Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ПОЯСНЮВАЛЬНА ЗАПИСКА

Дипломної роботи
магістра
(назва освітньо-кваліфікаційного рівня)

галузь знань *12 Інформацімайні технології*
(шифр і назва галузі знань)
спеціальність *125 Кібербезпека*
(код і назва спеціальності)
освітній ступень *магістр*
(назва освітньої програми)
освітньо-наукова програма *кібербезпека*

на тему: «Метод генерації паролів на основі псевдовипадкових послідовностей»

Виконавець: студент ІІ курсу, групи КБм-21

(підпис) **П’ятигор Віталій Петрович** (прізвище ім’я по-батькові)

	Прізвище, ініціали	Оцінка	Підпис
Науковий керівник	Бучик С. С.		
Рецензент	Самохвалов Ю. Я.		
Нормоконтроль	Даков С.Ю.		

Київ 2022

Міністерство освіти і науки України

Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
завідувач кафедри кібербезпеки
та захисту інформації
_____ Н.В. Лукова-Чуйко
«__» _____ 2021 р.

ЗАВДАННЯ
на виконання дипломної роботи

спеціальності *125 Кібербезпека*
(код і назва спеціальності)

студенту КБм-21 (група) П’ятигору Віталію Петровичу (прізвище ім’я по-батькові)

Тема дипломної роботи Метод генерації паролів на основі псевдовипадкових послідовностей

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Рішення засідання кафедри кібербезпеки та захисту інформації факультету інформаційних технологій протокол № 5 від 29.10.2021

2. МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об’єкт досліджень	процес генерації паролів на основі псевдовипадкових послідовностей.
Предмет досліджень	алгоритми генерації паролів на основі псевдовипадкових послідовностей, складність згенерованих ними паролів
Мета	модифікація алгоритму генерації паролів Diceware на основі псевдовипадкових послідовностей.
Вихідні дані для проведення роботи	алгоритми генерації паролів на основі псевдовипадкових послідовностей.

3. ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Наукова новизна удосконалено алгоритм генерації паролів Diceware на основі псевдовипадкових послідовностей, що дозволило зробити алгоритм більш стійким до атак ніж стандартна реалізація і підвищити значення ентропії на символ на 1,3 біта.

Практична цінність покращення алгоритму створення користувацьких паролів для використання у системах автентифікації

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Робота виконана у повному обсязі відповідно до теми.

5. ЕТАПИ ВИКОНАННЯ РОБОТИ

Найменування етапів робіт	**Строки виконання робіт (початок-кінець)**
Розробка плану для досягнення мети роботи	29.10.2021 – 23.01.2022
Аналіз літературних джерел	24.01.2022 – 14.02.2022
Розробка та аналіз алгоритму генерації паролів на основі псевдовипадкових послідовностей	15.02.2022 – 24.04.2022
Оформлення і друк пояснювальної записки	25.04.2022 – 19.05.2022

6. РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект Зниження збитків через викрадення даних атаками підбору паролів.

Соціальний ефект Покращення захисту даних користувачів у системах, що базуються на парольній аутентифікації.

7. ДОДАТКОВІ ВИМОГИ

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

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

Дата видачі завдання: _____
Термін подання дипломної роботи до ЕК _____

УДК 004.056.53

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Метод генерації паролів на основі псевдовипадкових послідовностей»: 66 сторінок, 8 рисунків, 10 таблиць, 7 лістингів коду, 45 літературних джерел та 1 додаток.

Мета роботи – модифікація алгоритму генерації паролів Diceware на основі псевдовипадкових послідовностей.

Об’єкт дослідження – процес генерації паролів на основі псевдовипадкових послідовностей.

Предмет дослідження – алгоритми генерації паролів на основі псевдовипадкових послідовностей, складність згенерованих ними паролів.

Наукова новизна: удосконалено алгоритм генерації паролів Diceware на основі псевдовипадкових послідовностей, що дозволило зробити алгоритм більш стійким до атак ніж стандартна реалізація і підвищити значення ентропії на символ на 1.4 біта.

У роботі розглянуто основні алгоритми генерації паролів, які існують на ринку. Здійснено аналіз їх переваг та недоліків. Оцінено складність згенерованих паролів на основі кінцевої ентропії та складності запам’ятовування. Розглянуто міжнародні вимоги до генерації паролів наведені у документі NIST.SP.800-63b та відповідність отриманих в паролів наведеним у документі вимогам. Оцінено стійкість паролю до атак грубої сили та атак за словником.

Також розроблено програмний застосунок, який реалізовує генерації за допомогою модифікованого алгоритму генерації паролів Diceware на основі заданої довжини та заданого набору слів та оцінює ентропію згенерованих паролів. Набір слів записаний в файлі. За замовчуванням використовується англійський набір слів. Основним завданням даної програми є генерація набору паролів для використання користувачами. Програма може працювати автономно без мережевого з’єднання. Пропонується використовувати модифікований алгоритм генерації паролів Diceware заміть стандартного через більшу ентропію першого.

Актуальність роботи: найпоширенішим типом автентифікації користувача є парольна автентифікація, проте користувачі схильні до створення слабких легкозламуваних паролів. Менеджери паролів дозволяють генерувати паролі, які є стійкими, проте неможливі до запам'ятовування, тому змушують користувачів використовувати їх на всіх сайтах та власних пристроях. Стандартний алгоритм генерації паролів Diceware є вразливим до атак словником та не відповідає більшості вимог, який вимагають сервіси автентифікації, а саме наявність цифри, спеціального символу, великої літери, тощо. Модифікований алгоритм Diceware дозволяє поєднати простоту запам’ятовування паролів оригінальної реалізації, та складністю до взлому паролів, які згенеровані менеджерами паролів.

Ключові слова: парольна автентифікація, пароль, генерація паролів, Diceware, ентропія, оцінка стійкості паролів.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

ІТ - Інформаційні технології;

ПІН - Персональний ідентифікаційний номер;

ПК - Персональний комп’ютер;

США - Сполучені штати Америки;

NIST - National Institute of Standards and Technology;

SP - Special Publication;

AES PBKDF2 SHA256 - Advanced Encryption Standard based on Password-Based Key Derivation Function with Secure Hash Algorithm Version 2;

ANSI - American National Standards Institute;

ASCII - American standard code for information interchange;

DES - Data Encryption Standard;

GPGPU - General-purpose computing on graphics processing units;

PIN - Personal Identification Number;

RFC - Request for Comments;

UTF - Unicode Transformation Format;

WEP - Wired Equivalent Privacy;

WPA PSK - Wi-Fi Protected Access Pre-Shared Key.

ВСТУП

В сучасних реаліях питанню захисту інформації від несанкціонованого доступу приділяється багато уваги. Одним із сервісів безпеки, які і забезпечують такий захист є сервіс автентифікації. Наразі в кожній системі використовується якийсь тип автентифікації користувача, найчастіше за все – парольна.

Проте без достатньої уваги до створюваних користувачами паролів, така система є достатньо вразливої для атак хакерів. Користувачі системи не створюють легкозламні паролі, такі як passw0rd, admin і т.ін. які можуть бути зламані за секунди. Тому для покращення стійкості паролів до зламу використовуються генератори паролів на основі випадкових або псевдовипадкових генераторів послідовності чисел, які можуть надавати користувачам складний пароль, який і буде використовуватися в системі для підтвердження користувача.

Без використання програмних засобів в системі може бути введено додаткові обмеження на довжину, символи та вміст самого паролю, проте дослідження [1][2] показують, що накладання занадто складних правил на паролі значно підвищують шанс користувачів забути або неправильно ввести пароль. Така ж ситуація з використанням паролів, які були згенеровані алгоритмами генерації паролів, які орієнтовані на складність і використання всіх символів у випадковій послідовності, а не на користувача.

Тому для одночасного покращення складності паролю і легкості його запам’ятовування рекомендується використовувати алгоритм генерації паролів на основі випадковості Diceware. Інакше, рекомендовано використовувати централізоване сховище ключів, де і будуть зберігатися паролі користувачів, проте це не тема даної роботи.

Мета роботи – модифікація алгоритму генерації паролів Diceware на основі псевдовипадкових послідовностей.

Об’єкт дослідження – процес генерації паролів на основі псевдовипадкових послідовностей.

Предмет дослідження – алгоритми генерації паролів на основі псевдовипадкових послідовностей, складність згенерованих ними паролів.

Наукова новизна: удосконалено алгоритм генерації паролів Diceware на основі псевдовипадкових послідовностей, що дозволило зробити алгоритм більш стійким до атак ніж стандартна реалізація і підвищити значення ентропії на символ на 1,3 біта.

У роботі розглянуто основні алгоритми генерації паролів, які існують на ринку. Здійснено аналіз їх переваг та недоліків. Оцінено складність згенерованих паролів на основі кінцевої ентропії та складності запам’ятовування. Розглянуто міжнародні вимоги до генерації паролів наведені у документі NIST SP.800-63b та відповідність отриманих в паролів наведеним у документі вимогам. Оцінено стійкість паролю до атак грубої сили та атак за словником.

Також розроблено програмний застосунок, який реалізовує генерації за допомогою модифікованого алгоритму генерації паролів Diceware на основі заданої довжини та заданого набору слів та оцінює ентропію згенерованих паролів. Набір слів записаний в файлі. За замовчуванням використовується англійський набір слів. Основним завданням даної програми є генерація набору паролів для використання користувачами. Програма може працювати автономно без мережевого з’єднання. Пропонується використовувати модифікований алгоритм генерації паролів Diceware заміть стандартного через більшу ентропію першого.

Актуальність роботи: найпоширенішим типом автентифікації користувача є парольна автентифікація, проте користувачі схильні до створення слабких легкозламуваних паролів. Менеджери паролів дозволяють генерувати паролі, які є стійкими, проте неможливі до запам'ятовування, тому змушують користувачів використовувати їх на всіх сайтах та власних пристроях. Стандартний алгоритм генерації паролів Diceware є вразливим до атак словником та не відповідає більшості вимог, який вимагають сервіси автентифікації, а саме наявність цифри, спеціального символу, великої літери, тощо. Модифікований алгоритм Diceware дозволяє поєднати простоту запам’ятовування паролів оригінальної реалізації, та складністю до взламу паролів, які згенеровані менеджерами паролів.

Апробація результатів роботи:

- VIII International conference “Information Technology and Implementation”, грудень, 2021 [3].
- Робота зайняла 29 місце у рейтингу Студентських наукових робіт Всеукраїнського конкурсу зі спеціальності «Кібербезпека» у 2020-2021 н.р.
- Подано до видання у Workshop on Cybersecurity Providing in Information and Telecommunication Systems [4].

РОЗДІЛ І
ЗАГАЛЬНИЙ ОГЛЯД АЛОРИТМІВ ГЕНЕРАЦІЇ ПАРОЛІВ НА ОСНОВІ ПСЕВДОВИПАДКОВОЇ ПОСЛІДОВНОСТІ ЧИСЕЛ

1.1.Сервіси та методи авторизації

Відповідно до Рекомендації X.800, автентифікація -, процедура встановлення належності користувачеві інформації в системі за допомогою пред'явленого ним ідентифікатора. З позицій інформаційної безпеки автентифікація є частиною процедури надання доступу для роботи в інформаційній системі, наступною після ідентифікації і передує авторизації [5].

Дослідження в сфері безпеки визначили, що для позитивної автентифікації слід перевірити елементи щонайменше з двох, а краще всіх трьох, факторів [6]. До цих трьох факторів належать [5]:

- фактор знання: те, що користувач знає. Як приклад, таким знанням може бути пароль, частковий пароль, фраза проходу або персональний ідентифікаційний номер (PIN), відповідь на запитання чи шаблон;
- фактор володіння: те, чим користувач володіє. Наприклад, ідентифікаційна картка, апаратний токен, мобільний телефон із вбудованим апаратним токеном, програмним токеном або мобільним телефоном, що містить програмне забезпечення, що генерує токен;
- фактор невід’ємності: те, яким користувач є. Тобто біометричні параметри такі як відбиток пальців, малюнок сітківки, послідовність ДНК, форма обличчя, голос, унікальні біоелектричні сигнали чи інший біометричний ідентифікатор.

Сервіси автентифікації забезпечують автентифікацію отримувача та джерела даних, як описано нижче [5]:

- Автентифікація отримувача. Ця послуга, коли надається (N) - шар, забезпечує підтвердження (N+1) - отримувача, що отримувач є заявленою (N+1) – сутністю;
- Автентифікація джерела походження даних. Ця послуга, коли надається (N) - шар, забезпечує підтвердження (N+1) - отримувача, що джерелом даних є заявлена (N+1) – сутність;

Сервіс управління доступом забезпечує захист від несанкціонованого використання ресурсів, доступних через OSI. Це можуть бути як ресурси OSI так і не OSI, до яких можна отримати доступ через протоколи OSI. Ця служба захисту може застосовуватися до різних типів доступу до ресурсу (наприклад, використання ресурсу зв'язку; читання, записування чи видалення інформаційного ресурсу; виконання ресурсу обробки) або до всіх доступів до ресурсу.

Найпростішим та найпопулярнішим способом автентифікації в веб-застосунках є парольна автентифікація. Така схема авторизації передбачає наявність у користувача ідентифікатора за фактором знання, наприклад пароль, частина паролю або ПІН. Користувач надсилає запит на авторизацію який містить пару логін – пароль. Паролем можуть виступати будь-яке відоме лише санкціонованому користувачу значення. Логіном має бути щось, що унікально ідентифікує користувача у системі, тобто в системі не може бути два однакових логіна. Прикладами логіна можуть виступати номер мобільного телефону, електронна пошта, унікальний набір символів, порядковий номер та інше.

Проблемою такого підходу є те, що створення паролю покладається на користувача і за результатами статистики взламаних паролей, більшість з них є слабкими паролями, які легко підбираються, містять персональну інформацію, що може біти зібрана з відкритих джерел або є дуже поширеними [7]. Для вирішення цієї проблеми все більше набирають популярність менеджери паролів [8], які надають можливість генерації стійких паролів, їх збереження та автозаповнення на сайтах. Проблема менеджерів в тому, що створювані ними паролі є випадковими наборами з 12-16 символів, які майже неможливі до запам’ятовування, тому такі

паролі прив’язують користувача до використовування цього менеджера на всіх девайсах. Також, такі менеджери не використовувують при вході до системи або мережі, яка створена на базі, наприклад, Active Directory.

1.2.Генератори паролів на основі псевдовипадкових послідовностей

Паролі, які використовуються в системах авторизації, необхідно їх якось створювати. Можна створювати їх власноруч, використовуючі свої методи, проте як було описано вище більшість таких паролів є слабкими. Тому для створення паролів існують генератори паролів, які можуть існувати самостійно або у складі менеджеру паролів.

Генератор випадкових паролів – це програмне забезпечення або апаратний пристрій, який приймає дані від генератора випадкових чи псевдовипадкових чисел і автоматично генерує пароль. Випадкові паролі можна генерувати вручну, використовуючи прості джерела випадковості, такі як кістки або монети, або їх можна створити за допомогою комп’ютера.

Псевдовипадкова послідовність – це послідовність чисел, яка була обчислена по деякому певному арифметичному правилу, але має всі властивості випадкової послідовності чисел в рамках розв'язуваної задачі.

Хоча псевдовипадкова послідовність на перший погляд позбавлена закономірностей, проте будь-який псевдовипадковий генератор з кінцевим числом внутрішніх станів повториться після дуже довгої послідовності чисел. Це може бути доведено за допомогою принципу Діріхле.

Для генерації випадкових чисел, які ніколи не може передбачити жоден спостерігач, потрібен причинно-недетермінований процес, коли події повністю не визначаються попередніми станами (наприклад, випускається фотон атомом у будь-якій заданій наносекунді). Через фізичну неможливість отримати достатню інформацію для прогнозування результату такої події, її результати гарантуються випадковим для всіх.

Хоча є багато прикладів «випадкових» програм генератора паролів, доступних в Інтернеті, генерування випадкових випадків може бути складним, і багато програм не генерують випадкові символи таким чином, щоб забезпечити високу безпеку. Поширена рекомендація – використовувати інструменти безпеки з відкритим кодом, де це можливо, оскільки вони дозволяють незалежно перевірити якість методів, які використовуються.

Слід зауважити, що просто згенерувати пароль випадковим чином не є гарантією, що пароль є надійним, оскільки можливо, хоча і дуже малоймовірно, що створений пароль буде легко зламним.

Генератори випадкових паролів зазвичай виводять рядок символів заданої довжини. Це можуть бути окремі символи з деякого набору символів або слова з якогось списку слів для формування парольної фрази. Програму можна налаштувати так, щоб отриманий пароль відповідав локальній політиці щодо паролів, скажімо, завжди створюючи суміш літер, цифр та спеціальних символів. Такі політики, як правило, трохи знижують стійкість паролю, формула для обрахування якої наведена нижче, оскільки символи вже не створюються незалежно.

Стійкість випадкового пароля проти конкретної атаки (атака грубої сили) можна обчислити, обчисливши інформаційну ентропію випадкового процесу, який згенерував пароль. Якщо кожен символ у паролі створюється незалежно та з рівномірною ймовірністю, ентропія у бітах задається наступною формулою [9]:

$$H = L \cdot log_2 N = L \cdot \frac{\log N}{\log 2} \qquad (1.1)$$

де N – кількість можливих символів, L – кількість символів у паролі.

Використовуючи формулу 1.1 в табл. 1.1 надано порівняння ентропії на символ для різних наборів символів, які використовуються в генераторах паролей.

Таблиця 1.1

Ентропія на символ для різних наборів символів

Набір символів	**Кількість символів N**	**Ентропія на символ H, біт**
Арабські цифри (0–9) (наприклад, PIN-код)	10	3.32
Шістнадцяткові цифри (0–9, A–F) (наприклад, ключ WEP)	16	4.00
Латинський алфавіт без урахування регістру (a–z або A–Z)	26	4.70
Алфавітно-цифровий регістр без урахування (a–z або A–Z, 0–9)	36	5.17
Латинський алфавіт з чутливістю до регістру (a–z, A–Z)	52	5.70
Буквенно-цифровий регістр (a–z, A–Z, 0–9)	62	5.95
Усі символи ASCII для друку	94	6.55
Список слів Diceware	7776	12.9

Будь-який генератор паролів обмежений простором станів генератора псевдовипадкових чисел, який використовується. Таким чином, пароль, згенерований за допомогою 32-бітового генератора псевдовипадкових чисел, обмежений 32-бітною ентропією, незалежно від кількості символів, які містить пароль.

В Інтернеті доступна велика кількість програм і веб-сайтів для генерування паролів. Їх якість варіюється, і важко оцінити, якщо немає чіткого опису джерела

випадковості, яке використовується, і якщо вихідний код не надано для перевірки. Крім того, і, мабуть, найважливіше, передача паролів через Інтернет викликає очевидні проблеми з безпекою, особливо якщо з'єднання з програмою сайту генерації паролів не захищене належним чином або якщо сайт якимось чином скомпрометований. Без безпечного каналу неможливо запобігти прослуховуванню, особливо через загальнодоступні мережі, такі як Інтернет. Можливим рішенням цієї проблеми є створення пароля за допомогою мови програмування на стороні клієнта, наприклад JavaScript. Перевага цього підходу полягає в тому, що згенерований пароль залишається на клієнтському комп'ютері і не передається на зовнішній сервер або з нього.

1.3.Алгоритми генерації паролів

Як приклади алгоритмів, які використовуються для генерації паролів, за допомогою генераторів паролю, було обрано декілька варіацій.

Механічний генератор симулює фізичний генератор паролів у програмному середовищі. Фізично цей метод можна описати як підкидання 2 шестигранних кубиків для визначення індексу символу у матриці символів. Додатково можна використати підкидання монети, якщо набору з 36 символів недостатньо.

Програмно цей метод реалізований за допомогою генератора псевдовипадкових чисел ANSI X9.17 та використовує 72 символи, які включають в себе англійські літери нижнього та верхнього регістру, цифри та деякі символи. Генерація здійснюється отриманням трьох псевдовипадкових чисел за допомогою генератора, які використовуються як індекси в трьохвимірному масиві для доступу до необхідного символу. Цей масив символів надано у табл. 1.2. Перші два згенеровані значення зазначають позицію символу у двохвимірній матриці, останній набуває значення 1 або 0 і симулює роботу клавіші Shift на клавіатурі: буквений алфавіт замінюється відповідної літерою верхнього регістру, числове значення замінюється спеціальним символом, наприклад 4 замінюється на $. Операція повторюється поки не буде досягнена бажана довжина пароля.

Як приклад комерційного продукту для генерації паролів був вибраний алгоритм з використанням AES-256 PBKDF2-SHA256, який надано програмним застосунком LastPass [10]. Цей застосунок є комерційним, тому вся реалізація даного генератора є закритою, так що інформація про використані засоби генерації є інформацією взятою з опису продукту та документації. Також зазначається, що дані шифруються та розшифровуються на рівні пристрою. Проте наявність можливості синхронізації паролів на різних пристроях показують, що паролі користувачів зберігаються на серверах цього застосунку. Таким чином користувачі мають довіряти рівню захисту цього застосунку, а також передачі даних. Іронічно, проте для використання цього застосунку генератора паролю, необхідно створити пароль.

Таблиця 1.2

Матриця символів, які застосовується в механічному алгоритмі генерації паролів

	1	**2**	**3**	**4**	**5**	**6**
1	a	b	c	d	e	f
2	g	h	i	j	k	l
3	m	n	o	p	q	r
4	s	t	u	v	w	x
5	y	z	0	1	2	3
6	4	5	6	7	8	9

Основним фокусом роботи є алгоритм Diceware. Це метод створення парольних фраз, паролів та інших криптографічних змінних, використовуючи звичайні кістки як апаратний генератор випадкових чисел [11]. На кожне слово в парольній фразі потрібно п’ять підкидань кубиків. Числа від 1 до 6, отримані в результаті, збираються у вигляді п’ятизначного числа, наприклад 43146. Це число потім використовується для пошуку слова в списку слів. Генеруючи кілька слів послідовно, можна побудувати довгий, простий для запам’ятовування та складний для підбору пароль.

Список слів Diceware — це будь-який список із $6^5 = 7776$ унікальних слів, бажано тих, які користувач легко напише й запам’ятає. Вміст списку слів не потрібно захищати чи будь-яким чином приховувати, оскільки безпека кодової фрази Diceware полягає у кількості вибраних слів і кількості слів, з яких можна взяти кожне вибране слово. Офіційні списки були складені для кількох мов, зокрема баскської, болгарської, каталонської, китайської, чеської, датської, голландської, англійської, есперанто, естонської, фінської, французької, німецької, грецької, івриту, угорської, італійської, японської, латинської, маорі, норвезької, польської, португальської, румунської, російської, словацької, словенської, іспанської, шведські та турецької.

Ентропія для цього алгоритму, як уже було зазначено в таблиці 1.1, становить 12.9 біт. Спочатку, у 1995 році, творець Diceware Арнольд Рейнхольд вважав, що п’ять слів (64,6 біт) мінімальна довжина, необхідна звичайним користувачам. Однак у 2014 році Рейнхольд почав рекомендувати використовувати щонайменше шість слів (77,5 біт) [12].

Цей рівень непередбачуваності передбачає, що потенційний зловмисник знає три речі: що Diceware було використано для створення парольної фрази, конкретний список слів, який використовується, і скільки саме слів складає парольну фразу. Якщо зловмисник має менше інформації, ентропія може перевищувати 12,9 біт на слово.

Наведені вище обчислення ентропії алгоритму Diceware припускають, що, як рекомендовано автором Diceware, кожне слово розділене пробілом. Якщо замість цього слова просто об’єднати, розрахована ентропія трохи зменшиться через надмірність; наприклад, фрази Diceware із трьох слів «in put clammy» та «input clam my» стають ідентичними, якщо видалити пробіли.

Арнольд Рейнхолд, винахідник алгоритму рекомендує використовувати тільки засоби «справжньої» випадковості і не використовувати засоби для програмної генерації паролів таким чином через використання псевдовипадковості у таких

програмах, тобто в генерації паролів є залежність. Проте з використанням сучасних засобів генерації псевдовипадкових чисел такі залежності дуже важко прослідити.

1.4.Вимоги до паролів

В Україні немає нормативно-правової бази, яка б регулювала б вимоги до застосовуваних паролів, тому як показник гарного паролю будемо використовувати вимоги наведені і стандарті NIST.SP.800-63b [13], який регулює властивості, які необхідно мати сертифікованому паролю.

Для даної роботи інтерес цікавить розділ цього документу призначений для паролів, які запам’ятовуються. До них висуваються наступні вимоги:

- довжина від 8 до 64 символів;
- мають бути доступним всі символи ASCII та UNICODE;
- для генерованих паролів мають бути використані затверджені генератори випадкових бітів зазначені у нормативному документі NIST.SP.800-90Ar1 [14];
- паролі не мають бути раніше компрометованими, в базі легких паролів, залежними від контексту та мати повторювані символи, а також слова зі словника.

Для порівняння, також розглянемо вимоги до паролів для користувачів у різних IT компаніях. Microsoft надає наступні вимоги щодо гарних паролів [15]:

- Мають бути складними для відгадування чи зламування.
- Не менше восьми символів.
- Не містить ім’я користувача, справжнє ім’я або назву компанії.
- Не містить цілих слів.
- Значно відрізняється від попередніх паролів.
- Містить великі та маленькі букви, цифри та символи.

Google надає наступні вимоги до паролів. Пароль може складатися з будь-якої комбінації літер, цифр і символів ASCII. Діакритичні знаки й символи з діакритиками не підтримуються. Не можна використовувати паролі, які [16]:

- Легко вгадати, наприклад, "пароль123".

- Використовувалися раніше для облікового запису.
- Починаються або закінчуються пробілом.

Також, компанією Google надається список рекомендацій щодо створення надійного пароля [16]:

- Надійний пароль має бути достатньо легким для вас, щоб його запам’ятати, але складним для інших, щоб ніхто не міг його вгадати чи підібрати.
- Створіть унікальний пароль

Використовуйте різні паролі для всіх важливих облікових записів, як-от електронної пошти й інтернет-банкінгу. Використовувати ті самі паролі небезпечно. Якщо хтось дізнається пароль для одного облікового запису, то зможе отримати доступ до вашої електронної пошти, адреси та навіть грошей.

- Створіть довгий пароль, який легко запам’ятати

Довгі паролі надійніші, тому ваш має складатися принаймні з 12 символів. Ці поради допоможуть створювати довгі паролі, які легко запам’ятати. Ось що можна спробувати:

- текст із пісні чи вірша;
- цитату з фільму чи промови;
- уривок із книги;
- набір слів, які асоціюються у вас із чимось;
- абревіатуру, наприклад, з перших літер кожного слова в реченні.

Уникайте паролів, які можуть легко вгадати ваші знайомі або люди з доступом до ваших відкритих даних, як-от профілю в соціальній мережі.

- Не використовуйте особисту інформацію й поширені слова

Не використовуйте особисту інформацію. Під час створення паролів не використовуйте інформацію, яку інші можуть знати або дізнатися. Наприклад:

- ваш псевдонім або ініціали;
- ім’я дитини або домашнього улюбленця;
- важливі дні народження чи роки;
- назва вулиці;

- цифри з адреси.

Не використовуйте поширені слова та комбінації. Уникайте простих слів, фраз і комбінацій, які легко вгадати. Наприклад:

- очевидні слова та фрази, як-от "пароль" або "дозволити_вхід";
- послідовності, як-от "abcd" або "1234";
- загальновідомі комбінації клавіш на клавіатурі, як-от "qwerty" або "qazwsx".

Як видно, багато у чому вимоги та рекомендації схожі між собою, проте все ще немає уніфікованого списку вимог до паролів. І кожен з веб-ресурсів вимагає свій набір вимог.

Висновок до першого розділу

В розділі було розглянуто основні принципи, на яких базується дана робота. Спочатку був розглянутий сервіс автентифікації, що є одним із сервісів веб-сервісів, який надає чи відмовляє користувачам у доступі до ресурсів сервісу. Базуючись на ньому було виділено парольну автентифікацію – автентифікацію на базі фактора знання користувача, що є найпоширенішим і найпростішим видом сервісу автентифікації.

В розділі був визначено, що вразливістю парольної автентифікації є слабкі паролі, які створюють користувачі. Для вирішення цього було описано генератори паролів, які дозволяють програмно або апаратно створювати надійні паролі. Також було описано як можна оцінити надійність паролів, а також основні терміни, на яких працюють генератори паролів, а саме поняття псевдовипадковості та пов’язаниї термінів.

Після розгляду термінів, розглянуто практичне застосування генераторів паролів – різні алгоритми генерації паролів. Описано механічний, AES-256 PBKDF2-SHA256 та Diceware алгоритми, надано рекомендації щодо їх застосування та порівняно набори слів, які використовуються.

Єдиної нормативно-правової бази паролів не існує, тому було надо декілька документів, що описують вимоги до паролів від NIST та інших IT компанії, а також рекомендації, щодо їх створення.

РОЗДІЛ ІІ
МОДИФІКАЦІЯ АЛГОРИТМУ ГЕНЕРАЦІЇ ПАРОЛІВ DICEWARE НА ОСНОВІ ПСЕВДОВИПАДКОВОЇ ПОСЛІДОВНОСТІ

2.1.Модифікований алгоритм

Для вирішення проблем алгоритму генерації паролів Diceware була створена модифікація, яка ускладнює паролі, які генеруються. Блок-схема алгоритму генерації одного слова паролю модифікованого алгоритму зображена на рис. 2.1.

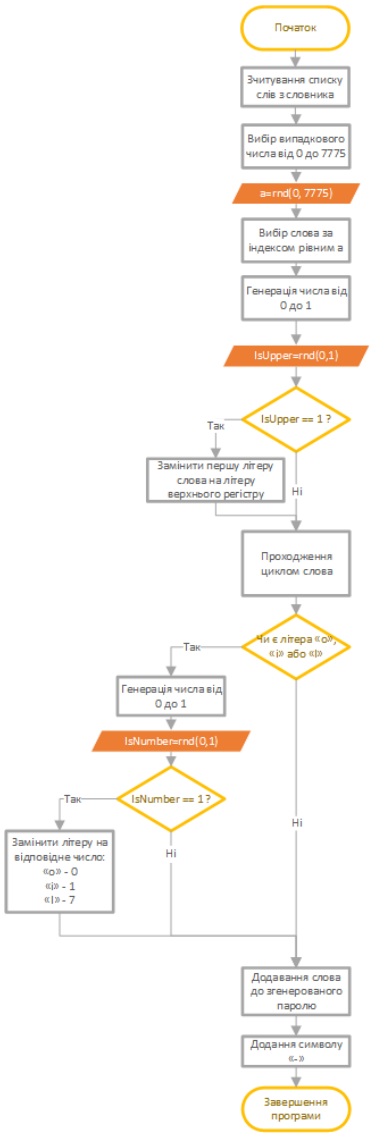

Рисунок 2.1 – Блок-схема алгоритму генерації одного слова паролю модифікованого алгоритму Diceware

Початковий етап модифікованого алгоритму Diceware складається з зчитування набору слів з файлу, за замовчанням – англійського набору. Наступним етапом є генерація слова. Спочатку генерується число від 0 до 7775, вибирається слово з набору слів, індекс якого відповідає згенерованому числу. Потім генерується число від 0 до 1, яке визначає чи матиме слово велику літеру чи ні. Наступним кроком є перебір решти символів. Якщо символ є «o», «i» або «l», то відбувається генерація числа від 0 до 1, яке визначає чи відбудеться заміна символу на «0», «1» або «7» відповідно чи ні. Процес повторюється поки не буде досягнуто бажаної заданої довжини паролю. В даному випадку довжина паролю визначається не кількістю символів, а кількістю згенерованих слів у отриманому паролі. Кінцевим етапом є об’єднання отриманих слів в єдине, причому слова розподілені рискою.

В результаті, згенеровані паролі мають наступний вигляд. Для прикладу, згенеровані паролі мають довжину 5 символів та використовують англійський набір слів.

Таблиця 2.1

Приклади паролів, які були згенеровані модифікованим алгоритмом Diceware

Aback-56-A7arm-Parr-Dunlop	76-pc-Were-Shrank-d0ze
Eff1e-6000-spunk-pr1nce-ht	bater-orate-found-mac-ram0
f0lk-wax-caro7-fr0st-Basil	Hick-0nly-Tampa-Je-Dance
7isp-Whir7-chuck-rob0t-kar7	jew-Dr1ft-saucy-Venom-rodeo
Ha-c0sta-car0l-Sw1m-Garry	Ruse-cipher-c00-W07ve-Pec0s

Як видно, паролі є доволі простими для запам’ятовування словами, хоча додання цифр додає рівень складності до паролю, проте він все ще є читабельним.

Для отримання цих паролей було створено програмний застосунок, який реалізує цей, та ще один алгоритм генерації паролів. Детальніше цей застосунок буде описано в цьому розділі, а далі надано програмний код методу, який генерує паролі, за допомогою модифікованого алгоритму Diceware. Мова програмування: C#, мінімальна версія .Net Runtime - .Net Standard 2.0.

```
public string Generate(int passwordLength)
{
    var builder = new StringBuilder();
    var wordCount = 0;
    while (wordCount < passwordLength)
    {
var dice = _random.Next() % _alphabet.DicewareAlph.Count;
var uppercase = _random.Next() % 2;
var currWord = _alphabet.DicewareAlph[dice];
var temp = currWord.ToCharArray();
if (uppercase == 1)
{
    temp[0] = temp[0].ToString().ToUpper()[0];
}
for(int i = 0; i < currWord.Length; i++)
{
    var currChar = currWord[i];
    switch (currChar)
    {
case 'o':
    var replaceO = _random.Next() % 2;
    if (replaceO == 1)
    {
temp[i] = '0';
    }
    break;
case 'i':
    var replaceI = _random.Next() % 2;
    if (replaceI == 1)
    {
```

```
temp[i] = '1';
    }
    break;
case 'l':
    var replaceL = _random.Next() % 2;
    if (replaceL == 1)
    {
temp[i] = '7';
    }
    break;
    }
}
builder.Append($"{new string(temp)}-");
wordCount++;
    }
    builder.Remove(builder.Length - 1, 1);
    return builder.ToString();
}
```

Лістинг 2.1

В коді застосовуються 2 змінні, які задаються ззовні: _random та _alphabet. Перша змінна визначає генератор випадкових чисел, який буде застосований для генерації та описаний далі. Друга – визначає набір слів, який використовується. В даному застосунку набір слів постачається у вигляді файлу. Користувачам надана можливість вибрати власний набір слів для генерації паролів. Вимоги до файлу є наступними:

- файл має мати тільки один набір індексу та слова на строку;
- індекс слова та слово має бути розділено символом табуляції;
- кодування файлу має бути UTF-8.

Процес генерації паролю відбувається так само для наборів слів з латинськими символами як і з стандартним набором слів. В випадку кирилиці замінюється тільки

велика літера та символ «о». В залежності від вибраного набору слів змінюється і ентропія та ентропія на символ. Це відбувається через різницю в кількості символів, які можуть бути замінені, в різних наборах слів. На жаль, українського набору слів не має у відкритому доступі та затвердженого розробником, проте за необхідності можна створити такий набір.

2.2.Генератор випадкових чисел

В якості генератора псевдовипадкових чисел використовується генератор ANSI X9.17, який прийнятий як Федеральний стандарт обробки інформації у США. Процес генерації складається з 4 кроків [17]:

1. отримати поточний час з максимально можливою точністю;
2. обрахувати тимчасове значення за допомогою шифрування поточної дати алгоритмом шифрування Triple DES;
3. обрахувати результат виключного або між тимчасовим значенням та випадковим початковим значенням. Результат шифрується за допомогою алгоритму Triple DES і є вихідним випадковим значенням;
4. обрахувати результат виключного або між отриманим випадковим значенням та тимчасовим значенням обрахованим в кроці 2. Результат шифрується за допомогою алгоритму Triple DES і замінює значення випадкового початкового значення.

Схематично алгоритм зображено на рисунку 2.2. На рисунку визначені наступні значення:

- DT_i: значення часу на початку і-го етапу генерації.
- V_i: Значення ключа на початку і-го етапу покоління.
- R_i: Псевдовипадкове число, отримане на етапі і-го покоління.
- K_1, K_2: ключі DES, що використовуються для кожного етапу.

Ці значення залежать між собою наступним чином:

$$R_i = EDE \cdot K_1, K_2 \cdot [V_i \oplus EDE \cdot K_1, K_2 \cdot [DT_i]] \quad (2.1)$$

$$V_{i+1} = EDE \cdot K_1, K_2 \cdot [R_i \oplus EDE \cdot K_1, K_2 \cdot DT_i] \quad (2.2)$$

де, EDE відноситься до послідовності шифрування-дешифрування-шифрування.

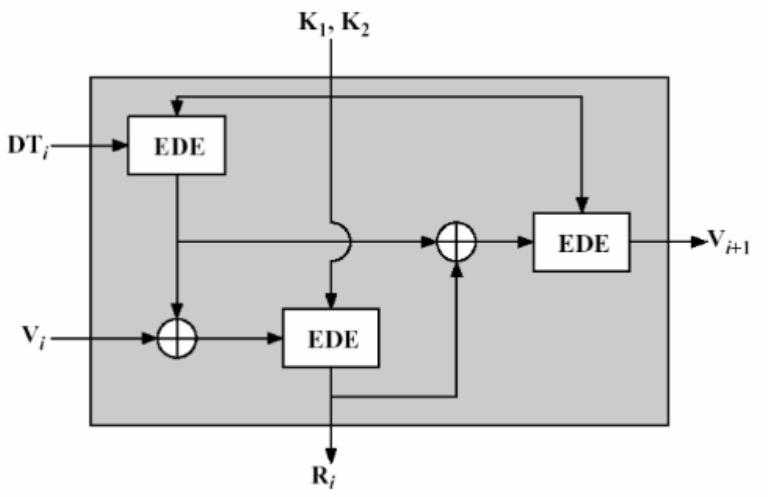

Рисунок 2.2 – Схема генерації випадкового числа алгоритмом ANSI X9.17

Так як в результаті, може буде згенероване число, яке є набагато більшим ніж кількість символів або слів в використовуваному для генерації паролю словнику, то в кінці генерації з вихідного випадкового числа, який відповідає максимальній кількості символів у словнику. Програмно генератор реалізовано наступним чином.

```
private byte[] _seed;
private readonly byte[] _key;
private readonly byte[] _iv;
public RandomGenerator(long seed)
{
   _seed = BitConverter.GetBytes(seed);
}
public RandomGenerator(long seed, long key, long iv)
{
   _seed = BitConverter.GetBytes(seed);
   _key = BitConverter.GetBytes(key);
```

```
    _iv = BitConverter.GetBytes(iv);
}
public int Next()
{
    using (var tdes = new TripleDESCryptoServiceProvider())
    {
        if (_key != null && _key.Length == 16 && _iv != null && _iv.Length ==
16)
        {
            tdes.Key = _key;
            tdes.IV = _iv;
        }
        var encryptor = tdes.CreateEncryptor();
        var time = BitConverter.GetBytes(DateTime.UtcNow.Ticks);
        var tempArr = encryptor.TransformFinalBlock(time, 0, time.Length);
        var xored = BitConverter.GetBytes(BitConverter.ToInt64(_seed, 0) ^
BitConverter.ToInt64(tempArr, 0));
        var x = encryptor.TransformFinalBlock(xored, 0, xored.Length);
        var xoredOut = BitConverter.GetBytes(BitConverter.ToInt64(x, 0) ^
BitConverter.ToInt64(tempArr, 0));
        _seed = encryptor.TransformFinalBlock(xoredOut, 0, xoredOut.Length);
        var res = BitConverter.ToInt32(x, 0);
        return res < 0 ? res * -1 : res;
    }}
```

Лістинг 2.2

Так як стандарт передбачає використання стандарту шифрування TripleDES, який вбудований за замовчання у .NET Standard 2.0 у бібліотеку System.Security.Cryptography, то використовуємо саме таку реалізацію. Також даний генератор має змогу генерувати від’ємні значення, що нам не потрібно, так як негативні індекси не пітримуються у данній мові програмування, то ми їх

інвертуємо. Дана реалізація генератора використовується у всіх алгоритмах генерації паролів, що представлено у роботі.

2.3.Механічний алгоритм генерації паролів.

Цей алгоритм було описано розділом вище, проте опишемо деталі. Механічний алгоритм генерації паролю відбувається за принципом генерації трьох випадкових значень: два з них мають значення від 0 до 5 та одне значення від 0 до 1. Використовуваний алфавіт має вигляд двох таблиць по 36 символів. Перші 36 символів включають символи нижнього регістру латинського алфавіту та арабські цифри від 0 до 9. Решта – символи верхнього регістру латинського алфавіту та спеціальні символи. Значення від 0 до 1 визначає який з двох алфавітів використовувати для пошуку символу, два інші значення визначають індекс символу в таблиці символів. Процес повторюється поки не буде досягнена бажана задана довжина паролю.

Програмно реалізація виглядає наступним чином.

```
public string Generate(int passwordLength) {
        var builder = new StringBuilder();
        while (builder.Length < passwordLength) {
           var dice1 = _random.Next() % 6;
           var dice2 = _random.Next() % 6;
           var coinFlip = _random.Next() % 2;
           var currChar = _alphabet.MechanicalAlph[coinFlip, dice1, dice2];
           builder.Append(currChar);
        }
        return builder.ToString();
    }
```

Лістинг 2.3

Значення _alphabet в даному випадку представляє собою трьохвимірний масив з описаними вище символами.

```
private static readonly char[,,] _mechanicalAlph =
    {
        {
            {'a', 'b', 'c', 'd', 'e', 'f' },
            {'g', 'h', 'i', 'j', 'k', 'l' },
            {'m', 'n', 'o', 'p', 'g', 'r' },
            {'s', 't', 'u', 'v', 'w', 'x' },
            {'y', 'z', '0', '1', '2', '3' },
            {'4', '5', '6', '7', '8', '9' },
        },
        {
            {'A', 'B', 'C', 'D', 'E', 'F' },
            {'G', 'H', 'I', 'J', 'K', 'L' },
            {'M', 'N', 'O', 'P', 'G', 'R' },
            {'S', 'T', 'U', 'V', 'W', 'X' },
            {'Y', 'Z', '_', '!', '@', '+' },
            {'$', '%', '=', '&', '-', '?' },
        }
    };
```

Лістинг 2.4

Так як у нас не має змоги програмно визначити на який спеціальний символ змінюється число, то цей алгоритм було реалізовано саме за допомогою трьохвимірного массиву. Генератор випадкових чисел, який використовується описаний вище. Приклади згенерованих паролів надано нище, кількість символів – 5.

Таблиця 2.2

Приклади паролів, згенерованих механічним алгоритмом

$Kj8=	E3a9A	h-7AZ	HypU1
DfNRX	Lcd$g	lDC?f	7w@04

Як видно, навіть такі короткі паролі є дуже складними до запам'ятовування, а рекомендована довжина таких паролів – 12-16 символів.

2.4.Графічний інтерфейс застосунку

Для показу роботи модифікованого алгоритму був розроблений програмний застосунок, який реалізує обидва алгоритми, а також вираховує ентропію та ентропію на символ для згенерованого паролю. Застосунок має можливість задати користувацьку довжину паролю від 3 до 20 символів, додати користувацький набір слів з списку оригінальних наборів слів та вибрати алгоритм генерації паролю. У правій половині застосунку виводиться результат роботи програми, а саме 5 згенерованих паролів, з якого файлу були вибрані слова для генерації паролю Diceware, значення ентропії та значення ентропії на символ для згенерованих паролів. Інтерфейс показано на рис. 2.3.

Рисунок 2.3 – Інтерфейс розробленого застосунку генерації паролів

Застосунок реалізує два способи генерації паролю: модифікований Diceware та механічний. Обидва алгоритми вже було описано перед цим, тому далі прдставлено результати виконання застосунку в різних режимах. Результат виконання застосунку в даному режимі з встановленим прапором «Механічний алгоритм» зображено на рис. 2.4.

Рисунок 2.4 – Результат роботи застосунку в режимі "Механічний алгоритм"

Результат виконання застосунку в режимі модифікованого алгоритму Diceware з встановленим прапором «Модифікація Diceware» зображено на рис. 2.5. В даному випадку, значення довжини визначає не кількість символів, а кількість сліві у згенерованому паролі.

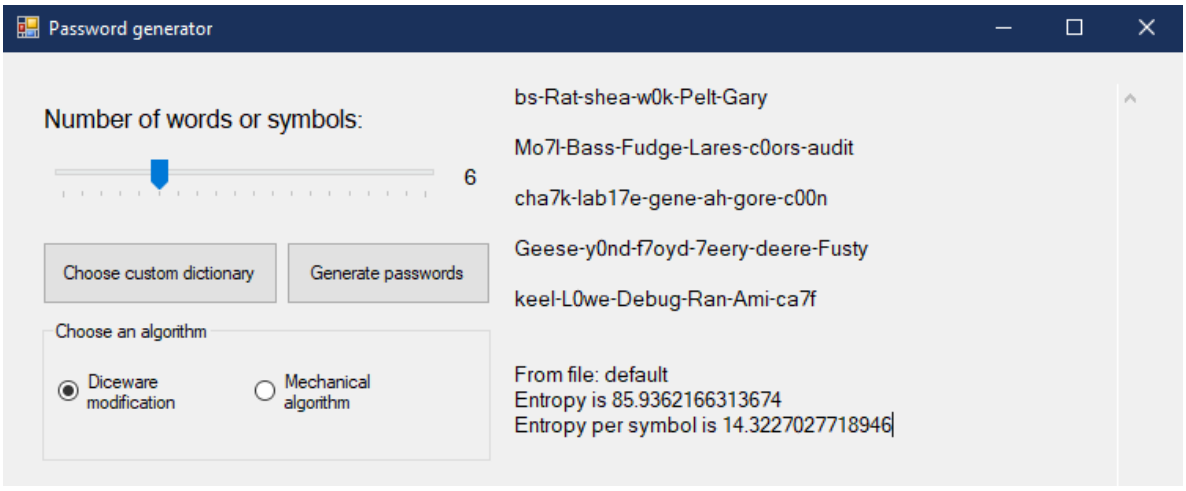

Рисунок 2.5 – Результат роботи застосунку в режимі "Модифікація Diceware"

Як можна побачити, першою строкою після згенерованих паролів є файл з якого були вибрані слова. За замовчуванням, як і зображено на рисунку, використовується набір англійських слів. Кнопка «Choose custom dictionary» дозволяє використовувати для генерації користувацький набір слів. Звичайно, краще

використовувати офіційні набори слів, які затвердженні розробником оригінального алгоритму Diceware, проте можливо створення власного набору слів, якщо мова не підтримується, з як мінімом такою ж кількістю слів. Результат роботи застосунку в режимі модифікованого алгоритму Diceware з використанням словника іспанської мови надано на рис. 2.6.

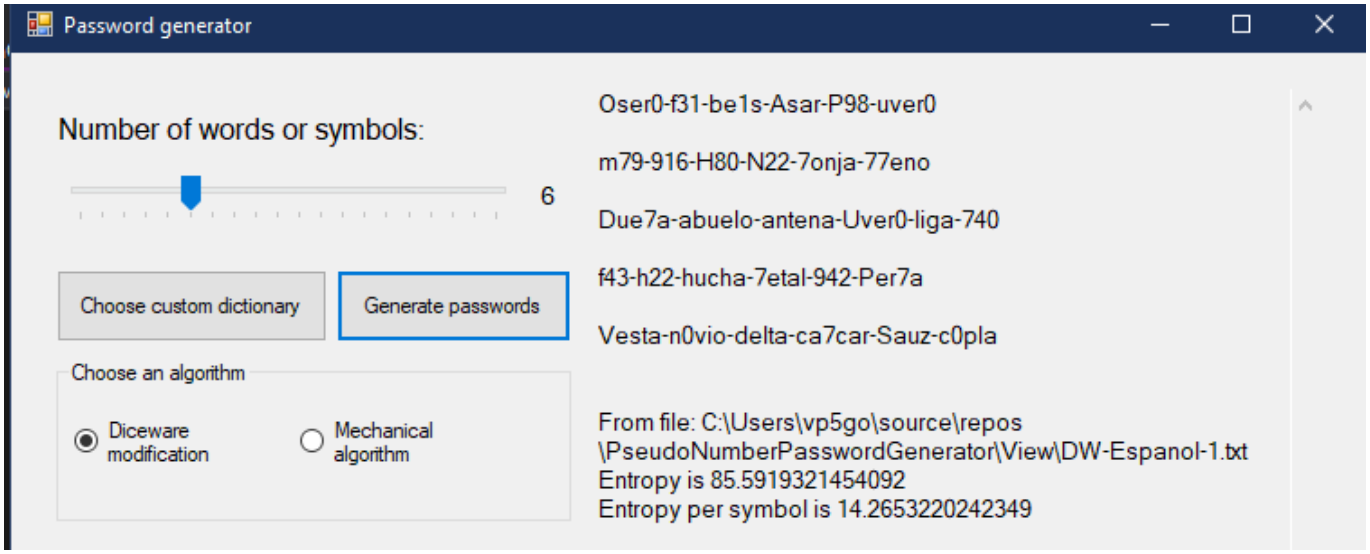

Рисунок 2.6 – Результат роботи застосунку в режимі "Модифікація Diceware" з використанням словника іспанської мови

Також, на рис. 2.3-2.6 можна побачити, що для кожного набору паролів обчислюється ентропія та ентропія на символ для даних паролів. Ентропія обчислюється за формулою 1.1, а ентропія на символ обчислюється за формулою 2.3.

$$H_{\text{символ}} = \frac{H}{L} \tag{2.3}$$

де H – загальна ентропія, L – кількість символів у паролі.

Видно, що ці два значення сильно варіюються залежно від алгоритму, а також навіть від словника, що використовується у модифікованому алгоритмі Diceware. Детальніше ці результати буде представлено у порівнянні та проаналізовано у наступному розділі.

Висновки за другим розділом

В даному розділі було описано запропонований модифікований алгоритм Diceware, який включає в себе особливості оригінального алгоритму, проте вирішує проблему стійкості паролів до атак за словником та задовільнення вимог веб-сервісів, щодо наявності числа та спеціального символу. Також описано програмну реалізацію цього алгоритму.

Так як будь-який алгоритм генерації паролів вимагає наявності якогось джерела випадковості, то було описано використовуваний для отримання паролів генератор випадкових чисел. Було використано генератор ANSI X9.17, який прийнятий як Федеральний стандарт обробки інформації у США та описано програмну реалізацію такого генератора.

Для порівняння було також запропоновано програмну реалізацію механічного алгоритму, яка використовує описаний генератор.

В роботі також було розроблено застосунок з графічним інтерфейсом, який виконує обидва алгоритми генерації, дозволяє вибирати один з них а також бажану довжину і виводить 5 згенерованих паролів, а також обраховує ентропію та ентропію на символ для згенерованих паролів. Також, для модифікованого алгоритму Diceware створена можливість завантаження користувацького набору слів.

В цьому розділі, було лише запропоновано покращення для алгоритму генерації паролів Diceware, а також програмну реалізацію його компонентів. Далі буде проаналізовано криптостійкість паролів та порівняння інших алгоритмів з запропонованою модифікацією Diceware.

РОЗДІЛ ІІІ
АНАЛІЗ ПАРОЛЕЙ ЗГЕНЕРОВАНИХ АЛГОРИТМАМИ ГЕНЕРАЦІЇ ПАРОЛЕЙ НА ОСНОВІ ПСЕВДОВИПАДКОВИХ ЧИСЕЛ

3.1.Методи оцінки надійності паролів

Міцність пароля – це міра ефективності пароля проти вгадування або атак грубої сили. У звичайній формі він оцінює, скільки спроб знадобиться в середньому зловмиснику, який не має прямого доступу до пароля, щоб його вгадати. Надійність пароля залежить від довжини, складності та непередбачуваності [18].

Використання надійних паролів знижує загальний ризик порушення безпеки, але надійні паролі не замінюють потреби в інших засобах безпеки [19]. Ефективність пароля заданої міцності сильно визначається розробкою та реалізацією факторів (знання, володіння, приналежність). Перший фактор є основним у цій роботі.

Швидкість, з якою зловмисник може передати системі вгадані паролі, є ключовим фактором у визначенні безпеки системи. Деякі системи накладають тайм-аут у кілька секунд після невеликої кількості (наприклад, трьох) невдалих спроб введення пароля. За відсутності інших уразливостей такі системи можна ефективно захистити за допомогою відносно простих паролів. Однак система повинна зберігати інформацію про паролі користувача в певній формі, і якщо ця інформація буде вкрадена, скажімо, через порушення системи безпеки, паролі користувача можуть бути під загрозою. У цій роботі при оцінці надійності припустимо, що у зловмисника немає обмеження на кількість спроб вгадування паролів.

Системи, які використовують паролі для аутентифікації, повинні мати спосіб перевірити будь-який введений пароль для отримання доступу. Якщо дійсні паролі просто зберігаються в системному файлі або базі даних, зловмисник, який отримає достатній доступ до системи, отримає всі паролі користувачів, надавши

зловмиснику доступ до всіх облікових записів у атакуваній системі та, можливо, до інших систем, де користувачі використовують такі самі або схожі паролі. Один із способів зменшити цей ризик — зберігати лише криптографічний хеш кожного пароля замість самого пароля. Стандартні криптографічні хеші, такі як серія Secure Hash Algorithm (SHA), дуже важко повернути назад, тому зловмисник, який отримав хеш-значення, не може безпосередньо відновити пароль. Однак знання хеш-значення дозволяє зловмиснику швидко перевірити припущення в автономному режимі. Широко доступні програми для злому паролів, які перевірять велику кількість пробних паролів на викрадений криптографічний хеш.

Для взлому паролів, які зберігаються у вигляді хеш значень, використовують райдужні таблиці. Райдужна таблиця – спеціальний варіант таблиць пошуку для звернення криптографічних хеш-функцій, які використовують механізм розумного компромісу між часом пошуку таблицею і займаною пам'яттю. Райдужні таблиці використовуються для розкриття паролів, перетворених за допомогою важкозворотної геш-функції, а також для атак на симетричні шифри на основі відомого відкритого тексту. Використання функції формування ключа з застосуванням солі робить цю атаку неможливою. Сіль поєднується з паролем під час обчислення хешу, тому зловмисник, який попередньо обчислює райдужну таблицю, повинен буде зберігати для кожного пароля свій хеш з усіма можливими значеннями солі. Це стає неможливим, якщо сіль має досить великий діапазон, скажімо, 32-бітове число. На жаль, багато систем аутентифікації, які широко використовуються, не використовують солі. Райдужні таблиці є розвитком більш раннього й простого алгоритму, запропонованого Мартіном Геллманом [20].

Удосконалення обчислювальних технологій продовжують збільшувати швидкість перевірки вгаданих паролів. Наприклад, у 2010 році Інститут технічних досліджень Джорджії розробив метод використання GPGPU для набагато швидше зламати паролі [21]. Elcomsoft винайшла використання звичайних графічних карт для швидкого відновлення пароля в серпні 2007 року і незабаром подала відповідний патент у США [22]. До 2011 року були доступні комерційні продукти, які передбачали можливість тестування до 112 000 паролів на секунду на

стандартному настільному комп’ютері з використанням високоякісного графічного процесора для того часу [23]. Такий пристрій зламає шестибуквенний пароль з одним регістром за один день. Слід зауважити, що роботу можна розподілити на багато комп’ютерів для додаткового прискорення, пропорційного кількості доступних комп’ютерів із порівнянними графічними процесорами. Доступні спеціальні хеші розтягування ключів, обчислення яких займає відносно багато часу, що зменшує швидкість здогадування. Хоча розтягування клавіш вважається найкращою практикою, багато поширених систем цього не роблять.

Інша ситуація, коли можливе швидке вгадування, це коли пароль використовується для формування криптографічного ключа. У таких випадках зловмисник може швидко перевірити, чи вгаданий пароль успішно декодує зашифровані дані. Наприклад, один комерційний продукт стверджує, що тестує 103 000 паролів WPA PSK в секунду [24].

3.2.Аналіз ентропії отриманих паролей

У комп’ютерній індустрії зазвичай вказують міцність пароля в термінах інформаційної ентропії, яка вимірюється в бітах і є поняттям з теорії інформації. Замість кількості здогадів, необхідної для точного пошуку пароля, надається логарифм цього числа з базою 2, який зазвичай називають кількістю «ентропійних бітів» у паролі, хоча це не зовсім та сама кількість. як інформаційна ентропія [25]. Пароль з ентропією в 42 біти, обчислений таким чином, буде таким же надійним, як рядок із 42 бітів, вибраний випадковим чином, наприклад, шляхом справедливого підкидання монети. Іншими словами, пароль з ентропією 42 біти вимагатиме 2^{42} (4 398 046 511 104) спроб, щоб вичерпати всі можливості під час грубого пошуку. Таким чином, збільшення ентропії пароля на один біт подвоює кількість необхідних здогадів, що робить завдання зловмисника вдвічі складнішим. У середньому зловмиснику доведеться спробувати половину можливої кількості паролів, перш ніж знайти правильний [26].

Як було описано вище, у розробленому програмному застосунку наявний функціонал автоматичного розрахунку імплементованих алгоритмів генерації паролів. Загальна ентропія розраховується за формулою 1.1, ентропія на біт – за формулою 2.3.

Для порівняння стійкості паролю, за допомогою розробленого застосунку було згенеровано паролі різної довжини. Використанний комерційний генератор паролів від LostPass було описано і прораховано за наданою документацією. У випадку механічного алгоритму та алгоритму AES-256 PBKDF2-SHA256 кількістю символів є фактична кількість символів у паролі, проте для Diceware та запропонованої модифікації кількістю символів у паролів є кількість слів. Результати аналізу ентропії та ентропії на символ для паролей згенерованих кожним алгоритмом наведені в табл. 3.1.

Таблиця 3.1

Значення ентропії для різних алгоритмів генерації паролів та довжини

Алгоритм	6 символів	8 символів	16 символів	Ентропія на біт
	Ентропія, біт	Ентропія, біт	Ентропія, біт	
Механічний	32,02	49,36	98,72	6,17
AES-256 PBKDF2-SHA256	39,33	52,44	104,87	6,55
Diceware	77,55	103,41	206,88	12,92
Модифікований Diceware	85,94	114,58	229,16	14,32

Якщо представити результати у вигляді залежності, то отримуємо графік зображений на рис. 3.1. Як видно, залежність у всіх алгоритмів лінійна, проте ентропія у алгоритмів, що базуються на Diceware зростає набагато швидше у порівнянні з механічним та AES-256 PBKDF2-SHA256. Ентропія, яка досягається у останніх двух алгоритмах при 16 символах, для алгоритму Diceware досягається у 8

символів, а для запропонованої модифікації у 7. Також, з таблиці видно, що у

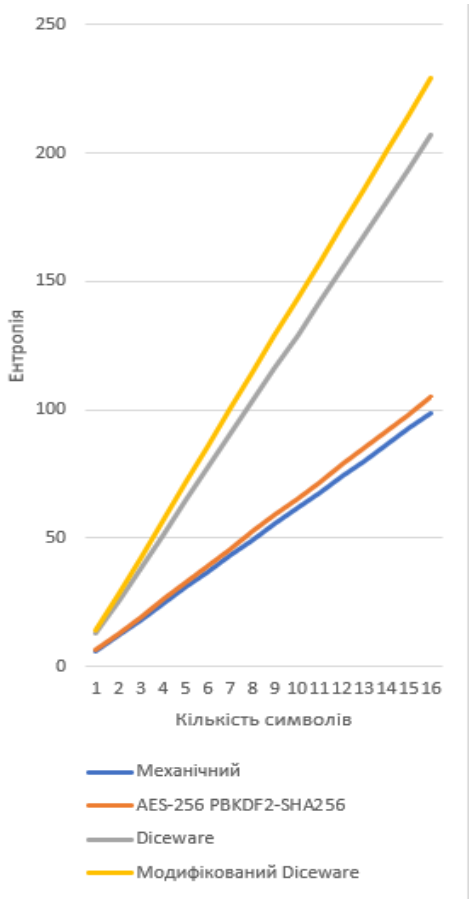

Рисунок 3.1 – Порівняння ентропії згенерованих паролей

порівнянні з оригінальним алгоритмом Diceware, запропонована модифікація алгоритму має на 1.4 біт більшу ентропію на біт.

Мінімальна кількість біт ентропії, необхідна для пароля, залежить від моделі загрози для даної програми. Якщо розширення ключа не використовується, потрібні паролі з більшою ентропією. RFC 4086, «Вимоги випадковості для безпеки», опублікований у червні 2005 року, надає деякі приклади моделей загроз і способи обчислення бажаної ентропії для кожної з них [27]. Їхні відповіді варіюються від 29 біт ентропії, необхідних, якщо очікуються лише онлайн-атаки, і до 96 біт ентропії, необхідних для важливих криптографічних ключів, які використовуються в таких програмах, як шифрування, де пароль або ключ мають бути безпечними протягом тривалого періоду часу та розширення не застосовується. Дослідження, проведене в

2010 році Джорджійським технічним дослідницьким інститутом, засноване на нерозширених ключах, рекомендувало випадковий пароль із 12 символів, як вимогу мінімальної довжини [21]. Слід зазначити, що обчислювальна потужність продовжує зростати, тому, щоб запобігти офлайн-атакам, необхідні біти ентропії також мають збільшуватися з часом.

У 1999 році проект Electronic Frontier Foundation зламав 56-бітове шифрування DES менш ніж за день за допомогою спеціально розробленого обладнання [28]. У 2002 році distributed.net зламав 64-бітний ключ за 4 роки, 9 місяців і 23 дні [29]. Станом на 12 жовтня 2011 року за оцінками distributed.net, злом 72-бітного ключа за допомогою поточного обладнання займе близько 45 579 днів або 124,8 років [30]. У зв'язку з наявними обмеженнями фундаментальної фізики, не можна очікувати, що будь-який цифровий комп'ютер (або комбінація) зможе зламати 256-бітне шифрування за допомогою атаки грубої сили [31]. Чи зможуть квантові комп'ютери зробити це на практиці, поки невідомо, хоча теоретичний аналіз припускає такі можливості [32].

3.3.Аналіз стійкості паролів до атак грубої сили

У криптографії атака грубої сили полягає в тому, що зловмисник подає багато паролів або парольних фраз з надією врешті-решт правильно вгадати. Зловмисник систематично перевіряє всі можливі паролі та парольні фрази, поки не знайде правильний. Крім того, зловмисник може спробувати вгадати ключ, який зазвичай створюється з пароля за допомогою функції отримання ключа. Це відомо як вичерпний пошук за ключами.

Атака грубої сили – це криптоаналітична атака, яка теоретично може бути використана для спроби розшифрувати будь-які зашифровані дані (крім даних, зашифрованих теоретично безпечним способом) [33]. Така атака може бути використана, коли неможливо скористатися іншими слабкостями в системі шифрування (якщо такі є), які полегшили б завдання.

Під час вгадування пароля цей метод дуже швидкий, коли використовується для перевірки всіх коротких паролів, але для більш довгих паролів використовуються інші методи, такі як атака по словнику, оскільки пошук методом грубої сили займає занадто багато часу. Довші паролі, парольні фрази та ключі мають більше можливих значень, що робить їх експоненціально важче зламати, ніж коротші [34].

Атаки грубої сили можна зробити менш ефективними, затуманюючи дані для кодування, що ускладнює розпізнавання зловмисника, коли код зламано, або змушуючи зловмисника виконувати більше роботи для перевірки кожного здогаду. Одним із показників міцності системи шифрування є те, скільки часу теоретично знадобиться зловмиснику, щоб здійснити успішну атаку грубою силою проти неї [35].

Атаки грубої сили – це застосування грубого пошуку, загальна техніка вирішення проблем, яка полягає в перерахуванні всіх кандидатів і перевірці кожного з них. Це реалізується в більшості випадків наступним чином. Множина всіх можливих паролів розбивається на непересічні підмножини. Застосунок перебирає ключі так, що процес здійснює перебір ключів з множини, що є відповідною цьому процесу і тільки йому. Система припиняє роботу, якщо один з процесів знайшов ключ. Найважче тут – це розділення вихідної множини. Але якщо кожен процесор почне обчислення з якогось довільного ключа, то час перебору збільшиться, але схема значно спроститься.

Окремим застосунком в данній роботі було реалізовано консольний застосунок для оцінки стійкості згенерованих паролів до атаки грубої сили. Цей застосунок фактично перебирає всі можливі варіанти паролів з алфавіту в 75 символи. Це передбачає, що зловмисник не знає ні довжини, ні алгоритму за допомогою якого було згенеровано пароль. Застосунок розділяє множину 75 символів на групи по 5 символів і починає перебирати паролі. Кожна з груп призначена як початковий символ для однієї асинхроної задачі, тобто у застосунку 15 паралелльних потоків підбирають паролі. Слід зауважити, що через необхідність

представлення результатів, час виконання застосунку збільшився, тому це варто брати до уваги. Нижче наведено код реалізації брутфорс атаки.

```
private static ulong guesses = 0;
private static bool guessed = false;
public async Task Execute(){
    var tasks = new List<Task>();
    Stopwatch stopWatch = new Stopwatch();
    stopWatch.Start();
    tasks.Add(Task.Run(() => { while (!guessed) { Console.Write("\nTotal guesses: {0}\nCracking speed: {1} per second", guesses, Math.Round(guesses/stopWatch.Elapsed.TotalSeconds)); Console.SetCursorPosition(0, Console.CursorTop-2); } }));
    foreach (var block in _mechanicalAlph){tasks.Add(Task.Run( () => Crack(block)));}
    await Task.WhenAny(tasks).ConfigureAwait(false);
    stopWatch.Stop();
    TimeSpan ts = stopWatch.Elapsed;
    Console.WriteLine($"\n\nCracked password {_password} in: "+String.Format("{0:00}:{1:00}:{2:00}.{3:00}",ts.Hours, ts.Minutes, ts.Seconds, ts.Milliseconds / 10));
}
private void Crack(char[] start) {
    int length = 1;
    while (!guessed) {
        foreach (var ch in start) {
            AddChar(ch.ToString(), length);
        }
        Length++;
    }
}
```

```
private void AddChar(string start, int depth){
    if (!guessed)    {
        if (depth == 1){
          foreach (var ch in _alph){
                if (_password == start + ch){
                   guessed = true;
                   return;
                }
                guesses++;
          }
        }
        else{
           depth--;
           foreach (var ch in _alph) {AddChar(start + ch, depth);}
        }
    }
    else{return;}
}
```

Лістинг 3.1

На рис. 3.2 також зображено результат роботи застосунку для тестового паролю.

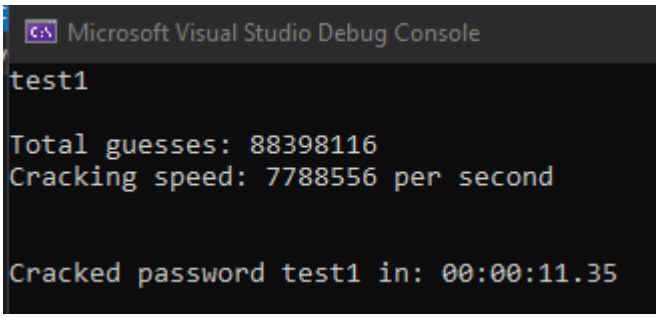

Рисунок 3.2 – Результат виконання атаки грубої сили на тестовий пароль

Застосунок досягає швидкості роботи у 7 788 556 паролів в секунду, що є дуже солідним показником як для одномашиного середовища взлому.

Тепер згенеруємо декілька паролів за допомогою досліджуваних методів і протестуємо їх застосунком. Для роботи було використано дуже малий набір символів через те, що час, який необхідний на взлом більш довгих паролів, дуже високий. В табл. 3.2 надано згенеровані паролі, час взлому та загальну кількість перебраних паролів. Для Diceware та запропонованої модифікації було обрано слово, з такою ж кількістю символів.

Таблиця 3.2

Час взлому паролів за допомогою власного застосунку

Метод	**Пароль**	**Час взлому**	**Кількість перебраних паролів**
Механічний	ySp	13мс	169 103
	L-9R	63мс	3 559 122
	e&KR=	1хв 6с	360 368 557
	nzM$uh	1год 14хв	31 057 847 445
AES-256 PBKDF2-SHA256	vY&	4мс	27 457
	uRR&	13мс	892 779
	IsA*H	1хв 17с	521 597 053
	P$JmwT	37хв 8с	15 300 226 048
Diceware	cot	7мс	29 292
	yuck	75мс	4 778 027
	jukes	1хв 2с	465 710 690
	coyote	38хв 28с	15 823 983 759
Запропонована модифікація Diceware	T0r	12с 54мс	74 251 884
	n0ra	67мс	2 950 870
	Metro	52с 16мс	263 277 089
	St0rey	1год 22хв	33 503 609 302

Як видно з отриманих результатів, час необхідний на взлом паролів експоненційно збільшується з збільшенням кількості символів у паролі. Час взлому між алгоритмами не сильно відрізняються між собою, варіація у результатах пояснюється розташуванням символів у блоці алфавіту: якщо початковий символ знаходиться на початку блоку, то він буде знайдений швидше ніж той, який починається з символу у кінці блоку. Тому для одних паролів довжиною в 6 символів кількість перебраних паролів становить 33,5 млрд а для інших – 15,8 млрд. Для порівняння було також перевірено час взлому цих самих паролів у стороньому застосунку Password Checker Online від компанії OnlineDomainTools [36]. Результати для різних конфігурації системи, яка виконує взлом, наданао у табл. 3.3.

Таблиця 3.3

Час взлому паролів за допомогою Password Checker Online

Метод	**Пароль**	**Час взлому**		
		Стандартний ПК	**GPU базований взлом**	**Середній ботнет**
Механічний	ySp	<1с	<1с	<1с
	L-9R	<1с	<1с	<1с
	e&KR=	42с	4с	<1с
	nzM$uh	1год	6хв	<1с
AES-256 PBKDF2-SHA256	vY&	<1с	<1с	<1с
	uRR&	<1с	<1с	<1с
	IsA*H	42с	4с	<1с
	P$JmwT	1год	6хв	<1с
Diceware	cot	<1с	<1с	<1с
	yuck	<1с	<1с	<1с
	jukes	<1с	<1с	<1с
	coyote	3с	<1с	<1с
Запропонована	T0r	<1с	<1с	<1с

модифікація Diceware	n0ra	<1c	<1c	<1c
	Metro	4c	<1c	<1c
	St0rey	9хв	57c	<1c

Ці результати також враховують кількість символів у наборі, що використовується, тому для Diceware базованих алгоритмів набір не містив спецсимволів. Проте результати власного застосунку та реузльтати надані сервісом Password Checker Online для стандартного ПК збігаються.

Слід зауважити, що кількість символів у паролі для механічного та AES-256 PBKDF2-SHA256 алгоритму становила максимум 6, тоді як для Diceware та запропонованої модифікації, фактично, кількість символів становить 1. Якщо теоретично розрахувати час, необхідний на взлам 16 символьного механічного паролю та 6 словного модифікованого алгоритму Diceware, які мають однакові значення ентропії при швидкості в 7 мільйонів паролів в секунду, яка досягається застосунком, то отримуємо наступне: для механічного паролю необхідно $3*10^5$ років, для модифікації Diceware для паролю довжиною в 30 символів (5 слів) – $4*10^{45}$ років. Слід зауважити, що для запропонованої модифікації кількість символів була усереднена через різну довжину слів, пароль може містити як меньше так і більше символів.

Але це неосяжний для розуміння проміжок часу, тому можна зробити висновок, що паролі є стійкими до атаки грубої сили, якщо використовується рекомендована довжина. Навіть якщо використовувати розподілені системи для взламу паролі зменшити час на 10^5 років, 6 років для механічного це все ще дуже великі витрати часу на взлам одного паролю, а модифікація Diceware все ще є неосяжно стійкою. Неможливо оцінити чи буде час настільки ж неосяжним при застосуванні квантових обчислень, але такі дані неможливо отримати на сьогодні, тому ми їх не розглядаємо.

Все це при умові, що зловмисник не знає ні метода генерації паролю, що застосовувався користувачем, ні не має обмежень на спроби. Це ідеальні умови, які

не існують в реальному житті. Для захисту від брутфорс атак адміністратори баз даних і каталогів можуть вживати заходів проти онлайн-атак, наприклад, обмежуючи кількість спроб введення пароля, вводячи часові затримки між послідовними спробами, збільшуючи складність відповіді (наприклад, вимагаючи відповіді CAPTCHA або коду підтвердження, надісланого через мобільний телефон) та/або блокування облікових записів після невдалих спроб входу. Адміністратори веб-сайту можуть перешкодити певній IP-адресі спробувати більше, ніж заздалегідь визначену кількість спроб введення пароля для будь-якого облікового запису на сайті.

3.4.Аналіз стійкості паролів до атак перебору за словником

У криптоаналізі та комп'ютерній безпеці атака перебору за словником — це атака, що використовує обмежену підмножину ключового простору для взлому шифру або механізму аутентифікації, намагаючись визначити його ключ дешифрування або парольну фразу, іноді намагаючись використовувати тисячі чи мільйони ймовірних можливостей [37], які часто отримують від списки минулих порушень безпеки [38].

Атака на словник заснована на випробувані всіх рядків у заздалегідь підготовленому списку. Спочатку такі атаки використовували слова, знайдені в словнику (звідси і словосполучення словникова атака) [39]; однак тепер у відкритому Інтернеті доступні набагато більші списки, що містять сотні мільйонів паролів, відновлених у результаті минулих порушень даних [40]. Існує також програмне забезпечення для злому, яке може використовувати такі списки та створювати звичайні варіації, наприклад, замінювати цифри на схожі літери. Атака на словник випробовує лише ті можливості, які вважаються найбільш успішними. Словникові атаки часто досягають успіху, оскільки багато людей мають тенденцію вибирати короткі паролі, які є звичайними словами або поширеними паролями; або варіанти, отримані, наприклад, шляхом додавання цифри або знаку пунктуації. Словникові атаки часто бувають успішними, оскільки багато поширених методів

створення паролів охоплюються доступними списками в поєднанні з генерацією шаблонів програмного забезпечення для злому. Більш безпечним підходом є випадкове створення довгого пароля (15 літер або більше) або багатослівної парольної фрази, використовуючи програму менеджера паролів або вводячи пароль вручну.

Можна досягти компромісу між часом і простором, попередньо обчисливши список хешів словникових слів і зберігаючи їх у базі даних, використовуючи хеш як ключ. Це вимагає значної кількості часу на підготовку, але це дозволяє здійснити реальну атаку швидше. Вимоги до зберігання попередньо обчислених таблиць колись були великою вартістю, але тепер вони є меншою проблемою через низьку вартість дискового сховища. Атаки на попередньо обчислені словники особливо ефективні, коли потрібно зламати велику кількість паролів. Попередньо обчислений словник потрібно створити лише один раз, і коли він буде завершений, хеш паролів можна знайти майже миттєво в будь-який момент, щоб знайти відповідний пароль. Більш витончений підхід передбачає використання райдужних таблиць, які зменшують вимоги до зберігання за ціною трохи довшого часу пошуку.

Попередньо обчислені атаки на словники або «атаки райдужної таблиці» можуть бути зірвані використанням солі, методики, яка змушує хеш-словник повторно обчислюватися для кожного шуканого пароля, що робить попереднє обчислення неможливим за умови, що кількість можливих значень солі дорівнює досить великий [41].

В роботі, для перевірки стійкості паролів також було реалізовано атаку за словником на згенеровані паролі на базі стандартного ПК. Механічний алгоритм та AES-256 PBKDF2-SHA256 не розглядалися, бо вони є стікими до атак за словником. У розробленій атаці передбачається, що зловмисник знає, який алгоритм використовується та має доступ для словника, який використовується для генерації паролю. Цей словник спочатку завантажується у пам'ять застосунку та потім використовується для взламу. Так як для запропонованої модифікації необхідно мати словник зі всіма варіаціями слова, тому навіть якщо слово не має літер, які можна замінити, в словнику має існувати 2 варіанти – з та без великої літери. Також,

як і для атаки грубої сили, список слів необхідно розділити на групи для того, що поділити початкові значення паролю на декілька потоків. Оптимальною по результатам атаки грубої сили виявилось значення у 15 потоків, тому розділяємо набор слів на 15 піднаборів. Нижче наведено код застосунку, який створює словники для запропонованої модифікації.

```
public static void Load() {
        var path = Directory.GetCurrentDirectory() + "\\diceware.wordlist";
        var _temp = new List<string>();
        using (var file = new StreamReader(File.OpenRead(path))) {
           var stream = file.ReadToEnd();
           var lines = stream.Split('\n');
           var result = lines.SkipWhile(x => x.Length < 5 || !x.Take(5).All(ch => Char.IsDigit(ch))).TakeWhile(x => x.Length > 5 && x.Take(5).All(ch => Char.IsDigit(ch))).Select(x => x.Split('\t')[1].TrimEnd()).ToList();
           _temp = result;
        }
        foreach (var word in _temp){
           AddWord(word, 0);
        }
        _modDicewareAlph = _alph.Select((x, i) => new { Index = i, Value = x })
        .GroupBy(x => x.Index / 2000)
        .Select(x => x.Select(v => v.Value).ToList())
        .ToList();
     }

private static string AddWord(string word, int index) {
   var temp = word.ToCharArray();
   if(index<word.Length){
      var currChar = word[index];
      var innerIndex = index+1;
```

```
            switch (currChar) {
               case 'o':
                    _alph.Add(AddWord(new string(temp), innerIndex));
                    temp[index] = '0';
                    _alph.Add(AddWord(new string(temp), innerIndex));
                    break;
               case 'i':
                    _alph.Add(AddWord(new string(temp), innerIndex));
                    temp[index] = '1';
                    _alph.Add(AddWord(new string(temp), innerIndex));
                    break;
               case 'l':
                    _alph.Add(AddWord(new string(temp), innerIndex));
                    temp[index] = '7';
                    _alph.Add(AddWord(new string(temp), innerIndex));
                    break;
               default:
                    _alph.Add(AddWord(new string(temp), innerIndex));
                    break;
            }
         }
         _alph.Add(new string(temp));
         if(char.IsLetter(temp[0])){
      _alph.Add(temp[0].ToString().ToUpperInvariant() + new string(temp).Remove(0,
1));
         }
      return new string (temp);
      }
```

Лістинг 3.2

В результаті створюється список з слів з яких будемо намагатися підбирати паролі. Для Diceware словника це 7776 слів, для запропонованої модифікації – 29 992 слова. Це значно більше ніж 75 алфавіт для інших розглянутих алгоритмів, тому довжина паролю буде знатно позначатися на часі, необхідним для взлому.

Після створення словників, переходимо до власне взламу. Код застосунку схожий до атаки грубої сили проте адаптований до генерації Diceware базованих паролів та наведено нижче.

```
public async Task Execute() {
    var tasks = new List<Task>();
    Stopwatch stopWatch = new Stopwatch();
    stopWatch.Start();
    tasks.Add(Task.Run(() => { while (!guessed) { Console.Write("\nTotal guesses: {0}\nCracking speed: {1} per second", guesses, Math.Round(guesses / stopWatch.Elapsed.TotalSeconds)); Console.SetCursorPosition(0, Console.CursorTop - 2); } }));
    foreach (var block in _modDicewareAlph){
      tasks.Add(Task.Run(() => Crack(block)));
    }
    await Task.WhenAny(tasks).ConfigureAwait(false);
    stopWatch.Stop();
    TimeSpan ts = stopWatch.Elapsed;
    Console.SetCursorPosition(0, Console.CursorTop + 3);
    Console.WriteLine($"\n\nCracked password {_password} in: " + String.Format("{0:00}:{1:00}:{2:00}.{3:00}",
    ts.Hours, ts.Minutes, ts.Seconds,
    ts.Milliseconds / 10));
}

private void Crack(List<string> start) {
    int length = 1;
```

```
    while (!guessed) {
        foreach (var word in start){
            if (_password == word){
                guessed = true;
                return;
            }
            if (length > 1){
                AddWordFromList(word, length);
                }
        }
        length++;
    }}
private void AddWordFromList(string start, int depth){
    if (!guessed){
        if (depth == 1){
            foreach (var word in _alph){
                if (_password == $"{start}-{word}"){
                    guessed = true;
                    return;
                }
                guesses++;
            }
        }
        else{
            depth--;
            foreach (var word in _alph){
                AddWordFromList($"{start}-{word}", depth);}
        }
    }
    else{
```

```
        return;
        }
}
```

Лістинг 3.3

Використовуючи даний застосунок, перевіримо час взлому паролів, які були згенеровані алгоритмом Diceware та запропонованою модифікацією. Для уникнення похибки паролі були однакові, проте для запропонованої модифікації було згенеровано аналог з заміною символів, великою літерою та дефісом між словами. Результати надано в табл. 3.4.

Таблиця 3.4

Час взлому паролів атакою словником за допомогою власного застосунку

Алгоритм	**Пароль**	**Час взлому**	**Кількість спроб**
Diceware	group	<1мс	1 114
	whoop cu	42с 34мс	236 923 542
	costa meaty watt	>5год	>108 000 000 000
Запропонована модифікація Diceware	Group	5с 27мс	20 085
	wh00p-Cu	1хв 12с	368 378 253
	C0sta-meaty-watt	>5год	>108 000 000 000

Як видно з результатів, навіть два слова займають близько хвилини, а аналогічні двухсимвольні паролі механічного та AES-256 PBKDF2-SHA256 взламувалися за мілісекунди. Паролі з трьох слов вже неможливо було взламати на моєму ПК. При швидкості в 6 мільйонів паролів в секунду, яка стабільно досягалася в процесі тестування, повний перебор набору трьох словних паролів для алгоритму Diceware зайняв би в середньому 10.5 годин, в той час як для запропонованої модифікації – 625 годин або 26 днів. І це половина рекомендованої довжини при

умові, що зловмисник знає, який алгоритм використовувався, а також має доступ до словника, за допомогою якого генерувалися паролі.

За результатами аналізу, набір слів, який застосовується для генерації паролів запропонованої модифікації збільшився з 7776 слів до 29992 тобто у чотири рази, що дозволило значно збільшити стійкість паролю до атак за словником. Теоритично, при рекомендованій довжині у 6 слів, середній час необхідний для взламу паролю, який був згенерований запропонованою модифікацією, збільшився з 584 мільйонів років оригінального алгоритму Diceware до $1.9*10^{12}$ років при швидкості в 6 мільйонів паролів в секунду. Проте, можна зробити висновок, що обидва алгоритми є стійкими до атак за словником.

3.5.Аналіз простоти запам’ятовування отриманих паролів

Однією із особливостей Diceware базованих паролів є те, що їх легко запам’ятовувати користувачу без використання додаткових застосунків, так як пароль складається з звичних слів. Для інших алгоритмів застосовуються методи запам’ятовування паролів:

- Мнемонічні паролі: деякі користувачі розробляють мнемонічні фрази і використовують їх для створення більш-менш випадкових паролів, які, тим не менш, відносно легко запам'ятати користувачеві. Наприклад, перша літера кожного слова у фразі, що запам’ятовується. Дослідження оцінюють надійність пароля таких паролів приблизно 3,7 біт на символ, у порівнянні з 6,6 біт для випадкових паролів із символів ASCII для друку [42].
- Мнемоніка після створення паролів: Після встановлення пароля варто придумати менімоніку, яка підходить [43]. Це не обов’язково має бути зв’язною фразою, поки користувач може її запам’ятати. Це дозволяє паролю бути повністю випадковими.

- Візуальне уявлення паролів: пароль запам’ятовується на основі послідовності натиснених клавіш, а не значень самих клавіш, напр. послідовність !qAsdE#2 представляє ромбоїд на клавіатурі. Метод створення таких паролів називається PsychoPass [44]; такі паролі з просторовим шаблоном можна покращити [45].
- Шаблони паролів: будь-який шаблон у паролі полегшує вгадування (автоматичне чи ні) і зменшує об’єм роботи зловмисника. Наприклад, паролі такої форми без урахування регістру: приголосний, голосний, приголосний, приголосний, голосний, приголосний, число, число (наприклад, pinray45) називаються паролями Environ. Шаблон чергування голосних і приголосних символів мав на меті зробити паролі більш імовірними для вимови і, таким чином, більш запам’ятовуваними. На жаль, такі шаблони суттєво знижують інформаційну ентропію пароля, роблячи атаки з використанням пароля значно ефективнішими.

Для оцінки алгоритмів генерації паролів, що ми розглядаємо в цій роботі, на запам’ятовуваність в табл. 3.5 надано 5 варіантів згенерованих паролів відповідними алгоритмами.

Таблиця 3.5

Приклади згенерованих паролів

Механічний алгоритм	**AES-256 PBKDF2-SHA256**	**Diceware**	**Модифікований Diceware**
bGyfZX?T	zaWDse%	bid net blunt snafu frenzy knurl	pr1ze-dewy-k0ng-p1ggy-wt-Tax
xO&4$6GU	iG5Q5Y*9egw1	deja frock hunk clime wise holm	k71ne-tone-basin-Ag0ny-Gt-B700p
rM?KLyMm	cbFyv94Rj*MC	ea zk bribe luxe world raul	Carb0n-Um-Pion-exact-thumb-Nasty
w0tKMh?V	1Mh*5XZbaLUi	bind ultra bethel septa texan onion	Coc0-Veery-ex-Chaos-P7ague-Avert
+91-kGnU	qdUiS5e@qDq$	siena chock er	laban-gab0n-M1st-cpa-

		gander stein 43	we77-pab7o

Як видно, найпростішим для читання і запам’ятовування є стандартний алгоритм Diceware. Не Diceware базовані алгоритми надто складно запам’ятати в чистому вигляді, тому їх краще всього використовувати разом з менеджером паролів. Якщо застосовувати принцип створення фрази для запам'ятовування паролю на основі перших літер, то для механічного та AES-256 PBKDF2-SHA256 така фраза буде становити 12-16 слів і вимагає від користувача самостійно її вигадати, в той час як обидва Diceware базовані алгоритми вже є такими фразами і складаються з 6 слів.

Хоча модифікований Diceware і є більш складним для запам’ятовування ніж стандартна реалізація проте не настільки складний як, наприклад, не Diceware базовані. Отримані результати показують, що збільшення ентропії призвело до збільшення складності читання та запам’ятовування паролю, хоча його досі може запам’ятати середньостатистичний користувач. Слід зазначити, що дана оцінка є суб’єктивною.

3.6.Аналіз відповідності згенерованих паролів вимогам

В першому розділі було розглянуто список вимог до використовуваних паролів, проте єдиного списку таких рекомендацій не було надано. Для початку розглянемо відповідність алгоритмів генерації паролів до стандарту NIST SP 800-63b. В таблиці 3.6 наведено таку матрицю відповідності. Були обрані тільки ті критерії, які залежать від ходу виконання програми або алгоритму, а не ті, які передбачають, наприклад, попередню перевірку паролів на компрометацію. Такі дії можуть бути легко додані до алгоритму, проте не є обов’язковими.

Таблиця 3.6

Відповідність алгоритмів генерації паролів стандарту NIST SP 800-63b

Критерій	**Алгоритм**

	Механічний алгоритм	**AES-256 PBKDF2-SHA256**	**Diceware**	**Модифікований Diceware**
Довжина	+	+	+	+
Всі символи ASCII та UNICODE	72	96	70	96
Повторювані слова	Можливі	Можливі	Можливі	Можливі
Словникові слова	-	-	+	+

Як видно, жодні з алгоритмів не відповідають стандартам NIST повністю. Насправді, одна з вимог про недопустимість повторювальних символів для мене не має сенсу, через те, що виключенням повторів ми зменьшуємо простір можливих значень тим самим робимо алгоритми математично менш стійкими. Також, як було доведено на практиці, якщо пароль навіть і містить словникові слова, то при достатній їх кількості взлам паролю досягає неосяжно великого часу.

Якщо подивитися з точки зору часто зустрічаємих вимог до паролів на веб-сервісах в мережі Інтернет (від 8 символів, має містити число, малу та велику літеру і спецсимвол), то всі паролі окрім оригінального алгоритму Diceware, де відсутні спецсимволи (пробіл не рахується, більш того, більшість сайтів не дозволяє використовувати пробіли в паролях) та великі літери задовільняють ці вимоги. Запропонована модифікація таких недоліків не має.

Для Diceware базованих паролів ще одним неприєним обмеженням паролів є обмеження по максимальній довжині паролю, зачасту у 32 символи. Через те, що згенеровані паролі можуть бути довшими за 32 символи, то їх необхідно або обрізати або перегенерувати, поки він не стане меньше за 32 символи. Зачасту наявність таких обмежень показує на застосування солі обмеженої довжини при хешувані паролів для зберігання, тому навіть у такому випадку паролі є стійкими.

Висновок за третім розділом

В розділі було розглянуто методи, за якими можна оцінити стійкість паролів до взлому. Серед них особливу увагу приділено оцінці паролів, що розглядаються у роботі (механічний, AES-256 PBKDF2-SHA256, Diceware, запропонована модифікація Diceware) на математичну стійкість за допомогою порівняння ентропії та стійкість паролів до атак грубої сили і атак за словником.

При оцінці ентропії, запропонованою модифікацією було досягнено покращення ентропії на біт на 1.4 біта у порівнянні з оригінальним алгоритмом Diceware. При цьому, значення ентропії, що досягається у механічному алгоритмі та алгоритмі AES-256 PBKDF2-SHA256 при 16 символах, що є рекомендованою до використання довжиною, для алгоритму Diceware досягається у 8 слів, а для запропонованої модифікації у 7.

Для оцінки стійкості до атак грубої сили та атак за словником були розроблені індивідуальні застосунки на програмній мові C#, які використовуючи 15 потоків намагалися взламати локально заданий пароль. При цьому, припускалося, що теоритичний зловмисник має необмежену кількість спроб, а при атаці за словником – знає словник, що використовувся для генерації паролю. При переборі швидкість підбору досягала 6-7 мільйонів паролів в секунду. В результаті теоритичного розрахунку часу, що необхідний на взлом паролю рекомендованої довжини, було зроблено висновок, що паролі є стійкими до обох видів атак і час взлому досягає неосяжно великих значень.

Все це при умові, що зловмисник не має обмежень на спроби. Це ідеальні умови, які не існують в реальному житті. Для захисту від таких атак адміністратори баз даних і каталогів можуть вживати заходів проти онлайн-атак, наприклад, обмежуючи кількість спроб введення пароля, вводячи часові затримки між послідовними спробами, збільшуючи складність відповіді та/або блокування облікових записів після невдалих спроб входу. Адміністратори веб-сайту можуть

перешкодити певній IP-адресі спробувати більше, ніж заздалегідь визначену кількість спроб введення пароля для будь-якого облікового запису на сайті.

Також було оцінено складність паролів до запам’ятовування. Для механічного алгоритму та алгоритму AES-256 PBKDF2-SHA256 необхідно було створювати мнемонічну фразу з кількості слів рівним довжині паролю, тобто при рекомендованій довжині, кількість слів у такій фразі досягає 16 слів. В Diceware базованих паролях таку фразу не потрібно видумувати, тому що самі паролі є такими фразами і містять 6-8 слів.

Оцінючи відповідність алгоритмів стандарту NIST SP 800-63, то жодні з алгоритмів не відповідають стандартам NIST повністю, але при оцінці відповідності часто зустрічаємим вимогам на веб-ресурсах, всі алгоритми крім оригінального Diceware ці вимоги задовільняють.

ВИСНОВКИ

Парольна аутентифікація є найпоширенішим видом аутентифікації користувачів у комп'ютерних системах, тому для надання належного рівня захисту, користувач має створити надійний та стійкий пароль. Генератори паролів покликані вирішіти цю проблему.

В даній роботі було розглянуто 4 алгоритми генерації паролів: механічний алгоритм, AES-256 PBKDF2-SHA256, Diceware та модифікований алгоритм Diceware. Запропонована у роботі модифікація базується на оригінальному алгоритмі Diceware та додає рівень складності, до згенерованих паролів, що збільшує складність генерованих паролів при цьому не сильно шкодить складності запам’ятовування паролю, що є однією з основних переваг оригінального алгоритму.

В першому розділі було описано основні теоритичні визначення та терміни, на яких базується робота. Також надано огляд генераторів псевдовипадкових чисел, алгоритмі генерації паролів, які розглядаються у роботі та документів, які можуть бути застосовані як вимоги та рекомендації до паролів та генераторів псевдовипадкових чисел.

В другому розділі було детально розглянуто запропоновану модифікацію алгоритму Diceware. Також, описано розроблений застосунок, на програмній мові C#, який реалізує механічний алгоритм генерації паролів та запропоновану модифікацію Diceware. В якості генератора в застосунку також реалізовано генератор ANSI X9.17. Застосунок має графічний інтерфейс на якому можна вибрати алгоритм генерації паролю, довжину та файл з користувацьким набором слів для модифікованого алгоритму. В результаті роботи користувачу буде виведено набір з 5 паролів обраного алгоритму та довжини, а також автоматично прораховано ентропію згенерованих паролів та ентропію на символ.

В третьому розділі була проведена оцінка стійкості згенерованих паролів. Була порівняна математична стійкість паролів та стійкість паролів до атак грубої

сили і атак за словником. Також розглянуто складність запам’ятовування паролів і відповідність вимогам та рекомендаціям.

При оцінці ентропії, запропонованою модифікацією Diceware було досягнено покращення ентропії на біт на 1.4 біта у порівнянні з оригінальним алгоритмом Diceware. При цьому, значення ентропії, що досягається у механічному алгоритмі та алгоритмі AES-256 PBKDF2-SHA256 при 16 символах, що є рекомендованою до використання довжиною, для алгоритму Diceware досягається у 8 слів, а для запропонованої модифікації у 7.

Для оцінки стійкості до атак грубої сили та атак за словником були розроблені індивідуальні застосунки на програмній мові C#, які використовуючи 15 потоків намагалися взламати локально заданий пароль. При цьому, припускалося, що теоритичний зловмисник має необмежену кількість спроб, а при атаці за словником – знає словник, що використовувся для генерації паролю. При переборі швидкість підбору досягала 6-7 мільйонів паролів в секунду. В результаті теоритичного розрахунку часу, що необхідний на взлом паролю рекомендованої довжини, було зроблено висновок, що паролі є стійкими до обох видів атак і час взлому досягає неосяжно великих значень при використанні паролів рекомендованої довжини.

Підсумовуючи, можна сказати, що використання даної модифікації алгоритму є більш простим ніж використання 12-16 – символьних паролів не Diceware базованих алгоритмів, через простоту запам’ятовування та надає більший рівень захисту паролів від атак «грубої сили». Тому в разі використання запам’ятовуваних паролів використання даної модифікації є бажаним.

Поставлені завдання були виконані у повному обсязі. Алгорим генерації паролів Diceware було покращено з вводом модифікації, що показала покращені результати математичної стійкості та стійкості до різних видів атак.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Yee-Yin Choong. Password Usability [Електронний ресурс] / Yee-Yin Choong. – 20151. – Режим доступу до ресурсу: https://csrc.nist.gov/CSRC/media/Presentations/Password-Usability/images-media/oct23_choong_password.pdf.

2. Saranga Komanduri1 , Richard Shay1 , Patrick Gage Kelley1 , Michelle L. Mazurek1 , Lujo Bauer1 , Nicolas Christin1 , Lorrie Faith Cranor1 , Serge Egelman. Of Passwords and People: Measuring the Effect of Password-Composition Policies : ACM.D.4.6; ACM.H.1.2 / Saranga Komanduri1 , Richard Shay1 , Patrick Gage Kelley1 , Michelle L. Mazurek1 , Lujo Bauer1 , Nicolas Christin1 , Lorrie Faith Cranor1 , Serge Egelman. – Vancouver, BC, Canada, 2011. – 10 с.

3. Serhii Buchyk PASSWORD COMPLEXITY ANALYSIS OF DICEWARE PASSWORD GENERATION ALGORITHM MODIFICATION BASED ON PSEUDO-RANDOM SEQUENCES / Serhii Buchyk, Vitalii Piatyhor // VIII International conference “Information Technology and Implementation”, December 1-3, 2021.

4. (CEUR Workshop Proceedings - Scopus) Buchyk S., Lukova-Chuiko N., Toliupa S., Piatyhor V. and Buchyk O. (2021) Diceware password generation algorithm modification based on pseudo-random sequences. Cybersecurity Providing in Information and Telecommunication Systems, 26 October 2021, Kyiv, Ukraine.

5. Recommendation X.800. INTERNATIONAL TELECOMMUNICATION UNION. SECURITY ARCHITECTURE FOR OPEN SYSTEM INTERCONNECTION FOR CCIT APPLICATIONS, 1991. – 46 с.

6. James Ohwofasa Akpeninor. Modern Concepts of Security / James Ohwofasa Akpeninor., 2013. – С. 135.

7. Bernard Meyer. Most common passwords: latest 2022 statistics [Електронний ресурс] / Bernard Meyer. – 2022. – Режим доступу до ресурсу: https://cybernews.com/best-password-managers/most-common-passwords/.

8. Password Manager and Vault 2021 Annual Report: Usage, Awareness, and Market Size [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://www.security.org/digital-safety/password-manager-annual-report/.

9. Shay, Richard; Kelley, Patrick Gage; Komanduri, Saranga; Mazurek, Michelle L.; Ur, Blase; Vidas, Timothy; Bauer, Lujo; Christin, Nicolas; Cranor, Lorrie Faith (2012). Correct horse battery staple: Exploring the usability of system-assigned passphrases (PDF). SOUPS '12 Proceedings of the Eighth Symposium on Usable Privacy and Security. doi:10.1145/2335356.2335366.

10. LastPass Password generator [Електронний ресурс] – Режим доступу до ресурсу: https://www.lastpass.com/ru/password-generator.

11. The Diceware Passphrase Home Page [Електронний ресурс] – Режим доступу до ресурсу: http://world.std.com/~reinhold/diceware.html.

12. Internet Secrets, 2nd Edition, John R. Levine, Editor, Chapter 37, IDG Books, 2000, ISBN 0-7645-3239-1

13. Digital Identity Guidelines Authentication and Lifecycle Management [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://pages.nist.gov/800-63-3/sp800-63b.html#memsecret.

14. Elaine Barker. Recommendation for Random Number Generation Using Deterministic Random Bit Generators [Електронний ресурс] / Elaine Barker, John Kelsey. – 2015. – Режим доступу до ресурсу: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf.

15. Створення надійного пароля для облікового запису Microsoft [Електронний ресурс] – Режим доступу до ресурсу: https://support.microsoft.com/en-us/account-billing/how-to-create-a-strong-password-for-your-microsoft-account-f67e4ddd-0dbe-cd75-cebe-0cfda3cf7386.

16. Як створити надійний пароль і краще захистити обліковий запис [Електронний ресурс] – Режим доступу до ресурсу: https://support.google.com/accounts/answer/32040.

17. W. Stallings, “Cryptography and Network Security Principles and Practice”, Prentice Hall, 2003

18. CISA. Security Tip (ST04-002) Choosing and Protecting Passwords / CISA. – 2009.

19. Torsten George. Why User Names and Passwords Are Not Enough [Електронний ресурс] / Torsten George. – 2019. – Режим доступу до ресурсу: https://www.securityweek.com/why-user-names-and-passwords-are-not-enough.

20. M. Hellman. A cryptanalytic time-memory trade-off / M. Hellman // IEEE Transactions on Information Theory / M. Hellman., 1980. – С. 401 – 406.

21. Georgia Tech Research Institute. Teraflop Troubles: The Power of Graphics Processing Units May Threaten the World's Password Security System [Електронний ресурс] / Georgia Tech Research Institute. – 2010. – Режим доступу до ресурсу: https://web.archive.org/web/20101230063449/http://www.gtri.gatech.edu/casestudy/Teraflop-Troubles-Power-Graphics-Processing-Units-GPUs-Password-Security-System.

22. BELENKO ANDREY V. Use of graphics processors as parallel math co-processors for password recovery / BELENKO ANDREY V., 2011.

23. ElcomSoft Password Recovery Speed table [Електронний ресурс]. – 2011. – Режим доступу до ресурсу: https://web.archive.org/web/20061017173506/http://www.elcomsoft.com/eprb.html.

24. Elcomsoft Wireless Security Auditor [Електронний ресурс]. – 2011. – Режим доступу до ресурсу: https://web.archive.org/web/20110219131825/http://www.elcomsoft.com/ewsa.html.

25. James L. Massey. Guessing and entropy / James L. Massey // Proceedings of 1994 IEEE International Symposium on Information Theory / James L. Massey. – Zurich, Switzerland: 1994.

26. William E. Burr. NIST Special Publication 800-63 Electronic Authentication Guideline / William E. Burr, Donna F. Dodson, W. Timothy Polk., 2004.

27. J. Schiller. RFC 4086 Randomness Requirements for Security [Електронний ресурс] / J. Schiller. – 2005. – Режим доступу до ресурсу: https://datatracker.ietf.org/doc/html/rfc4086.

28. "EFF DES CRACKER" MACHINE BRINGS HONESTY TO CRYPTO DEBATE [Електронний ресурс]. – 1998. – Режим доступу до ресурсу:

https://web.archive.org/web/20100101001853/http://w2.eff.org/Privacy/Crypto/Crypto_misc/DESCracker/HTML/19980716_eff_descracker_pressrel.html.

29. RC5-64 / Overall Project Stats [Електронний ресурс]. – 2013. – Режим доступу до ресурсу: https://web.archive.org/web/20130910051812/http://stats.distributed.net/projects.php?project_id=5.

30. RC5-72 / Overall Project Stats [Електронний ресурс]. – 2011. – Режим доступу до ресурсу: https://stats.distributed.net/projects.php?project_id=8.

31. Bruce Schneier. Crypto-Gram [Електронний ресурс] / Bruce Schneier. – 1999. – Режим доступу до ресурсу: https://www.schneier.com/crypto-gram/archives/1999/0215.html.

32. Quantum Computing and Encryption Breaking [Електронний ресурс] – Режим доступу до ресурсу: https://stackoverflow.com/questions/2768807/quantum-computing-and-encryption-breaking.

33. Paar, Christof. Understanding Cryptography: A Textbook for Students and Practitioners. / Paar, Christof, Pelzl, Jan, Preneel, Bart., 2010.

34. Ian Urbina. The Secret Life of Passwords [Електронний ресурс] / Ian Urbina. – 2014. – Режим доступу до ресурсу: https://www.nytimes.com/2014/11/19/magazine/the-secret-life-of-passwords.html.

35. Sebastian Schrittwieser. Code Obfuscation against Static and Dynamic Reverse Engineering / Sebastian Schrittwieser, Stefan Katzenbeisser. // International Workshop on Information Hiding. – 2011. – С. 270–284.

36. Password Checker Online [Електронний ресурс] – Режим доступу до ресурсу: http://password-checker.online-domain-tools.com/.

37. An off-line dictionary attack on a simple three-party key exchange protocol / Junghyun Nam, Juryon Paik, Ung Kim, Dongho Won // IEEE Communications Letters / Junghyun Nam, Juryon Paik, Ung Kim, Dongho Won., 2009. – С. 205–207.

38. Oxford Languages and Google [Електронний ресурс] – Режим доступу до ресурсу: https://languages.oup.com/google-dictionary-en/.

39. Dictionary Attacks 101 [Електронний ресурс]. – 2009. – Режим доступу до ресурсу: https://blog.codinghorror.com/dictionary-attacks-101/.

40. CrackStation's Password Cracking Dictionary [Електронний ресурс] – Режим доступу до ресурсу: https://crackstation.net/crackstation-wordlist-password-cracking-dictionary.htm.

41. CAPEC-55: Rainbow Table Password Cracking [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://capec.mitre.org/data/definitions/55.html.

42. Johannes Kiesel. A Large-scale Analysis of the Mnemonic Password Advice / Johannes Kiesel, Benno Stein, Stefan Lucks // NDSS Symposium 2017 / Johannes Kiesel, Benno Stein, Stefan Lucks., 2017.

43. Remembering passwords [Електронний ресурс] – Режим доступу до ресурсу: http://changingminds.org/techniques/memory/remembering_passwords.htm.

44. Cipresso P. How to Create Memorizable and Strong Passwords / Cipresso P, Gaggioli A, Serino S // J Med Internet Res / Cipresso P, Gaggioli A, Serino S., 2012.

45. Bostjan Brumen. Security Analysis and Improvements to the PsychoPass Method / Bostjan Brumen, Marjan Heričko, Ivan Rozman // J Med Internet Res / Bostjan Brumen, Marjan Heričko, Ivan Rozman., 2013.

ДОДАТОК А

СПИСОК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Статті у наукових фахових виданнях України:

1. (CEUR Workshop Proceedings - Scopus) Buchyk S., Lukova-Chuiko N., Toliupa S., Piatyhor V. and Buchyk O. (2021) Diceware password generation algorithm modification based on pseudo-random sequences. Cybersecurity Providing in Information and Telecommunication Systems, 26 October 2021, Kyiv, Ukraine.

2. Робота зайняла 29 місце у рейтингу Студентських наукових робіт Всеукраїнського конкурсу зі спеціальності «Кібербезпека» у 2020-2021 н.р.

Тези наукових доповідей:

1. Serhii Buchyk PASSWORD COMPLEXITY ANALYSIS OF DICEWARE PASSWORD GENERATION ALGORITHM MODIFICATION BASED ON PSEUDO-RANDOM SEQUENCES / Serhii Buchyk, Vitalii Piatyhor // VIII International conference “Information Technology and Implementation”, December 1-3, 2021.