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 № 4

**Режим роботи шифрів та особливості реалізації**

**Мета лабораторної роботи**: перевірити ваше розуміння режиму CBC та проблеми що виникають з ним через некоректну реалізацію та відсутність автентифікації.

Завдання № 1

**«Кепський CBC»**

Ваша мета – віднайти гасло (FLAG) до задачі [Lazy CBC](https://aes.cryptohack.org/lazy_cbc/) на платформі CryptoHack.

На сторінці задачі наданий код (Python скрипт) що реалізує функції зашифрування і розшифрування алгоритмом AES-128 в режимі CBC, які використовує сервер для відповіді на запити. Проте реалізація має критичну вразливість. Ваша задача – скористатися цією вразливістю для отримання значення секретного гасла (FLAG).

Опис функцій бекенду сервера:

1. encrypt (https://aes.cryptohack.org/lazy\_cbc/encrypt/<plaintext>/) - функція приймає на вхід повідомлення (строка у форматі hex) та перевіряє що довжина повідомлення кратна 16 байтам (розмір блоку AES). Якщо довжина коректна, функція зашифровує повідомлення використовуючи секретний ключ KEY в режимі CBC. При цьому значення ключа також використовується у якості вектора ініціалізації IV. Функція повертає отриманий шифртекст у hex форматі.

2. get\_flag: (https://aes.cryptohack.org/lazy\_cbc/get\_flag/<key>/) - функція приймає на вхід строку у hex форматі. Якщо значення вхідного значення співпадає з ключем шифрування KEY, функція повертає необхідний вам флаг FLAG закодований у hex форматі.

3. receive: (https://aes.cryptohack.org/lazy\_cbc/receive/<ciphertext>/) - функція приймає на вхід шифртекст (у форматі hex) та перевіряє чи його довжина кратна 16 байтам. Якщо довжина коректна, функція розшифровує шифртекст на ключі KEY в режимі AES-128-CBC (знову використовуючи ключ у якості вектора ініціалізації IV). Якщо розшифровані дані коректно декодуються в ASCII текст, функція повертає успішне повідомлення success. В протилежному випадку повертається помилка разом з розшифрованими даними в hex форматі.

**Вразливість**: головна вразливість цієї схеми полягає в повторному використанні ключа замість вектора ініціалізації в режимі CBC. Це дозволяє маніпулювати дані передбачливим чином для розкриття зашифрованих даних.

**Ваша задача:** використовуючи запити до функцій encrypt та receive сформувати спеціальний шифртекст таким чином, що це дозволить вам віднайти значення ключа KEY й отримати гасло.

**Додаткова інформація:**

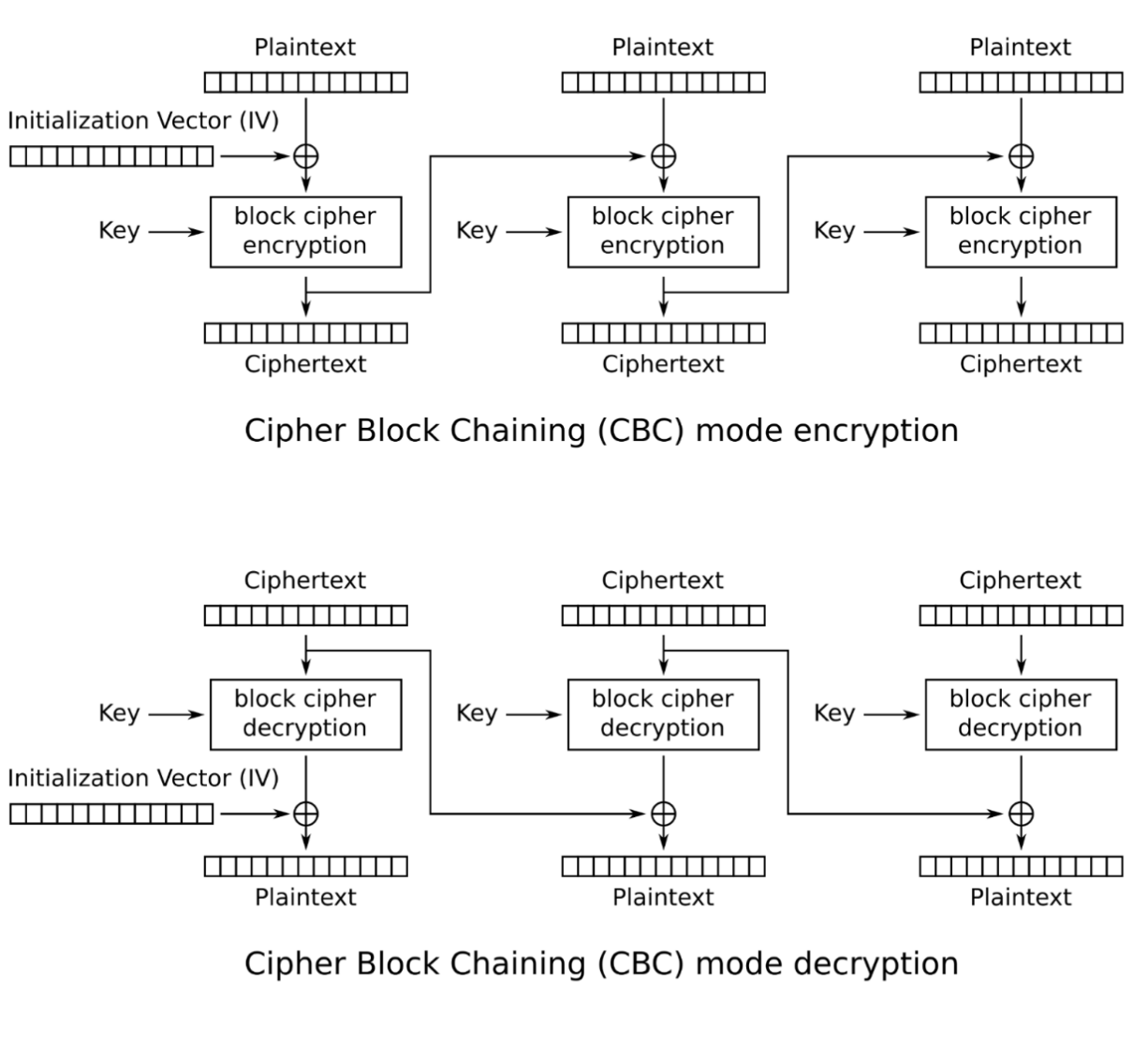
1. Проаналізуйте властивості CBC режиму та як вектор ініціалізації IV впливає на дані під час зашифровання та розшифрування.

2. Подумайте як маніпуляція шифртексту може вплинути на розшифроване повідомлення.

3. Дослідить надсилання спеціально сформатованих шифртекстів у функцію receive та проаналізуйте відповіді сервера.

4. Для надислання запитів на сервер рекомендуємо використовувати бібліотеку [requests](https://docs.python-requests.org/en/latest/index.html).

5. Для криптографічних операцій можете користуватись бібліотекою [pycryptodome](https://pycryptodome.readthedocs.io/en/latest/src/introduction.html) або [cryptography](https://cryptography.io/).



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

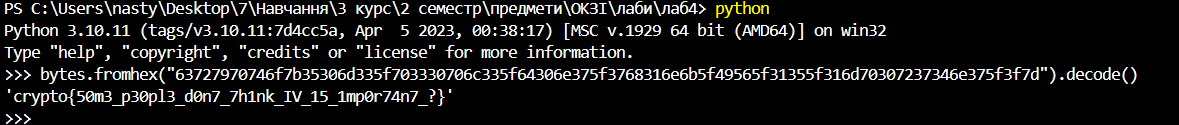


Рисунок 1 – Значення отриманого гасло

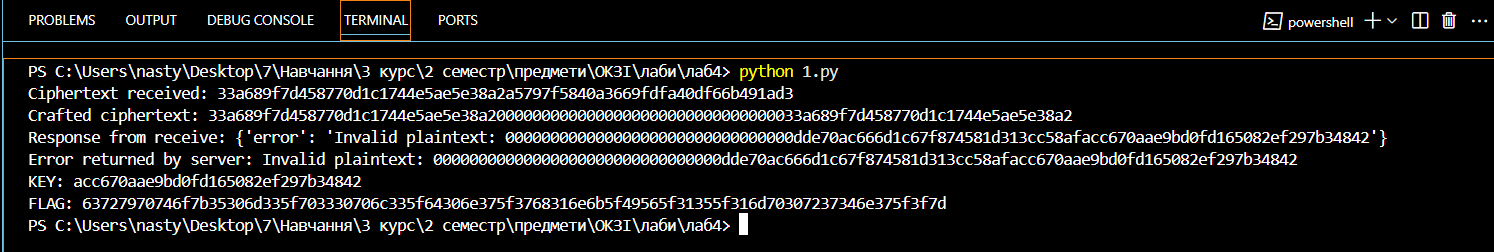


Рисунок 2 – Результат

**Контрольні запитання**

**Режими роботи блокового шифру**

1. Яка різниця між режимами ECB та CBC? У яких випадках не варто використовувати ECB?  
ECB (Electronic Codebook) шифрує кожен блок окремо. Недолік — однакові блоки відкритого тексту шифруються в однакові шифроблоки. Не варто використовувати ECB у випадках, коли важлива конфіденційність структури даних, наприклад, у зображеннях.  
CBC (Cipher Block Chaining) використовує XOR кожного блоку з попереднім шифроблоком перед шифруванням. Потрібен ініціалізаційний вектор (IV). Надійніший, оскільки знищує повторення.

2. Що таке режим CTR, і як він забезпечує потокове шифрування?  
CTR (Counter Mode) перетворює блочний шифр на потоковий. Шифруються послідовні значення лічильника (Nonce + Counter), які потім XOR-яться з відкритим текстом. Це дозволяє паралельну обробку блоків і підходить для потокового шифрування.

3. Як працює режим GCM, і чому він популярний у сучасних протоколах?  
GCM (Galois/Counter Mode) поєднує шифрування у режимі CTR з автентифікацією на основі множення у полі GF(2¹²⁸). Забезпечує конфіденційність і цілісність даних. Дуже швидкий, підтримує паралелізм, активно використовується в TLS, IPsec, SSH.

4. У чому недоліки повторного використання Nonce в режимах CTR, GCM або CBC?  
У CTR і GCM повторний Nonce призводить до повторення ключового потоку, що дозволяє атакувальнику аналізувати XOR і потенційно розкрити дані. У CBC повторний IV створює повторювані шифроблоки, що компрометує безпеку.

5. Для чого потрібні AAD (додаткові автентифіковані дані), і в яких режимах вони використовуються?  
AAD — це нешифрована, але автентифікована інформація, наприклад, заголовки протоколів. Вони дозволяють перевірити цілісність критичних метаданих. Використовуються в режимах GCM, CCM, EAX, ChaCha20-Poly1305.

**Автентифіковане шифрування**

1. Що таке автентифіковане шифрування і які його основні переваги?  
Автентифіковане шифрування (AE) забезпечує одночасно конфіденційність, цілісність і автентичність. Основна перевага — захист від модифікації зашифрованих даних.

2. Як відрізняється підхід Encrypt-then-Authenticate від Authenticate-then-Encrypt?  
Encrypt-then-Authenticate: спочатку шифрується, потім створюється тег автентичності — вважається найбезпечнішим.  
Authenticate-then-Encrypt: спочатку створюється MAC, потім усе шифрується — має слабкі місця.  
Encrypt-and-Authenticate: обидві операції паралельно — прикладом є GCM і ChaCha20-Poly1305.

3. Назвіть і порівняйте три основні режими автентифікованого шифрування: CCM, GCM та EAX.  
CCM поєднує CTR та CBC-MAC, не підтримує паралелізм, застосовується у TLS.  
GCM використовує CTR та GHASH, дуже швидкий і підтримує паралельну обробку.  
EAX — комбінація CTR та OMAC, не має патентних обмежень, частково підтримує паралелізм.

4. У чому перевага ChaCha20-Poly1305 порівняно з AES-GCM?  
ChaCha20-Poly1305 краще працює на пристроях без апаратної підтримки AES. Стійкий до атак через побічні канали, швидкий і простий в реалізації. Використовується у TLS 1.3, WireGuard, SSH.

5. Чому повторне використання Nonce є критичною помилкою для автентифікованого шифрування?  
Повторення Nonce в AE, особливо в GCM або ChaCha20-Poly1305, може розкрити ключовий потік або дозволити фальсифікацію тегів MAC. Це повна компрометація безпеки.

**Рекомендації**

1. Які практичні рекомендації можна надати для безпечного використання блокових шифрів у системах?  
Уникайте ECB. Завжди використовуйте унікальні Nonce або IV. Віддавайте перевагу автентифікованому шифруванню. Користуйтеся перевіреними бібліотеками. Ключі зберігайте у безпечних сховищах або апаратних модулях.

2. Чому важливо використовувати криптографічно безпечні генератори випадкових чисел для створення Nonce?  
Nonce має бути унікальним і непередбачуваним. Використання слабких генераторів призводить до повторного використання Nonce або передбачуваності ключового потоку. Застосовуйте криптографічні ГВЧ (наприклад, secrets, /dev/urandom).

3. У яких випадках потрібно віддавати перевагу автентифікованому шифруванню замість простого шифрування?  
Завжди, коли важлива не лише конфіденційність, а й цілісність — при передачі повідомлень, збереженні конфігурацій, захисті транзакцій. Просте шифрування не захищає від змін даних.

4. Чи можна використовувати той самий ключ для різних режимів шифрування? Чому?  
Ні. Різні режими мають різні криптографічні властивості. Використання одного ключа для різних режимів послаблює безпеку і може призвести до витоку інформації. Краще використовувати розділення ключів за допомогою KDF.

5. Як правильно зберігати та обмінюватися ключами для забезпечення безпеки?  
Передача ключів має здійснюватись через захищені канали (TLS, SSH). Для зберігання — апаратні модулі безпеки (HSM, TPM) або зашифровані сховища. Не зберігайте ключі у відкритому вигляді чи в коді.