МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Чернігівська політехніка»

Навчально-науковий інститут електроних та інформаційних технологій

Кафедра кібербезпеки та математичного моделювання

**ЗВІТ**

про виконання лабораторної роботи

з дисципліни «Основи криптографічного захисту інформації»

Виконав: здобувач 3 курсу, групи КБ-221 Регент Анастасія

Перевірив: Шелест Михайло Євгенович

НУ «Чернігівська політехніка» 2025

Лабораторна робота № 5

**Хешування**

**Мета завдання:** Вам необхідно захешувати надані значення, вибираючи метод хешування що найкраще підходить для досягнення мети описаної в кожному завданні.

Завдання № 1

Необхідно підготувати паролі користувачів для збереження у базі даних.

qwertyuiop

sofPed-westag-jejzo1

f3Fg#Puu$EA1mfMx2

TIMCfJDkKBRm9/zwcFbHhE6zaMcSxR7nke1mJKcVqXpvCzg69d7Mf2quanMoAfmPJXyqT4gyGpLoL1lTHoqmwVmaUwrpOPRecB8GAU17eUJJHiksv3qrqcVxhgpMkX/UlKaLdFSwFIr7cVoJmBqQ/buWzxJNCIo7qbtIi3fSi62NwMHh

**Додаткова інформація**

Подумайте, чи вхідні значення з якими ви працюєте мають високу ентропію? Це є вирішальним фактором для вибору алгоритму.

Пам’ятаємо, що ми працюємо з паролями користувачів до якогось веб-сервісу. Відповідно, серверна частина повинна бути спроможна перевіряти правильність паролів тисяч, а то й мільйонів користувачів, що автентифікуються на сервісі щодня. Це впливає на обрані параметри хешування.

Окремо опишіть обраний метод хешування та використані параметри. Чому обрали саме цей метод та параметри?

**Обраний метод:** Argon2id (через бібліотеку argon2-cffi)

**Параметри:**

time\_cost=2 — баланс між безпекою та продуктивністю

memory\_cost=65536 — захист від атак типу ASIC/FPGA

parallelism=2 — дозволяє паралельне обчислення

salt\_len=16, hash\_len=32

**Чому саме цей метод:**

1. Стійкий до сучасних атак (включаючи атаки по часу, на GPU)

2. Відповідає рекомендаціям OWASP та NIST

3. Добре масштабується для великої кількості користувачів

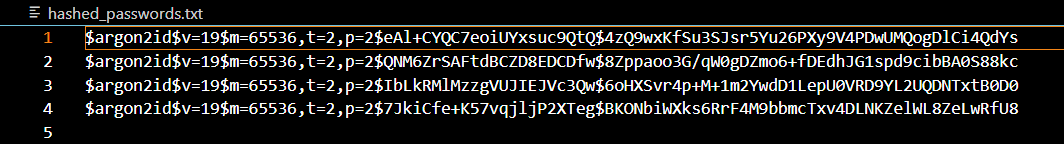


Рисунок 1 – Результат

Завдання № 2

**Згенерувати імітовставку (Message Authentication Code, MAC) для наданого шифртексту:**

76c3ada7f1f7563ff30d7290e58fb4476eb12997d02a6488201c075da52ff3890260e2c89f631e7f919af96e4e47980a

**Головний ключ, використаний для зашифрування:**

63e353ae93ecbfe00271de53b6f02a46

**IV**: 75b777fc8f70045c6006b39da1b3d622

**Вам необхідно обрати:**

1. алгоритм, що забезпечує автентифікацію даних;

2. ключ для обчислення імітовставки;

3. які саме дані включати до генерування імітовставки.

**Додаткова інформація**

Вам непотрібно реалізовувати сам алгоритм хешування, можна використовувати готові реалізації з Python бібліотек pycryptodome або cryptography.

Опис того, чому вибрали саме такий алгоритм та ключ для генерування імітовставки (MAC).

**Для створення імітовставки було використано алгоритм AES-CMAC, оскільки:**

1. Він сумісний із шифруванням AES, яке застосовувалось у завданні.

2. Є стандартом NIST і забезпечує високий рівень захисту.

3. Підтримується бібліотекою pycryptodome, що спрощує реалізацію.

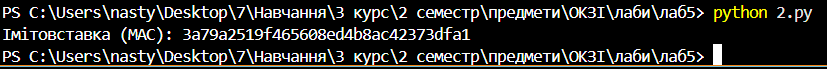


Рисунок 2 – Результат

**Завдання № 3**

Вам необхідно реалізувати функцію, що приймає на вхід пароль користувача та на основі нього генерує для користувача криптографічний ключ придатний для шифрування даних алгоритмом AES-128.

**Сигнатура функції може виглядати наступним чином:**

def derive\_key(username, password)

**Ви можете зберігати додаткові метадані для кожного користувача у текстовому файлі, наприклад у форматі JSON:**

[

{

"username": "John Doe",

"param": "value",

"salt": "75b777fc8f70045c6006b39da1b3d622",

...

},

...

]

Варто пам’ятати, що паролі користувачів зазвичай мають низьку ентропію. При цьому ключі шифрування повинні бути псевдовипадковими, і тому вимагають максимальної ентропії. У реалізації необхідно використати відповідні хеш-функції, що здатні забезпечити ці вимоги.

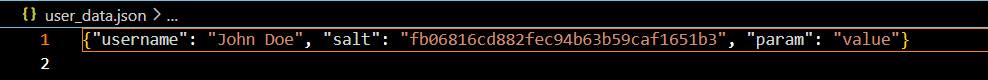


Рисунок 3 – Результат 1.1



Рисунок 4 – Результат 1.2

**Посилання на репозиторій:** <https://github.com/Anasteishha/-/tree/main>

Окремо опишіть чому вибрали саме такі алгоритми для забезпечення описаних вимог.

SHA-256 вибрано через високу безпеку, оскільки це частина сімейства SHA-2, яке є одним із найнадійніших на сьогоднішній день. Має довжину хешу 256 біт, що забезпечує високий рівень криптографічної стійкості, а також широко використовується в криптографічних протоколах.

os.urandom() обрано для генерації криптографічно стійких випадкових чисел. Воно використовує системні джерела випадковості для створення унікальних і непередбачуваних солей, що забезпечує додаткову безпеку для кожного користувача.

PBKDF2 (Password-Based Key Derivation Function 2) вибрано для перетворення паролів у криптографічні ключі, оскільки він включає велику кількість ітерацій (100000), що ускладнює атаки типу "brute force". Це є стандартним і перевіреним методом, рекомендованим для таких задач.

JSON використовується для зберігання метаданих, оскільки це зручний і зрозумілий формат для зберігання структурованих даних, таких як ім'я користувача, сіль і додаткові параметри.

**Контрольні запитання**

1. Які основні задачі вирішують хеш-функції у криптографії?

Хеш-функції в криптографії вирішують кілька важливих задач: перевірка цілісності даних, генерація цифрових підписів, а також використовуються для захисту паролів та створення унікальних ідентифікаторів для повідомлень чи файлів.

2. Яка різниця між хешуванням і шифруванням?

Хешування — це процес перетворення вхідних даних у фіксований розмір вихідного хешу, який є одностороннім і не дозволяє відновити вихідні дані. Шифрування ж призначене для того, щоб захистити інформацію таким чином, щоб її можна було відновити за допомогою певного ключа.

3.Які основні відмінності між криптографічними та некриптографічними хеш-функціями?

Основна відмінність між криптографічними та некриптографічними хеш-функціями полягає в тому, що криптографічні функції повинні мати властивості, як стійкість до колізій (не існує двох різних вхідних даних, що дають однаковий хеш), односторонність та ефективність у обчисленнях. Некриптографічні функції можуть не мати таких властивостей і використовуються для менш критичних задач.

4. У чому полягає проблема колізій у хеш-функціях?

Проблема колізій у хеш-функціях полягає в тому, що два різні набори даних можуть дати однаковий хеш. Це дозволяє зловмиснику підробити дані, не змінюючи їх хеш, що може призвести до серйозних уразливостей у системах безпеки.

5. Чому функція SHA-1 вважається небезпечною для використання?

SHA-1 вважається небезпечною для використання, оскільки були виявлені вразливості, що дозволяють знайти колізії, тобто два різних повідомлення, які мають однаковий хеш. Це ставить під загрозу її використання в криптографічних протоколах, таких як цифрові підписи.

6. Що таке SHA-3, і чим вона відрізняється від SHA-2?

SHA-3 — це останній стандарт хешування в криптографії, розроблений для покращення стійкості до атак. Відмінність між SHA-2 та SHA-3 полягає в тому, що SHA-3 використовує інший алгоритм (Keccak) і не є прямим покращенням SHA-2, а новим стандартом з іншою архітектурою.

7. Які криптографічні стандарти використовують MD5 і чому вони більше не рекомендовані?

MD5 використовувався в багатьох криптографічних стандартах, але зараз більше не рекомендується через виявлені уразливості, що дозволяють знаходити колізії. Через це багато криптографічних систем більше не використовують MD5 для підписів або хешування.

8. Що таке HMAC, і як він забезпечує автентифікацію повідомлень?

HMAC (Hash-based Message Authentication Code) — це алгоритм для перевірки цілісності та автентичності повідомлень. Він використовує секретний ключ разом із хеш-функцією для створення автентифікаційного коду, що дозволяє перевірити, чи не було змінено повідомлення.

9. Що таке "сіль" (salt) у хешуванні паролів і навіщо вона потрібна?

Сіль (salt) — це випадкові дані, які додаються до пароля перед хешуванням, щоб уникнути використання попередньо згенерованих таблиць для швидкої атаки (rainbow tables). Сіль робить хешування пароля унікальним для кожного користувача.

10. Чому небезпечно зберігати хеш пароля без додавання солі?

Зберігання хешу пароля без солі є небезпечним, оскільки це дозволяє зловмиснику використовувати попередньо обчислені таблиці (rainbow tables) для швидкого пошуку пароля. Сіль гарантує, що навіть однакові паролі будуть мати різні хеші.