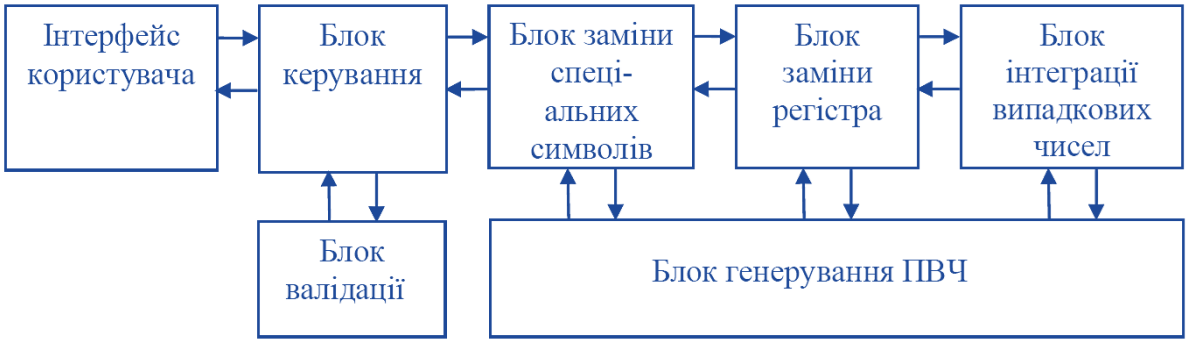


Рисунок 1.9. Структурна схема застосунку

Алгоритм роботи застосунку узагальнюється наступними кроками:

1. Введення даних: Користувач через створений інтерфейс вводить ключове слово та обирає методи розмивання з відповідними ймовірнісними параметрами;

2. Формування параметричного алфавіту: На основі введених налаштувань визначається список доступних символів (заміна символів, зміна регістра тощо);

3. Розмивання ключового слова: Блоки заміни символів, регістра та інтеграції випадкових чисел послідовно обробляють початкове слово, використовуючи надійний генератор випадкових чисел для визначення випадкових замін та вставок;

4. Валідація: Згенеровані варіанти паролів проходять перевірку за заданими критеріями (довжина, наявність спеціальних елементів) та валідацію хешу згідно з алгоритмом SHA-256. Слабкі варіанти відсіюються;

5. Виведення результатів: Кінцеві, перевірені паролі демонструються користувачу для подальшого використання.

Завдяки комплексному підходу до генерації, який об'єднує декілька методів розмивання пароля, із застосуванням сучасних технологій псевдовипадкового генерування чисел та валідації за допомогою хешування, запропонований програмний засіб дозволяє значно підвищити стійкість отриманих парольних фраз. Це забезпечує ефективний захист від словникових атак і атак грубої сили, зберігаючи при цьому зручність для користувачів за рахунок адаптивного налаштування параметрів генерації.

1.4.1 Розробка БСА генерування паролів

Базова парольна фраза створюється з використанням стандартного алфавіту, який містить символи, що вводяться з клавіатури відповідного стандарту. Далі для підвищення її стійкості застосовуються кілька методів «розмивання». Серед цих методів – заміна окремих літер на подібні спеціальні символи, зміна регістра та вставлення випадкових цифр у формується автоматично відповідно до заданих параметрів.

Метод заміни літер на спеціальні символи (рис.1.10) дозволяє посилити захист шляхом модифікації базової парольної фрази. На вході блоку не подається жодне слово від користувача, оскільки пароль генерується автоматично. Натомість користувач задає параметри процедури, зокрема рівень ймовірності заміни символу (наприклад, 40 %):

- Для кожного символу автоматично згенерованої фрази система генерує псевдовипадкове число в діапазоні від 0 до 100;
- Якщо значення потрапляє в межі заданої ймовірності (наприклад, від 0 до 40), відповідний символ замінюється на один із альтернативних варіантів, що вибрані з попередньо відсортованої таблиці заміни (табл. 1.2);
- Варіанти заміни формуються на основі статистично важливих даних, що забезпечують візуальну схожість із оригіналом, але значно ускладнюють послідовність символів для потенційних атак.

Для подальшого підвищення криптографічного захисту застосовується метод

випадкової інверсії регістра окремих символів (рис.1.11):

- При обході базової згенерованої парольної фрази для кожної літери генерується випадкове число (у тому ж діапазоні від 0 до 100);
- Якщо згенероване число потрапляє в заздалегідь встановлений користувачем інтервал (наприклад, 30 %), система автоматично змінює регістр цієї літери (з великого на малий або навпаки);
- Цей процес також забезпечується за допомогою надійного генератора псевдовипадкових чисел.

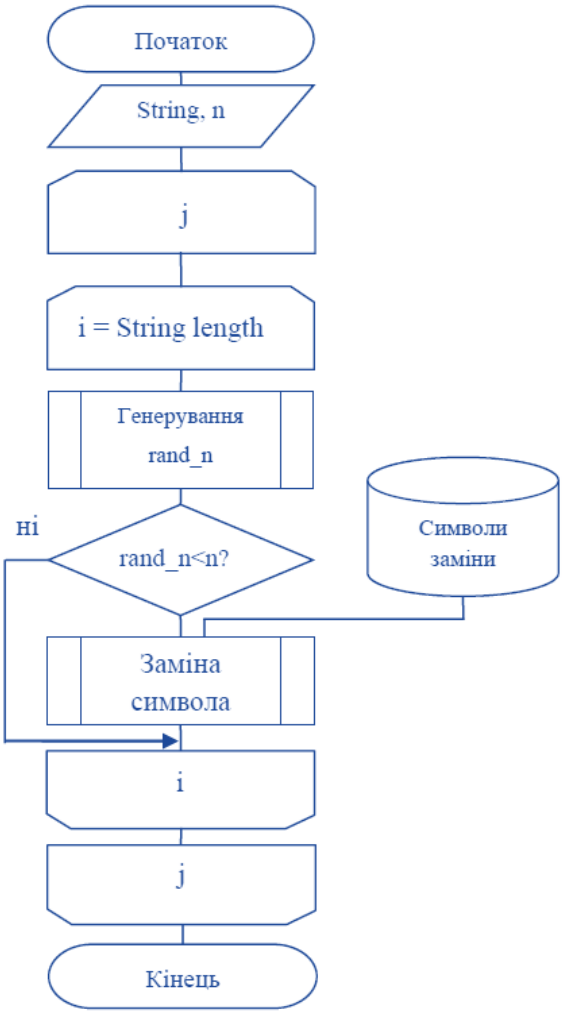


Рисунок 1.10. БСА методу заміни літер на спеціальні символи у паролі

Для подальшої ускладнення структури парольного рядка і запобігання словниковим атакам передбачається вставлення випадкової цифри у середину згенерованої фрази (рис.1.12):

- Цей метод застосовується окремо один раз для кожної згенерованої парольної фрази;

- За допомогою генератора випадкових чисел визначається, чи буде вставлятися цифра, виходячи з заданого користувачем відсоткового порогу;
- Якщо умова задовольняється, у випадково обрану позицію в рядку вставляється цифра (від 0 до 9).

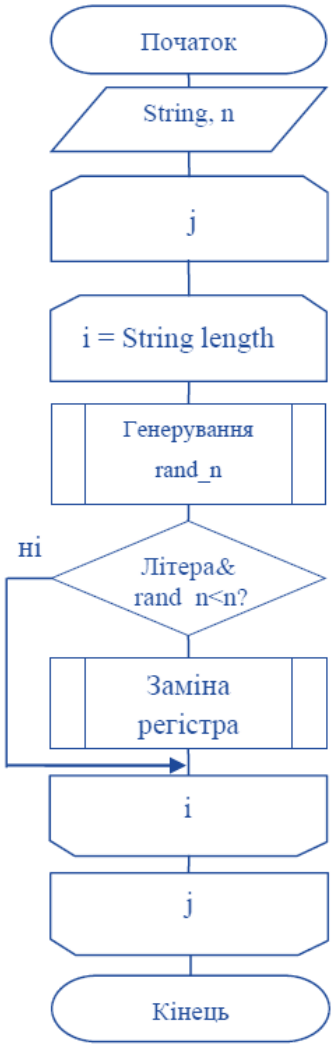


Рисунок 1.11. БСА методу заміни регістру літер у паролі

Всі методи модифікації ґрунтуються на одному стабільному генераторі випадкових чисел, реалізованому за методом BBS. Сам алгоритм BBS забезпечує високий рівень непередбачуваності та криптографічну стійкість завдяки своїм математичним властивостям. Для кожного виклику генератора кількість отриманих чисел гарантує рівномірний розподіл в межах від 0 до 100, що дозволяє точно застосовувати ймовірнісні пороги, встановлені користувачем.

Оскільки алгоритми розмивання, будучи побудованими на випадковості, можуть створити варіанти без застосування жодного з методів (що знижує

остаточну стійкість), після етапу розмивання здійснюється перевірка згенерованих паролів. Блок перевірки аналізує кожну отриману парольну фразу за базовими критеріями безпеки: відповідність заданій довжині, наявність спеціальних символів, цифр та використання різних регістрів. Слабкі варіанти відсіюються, а найкращі результати представлені користувачу.

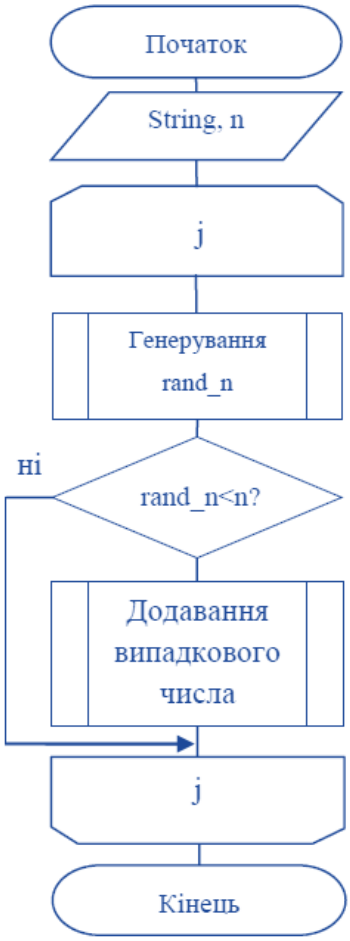


Рисунок 1.12. БСА методу вставлення випадкової цифри у паролі

1.4.2 Розробка БСА валідації паролів

Основною метою алгоритму для валідації згенерованих паролів є оцінка якості пароля та відсіювання слабких варіантів, які не відповідають встановленим стандартам криптографічної стійкості. Процес валідації інтегрує кілька ключових перевірок, що дозволяють гарантувати відповідність паролів сучасним вимогам до безпеки.

Під час перевірки пароля враховуються такі параметри (рис.1.13):

- Довжина пароля: Пароль повинен відповідати мінімальним вимогам щодо довжини, встановленим користувачем (наприклад, від 8 до 50 символів);

- Наявність спеціальних символів: Система перевіряє, чи містить пароль хоча б один спеціальний символ із множини допустимих символів (!, @, #, $, %, ^, &);
- Наявність цифр: Перевіряється, чи містить пароль хоча б одну цифру з діапазону 0-9;
- Використання літер верхнього та нижнього регістру: Пароль повинен містити хоча б одну літеру у верхньому регістрі та одну літеру у нижньому регістрі;
- Унікальність структури: Перевіряється, чи пароль не складається виключно з повторюваних символів або простих комбінацій, які можуть бути вразливими для атак методом перебору.

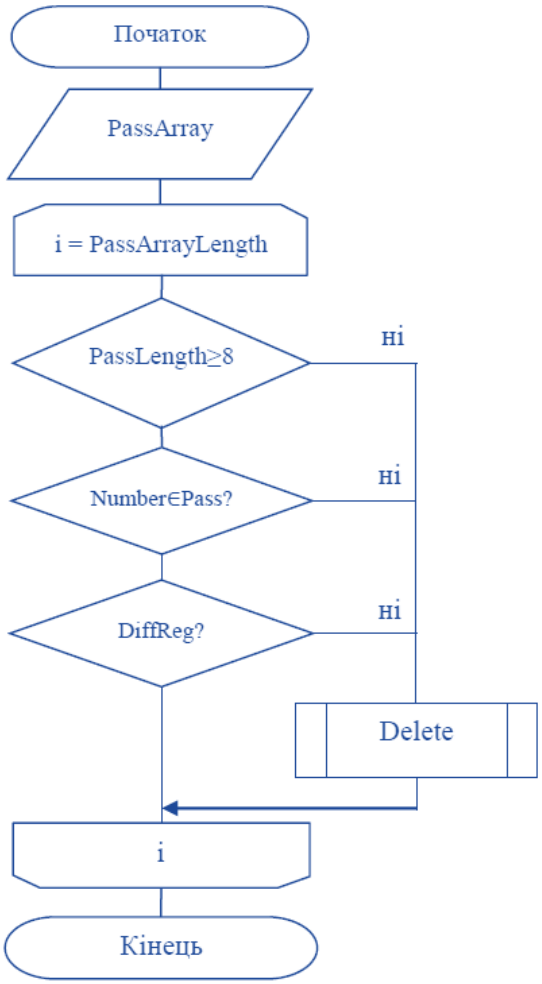


Рисунок 1.13. БСА перевірки пароля

Оскільки система передбачає генерацію паролів із високим рівнем захисту, кожен пароль додатково проходить хешування за допомогою SHA-256 та валідацію:

1. Обчислення хешу: Для кожного пароля обчислюється його хеш-функція

SHA-256: $h = \mathrm{SHA256}(p)$, де p — згенерований пароль, а h — хеш його значення.

2. Перевірка на відповідність раніше збереженим значенням: Якщо система застосовується для автентифікації, то перевіряється, чи отриманий хеш співпадає з збереженими значеннями у базі даних.

verifyPassword(p, s, h_stored) ={ true, якщо H(s || p) = h_stored false, інакше }

де *s* — унікальна сіль для пароля, що використовується для запобігання атакам через попередньо обчислені таблиці (rainbow tables).

Процес валідації реалізується через послідовний аналіз кожного згенерованого пароля. Алгоритм перевірки складається з таких етапів:

1. Перевірка довжини пароля: Якщо пароль коротший за мінімальну довжину (наприклад, 8 символів), він відкидається;

2. Перевірка наявності спеціальних символів: Виконується пошук хоча б одного символу із допустимого списку;

3. Аналіз використання цифр та регістрів літер: Система перевіряє, чи містить пароль хоча б одну цифру та літери в різних регістрах;

4. Генерація хешу та його валідація: Під час створення пароля система обчислює його SHA-256 хеш та перевіряє його криптографічну стійкість.

5. Остаточний фільтр: Якщо пароль відповідає всім критеріям, він приймається як надійний; у іншому випадку він відкидається, і здійснюється повторна генерація пароля.

Після проходження всіх перевірок система формує список найкращих паролів, що відповідають критеріям стійкості. Слабкі варіанти повністю виключаються із набору, а перевірені паролі пропонуються користувачу для збереження та подальшого використання.

Розроблена БСА валідації паролів дозволяє забезпечити відповідність кожного згенерованого пароля сучасним вимогам до безпеки. Використання алгоритму SHA-256 для перевірки криптографічної стійкості, а також суворе тестування параметрів (довжина, спеціальні символи, цифри, регістр літер) гарантує, що результати генерації є максимально надійними та захищеними від атак методом перебору.

1.5 Реалізація програмної моделі та інтерфейсу застосунку

Програмний застосунок реалізовано мовою програмування C# у інтегрованому середовищі розробки Visual Studio 2022 у двох окремих модулях: перший забезпечує генерування надійних паролів, а другий – їх валідацію за допомогою криптоалгоритму SHA-256. Модуль генерування паролів автоматично створює надійні паролі із застосуванням криптографічно стійкого генератора випадкових чисел на основі алгоритму BBS (Blum Blum Shub) та декількох методів «розмивання» базової парольної фрази. Модуль валідації паролів, який перевіряє згенеровані паролі за допомогою хешування алгоритмом SHA-256.

1.5.1 Розробка модуля генерування паролів

Перший модуль повністю автоматизує процес генерації паролів. Основні завдання цього модуля – створення базового рядка символів із заданого алфавіту та його подальше «розмивання» з використанням трьох ключових методів, що впроваджуються за допомогою надійного генератора випадкових чисел BBS. Основні функціональні блоки модуля такі:

1. Базова генерація рядка: Створюється початковий рядок, який формується за допомогою символів, що доступні за стандартним набором клавіатури. Цей базовий рядок є відправною точкою для подальших модифікацій, і його довжина визначається параметром, що задається користувачем;

2. Методи розмивання:

- Заміна символів на спеціальні: Для кожного символу базового рядка генерується число за допомогою алгоритму BBS у діапазоні від 0 до 100. Якщо номер потрапляє в заданий користувачем інтервал (наприклад, 0–40 при 40%-й заміні), символ замінюється на альтернативний, обраний із попередньо сформованої таблиці заміни. Ця таблиця містить варіанти, які статистично відібрані так, щоб бути візуально схожими з оригінальними літерами;
- Зміна регістра: Для кожного символу, що є літерою, здійснюється перевірка за допомогою BBS. При введенні певного відсотка

(наприклад, 30%) і в разі відповідного випадкового числа символ інвертується (з великої літери перетворюється на малу і навпаки). Таким чином, додатково ускладнюється послідовність символів;

- Вставлення випадкових цифр: Цей метод застосовується один раз для кожного згенерованого пароля. За допомогою BBS визначається, чи відбудеться вставлення цифри (з урахуванням встановленого порогу, наприклад, 20%). Якщо умова виконується, випадково обрана цифра (від 0 до 9) вставляється у середину рядка, що порушує звичну послідовність символів.

3. Фільтрація та перевірка: Після застосування всіх методів «розмивання» отримані паролі проходять первинну валідацію за критеріями: відповідність заданій довжині, наявність спецсимволів, цифр і використання літер різного регістру. Слабкі варіанти відсіюються, а лише ті, що відповідають мінімальним вимогам, подаються як кінцевий результат.

Основний клас модуля, PasswordGenerator, містить такі ключові методи:

- GeneratePassword(int length, bool includeDigits, bool includeSpecials, bool includeUpperCase). Цей метод відповідає за створення вихідного (базового) рядка з потрібною довжиною, а також за виклик процедури «розмивання». Параметри includeDigits, includeSpecials та includeUpperCase дозволяють ввімкнути або відключити відповідні методи зміни;
- ApplySpecialCharSubstitution(string input, int substitutionProbability). Метод, що реалізує алгоритм заміни символів на спеціальні. Для кожного символу викликається BBS-генератор для перевірки, чи повинен символ бути замінений, а далі здійснюється підстановка альтернативного символу;
- ApplyCaseInversion(string input, int inversionProbability). Метод для випадкової зміни регістра символів на основі порогової величини, заданої користувачем;
- InsertRandomDigit(string input, int insertionProbability). Метод, що відповідає за вставлення випадкової цифри у середину базового рядка згідно із встановленою ймовірністю;
- BBSGenerate(int min, int max). Допоміжний метод, що реалізує генератор

випадкових чисел за алгоритмом BBS для отримання значень у заданому діапазоні (від 0 до 100). Завдяки його використанню забезпечується високий рівень криптографічної стійкості у всіх алгоритмах розмивання.

Перший модуль має графічний інтерфейс, розроблений у Windows Forms з використанням Visual Studio 2022. Основні елементи інтерфейсу наступні:

- Основне вікно застосунку. Вікно містить велике текстове поле, у якому автоматично відображається згенерований надійний пароль. Це поле дозволяє користувачеві швидко скопіювати пароль для подальшого використання;
- Перемикачі (Switch/Toggle-Buttons). Три перемикача дозволяють увімкнути або вимкнути режими розмивання: використання цифр – вмикає метод вставлення випадкових цифр у пароль; використання спеціальних символів – забезпечує застосування методу заміни символів; використання великих літер – активує режим зміни регістра літер;
- Повзунок (Slider). За допомогою повзунка користувач може задати довжину генерованого пароля в інтервалі від 5 до 50 символів. Значення повзунка безпосередньо передається до функції GeneratePassword.

Розробка модуля здійснена у C# з використанням підходу об’єктно-орієнтованого програмування. Структура класів логічно розділена на допоміжні класи (для реалізації генератора BBS) та класи, що працюють безпосередньо з інтерфейсом. Виклики до методу генерації відбуваються у відповідь на події (наприклад, зміна положення повзунка або натискання кнопки "Генерувати"), після чого згенерований пароль відображається у текстовому полі.

Базова реалізація метода генерації виглядає наступним чином:

```
public string GeneratePassword(int length, bool includeDigits,
bool includeSpecials, bool includeUpperCase)
{
   // 1. Формування базового рядка із стандартного алфавіту.
   string basePassword = BaseAlphabetGenerator.Generate(length);
   // 2. Застосування методу заміни символів (якщо увімкнено).
   if(includeSpecials)
      basePassword = ApplySpecialCharSubstitution(basePassword,
      substitutionProbability);
      // 3. Застосування методу інверсії регістра (якщо увімкнено).
```

```
        if(includeUpperCase)
            basePassword = ApplyCaseInversion(basePassword, inversionProbability);
            // 4. Вставлення випадкових цифр (якщо увімкнено).
        if(includeDigits)
            basePassword = InsertRandomDigit(basePassword,
            digitInsertionProbability);
            return basePassword;
    }
```

Всі псевдовипадкові числа генеруються за допомогою методу BBSGenerate(), що гарантує високий рівень непередбачуваності.

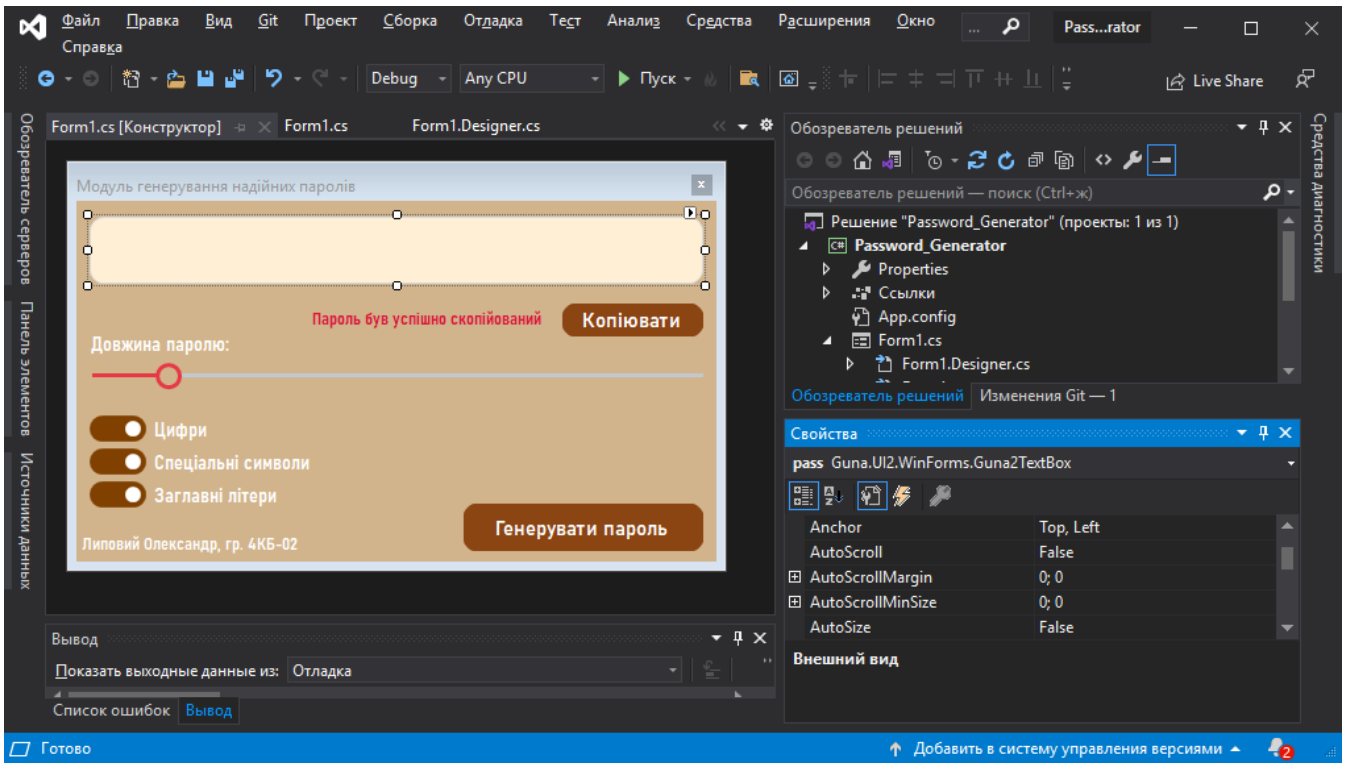


Рисунок 1.14. Розробка модулю генерування паролів у ICP MS Visual Studio

1.5.2 Розробка модуля валідації паролів

Цей модуль працює окремо від модуля генерації паролів, приймаючи як вхід дані, що збережені в системі, і порівнюючи їх з введеним користувачем значенням для автентифікації.

Модуль валідації паролів складається з наступних компонентів:

- Інтерфейс користувача: Просте вікно, яке містить текстове поле для введення пароля користувачем, кнопку «Валідувати» та поле для відображення результату перевірки. При натисканні кнопки перевірка розпочинається, і користувачу відображається повідомлення про успішну або невдалу

автентифікацію;

- Клас для обчислення хешу: Основна логіка модуля побудована навколо методу, який отримує введений пароль і, при необхідності, додає унікальне сольове значення. Даний метод обчислює хеш парольної фрази за алгоритмом SHA-256, використовуючи засоби .NET (клас SHA256.Create());
- Механізм порівняння хешів: Отриманий хеш порівнюється зі збереженим значенням хешу (яке, як правило, зберігається в базі даних або конфігураційному файлі) для підтвердження автентичності. Для цього використовується точне порівняння бітових послідовностей. Якщо хеші збігаються, автентифікація вважається успішною;
- Обробка помилок та валідація введення: Модуль перевіряє, чи не є поле пароля порожнім. У випадку порожнього введення користувачу виводиться повідомлення із проханням ввести пароль. Це забезпечує коректну роботу механізму валідації та запобігає непередбачуваним помилкам.

Основний клас модуля PasswordValidator включає наступні методи:

- ComputeHash(string password, string salt). Цей метод приймає пароль та додаткове сольове значення і обчислює хеш за алгоритмом SHA-256. Реалізація може використовувати об’єкт класу SHA256, що забезпечує одностороннє перетворення, гарантуючи неможливість відновлення вихідного тексту за його хешем;
- ValidatePassword(string inputPassword, string storedHash, string salt). Метод порівнює обчислений хеш введеного пароля (з додаванням сольового значення) із збереженим хешем. Якщо вони співпадають, повертається логічне значення true, інакше — false;
- Обробка порожнього введення: Перед виконанням перевірки модуль виконує валідацію введеного значення. Якщо поле порожнє, або містить лише пробіли, користувачу виводиться відповідне повідомлення, і процес перевірки припиняється.

Нижче наведено код, що ілюструє основну логіку валідації:

```
public class PasswordValidator
{
```

```
    public bool ValidatePassword(string inputPassword, string storedHash, string salt)
    {
        if (string.IsNullOrWhiteSpace(inputPassword))
        {
            MessageBox.Show("Будь ласка, введіть пароль для перевірки.",
            "Помилка введення",
                        MessageBoxButtons.OK, MessageBoxIcon.Warning);
            return false;
        }
        string computedHash = ComputeHash(inputPassword, salt);
        return computedHash.Equals(storedHash);
    }
    private string ComputeHash(string password, string salt)
    {
        // Формуємо комбінований рядок за допомогою сольового значення
        string combined = salt + password;
        using (SHA256 sha256 = SHA256.Create())
        {
            byte[] hashBytes = sha256.ComputeHash(Encoding.UTF8.GetBytes(combined));
            return BitConverter.ToString(hashBytes).Replace("-", "").ToLower();
        }
    }
}
```

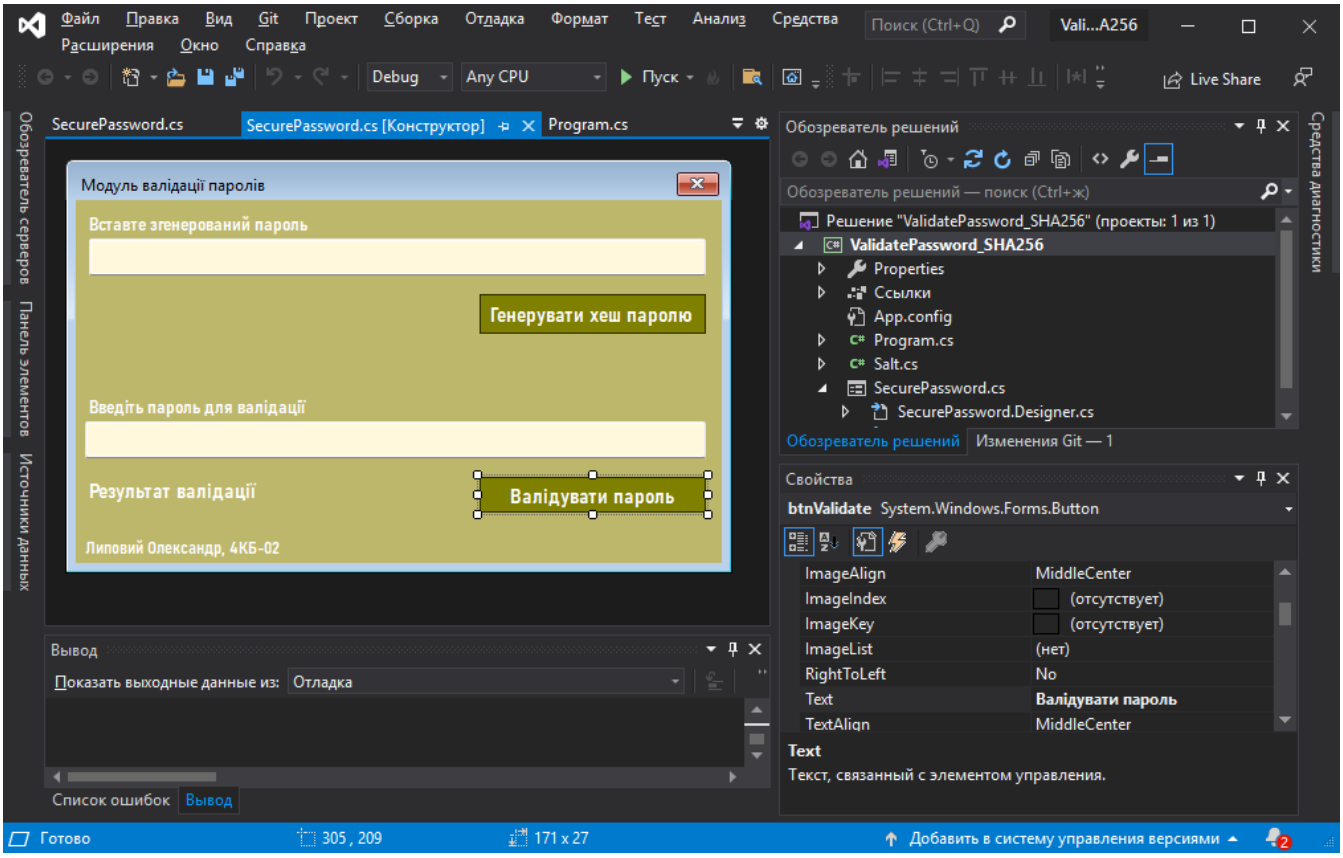


Рисунок 1.15. Розробка модулю валідації паролів у ICP MS Visual Studio

Інтерфейс користувача для модуля валідації розроблено із простим дизайном, що містить:

- поле для введення пароля: Користувач вводить пароль, який потрібно перевірити;
- кнопку «Валідувати пароль»: Після натискання цієї кнопки запускається процес хешування введеного пароля та його порівняння зі збереженим значенням;
- мітку для відображення результатів: Результат перевірки (успішна автентифікація або невідповідність) відображається у відповідній мітці на формі.

1.6 Тестування програмної моделі генерування та валідації надійних паролів

У даному підрозділі описується результати тестування програмної моделі застосунку, що складається з двох окремих модулів: один відповідає за автоматичне генерування надійних паролів, а другий – за їх валідацію з використанням криптографічного алгоритму SHA-256. Нижче наведено опис тестування першого модуля, що стосується процесу генерування паролів.

1.6.1 Тестування модуля генерування паролів

На основі проведеного тестування розробленого застосунку, фрагменти коду якого наведені у Додатку А, можна зазначити, що інтерфейс модуля генерування має наступні інтуїтивно зрозумілі елементи (рис.1.16):

- Поле виведення згенерованого пароля. У верхній частині вікна розміщене текстове поле, де відображається автоматично згенерований пароль. Це поле дозволяє користувачеві швидко переглянути результат роботи алгоритму та скопіювати його за потреби;
- Кнопка для копіювання пароля. Рядом із згенерованим паролем знаходиться кнопка «Копіювати», яка надає можливість швидко скопіювати пароль у буфер обміну для подальшого використання;
- Повзунок для зміни довжини пароля. Інтерфейс передбачає повзунок, за допомогою якого користувач може встановити бажану довжину пароля.

Наразі значення задане на 32 символи, проте інтерфейс дозволяє варіювати довжину в межах від 5 до 50 символів;

- Перемикачі режимів розмивання. Три окремих перемикачі дозволяють активувати чи деактивувати окремі режими модифікації згенерованого пароля: Цифри – режим вставлення випадкових цифрових символів. Спеціальні символи – режим заміни стандартних літер на спеціальні знаки. Заглавні літери – режим застосування інверсії регістра окремих символів. На скріншоті всі ці перемикачі знаходяться у вимкненому положенні, що дає змогу порівняти вихідну базову парольну фразу з модифікованою версією, коли необхідно – просто ввімкнути бажані опції;
- Кнопка «Генерувати пароль». Натискання на кнопку «Генерувати пароль» ініціює процес створення нової парольної фрази з урахуванням заданих налаштувань. Це дозволяє оновлювати пароль у режимі реального часу, що є важливим для тестування ефективності алгоритмів розмивання.

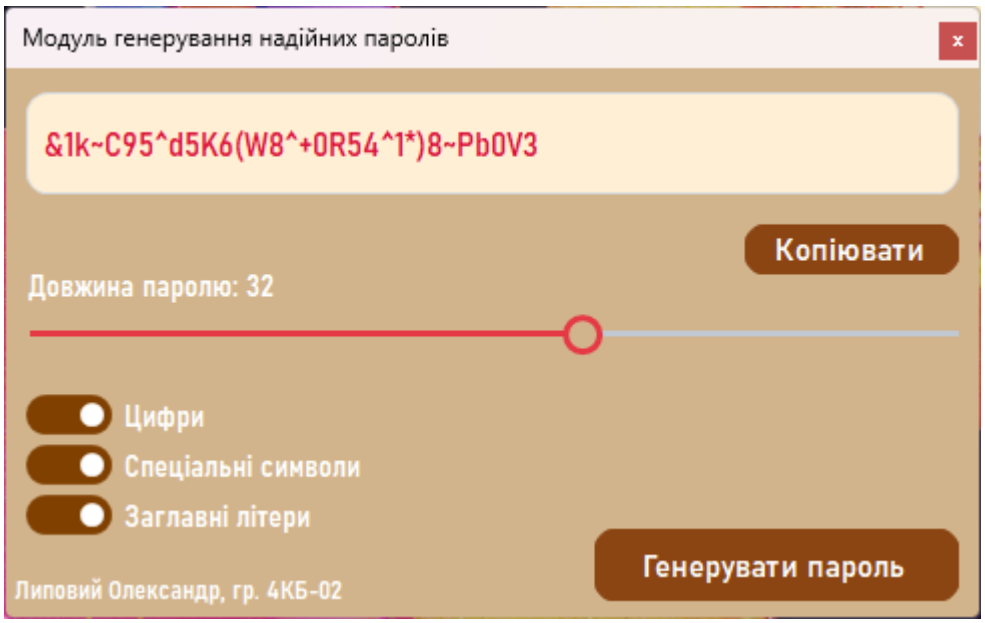


Рисунок 1.16. Вікно модулю генерування паролів

При кожному натисканні кнопки генерації пароль формується автоматично із базового алфавіту, а потім піддається різним методам розмивання:

- Заміна символів на спеціальні знаки. Для кожного символу згенерованого пароля алгоритм за допомогою генератора випадкових чисел BBS визначає, чи підлягає символ заміні за умовами, встановленими користувачем (за

умовою заданого відсотка). Якщо умова виконується, символ замінюється на альтернативу з попередньо визначеної таблиці;

- Зміна регістра літер. В залежності від параметрів, заданих через перемикач, окремі літери можуть бути інвертовані (з великої на малу або навпаки), що значно ускладнює аналіз послідовності символів зловмисниками;
- Вставлення випадкових цифр. Модуль вставляє випадкову цифру в середину парольної фрази, що перериває звичайну послідовність і ускладнює атаки словникового типу.

Під час тестування було перевірено коректність роботи кожного елементу інтерфейсу та алгоритмів розмивання. Реакція системи миттєва: після натискання кнопки «Генерувати пароль» одразу ж у текстовому полі відображається новостворений пароль, який відповідає заданим параметрам. Перемикачі дозволяють легко перемикати режими модифікації, а повзунок – змінювати довжину пароля з високою точністю. Приклади згенерованих паролів підтверджують, що алгоритми BBS забезпечують достатній рівень випадковості та стійкості, що є критично важливим для захисту від атак грубої сили та словникових атак.

1.6.2 Модуль валідації паролів

Інтерфейс модуля валідації паролів, призначеного для перевірки правильності згенерованого пароля за допомогою алгоритму SHA-256, наведений на рис.1.17.

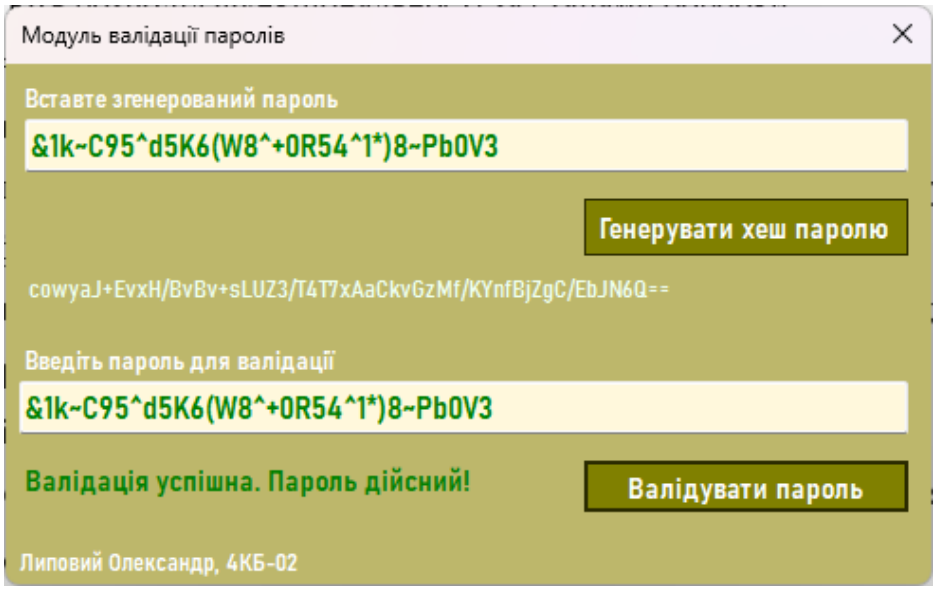


Рисунок 1.17. Вікно модулю валідації паролів: пароль співпадає

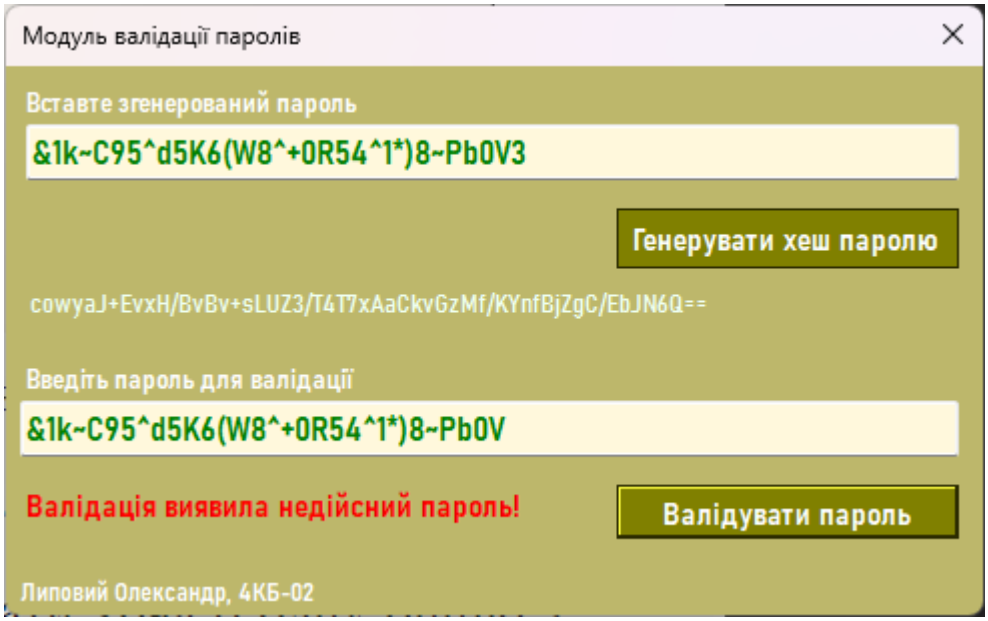


Рисунок 1.18. Вікно модулю валідації паролів: пароль не співпадає

Основні компоненти інтерфейсу включають:

- Поле «Вставте згенерований пароль»: У верхній частині інтерфейсу розташовано текстове поле, в яке копіюється згенерований пароль. Це забезпечує подальшу роботу з перевіркою;
- Кнопка для генерації хешу паролю: Розташована поруч із полем – кнопка ініціює процес обчислення хешу з використанням алгоритму SHA-256;
- Поле для введення пароля для валідації: Нижче розташоване ще одне поле, в яке користувач може ввести або підтвердити пароль, що підлягає перевірці. Зазвичай воно заздалегідь заповнене згенерованим значенням для зручності порівняння;
- Повідомлення з результатом валідації: Після запуску процесу перевірки в інтерфейсі відображається повідомлення, яке засвідчує успішність або невідповідність пароля. Повідомлення «Валідація успішна. Пароль дійсний!» свідчить про те, що введений пароль відповідає збереженому хешу. У зворотному випадку (рис.1.18) з’явиться повідомлення «Валідація виявила недійсний пароль».

Модуль валідації реалізовано на основі алгоритму SHA-256. Основною логікою є наступне:

1. Обчислення хешу: При запуску процесу для введеного (або скопійованого)

пароля, з додаванням сольового значення, обчислюється його SHA-256 хеш за допомогою відповідної бібліотеки .NET;

2. Порівняння: Обчислений хеш порівнюється з попередньо збереженим значенням. Якщо вони співпадають, система видає повідомлення про успішну валідність, інакше повідомляє про невідповідність;

3. Валідація введення: Перед обчисленням проводиться перевірка на порожнє чи невірне значення у полі введення (рис.1.19), що дозволяє уникнути непередбачуваних помилок. У разі відсутності даних користувача повідомляється про необхідність введення пароля.

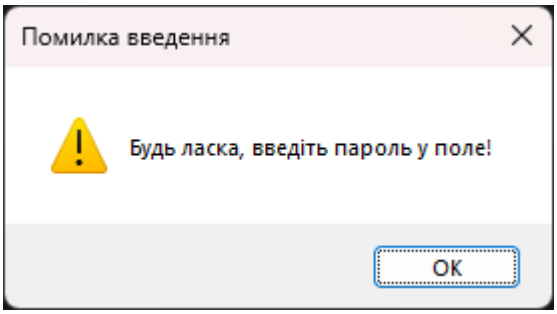


Рисунок 1.19. Вікно повідомлення про відсутність паролю у полі введення

Під час тестування було перевірено коректність роботи всіх елементів інтерфейсу та алгоритмічну точність:

- Після генерації пароля, його коректне відображення у полі «Вставте згенерований пароль» дає можливість миттєво копіювати значення;
- При натисканні кнопки генерації хешу обчислення відбувається без затримок, а результат перевірки відображається на екрані;
- Валідація здійснюється згідно з високими криптографічними стандартами завдяки використанню SHA-256, що гарантує захист від атак методом перебору та словникових атак.

Тестування програмної моделі показало, що обидва модулі – генерації та валідації паролів – працюють ефективно та інтегровані між собою. Інтерфейси модулів виконані у зрозумілому та інтуїтивному стилі. Модуль валідації забезпечує простий спосіб перевірки, де користувач може за потреби скопіювати або ввести пароль, а система за допомогою алгоритму SHA-256 визначає його криптографічну стійкість.

2 ЕКОНОМІЧНИЙ РОЗДІЛ

2.1 Резюме

У дипломному дослідженні розроблено програмну модель, здатну генерувати та перевіряти складні паролі з високою ентропією, що є критично важливим аспектом для забезпечення захисту від несанкціонованого доступу. Представлена система поєднує математичні моделі генерації паролів, алгоритмічні методи підвищення їхньої стійкості та передові криптографічні технології для хешування. В умовах сучасного розвитку кіберзагроз слабкі паролі стають однією з основних вразливостей інформаційної безпеки, а питання використання надійних облікових даних набуває дедалі більшої актуальності через збільшення кількості атак, заснованих на методах перебору та словникових атаках.

Ефективність програмного продукту залежить не лише від його функціональних можливостей, а й від оптимальності процесу розробки. Оцінка якості програмного забезпечення здійснюється за декількома параметрами, включаючи перспективу кінцевого користувача, ефективність використання ресурсів та відповідність заданим вимогам. Для користувача важливим є не лише якість продукту, а й загальні витрати на його створення, зокрема трудові ресурси та фінансові витрати. У межах цього розділу проводиться аналіз вартості розробленого програмного забезпечення.

2.2 Визначення трудомісткості розробки програмного забезпечення

Тривалість розробки залежить від ряду факторів, включаючи обсяг робіт, рівень складності, кваліфікацію команди розробників та часові обмеження, зумовлені ринковими умовами. Для оцінки масштабу програмного продукту використовується метод структурної аналогії, що передбачає звернення до спеціалізованих каталогів схожих проєктів. На їхній основі визначається кількість умовних машинних команд, необхідних для реалізації аналогічного програмного рішення (у тисячах команд).

Таблиця 2.1. Каталог аналогів

Найменування ПП	**Обсяг функції ПП – V_o, усл. машинних командах.**
1. ПП автоматизації засобів по каталогу	680 – 7000
2. ПП автоматизованих розрахунків	1300 – 8600
3. ПП оптимізації розрахунків	1300 – 4200

У таблиці 2.1 представлені аналоги програмного забезпечення, функції яких, у більшому або меншому ступені, виконує розроблений програмний продукт. Для нашого варіанта виділено сірим кольором.

Вибравши аналог ПЗ, що містить Vo в умовних машинних командах, трудомісткості визначати на основі табл.2.2

Таблиця.2.2. Норма часу

Обсяг ПЗ, тис.умов.машинних команд	**Норма часу, люд/год**
1.00	229
2.00	244
3.00	262

На підставі отриманого значення, по довіднику, визначається укрупнена норма часу на розробку аналога програмного забезпечення (коректується поправочним коефіцієнтом враховуючої умови розробки ПП, тобто в умовах комп'ютера, Кк=0,7÷0,8): Тар = 229 х 0,8 = 183,2 (люд/годин).

Трудомісткість програмного продукту визначається окремо для кожного етапу розробки, виходячи з показників трудомісткості відповідного аналога. При цьому враховують рівень складності розробки, ступінь інноваційності та частку використання стандартних модулів. Розрахунки проводяться згідно з наступними формулами:

$$T_{ТЗ} = T^a p \times L_1 \times K_H \qquad (2.1)$$

$$T_{ТП} = T^a p \times L_2 \times K_H \qquad (2.2)$$

$$T_{РП} = T^a p \times L_3 \times K_H \times K_T \qquad (2.3)$$

Для розрахунку необхідні наступні коефіцієнти:

Li – питома вага і-го етапу розробки (див. табл. 2.3.);

Кн – поправочний коефіцієнт, що враховує ступінь новизни (див. табл. 2.4.);

Кт – поправочний коефіцієнт, що враховує ступінь використання в розробці типових програм (див. табл. 2.5)

Таблиця 2.3. Значення питомих коефіцієнтів трудомісткості стадії в загальній трудомісткості розробки ПП

Код стадії	**Ступінь новизни**		
	А	Б	В
ТЗ (L_1)	0,15	0,12	0,12
ТП (L_2)	0,16	0,15	0,11
РП (L_3)	0,55	0,58	0,61

Для нашого варіанта виділено сірим кольором.

Таблиця 2.4. Значення поправочного коефіцієнта, що враховує ступінь новизни

Код ступеня новизни	**Ступінь новизни**	**Значення $К_н$**
А	Принципово нові ПП	1,75 – 1,2
Б	ПП – розвиток визначеного параметричного ряду	1,0 – 0,8
В	ПП маючий аналог	0,7

Для нашого варіанта виділено сірим кольором.

Таблиця 2.5. Значення коефіцієнта ступеня використання в розробці типових програм

Ступінь охоплення реалізованих функцій розроблювального ПП типовими програмами, %	**Значення $К_т$**
60 і вище	0,6
40-60	0,7
20-40	0,8
До 20	0,9

Для нашого варіанта виділено сірим кольором.

Тепер розраховуємо трудомісткість по кожному етапу окремо:

Трудомісткість технічного завдання

$$T_{тз}=T^{а} *L_1*К_н= 183,2 \ *0,12*0,7= 15,38 \text{ (люд/годин)} \quad (2.4)$$

Трудомісткість розробки технічного проекту

$$T_{тп}=T^{а} *L_2*К_н =183,2 \ *0,11*0,7 = 14,11 \text{ (люд/годин)} \quad (2.5)$$

Трудомісткість розробки робочого проекту

$$T_{рп}=T^{а} *L_3*К_н*К_т=183,2 \ *0,61*0,7*0,6= 46,94 \text{ (люд/годин)} \quad (2.6)$$

Для подальших розрахунків визначили кількість папера, витраченого на кожен етап: технічне завдання $N_{тз}$= 2 (стр), розробка ТП $N_{тп}$=27 (стр), розробка робочого проекту $N_{рп}$=9(стр), пояснювальна записка відповідно $N_{пз}$=21(стр) Розрахунок зведений у таблицю 2.6.

Таблиця 2.6. Розрахунок трудомісткості ПП

Найменування етапів	**Розрахунок, годин**		
1.ТЗ	$T_{Ртз}$=15,38	$T_{кк}=0,7*N_{тз}$= 0,7*2=1,4	$T_{нк}=0,15*N_{тз}=0,15*2=0,3$
2.Розробка ТП	$T_{Ртп}$=14,11	$T_{кк}=0,7*N_{тп}=0,7*27=18,9$	$T_{нк}=0,15*N_{тп}=0,15*27=4,05$
3.Розробка РП	$T_{Ррп}$= 46,94	$T_{кк}=0,7*N_{рп}=0,7*9=6,3$	$T_{нк}=0,15*N_{рп}=0,15*9=1,35$
4.Розробка ПЗ	$T_{пз}=1,5*N_{пз}$= 1,5*21 =31,5	$T_{кк}=0,7*N_{тз}=0,7*21=14,7$	$T_{нк}=0,15*N_{пз}=0,15*21$ =3,15
Усього, в т.ч.:	158,08		
- на розробку	$\Sigma T_р$=107,93		
- контроль керівника		$\Sigma T_{кк}$=41,3	
- нормоконтроль			$\Sigma T_{нк}$=8,85

2.3 Розрахунок ціни програмного продукту

Для оцінки вартості програмного продукту розглядаються основна заробітна плата виконавців, матеріальні витрати та загальні витрати на розробку ПЗ. Детальний розрахунок основної заробітної плати наведено у таблиці 2.7. Згідно зі статтею 8 «Закону про Державний бюджет України на 2025» з 1 січня 2025 року встановлено мінімальну місячну заробітну плату у розмірі 8000 гривень, а також мінімальну погодинну тарифну ставку – 48,00 грн.

Таблиця 2.7. Розрахунок основної заробітної плати виконавців

Найменування робіт	Трудомісткість робіт, години	Погодинна тарифна ставка, грн.	Розрахунок, грн.
1.Розробка ПП	107,93	48,00	5180,64
2.Контроль керівника	41,3	108,00	4460,4
3.Нормоконтроль	8,85	120,00	1062,0
Усього	-	-	ΣЗо= 10703,04

Зробимо розрахунок матеріальних витрат на розробку ПП (табл.2.8).

Таблиця 2.8. Розрахунок матеріальних витрат на розробку ПЗ

Найменування матеріальних витрат	Тип, модель	Кількість	Ціна одиниці, грн.	Вартість, грн.
Папір	Лист А4	59	5.0	295,00
Разом	-	-	-	$В_{м\,і}$ =295,00
Транспортно – заготівельні Витрати (10%)				$Втр\_з = 0{,}1 \times Вм1$ = 0,1*280=28,0
Усього				$В_м=В_{мі}+В_{тр\_з=}$ 308,00

На підставі отриманих даних по окремих статтях витрат складена калькуляція планової собівартості в цілому ПП за формою, приведеною в таблиці 2.9.

Таблиця 2.9. Розрахунок статей витрат планової собівартості

Стаття витрат	Значення, грн.	Формула розрахунку
1. Матеріали	324,5	$В_м$ (див. табл. 2.8.)
2. Основна заробітна плата	10703,04	$З_о$ (див. табл. 2.7.)
3.Додаткова заробітна плата	1070,30	$Зд = 0{,}1 \times Зо$ = 10703,04*0,1
4.Відрахування до єдиного фонду соціального внеску	2590,13	$Вє.с.в. = 0{,}22 \times (Зо + Зд)$ =0,22* (10703,04+1070,30)
5. Накладні витрати	4281,22	$Внак. = 0{,}4 \times Зо$ =0,4 * 10703,04
6. Повна собівартість	18969,19	$С_{пов}$= $В_м$ +$З_о$ + $З_д$ +Вє.с.в. + Внак.= 324,5+10703,04+1070,30+2590,13 +4281,22

Розмір прибутку, що включається в ціну, визначаємо по наступній формулі:

$П=(С_{пов}*Р)/100=(18969,19*15)/100=2845,38$ грн (2.7)

Де р – плановий рівень рентабельності (10-15%).

Оптова ціна (кошторисна вартість) визначається по формулі:

$Ц_о = С_{пов}+П$ = 18969,19+2845,38 = 21833,38 грн; (2.8)

Ціна реалізації розробленого програмного забезпечення становитиме:

$Ц_р=Ц_о+ПДВ$ = = 21833,38 +21833,38 *0.2 = 26200,06 грн; (2.9)

3 РОЗДІЛ ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

Сучасне застосування комп'ютерних технологій відіграє ключову роль у створенні ефективних механізмів забезпечення безпеки даних, зокрема в розробці методів генерації та валідації складних паролів. Автоматизація процесу створення захищених паролів дозволяє значно підвищити рівень інформаційної безпеки, зменшити ризик несанкціонованого доступу та оптимізувати управління обліковими записами.

У межах даного дипломного проєкту передбачається створення програмної моделі, яка поєднує алгоритмічні методи генерування паролів із високою ентропією та механізми їхньої перевірки на відповідність сучасним стандартам захисту. В основі розробки лежить використання математичних принципів для формування стійких до перебору паролів, а також застосування криптографічних методів хешування для надійного збереження даних.

Функціональність створеного програмного продукту охоплює не лише генерацію паролів, але й оцінку їхньої міцності з урахуванням загальних кібербезпекових загроз, зокрема атак методом грубої сили та словникових атак. Використання даної моделі сприяє підвищенню рівня захисту інформаційних систем, забезпечуючи користувачам інструменти для формування та перевірки паролів відповідно до найкращих практик безпеки.

Таким чином, розробка такої моделі дозволяє комплексно вирішити проблему ненадійних паролів, автоматизуючи процес їх створення та валідації, що є важливим кроком у зміцненні сучасних систем аутентифікації.

3.1 Аналіз небезпечних і шкідливих факторів, що впливають на користувача ПК

До основних критеріїв забезпечення гігієни робочого середовища належать інтенсивність освітлення, температура повітря, вологість, рівень шумового забруднення, ступінь вібраційного впливу, токсичність, загазованість, а також обмеження загальної м'язової активності. Крім цього, враховується дія електростатичного поля та вплив як неіонізуючих.

3.2 Гігієнічні вимоги до виробничого середовища

Державні санітарні норми, зокрема ДСАНПіН 3.3.2.007-98 «Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин», спрямовані на запобігання негативного впливу шкідливих чинників, що супроводжують роботу з візуальними дисплейними терміналами, на здоров’я працівників.

3.2.1 Вимоги до приміщення

Розміщення робочих місць із використанням ВДТ, ЕОМ і ПЕОМ заборонено у підвальних приміщеннях та на цокольних поверхах. Для кімнат, призначених для роботи з візуальними дисплейними терміналами, рекомендується орієнтувати вікна у напрямку півночі або північного сходу. На вікнах повинні бути встановлені регульовані жалюзі або штори, що дозволяють їх повністю закривати для забезпечення оптимальних умов освітлення.

Планувальні рішення будівель і приміщень, де розташовано відеодисплейні термінали, мають відповідати вимогам ДСАНПіН 3.3.2.007-98. Для робочого місця програміста передбачено мінімальну площу не менше 6 кв. м та об’єм приміщення не менше 20 куб. м. Крім того, стіни приміщень повинні бути пофарбовані матовою фарбою, а в приміщеннях з ВДТ обов’язково мають бути передбачені зони для відпочинку та психологічного розвантаження.

3.2.2 Освітлення

Для створення комфортного середовища для роботи програміста застосовується комбінована система освітлення, що поєднує природне світло з додатковим штучним освітленням. Загальне освітлення приміщення реалізується за допомогою газорозрядних ламп типу ЛД. Відповідно до встановлених нормативів, робоче місце для виконання високоточних операцій (де розмір найдрібнішого об'єкта розрізнення становить 0,3–0,5 мм) має забезпечувати рівномірну освітленість не менш ніж 300 лк. Загалом, ці вимоги щодо рівня освітлення дотримані.

3.2.3 Шум

У робочих приміщеннях основним джерелом шумового навантаження є звуки, що генеруються ПЕОМ. Крім того, значну частину шуму створюють джерела електромагнітного походження – це коливання компонентів електромеханічних пристроїв під впливом змінних магнітних полів. До того ж, в приміщеннях виникає структурний шум, який випромінюють поверхні конструктивних елементів (стіни, перекриття, перегородки) у звуковому спектрі частот. Для зниження або усунення негативного впливу шуму доцільно ізолювати робочі зони, розташовуючи їх у частинах будівлі, що знаходяться в глибині та ведуть своїми вікнами у двір – таким чином мінімізується вплив міського шуму. Крім цього, необхідно регулярно перевіряти герметичність корпусів комп'ютерної техніки та своєчасно здійснювати заміну вентиляторів охолодження.

3.3 Вимоги до організації робочого місця працівника

Конструкція робочого місця користувача комп'ютера, з урахуванням розташування сидіння, засобів керування та засобу відображення інформації, розроблена згідно з антропометричними, фізіологічними та психологічними вимогами, а також відповідно до специфіки виконуваної роботи. Робоче меблеве обладнання повинно бути оснащене можливістю індивідуального регулювання, що дозволить адаптувати його під зріст кожного користувача й підтримувати оптимальну, зручну поставу. Робочий стіл рекомендовано обробляти матовим покриттям, що сприяє зменшенню небажаних відблисків. > > Розміщення дисплея організовано таким чином, щоб його верхня межа відповідала рівню очей, а відстань до екрану становила приблизно 70 см – що повністю входить у допустимий інтервал від 60 до 90 см. Частота мерехтіння екрану fмер дорівнює 100 Гц, що значно перевищує мінімальне рекомендоване значення у 70 Гц. Крім цього, робоче місце розташовано перпендикулярно до віконних прорізів, що дозволяє уникнути прямого та відбитого світлового мерехтіння від вікон та джерел штучного освітлення.

3.4 Мікроклімат

Показники мікроклімату, складу іонів у повітрі, а також рівень шкідливих речовин у робочих зонах, де використовуються ПК, мають відповідати вимогам ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень».

Для підтримки нормативних значень мікроклімату та забезпечення оптимального співвідношення позитивних і негативних іонів слід передбачити установку пристроїв зволоження, штучної іонізації або кондиціювання повітря. Крім того, рівні інфрачервоного випромінювання не повинні перевищувати встановлених нормативних меж згідно з ГОСТ 12.1.005. Також вміст озону в робочій зоні не має перевищувати 0,1 мг/м³, оксидів азоту – 5 мг/м³, а концентрація пилу повинна залишатися в межах 4 мг/м³.

3.5 Електробезпека

Приміщення, де використовуються імпульсні джерела живлення згідно з ОНТП24-86 і ПУЕ-87, віднесено до категорії об'єктів, де ризик ураження персоналу електричним струмом не є підвищеним. Це пояснюється тим, що відносна вологість повітря не перевищує 75%, температура залишається нижчою за 35°С, а хімічно агресивні середовища відсутні. Електроживлення обладнання організовано від двофазної мережі з заземленою нейтраллю, при напрузі 220 В і частоті 50 Гц, із застосуванням автоматичних пристроїв токового захисту.

В приміщенні обов’язково має бути встановлена схема заземлення. Ураження електричним струмом може виникнути у випадках: 1) при контакті з відкритими струмоведучими елементами; 2) при торканні неструмоведучих частин обладнання, які, через порушення ізоляції або інші причини, опинилися під напругою.

Відповідно до вимог ГОСТ-12.2.007.0-75 устаткування (за винятком ЕОМ II класу) відноситься до I класу та оснащене робочою ізоляцією згідно з ГОСТ 12.1.009-76. Підключення обладнання здійснено згідно з нормативами ПБЕ та ПУЕ, тому додаткових заходів щодо електробезпеки не вимагається.

3.6 Пожежна безпека

Робоче приміщення, що відповідає вимогам ПБЕ та ОНТП 24–86 у сфері вибухово-пожежної безпеки, класифікується як об’єкт категорії «В». Основними потенційними причинами виникнення пожежі в такому приміщенні є: 1. Коротке замикання електропроводки; 2. Використання побутових електрорадіоприладів; 3. Недотримання встановлених норм протипожежного захисту. Відповідно до ПУЕ, для зниження ризику виникнення пожежі необхідно забезпечити комплекс заходів, зокрема: ретельну ізоляцію всіх струмоведучих проводів, що підключені до робочих місць, регулярний огляд та перевірку стану їх ізоляції, а також суворе дотримання норм безпечної експлуатації обладнання. Для гасіння пожеж на робочому місці користувача ПК застосовують як вуглекислотні, так і порошкові вогнегасники. – Вуглекислотні вогнегасники випускаються у варіанті ручних пристроїв (наприклад, ВВК-5); – Порошкові вогнегасники представлені моделями ВП-2, ВП-5, ВП-10 та іншими перефразируй

Рисунок 3.1. Засоби пожежогасіння

З метою своєчасного оповіщення, на дільниці необхідно встановити протипожежну сигналізацію. Проходи та запасні виходи повинні бути вільними. Пожежний щит повинен розміщуватись в доступному місці та містити первинні засоби пожежогасіння (вогнегасник, лопату, відро, простирадло, ящик з піском)

ВИСНОВКИ

У дипломному проекті було виконано аналіз сучасних методів і засобів автентифікації користувача, реалізовано програмне рішення, яке поєднує автоматичну генерацію паролів із високою криптографічною стійкістю та їхню подальшу валідацію за сучасними стандартами безпеки. Протягом роботи над проектом було виконано кілька ключових етапів, починаючи від аналізу вимог і розробки концептуальної моделі, до впровадження алгоритмів у програмне забезпечення на мові C# в середовищі Visual Studio 2022. Застосунок розбитий на два незалежні модулі: модуль генерування паролів і модуль їх валідації. Перший модуль базується на використанні криптографічно стійкого генератора випадкових чисел за алгоритмом BBS (Blum Blum Shub), що дозволяє автоматично створювати базову парольну фразу із символів стандартного алфавіту, а далі піддавати її модифікації через застосування декількох методів «розмивання» — заміни літер на спеціальні символи, зміни регістра та вставлення випадкових цифр. Завдяки інтуїтивно зрозумілому графічному інтерфейсу, який включає текстове поле для показу згенерованого паролю, перемикачі для активації відповідних режимів, а також повзунок для налаштування довжини паролю (від 5 до 50 символів), користувач може легко та оперативно отримувати унікальні, високостійкі паролі. Другий модуль реалізовано для валідації згенерованих паролів шляхом застосування одностороннього хешування алгоритмом SHA-256 у поєднанні з сольовим значенням, що гарантує ризик відновлення вихідного пароля практично нульовим і дозволяє ефективно виявляти невідповідності при автентифікації.

Проведене тестування програмної моделі показало, що обидва модулі працюють коректно, алгоритми генерації забезпечують високий рівень випадковості, а модуль валідації надійно перевіряє автентичність створених паролів. Отримані результати свідчать про те, що застосування сучасних криптографічних методів є ефективним підходом для створення систем захисту інформації. Розроблена модель має значний практичний потенціал, може бути інтегрована у сучасні системи автентифікації, і відкриває перспективи для подальшого вдосконалення алгоритмів генерування та перевірки паролів.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Афанасьєв В. Основи криптографії. – Київ: Наукова думка, 2010. – 320 с.
2. Андрієнко Ю. Сучасні методи автентифікації. – Львів: ЛІГА, 2012. – 250 с.
3. Бабій І. Захист інформації в комп'ютерних системах. – Дніпро: ДНІПРО-Прес, 2008. – 308 с.
4. Веремій Г. Криптографічні алгоритми та їх застосування. – Харків: Харківський національний університет, 2015. – 200 с.
5. Гнатюк О. Основи цифрової безпеки. – Вінниця: Вид. «Безпека», 2016. – 276 с.
6. Дмитренко Л. Інформаційна безпека: теорія і практика. – Одеса: Одеська політехніка, 2013. – 350 с.
7. Єфремов С. Основи криптографії та захисту даних. – Київ: Видавничий дім НТУ, 2014. – 310 с.
8. Журавльов П. Методи криптографічного аналізу. – Львів: Видавництво ЛНУ, 2011. – 250 с.
9. Зозуля М. Сучасні підходи до захисту інформації. – Харків: Харьківський центр інформаційної безпеки, 2018. – 290 с.
10. Іваненко О. Криптографія у цифрових мережах. – Київ: Фенікс, 2009. – 334 с.
11. Коваленко Р. Основи криптографічної безпеки. – Київ: Літопис, 2017. – 280 с.
12. Литвин А. Інформаційна безпека систем: підручник. – Львів: Видавництво «Світ», 2015. – 400 с.
13. Мельник В. Базові принципи захисту даних. – Київ: Центр учбової літератури, 2012. – 310 с.
14. Никитенко Г. Методи криптографічного захисту інформації. – Дніпро: Академіка, 2016. – 275 с.
15. Олійник Т. Комп'ютерна безпека: теорія та практика. – Вінниця: Фенікс, 2010. – 290 с.
16. TechInfo. Шифрування та хешування в сучасних системах [Електронний ресурс]. – Режим доступу: https://techinfo.ua/crypto-hash (дата звернення: 15.05.2025).

ДОДАТОК А. Фрагменти коду мовою C# модулів генерування та валідації паролів

Фрагмент коду модуля генерування паролів

```
using System;
using System.Numerics;
using System.Text;

// Клас, що реалізує генератор псевдовипадкових чисел за алгоритмом BBS
public class BBSRandom
{
    private BigInteger p, q, n, state;

    // Конструктор приймає два простих числа p, q та початкове значення
(seed)
    public BBSRandom(BigInteger p, BigInteger q, BigInteger seed)
    {
        this.p = p;
        this.q = q;
        n = p * q;
        // Встановлюємо початковий стан: seed^2 mod n
        state = BigInteger.ModPow(seed, 2, n);
    }

    // Метод Next() обчислює наступне число у послідовності
    public BigInteger Next()
    {
        state = BigInteger.ModPow(state, 2, n);
        return state;
    }

    // Метод для отримання цілих чисел у заданому діапазоні [min, max]
    public int NextInt(int min, int max)
    {
        int range = max - min + 1;
        // Простіший спосіб – використання отриманого числа за модулем range
        int value = (int)(Next() % range);
        return min + value;
    }
}

// Клас для генерації паролів із застосуванням різних методів розмивання
public class PasswordGenerator
{
    // Визначення наборів символів
    private static readonly string Letters = "abcdefghijklmnopqrstuvwxyz";
    private static readonly string UpperLetters =
"ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    private static readonly string Digits = "0123456789";
    private static readonly string SpecialSymbols = "!@#$%^&*";

    // Статичний екземпляр генератора BBS
    public static BBSRandom Bbs = new BBSRandom(new BigInteger(101), new
BigInteger(113), new BigInteger(1234567));

    // <summary>
    // Генерує пароль із зазначеною довжиною та параметрами модифікації.
    // </summary>
    // <param name="length">Бажана довжина пароля</param>
    // <param name="useDigits">Використовувати вставлення цифр</param>
    // <param name="useSpecial">Використовувати заміну символів на
    //  спеціальні</param>
    // <param name="useUpperCase">Використовувати зміну регістру</param>
```

```
    // <param name="substitutionProb">Ймовірність заміни
    //  символу на спеціальний (у %)</param>
    // <param name="caseInversionProb">Ймовірність інверсії регістру (
    // у %)</param>
    // <param name="digitInsertionProb">Ймовірність вставлення цифри
    // (у %)</param>
    // <returns>Згенерований пароль</returns>
    public static string GeneratePassword(int length, bool useDigits, bool
useSpecial, bool useUpperCase, int substitutionProb, int caseInversionProb,
int digitInsertionProb)
    {
// 1. Формування базової парольної фрази (вибираємо символи з набору Letters)
        StringBuilder basePassword = new StringBuilder();
        for (int i = 0; i < length; i++)
        {
            int index = Bbs.NextInt(0, Letters.Length - 1);
            basePassword.Append(Letters[index]);
        }

        char[] pwdChars = basePassword.ToString().ToCharArray();
        // 2. Метод заміни символів на спеціальні
        if (useSpecial)
        {
            for (int i = 0; i < pwdChars.Length; i++)
            {
                int chance = Bbs.NextInt(0, 100);
                if (chance < substitutionProb)
                {
             // Замінюємо символ на випадковий зі списку спеціальних символів
                    int spIndex = Bbs.NextInt(0, SpecialSymbols.Length - 1);
                    pwdChars[i] = SpecialSymbols[spIndex];
                }
            }
        }
        // 3. Метод зміни регістра літер (інверсія регістру)
        if (useUpperCase)
        {
            for (int i = 0; i < pwdChars.Length; i++)
            {
                if (Char.IsLetter(pwdChars[i]))
                {
                    int chance = Bbs.NextInt(0, 100);
                    if (chance < caseInversionProb)
                    {
                        // Якщо літера малі, переводимо в великі і навпаки
                        pwdChars[i] = Char.IsLower(pwdChars[i]) ?
                     Char.ToUpper(pwdChars[i]) : Char.ToLower(pwdChars[i]);
                    }
                }
            }
        }
        // 4. Метод вставлення випадкових цифр (один раз вставляємо цифру у
        //  середину, якщо умова виконується)
        string finalPassword = new string(pwdChars);
        if (useDigits)
        {
            int chance = Bbs.NextInt(0, 100);
            if (chance < digitInsertionProb)
            {
                int midPos = finalPassword.Length / 2;
                int digitIndex = Bbs.NextInt(0, Digits.Length - 1);
                finalPassword = finalPassword.Insert(midPos,
Digits[digitIndex].ToString());
            }
        }
        return finalPassword;
    }
}
```

Фрагмент коду модуля валідації паролів

```
using System;
using System.Security.Cryptography;
using System.Text;

public class PasswordValidator
{
    // <summary>
    // Обчислює SHA-256 хеш для поєднаного рядка "сіль + пароль"
    // та повертає його як шістнадцятковий рядок.
    // </summary>
    // <param name="data">Вхідний рядок (сіль + пароль)</param>
    // <returns>Хеш SHA-256</returns>
    private static string ComputeSHA256(string data)
    {
        using (SHA256 sha256 = SHA256.Create())
        {
            byte[] bytes = sha256.ComputeHash(Encoding.UTF8.GetBytes(data));
            StringBuilder builder = new StringBuilder();
            foreach (byte b in bytes)
            {
                builder.Append(b.ToString("x2"));
            }
            return builder.ToString();
        }
    }

    // <summary>
    // Валідовує введений пароль, порівнюючи обчислений хеш
    // (з доданою сіллю) з збереженим значенням.
    // </summary>
    // <param name="inputPassword">Пароль, введений користувачем</param>
    // <param name="storedHash">Збережений хеш</param>
    // <param name="salt">Сіль</param>
    // <returns>True, якщо хеші співпадають; інакше false</returns>
    public static bool ValidatePassword(string inputPassword, string
storedHash, string salt)
    {
        // Обчислюємо хеш для поєднаного рядка "сіль + введений пароль"
        string computedHash = ComputeSHA256(salt + inputPassword);
        return computedHash.Equals(storedHash,
StringComparison.OrdinalIgnoreCase);
    }
}
```

ДОДАТОК Б. Слайди мультимедійної презентації

Багатофакторна автентифікація і взаємодія різних методів захисту

Порівняльна характеристика методів автентифікації користувача

Метод автентифікації	Основні характеристики	Переваги	Недоліки
Парольна автентифікація	Використання знань користувача (паролі, PIN-коди). Надійність залежить від складності секретної інформації та її конфіденційності.	Простота впровадження та низька вартість.	Схильність до соціальної інженерії, атаки перебором, недостатня стійкість при використанні слабких паролів.
Фізичні засоби	Аутентифікація через наявність у користувача фізичних носіїв (смарт-карти, токени).	Додатковий фізичний бар'єр, що ускладнює несанкціонований доступ при компрометації секретної інформації.	Ризик втрати/крадіжки пристрою, необхідність апаратного забезпечення, можливість підміни при фізичному доступі до пристрою.
Біометричні дані	Використання унікальних фізичних або поведінкових ознак користувача (відбитки пальців, розпізнавання обличчя, голос).	Висока точність завдяки унікальності біометричних даних, неможливість точного копіювання.	Висока вартість обладнання, потенційні помилки розпізнавання, питання конфіденційності та зберігання особистих даних.
Цифрові сертифікати	Використання засобів криптографії з відкритими та закритими ключами для підтвердження автентичності користувача.	Забезпечення високої конфіденційності і цілісності даних, складність фальсифікації за допомогою криптографічних алгоритмів.	Складність управління сертифікатами, високі витрати на впровадження та обслуговування системи.
Одноразові паролі (OTP)	Генерація динамічних паролів, які використовуються лише один раз для кожного сеансу автентифікації.	Захищеність від повторного використання викрадених даних, підвищення безпеки за рахунок змінності пароля в реальному часі.	Необхідність синхронізації між пристроями користувача та системою, додаткові витрати на реалізацію алгоритмів генерації OTP.

Алгоритм генерації паролів із застосуванням трансформації символів та перевірки ентропії

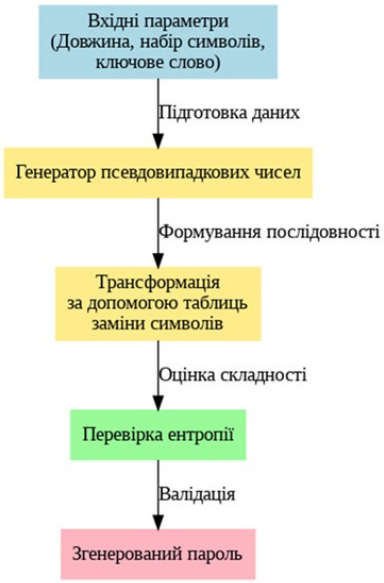


Схема вихорю Мерсенна

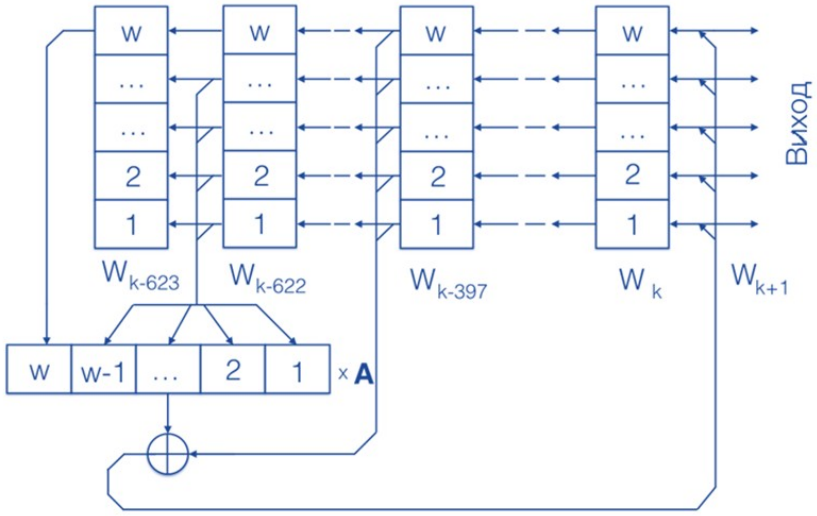
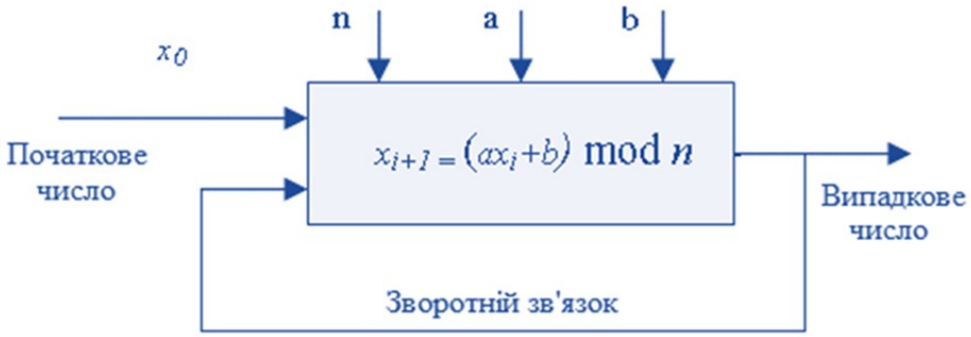
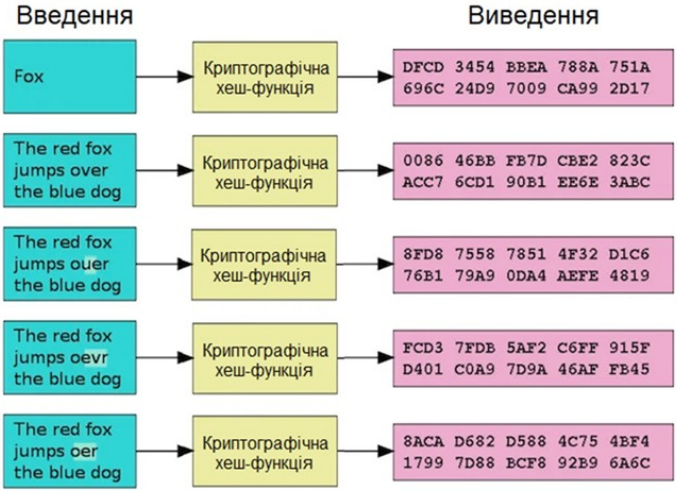


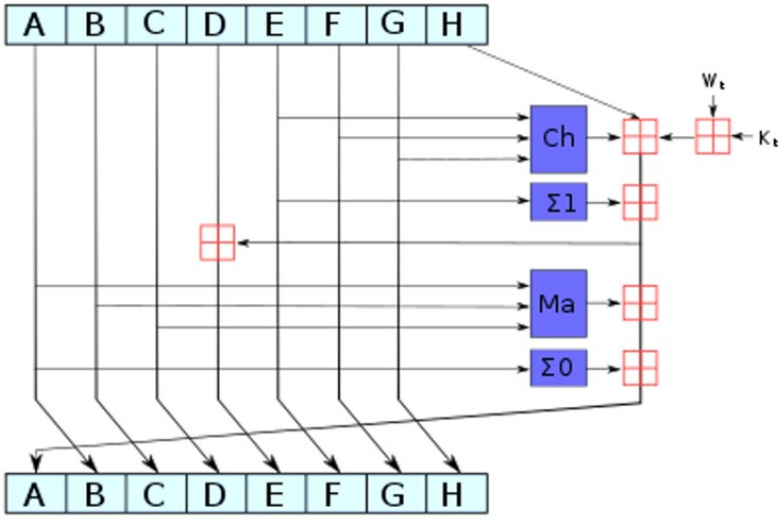
Схема роботи алгоритму BBS



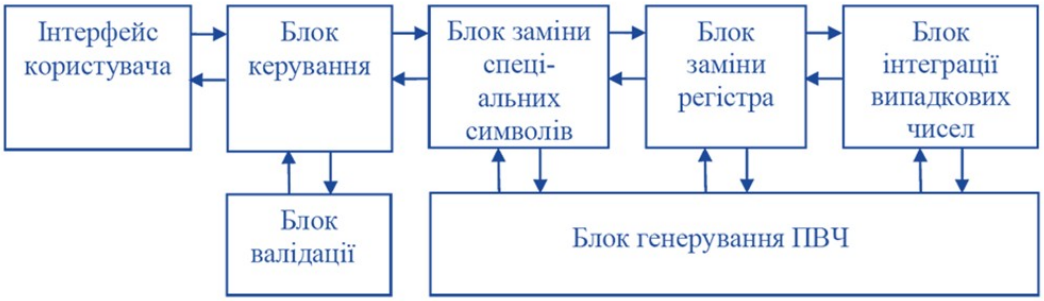
Принцип хешування за допомогою хеш-функцій

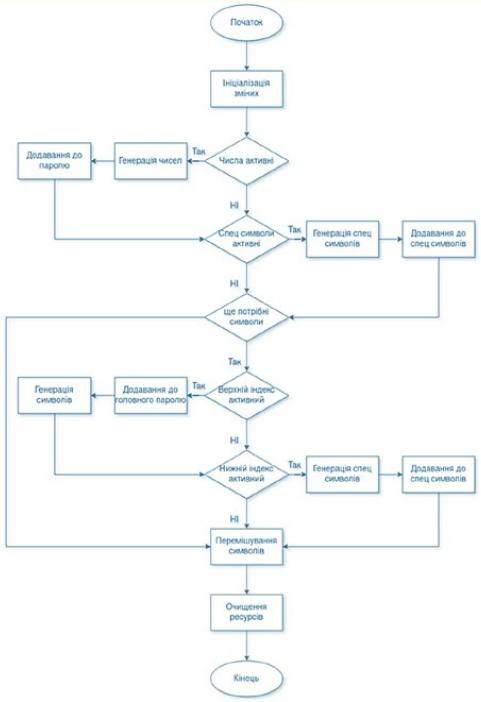


Графічне представлення одної ітерації обробки блоку даних SHA-256



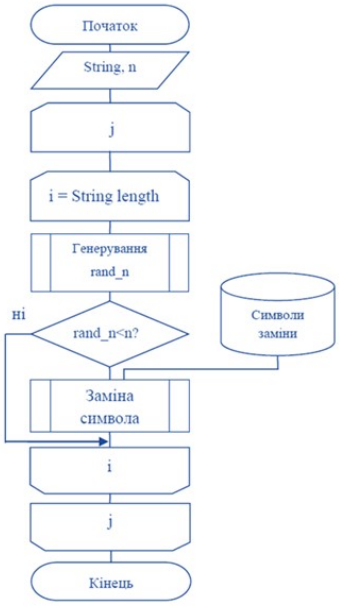
Структурна схема застосунку
для генерування та валідації надійних паролів



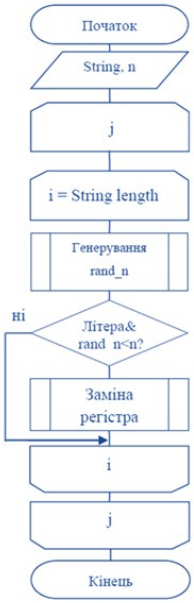


Загальна БСА роботи системи генерації паролів

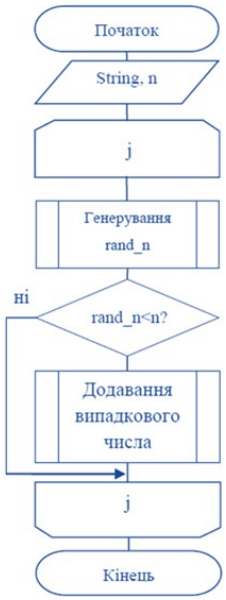
БСА методу заміни літер на спеціальні символи у паролі



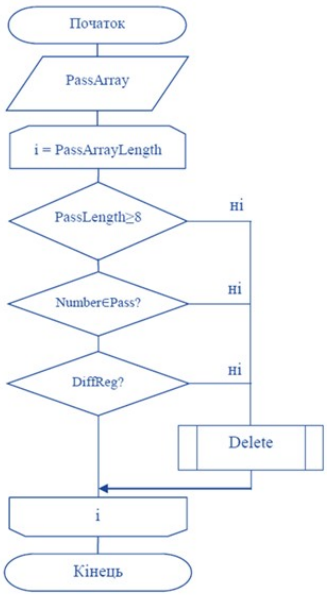
БСА методу заміни регістру літер у паролі



БСА методу вставлення випадкової цифри у паролі

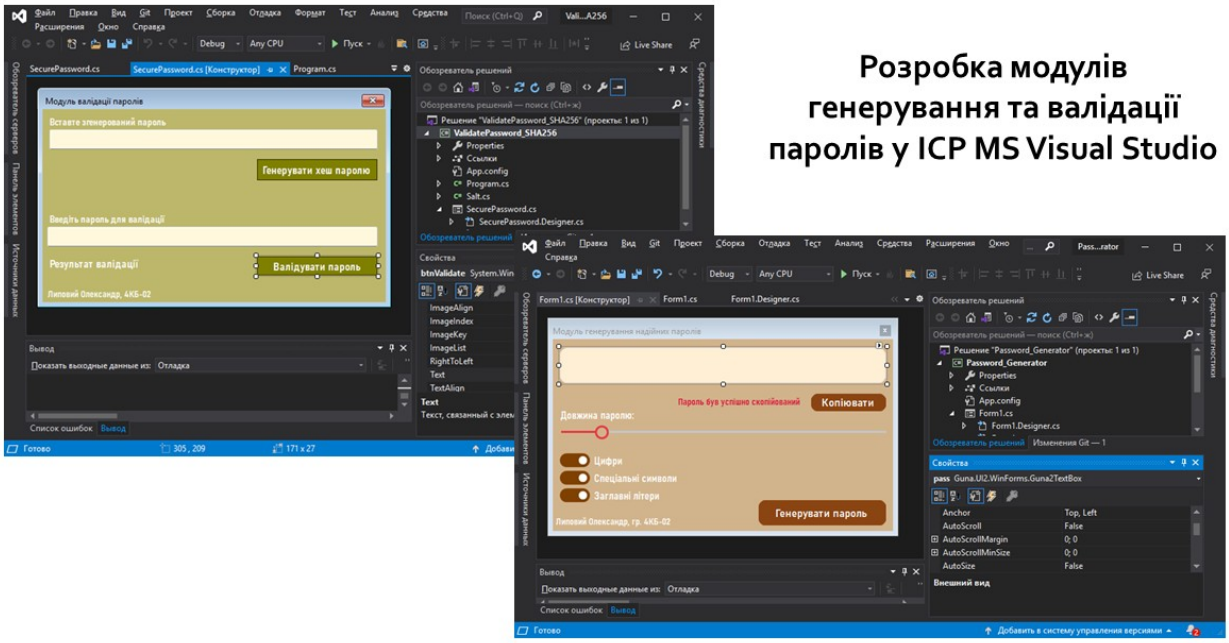


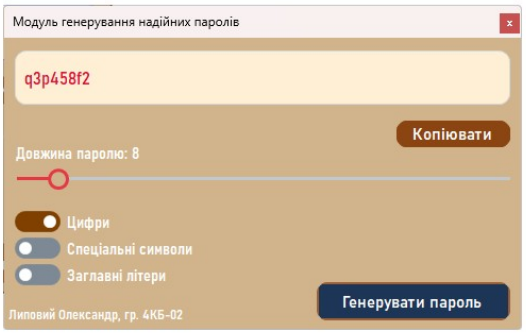
БСА перевірки надійності пароля

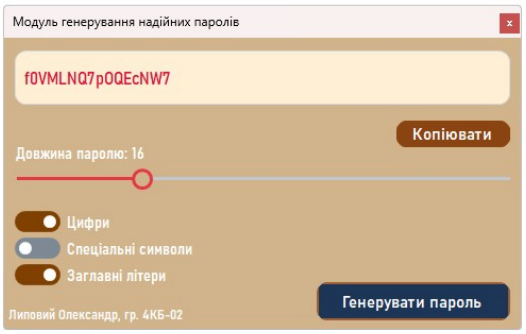


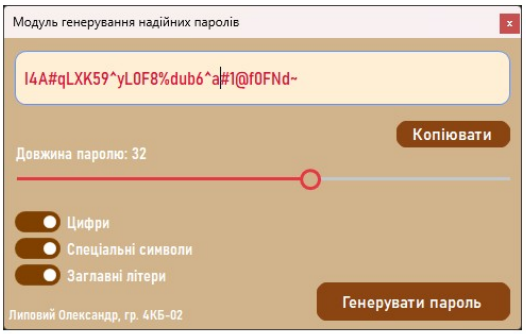
БСА валідації паролю

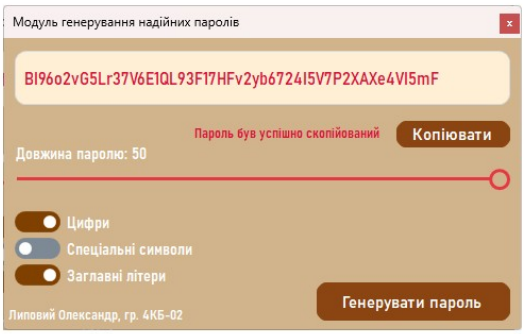
Розробка модулів генерування та валідації паролів у ICP MS Visual Studio

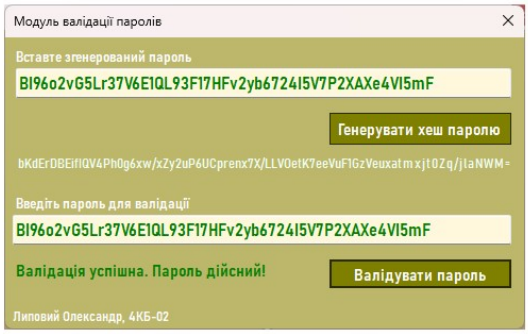


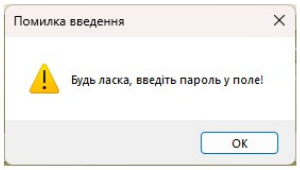

Модуль генерування надійних паролів
q3p458f2
Копіювати
Довжина паролю: 8
Цифри
Спеціальні символи
Заглавні літери
Генерувати пароль
Липовий Олександр, гр. 4КБ-02

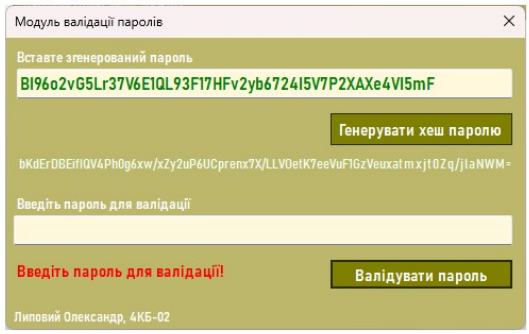

Модуль генерування надійних паролів
f0VMLNQ7p0QEcNW7
Копіювати
Довжина паролю: 16
Цифри
Спеціальні символи
Заглавні літери
Генерувати пароль
Липовий Олександр, гр. 4КБ-02

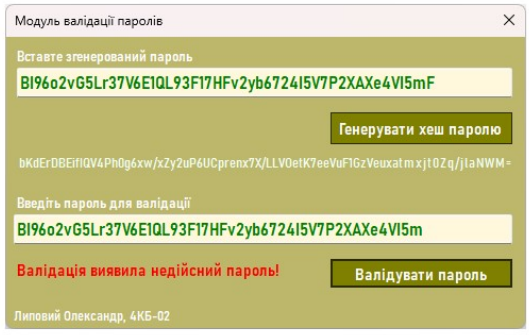

Модуль генерування надійних паролів
I4A#qLXK59^yL0F8%dub6^a#1@f0FNd~
Копіювати
Довжина паролю: 32
Цифри
Спеціальні символи
Заглавні літери
Генерувати пароль
Липовий Олександр, гр. 4КБ-02


Модуль генерування надійних паролів
Bl96o2vG5Lr37V6E1QL93F17HFv2yb6724I5V7P2XAXe4VI5mF
Пароль був успішно скопійований
Копіювати
Довжина паролю: 50
Цифри
Спеціальні символи
Заглавні літери
Генерувати пароль
Липовий Олександр, гр. 4КБ-02


Модуль валідації паролів
Вставте згенерований пароль
Bl96o2vG5Lr37V6E1QL93F17HFv2yb6724I5V7P2XAXe4VI5mF
Генерувати хеш паролю
bKdErDBEifIQV4Ph0g6xw/xZy2uP6UCprenx7X/LLVOetK7eeVuF1GzVeuxatmxjt0Zq/jlaNWM=
Введіть пароль для валідації
Bl96o2vG5Lr37V6E1QL93F17HFv2yb6724I5V7P2XAXe4VI5mF
Валідація успішна. Пароль дійсний!
Валідувати пароль
Липовий Олександр, 4КБ-02


Помилка введення
Будь ласка, введіть пароль у поле!
OK


Модуль валідації паролів
Вставте згенерований пароль
Bl96o2vG5Lr37V6E1QL93F17HFv2yb6724I5V7P2XAXe4VI5mF
Генерувати хеш паролю
bKdErDBEifIQV4Ph0g6xw/xZy2uP6UCprenx7X/LLVOetK7eeVuF1GzVeuxatmxjt0Zq/jlaNWM=
Введіть пароль для валідації
Введіть пароль для валідації!
Валідувати пароль
Липовий Олександр, 4КБ-02


Модуль валідації паролів
Вставте згенерований пароль
Bl96o2vG5Lr37V6E1QL93F17HFv2yb6724I5V7P2XAXe4VI5mF
Генерувати хеш паролю
bKdErDBEifIQV4Ph0g6xw/xZy2uP6UCprenx7X/LLVOetK7eeVuF1GzVeuxatmxjt0Zq/jlaNWM=
Введіть пароль для валідації
Bl96o2vG5Lr37V6E1QL93F17HFv2yb6724I5V7P2XAXe4VI5m
Валідація виявила недійсний пароль!
Валідувати пароль
Липовий Олександр, 4КБ-02

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Липового Олександра Сергійовича

(прізвище, ім`я та по батькові)

Спеціальність *123 «Комп'ютерна інженерія»*

Освітньо-професійна програма *«Безпека комп'ютерних систем і мереж»*

Керівник дипломного проекту (роботи) *Скорняков В'ячеслав Сергійович*

(прізвище, ім`я та по батькові)

Тема дипломного проекту (роботи) *Розробка програмної моделі генерування та валідації надійних паролів*

Обсяг розрахунково-пояснювальної записки *77* сторінок

Обсяг графічної (презентаційної) частини *15* аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) заключення про ступінь відповідності виконаного дипломного проекту завданню

Представлений дипломний проект відповідає затвердженій темі та виконаний відповідно технічному завданню. Дипломний проект присвячений створенню програмної моделі генерування та валідації надійних паролів і складається з пояснювальної записки та мультимедійної презентації з відповідними схемами.

б) характеристика виконання кожного розділу дипломного проекту

Пояснювальна записка складається з основного розділу (Аналітичний огляд засобів генерування надійних паролів; Аналіз криптографічних засобів захисту паролів; Складання математичної моделі; Розробка структури та алгоритмів роботи ПЗ; Реалізація програмної моделі та інтерфейсу застосунку; Тестування програмної моделі генерування та валідації надійних паролів), економічного розділу, розділу охорони праці та додатків. Перелічені розділи поетапно охоплюють розробку, виконані докладно та обґрунтовано.

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту

Графічна частина складається з 15 слайдів мультимедійної презентації, виконаної у програмному продукті MS PowerPoint, які містять структурні та функціональні схеми, діаграми та скріншоти, блок-схеми алгоритмів, передбачені технічним завданням. Пояснювальна записка виконана акуратно та у відповідності до норм. Якість виконання пояснювальної записки відмінна, розробку виконано у повному обсязі.

г) перелік позитивних якостей дипломного проекту *Проблема створення надійних паролів є важливою для сучасної кібербезпеки, особливо в контексті зростаючих загроз, таких як атаки методом перебору та словникові атаки; Використання SHA-256 та BBS забезпечує високий рівень криптографічної стійкості згенерованих паролів; Програмний проект має добре структурований код*

д) основні недоліки дипломного проекту *Бажано було глибше розглянути питання тестування згенерованих паролів щодо їхньої захищеності від реальних атак, зокрема, аналіз ентропії; Відсутність аналізу альтернативних алгоритмів – у проекті розглядається SHA-256, але не аналізується, наприклад, bcrypt або Argon2; Інтерфейс користувача має базову функціональність*

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Добре*

Загальна оцінка *Відмінно*

Прізвище, ім'я, по батькові рецензента *к.т.н. Шибаєва Наталя Олегівна*

Місце роботи і посада рецензента *Національний університет «Одеська політехніка», доцент кафедри інформаційних технологій*

Підпис

«*20*» *червня* *2025 р.*

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на дипломний проект здобувача (здобувачки) освіти
відділення комп'ютерних систем

Липового Олександра Сергійовича

(прізвище, ім`я та по батькові)

Спеціальність: *123 «Комп'ютерна інженерія»*

Освітня програма: *«Безпека комп'ютерних систем і мереж»*

Тема дипломного проекту: *Розробка програмної моделі генерування та валідації надійних паролів*

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ

а) обсяг і якість виконання проекту (графічного матеріалу і розрахунково-пояснювальної записки) *Дипломний проект виконано відповідно технічному завданню. Пояснювальна записка до дипломного проекту містить 77 сторінок. У пояснювальній записці описано створення програмної моделі генерування та валідації надійних паролів за допомогою мови програмування C#. Графічна частина складається з 15 слайдів, оформлених у вигляді презентації, передбачених технічним завданням. Якість виконання пояснювальної записки та слайдів добра.*

б) самостійність роботи над проектом: *Протягом виконання дипломного проекту здобувач освіти Липовий Олександр поступово та послідовно виконував всі етапи, проявив ініціативу в створенні загальної концепції та реалізації роботи. Всі роботи здобувач освіти виконував самостійно, з оглядом на рекомендації керівника.*

в) теоретична підготовка випускника (випускниці): *Здобувач освіти Липовий Олександр під час роботи над дипломним проектом вивчив достатньо багато літературних та інтернет-джерел за даною тематикою.*

Вважаю, що теоретична підготовка дипломника добра і він готовий до захисту проекту.

г) вміння розв'язувати виробничі та конструкторські питання *Під час виконання дипломного проекту здобувач освіти Липовий Олександр показав вміння організовано працювати над поставленим завданням, застосовувати знання у сфері безпеки комп'ютерних систем і мереж, програмування, використовуючи сучасні комп'ютерні програмні засоби розробки, такі як Microsoft Visual Studio.*

Оцінка розрахункової частини *Добре*

Оцінка графічної частини *Добре*

Загальна оцінка *Добре*

Прізвище, ім'я, по батькові керівника дипломного проекту *Скорняков В'ячеслав Сергійович*

Місце роботи і посада керівника дипломного проекту *ВСП «Одеський технічний фаховий коледж ОНТУ», викладач спецдисциплін циклової комісії комп'ютерних технологій та програмної інженерії*

Підпис

«16.» 06 2025 р.

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
(ДИПЛОМНОГО ПРОЕКТУ)
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Липовий О.С.,
здобувач освіти гр. 4КБ-02, та

Скорняков В.С.,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до дипломного проекту фахового молодшого бакалавра на тему:

«Розробка програмної моделі генерування та валідації надійних паролів» (автор роботи – Липовий О.С., керівник роботи – Скорняков В.С.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2025 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи і даємо згоду на обробку персональних даних.

Виконавець _____ / Липовий О.С. /

Керівник _____ / Скорняков В.С. /

«16» червня 2025 р.

ДОВІДКА

циклової комісії КТ та ПІ
про допуск до захисту дипломного проєкту
здобувача (здобувачки) освіти IV курсу
відділення комп'ютерних систем групи 4КБ-02

Липового Олександра Сергійовича

на тему *Розробка програмної моделі генерування та валідації надійних паролів*

Висновок відповідальної особи за проведення нормоконтролю:
пояснювальна записка до дипломного проєкту виконана з некритичними порушеннями ДСТУ та оформлена відповідно до вимог Положення про дипломне проєктування

(підпис) *16.06.2025* (дата) *Петрашова В.І.* (П.І.Б.)

Висновок відповідальної особи за перевірку роботи на наявність академічного плагіату *згідно звіту про перевірку від 28.05.2025 р. значення коефіцієнту подібності в роботі становить 13,36%, коефіцієнт цитування – 0,94%.*

(підпис) *16.06.2025* (дата) *Краснокутська К.Г.* (П.І.Б.)

Попередня експертиза (малий захист) дипломного проєкту

здобувача (здобувачки) освіти *Липового О.С.* (П.І.Б.)

проведена « *16* » *червня* 2025 р.

Висновки *Пояснювальна записка до дипломного проєкту виконана у повному обсязі. Випускна кваліфікаційна робота (дипломний проєкт) відповідає вимогам Положення про дипломне проєктування та рекомендована до захисту.*

Голова ЦК КТ та ПІ (підпис) *Кривченко Ю.В.* (П.І.Б.)

StrikePlagiarism.com

Дата звіту 5/28/2025
Дата редагування 5/28/2025

Документ прийнятий

Звіт подібності

метадані

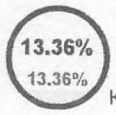
Назва організації
Odesa Technical Professional College of Odesa National University of Technology
Заголовок
Розробка програмної моделі генерування та валідації надійних паролів

Автор
Липовий Олександр Сергійович
Науковий керівник / Експерт
Скорняков В'ячеслав Сергійович

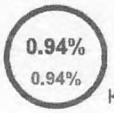
підрозділ
Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету"

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1



КЦ

25
Довжина фрази для коефіцієнта подібності 2

15277
Кількість слів

129694
Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв		30
Інтервали	A→	0
Мікропробіли		0
Білі знаки		0
Парафрази (SmartMarks)	a	72

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Розробка програмної моделі оцінки рівня стійкості комп'ютерних систем і мереж [redacted] 5/25/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	230 1.51 %

2	Розробка програмної моделі оцінки рівня стійкості комп'ютерних систем і мереж [redacted] 5/25/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	207 1.35 %
3	Розробка програмної моделі оцінки рівня стійкості комп'ютерних систем і мереж [redacted] 5/25/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	131 0.86 %
4	https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content	90 0.59 %
5	https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content	82 0.54 %
6	Розробка програмної моделі оцінки рівня стійкості комп'ютерних систем і мереж [redacted] 5/25/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	81 0.53 %
7	https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download	72 0.47 %
8	https://card-file.ontu.edu.ua/server/api/core/bitstreams/995bdcec-4e4d-4321-8070-4d6badcb8e49/content	62 0.41 %
9	https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content	59 0.39 %
10	https://card-file.ontu.edu.ua/bitstreams/5240e379-7721-49f0-8ee8-27140b0b473a/download	55 0.36 %

з домашньої бази даних (5.45 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Розробка програмної моделі оцінки рівня стійкості комп'ютерних систем і мереж [redacted] 5/25/2025 **Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")**	**827 (15) 5.41 %**
2	Розробка цифрового тестеру ємності акумуляторних батарей [redacted] 5/25/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	6 (1) 0.04 %

з програми обміну базами даних (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з Інтернету (7.91 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content	**317 (9) 2.08 %**
2	https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download	**121 (4) 0.79 %**
3	https://card-file.ontu.edu.ua/bitstreams/1dff552d-7200-49b8-ae1d-ba76a1335685/download	**96 (6) 0.63 %**
4	https://card-file.ontu.edu.ua/server/api/core/bitstreams/44c16132-5f53-48e2-b6c0-61e9a2f0fd75/content	**91 (7) 0.60 %**

5	https://card-file.ontu.edu.ua/server/api/core/bitstreams/995bdcec-4e4d-4321-8070-4d6badcb8e49/content	88 (4) 0.58 %
6	https://card-file.ontu.edu.ua/bitstreams/5240e379-7721-49f0-8ee8-27140b0b473a/download	78 (2) 0.51 %
7	https://card-file.ontu.edu.ua/bitstreams/29489599-0581-4ce6-8890-c3b13d9f2e0e/download	51 (3) 0.33 %
8	https://blog.csdn.net/weixin_42098295/article/details/139535516	51 (2) 0.33 %
9	https://card-file.ontu.edu.ua/bitstreams/035f6436-20b4-4ee6-8e99-bede670e308b/download	45 (2) 0.29 %
10	https://card-file.ontu.edu.ua/bitstreams/6cf43324-8f08-4031-ba42-f80b18efbbc8/download	34 (3) 0.22 %
11	https://card-file.ontu.edu.ua/bitstreams/549ee9fe-7574-4ae5-b500-9fe2711f33e6/download	31 (2) 0.20 %
12	https://card-file.ontu.edu.ua/bitstreams/bbaf3f38-16a8-4070-bead-5562769b7c71/download	31 (2) 0.20 %
13	https://card-file.ontu.edu.ua/server/api/core/bitstreams/4bb7255e-46d4-4349-9726-9698476da02d/content	27 (4) 0.18 %
14	https://card-file.ontu.edu.ua/bitstreams/c58b0ff5-46e0-49f8-8cbe-65c32256665d/download	26 (2) 0.17 %
15	https://card-file.ontu.edu.ua/bitstreams/21173711-5b67-4b87-b17f-6302c25e7a31/download	25 (2) 0.16 %
16	https://card-file.ontu.edu.ua/bitstreams/2b75599e-e1ac-412d-bf09-10d2eb49022f/download	21 (1) 0.14 %
17	https://blog.csdn.net/qq_33790894/article/details/138904901	18 (1) 0.12 %
18	https://card-file.ontu.edu.ua/bitstreams/82a6d375-2b69-4233-b80f-fbfd149b7747/download	12 (1) 0.08 %
19	https://card-file.ontu.edu.ua/bitstreams/34a6756b-592f-4b77-a805-183aa03a6a26/download	10 (1) 0.07 %
20	https://card-file.ontu.edu.ua/bitstreams/0e72a3b9-bdd7-4711-a3c6-dedc1d4287cc/download	10 (1) 0.07 %
21	https://card-file.ontu.edu.ua/bitstreams/e4afae26-0a7e-4a4d-afc2-94341838de2a/download	10 (1) 0.07 %
22	https://card-file.ontu.edu.ua/bitstreams/bbed74c8-2ea7-44c5-8d00-0fe3fd9790ee/download	9 (1) 0.06 %
23	https://card-file.ontu.edu.ua/server/api/core/bitstreams/a141b658-5fa7-4f90-b0bd-7f0ccaed21e5/content	6 (1) 0.04 %

Список прийнятих фрагментів (немає прийнятих фрагментів)

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»_
Освітньо-професійна програма: **«Безпека комп'ютерних систем і мереж» Група:** 4КБ- 02

Дипломний **проект здобувача освіти денної форми навчання КБ.** 02.11.000.ДП

ЛИПОВОГО
ОЛЕКСАНДРА СЕРГІЙОВИЧА м. Одеса_
2025 р. **МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»**

Спеціальність: 123 «Комп'ютерна інженерія»
Освітньо-професійна програма: **«Безпека комп'ютерних систем і мереж»**
Група: 4 КБ-02

ПОЯСНЮВАЛЬНА ЗАПИСКА до дипломного проекту на тему:

Проектний матеріал складається з