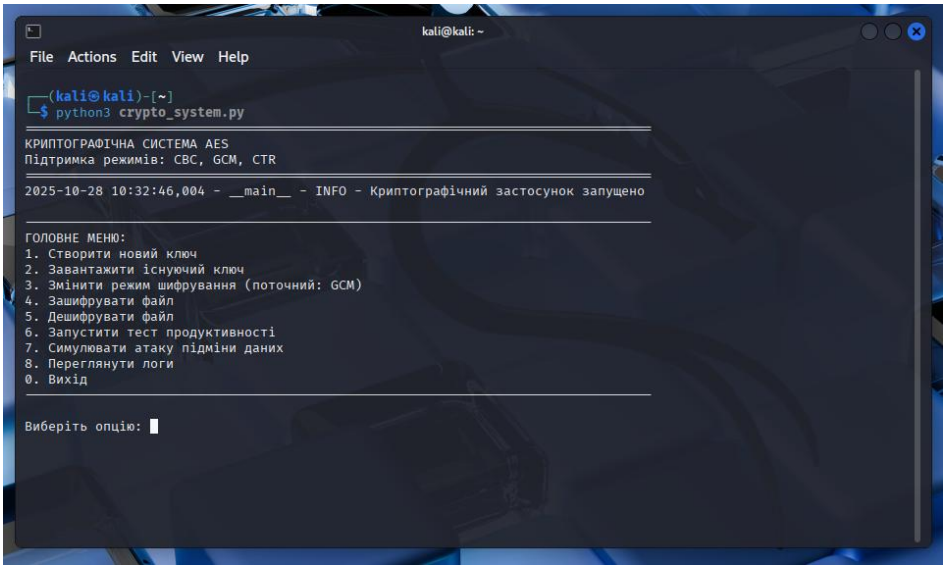
1. Запуск програми

python3 crypto_system.py



2. Створення ключа

Меню → 1

Key ID: project_1

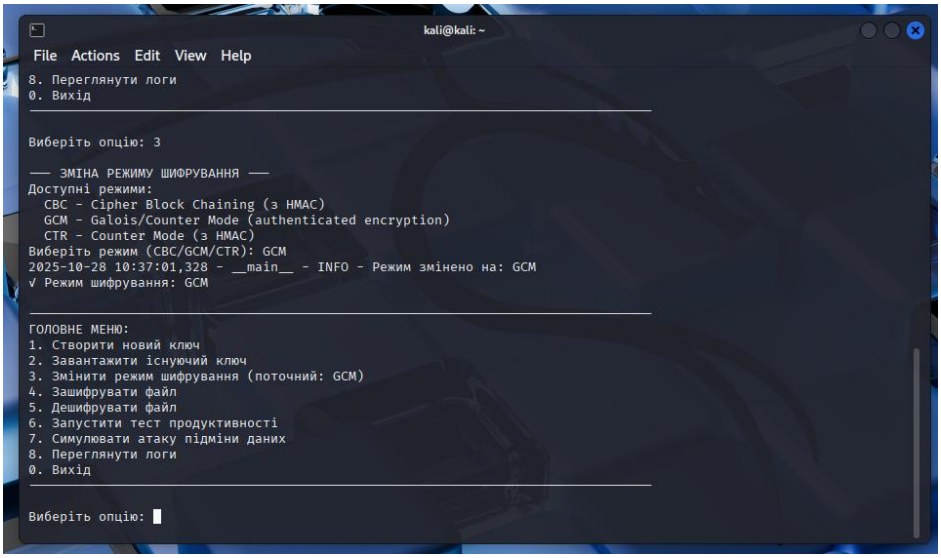
Password: MyS3cur3P@ssw0rd!

Length: 32

3. Вибір режиму

Меню → 3

Mode: GCM

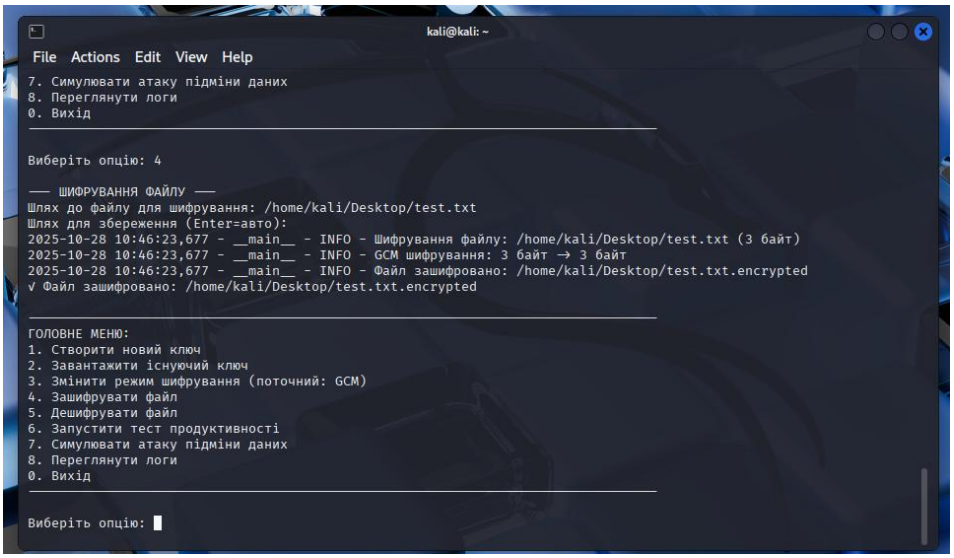


4. Шифрування файлу

Меню → 4

Input: test.txt

Output: test.txt.encrypted

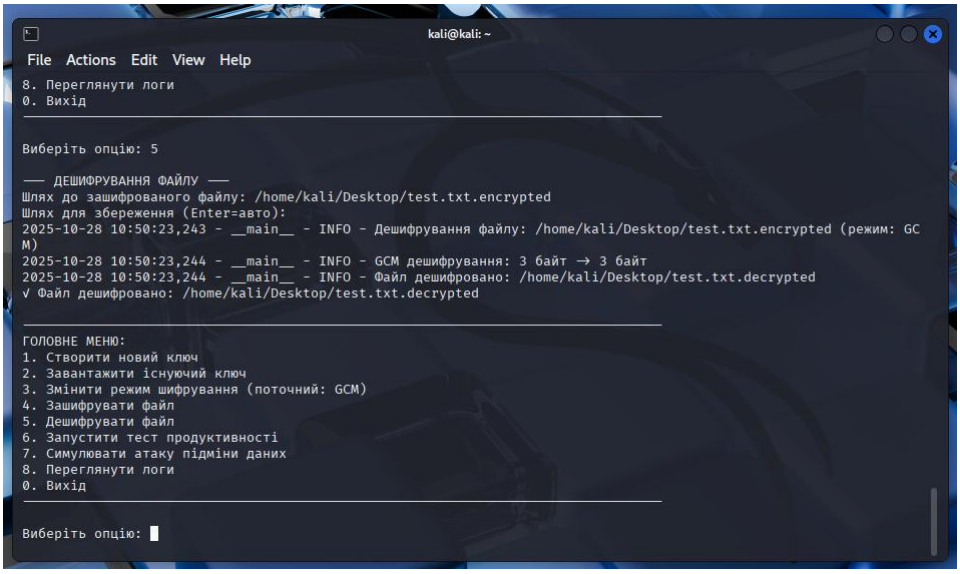


5. Дешифрування

Меню → 5

Input: test.txt.encrypted

Output: test.txt.decrypted

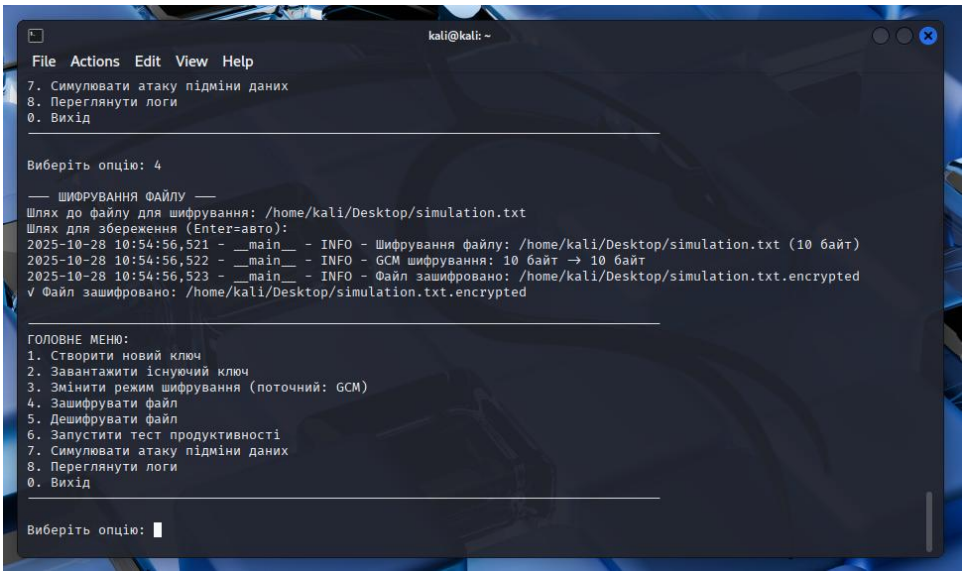


Тестування безпеки

1. Шифрування тестового файлу

Меню → 4

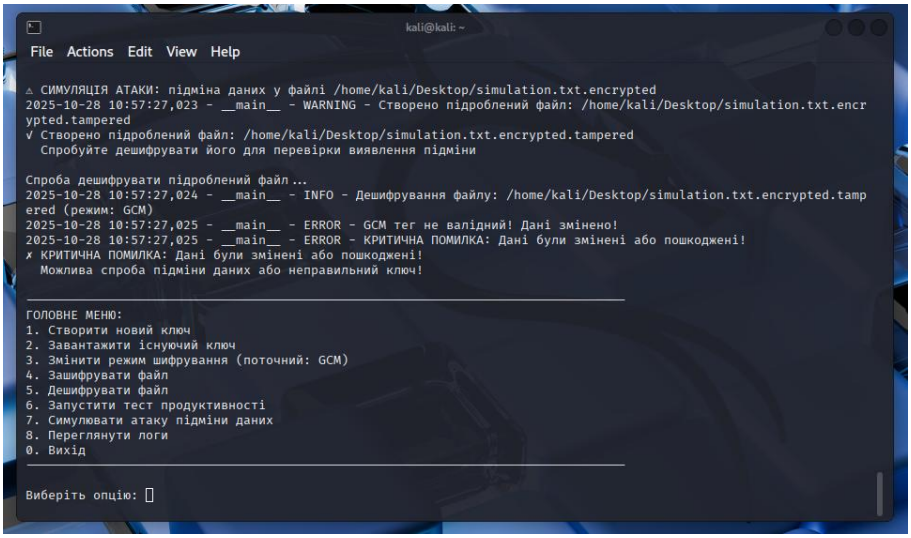
Input: simulation.txt



2. Симуляція атаки

Меню → 7

File: simulation.txt.encrypted

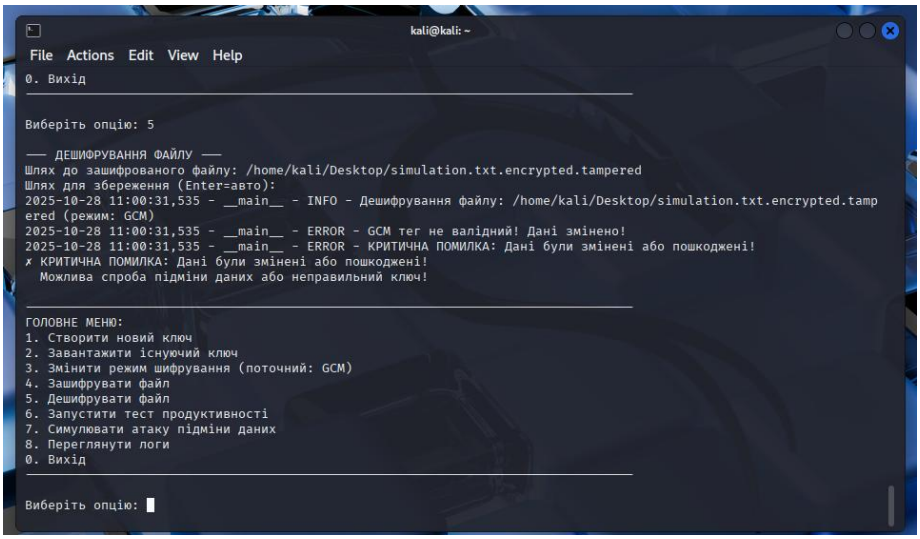


3. Спроба дешифрування підробленого файлу

Меню → 5

Input: simulation.txt.encrypted.tampered

Очікується помилка автентифікації

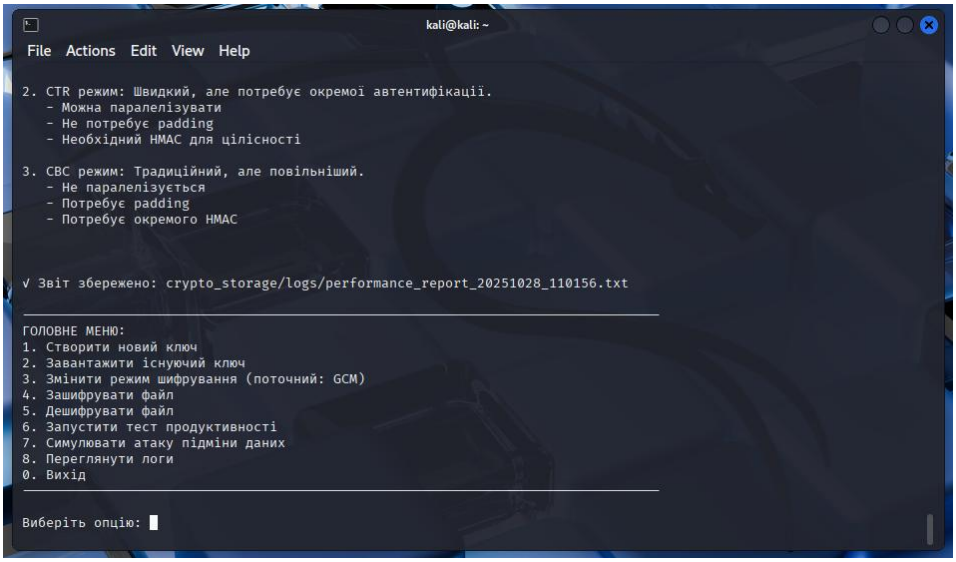


Аналіз продуктивності

Меню → 6

Система протестує всі режими на різних розмірах

Звіт буде збережено в crypto_storage/logs/


kali@kali: ~
File Actions Edit View Help
2. CTR режим: Швидкий, але потребує окремої автентифікації.
- Можна паралелізувати
- Не потребує padding
- Необхідний HMAC для цілісності
3. CBC режим: Традиційний, але повільніший.
- Не паралелізується
- Потребує padding
- Потребує окремого HMAC
√ Звіт збережено: crypto_storage/logs/performance_report_20251028_110156.txt
ГОЛОВНЕ МЕНЮ:
1. Створити новий ключ
2. Завантажити існуючий ключ
3. Змінити режим шифрування (поточний: GCM)
4. Зашифрувати файл
5. Дешифрувати файл
6. Запустити тест продуктивності
7. Симулювати атаку підміни даних
8. Переглянути логи
0. Вихід
Виберіть опцію: