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 № 3

**Симетричні шифри**

Завдання № 1

**Реалізація та дослідження алгоритму "Simplified AES"**

**Мета завдання:**

1. Ознайомитися з принципами роботи блочного шифру AES на прикладі спрощеної версії S-AES.

2. Реалізувати алгоритм S-AES для шифрування та дешифрування коротких повідомлень.

3. Зрозуміти базові криптографічні операції: SubNibble, ShiftRows, MixColumns, AddRoundKey.

**Теоретичні відомості**

Simplified AES (S-AES) — це спрощена модель стандартного AES, яка зберігає структуру шифрування, але використовує менші блоки та простіші операції для полегшеного аналізу.

|  |  |
| --- | --- |
| Параметр | S-AES |
| Розмір блоку | 16 біт (2 байти) |
| Ключ | 16 біт |
| Кількість раундів | 2 |
| S-блок | 4-бітний |
| Математичне поле | GF(2⁴) |

**Опис етапів алгоритму**

**Етапи шифрування:**

1. AddRoundKey — XOR блоку з ключем.

2. SubNibble — заміна ніблів за допомогою S-блоку.

3. ShiftRows — зсув рядків.

4. MixColumns — перемішування стовпців.

5. AddRoundKey — знову додаємо ключ.

**Фінальний раунд:**

1. SubNibble → ShiftRows → AddRoundKey (без MixColumns).

**S-блок (SubNibble):**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 9 | 4 | A | B | D | 1 | 8 | 5 | 6 | 2 | 0 | 3 | C | E | F | 7 |

**Матричне перетворення MixColumns:**

[1 4]

[4 1]

Перемноження здійснюється у полі GF(2⁴) з незвідним поліномом x⁴ + x + 1.

**Постановка завдання**

**Необхідно реалізувати S-AES для:**

1. Шифрування 16-бітного повідомлення.

2. Дешифрування зашифрованого тексту.

3. Перевірки коректності роботи шляхом шифрування та зворотного дешифрування.

**Кроки та виконання лабораторної роботи**

**Крок 1: Ініціалізація даних**

Задати вхідний текст (16 біт) у вигляді двійкового рядка або шістнадцяткового значення.

Вказати ключ шифрування (16 біт).



Рисунок 1 – Код до кроку 1

**Крок 2: Реалізація функцій**

AddRoundKey — XOR блоку даних з ключем.



Рисунок 2 – Код до кроку 2.1

SubNibble — заміна кожного ніблу через S-блок



Рисунок 3 – Код до кроку 2.2

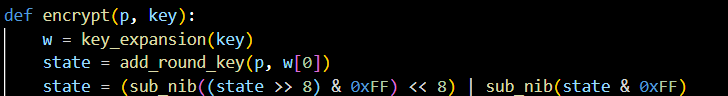


Рисунок 4 – Код до кроку 2.3

ShiftRows — зсув рядків:

Перший рядок — без змін.

Другий рядок — зсувається на 1 нібл вліво.

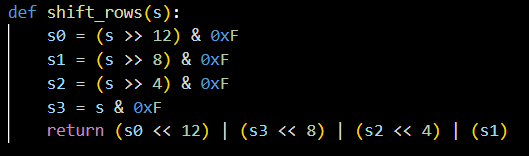


Рисунок 5 – Код до кроку 2.4

MixColumns — виконати перемноження матриці стану на матрицю MixColumns у полі GF(2⁴).

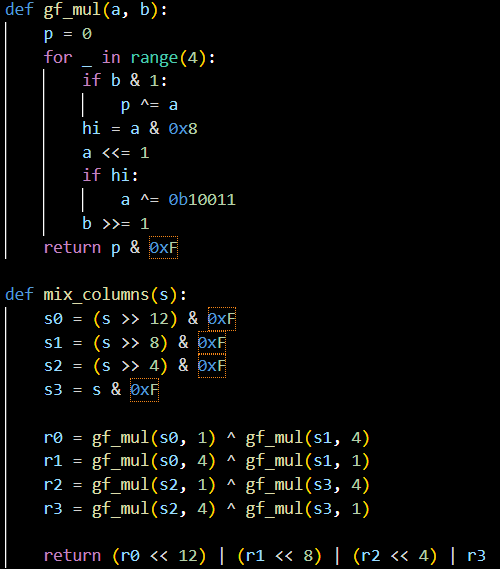


Рисунок 6 – Код до кроку 2.5

KeyExpansion — створити підключі для кожного раунду.

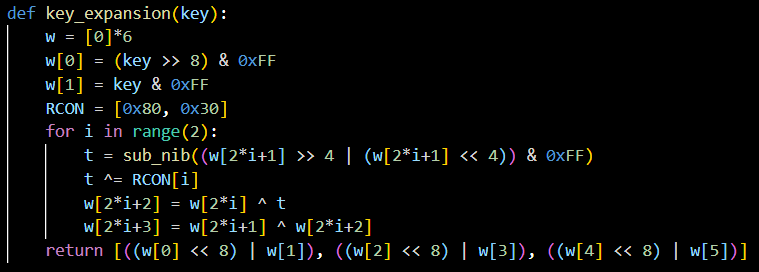


Рисунок 7 – Код до кроку 2.6

**Крок 3: Шифрування даних**

**Виконати всі етапи шифрування:**

AddRoundKey → SubNibble → ShiftRows → MixColumns → AddRoundKey → SubNibble → ShiftRows → AddRoundKey.

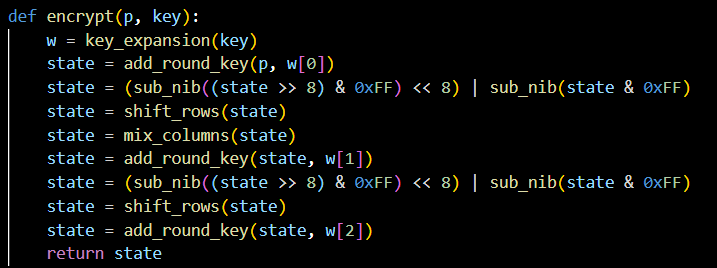


Рисунок 8 – Код до кроку 3

**Крок 4: Дешифрування даних**

**Реалізувати обернені функції:**

1. InvSubNibble

2. InvShiftRows

3. InvMixColumns

Переконатися, що результат збігається з оригінальним текстом.

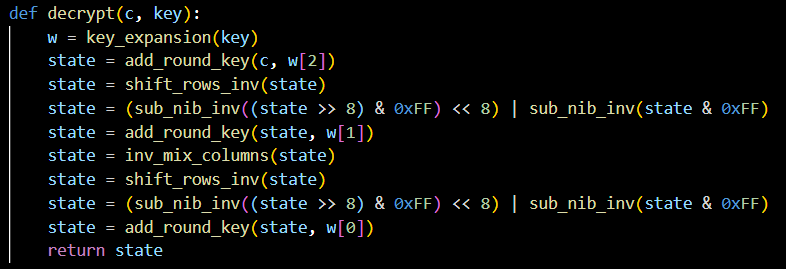


Рисунок 9 – Код до кроку 4

**Крок 5: Перевірка правильності**

Зашифрувати та розшифрувати кілька повідомлень.

Порівняти результати з очікуваними.

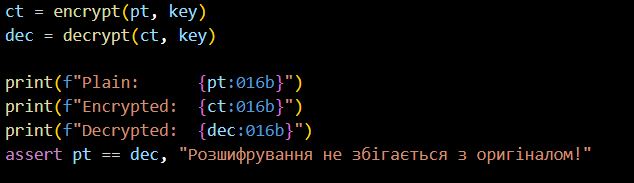


Рисунок 10 – Код до кроку 5

**Контрольні запитання**

1. Чому використовується поле GF(2⁴) в S-AES?  
Тому що S-AES працює з 4-бітовими ніблами. Поле GF(2⁴) ідеально підходить для операцій над такими блоками, зокрема в MixColumns.

2. Яка роль S-блоків у забезпеченні криптостійкості?  
S-блоки вводять нелінійність і плутанину, що робить зв’язок між відкритим текстом і шифротекстом складнішим, захищаючи від атак.

3. Чому на останньому раунді не використовується MixColumns?  
Щоб спростити зворотне дешифрування та завершити шифрування менш дифузним, завершальним перетворенням.

4. Чи можна збільшити кількість раундів для підвищення стійкості S-AES?  
Так, більше раундів підвищує стійкість, але для навчальних цілей 2 раунди достатні.

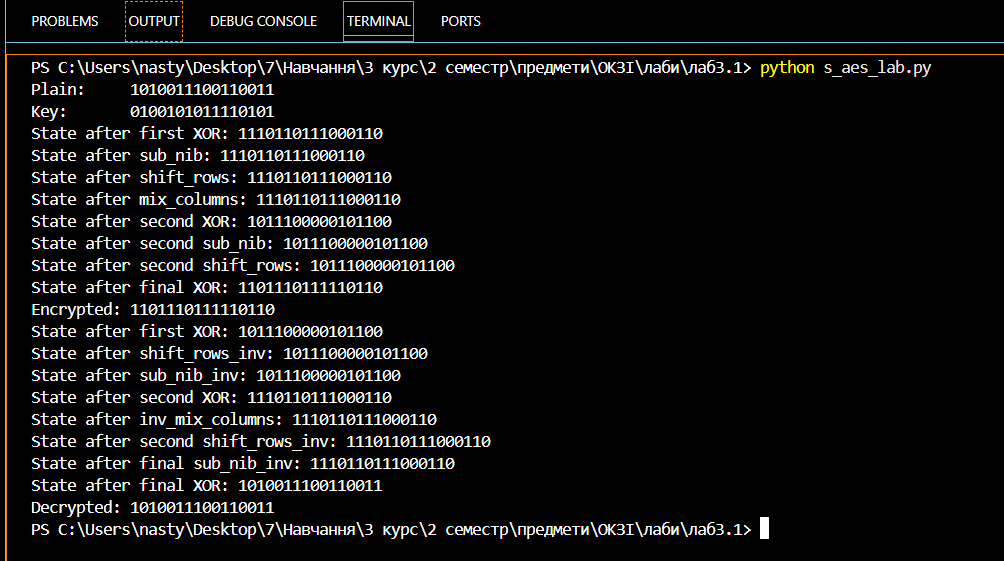


Рисунок 11 – Результат

Завдання № 2

**Реалізація та дослідження шифру ChaCha20**

**Мета роботи:**

1. Ознайомитися з принципами роботи потокового шифру ChaCha20.

2. Реалізувати спрощену версію ChaCha20 для генерації ключового потоку.

3. Виконати шифрування та дешифрування тексту.

4. Дослідити вплив зміни nonce та лічильника на результат шифрування.

**Теоретичні відомості**

ChaCha20 — це потоковий шифр, який працює шляхом генерації ключового потоку і застосування операції XOR з відкритим текстом. Основою шифру є проста операція Quarter Round, яка забезпечує дифузію та криптостійкість.

**Основні параметри ChaCha20:**

**Ключ:** 256 біт (32 байти)

**Nonce:** 96 біт (12 байт)

**Лічильник:** 32 біти

**Кількість раундів:** 20 (10 подвійних)

**Постановка завдання**

**Реалізувати спрощену версію ChaCha20**:

1. Ініціалізація стану.

2. Реалізація операції **Quarter Round**.

3. Генерація ключового потоку.

**Зашифрувати/розшифрувати текст**:

1. Використати згенерований ключовий потік для шифрування відкритого тексту.

2. Перевірити правильність дешифрування.

**Перевірити вплив nonce та лічильника**:

1. Змінити **nonce** та **лічильник** і порівняти результати.

2. З’ясувати, як зміни впливають на ключовий потік і шифртекст.

**Етапи виконання лабораторної роботи**

**Крок 1: Ініціалізація стану ChaCha20**

Стан складається з 16 слів по 32 біти:

Стан = [Константа (4 слова) | Ключ (8 слів) | Лічильник (1 слово) | Nonce (3 слова)]

Використати константу: "expand 32-byte k".

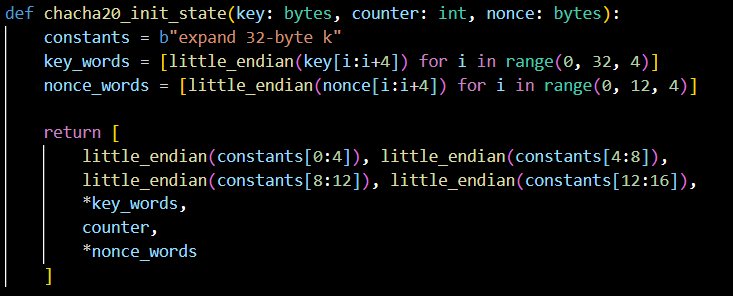


Рисунок 12 – Код до кроку 1

**Крок 2: Реалізація основної операції (Quarter Round)**

**Quarter Round(a, b, c, d):**

a += b; d ^= a; d <<<= 16

c += d; b ^= c; b <<<= 12

a += b; d ^= a; d <<<= 8

c += d; b ^= c; b <<<= 7

Операції виконуються по модулю 2³².

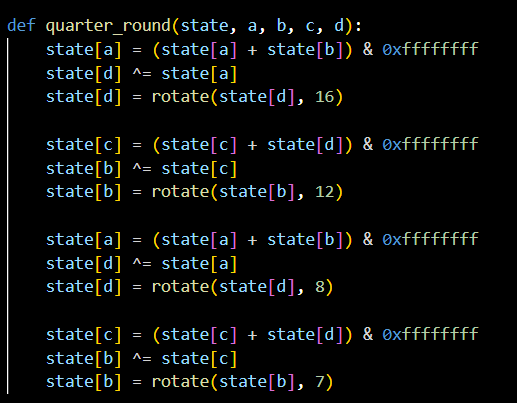


Рисунок 13 – Код до кроку 2

**Крок 3: Реалізація повного циклу шифрування**

Виконати **20 раундів** (10 подвійних):

**Column Round** → **Diagonal Round**

Після 20 раундів додати результат до початкового стану.

Згенерований ключовий потік XOR'ити з відкритим текстом.

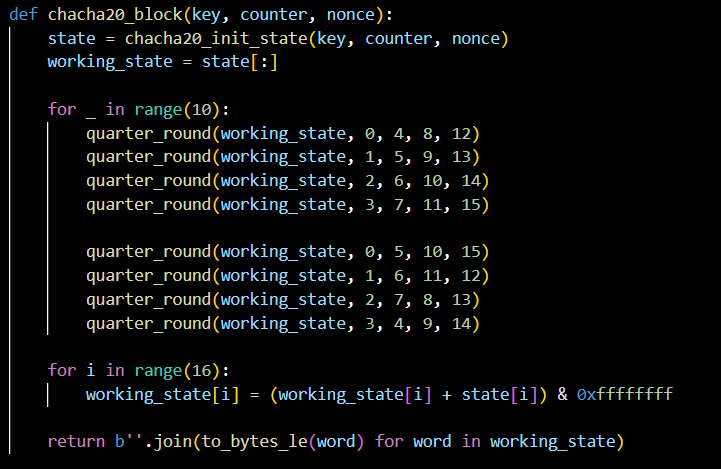


Рисунок 14 – Код до кроку 3

**Крок 4: Дешифрування тексту**

Виконати операцію XOR між шифртекстом та тим же ключовим потоком.

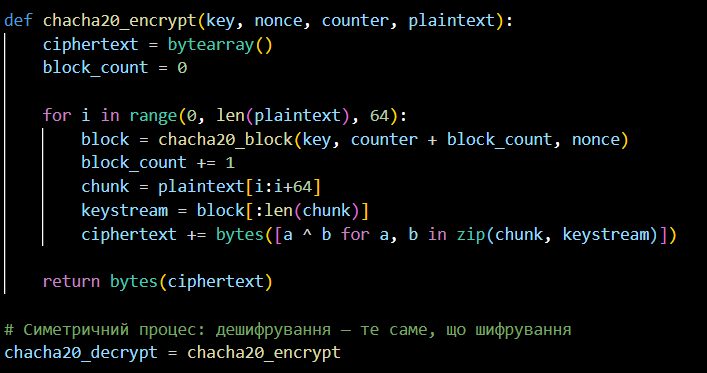


Рисунок 15 – Код до кроку 4

**Крок 5: Аналіз впливу nonce та лічильника**

Змінити nonce та повторити шифрування.

Змінити лічильник та повторити шифрування.

Порівняти результати шифртекстів та оцінити вплив змін.

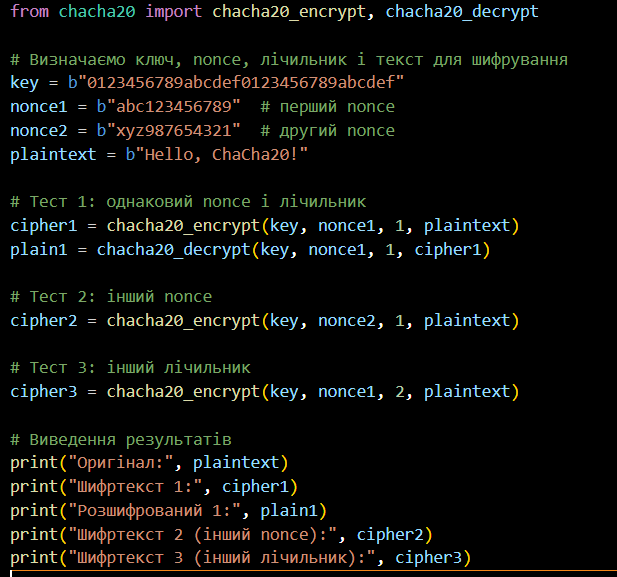


Рисунок 16 – Код до кроку 5

**Контрольні питання**

1. Що таке nonce у ChaCha20 і чому важливо уникати його повторень?

Nonce — унікальний "одноразовий" вектор, який запобігає повторному використанню ключового потоку. Повторення — критична вразливість.

2. Як впливає зміна лічильника на ключовий потік?

Лічильник змінює блок ключового потоку (новий шифртекст).

3. Чому шифр ChaCha20 вважається стійким до відомих атак?

ChaCha20 захищений від атак типу битових зіставлень, має сильну дифузію, не має слабких ключів.

4. Порівняйте ChaCha20 з AES у режимі CTR. Які основні відмінності?

AES-CTR базується на блочному шифрі + лічильнику, але ChaCha20 краще в ARM/мобільних системах і не потребує апаратного прискорення.

Висновки:

1. Якщо ми змінюємо nonce або лічильник, то шифртекст змінюється.

2. Оскільки ChaCha20 використовує унікальні значення nonce та counter, шифрування з однаковими параметрами дає однакові результати.

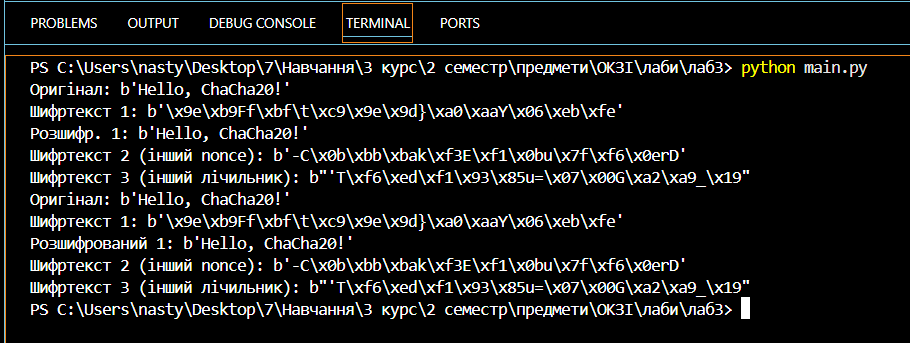
****

Рисунок 17 – Результат

Завдання № 3

**Мета завдання:** Вам необхідно вирішити задачу [ECB Oracle](https://aes.cryptohack.org/ecb_oracle/) на платформі [CryptoHack](https://cryptohack.org/).

На сторінці задачі наведено серверний код, що опрацьовує надіслані запити. Позначено невідомі значення KEY та FLAG.

Значення FLAG має наступний формат: crypto{...} де ... – випадкова послідовність символів. Вам необхідно віднайти FLAG користуючись вразливістю шифрування в режимі ECB.

Для успішного вирішення не обовʼязково реєструватись на платформі – API доступний публічно. У файлі task\_1.py наведено код для взаємодії з API задачі.

Додаткова інформація

Для виконання завдання вам знадобляться додатково 2 пакети залежностей Python, які ви можете інсталювати наступним чином:

pip install requests pycryptodome

**Формат виконання:**

Реалізуйте алгоритм знаходження невідомого FLAG. Вам треба надати код зі знайденим рішенням.

Видає «Помилку 404» – в коді не спрацьовує посилання, хоча воно є вірним. Через це не можу перевірити правильність кода.

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

**Контрольні запитання**

**Потокові шифри**

1. Що таке потоковий шифр, і для яких завдань він використовується?  
Потоковий шифр — це симетричний шифр, який шифрує дані побітово або побайтово, використовуючи псевдовипадкову послідовність (ключовий потік).  
**Використовується для:**

1. шифрування потоків даних у реальному часі (наприклад, голосовий або відеозв’язок);

2. коли потрібна мала затримка обробки;

3. у безпровідних протоколах (наприклад, WEP, Bluetooth).

2. Як працюють лінійні регістри зсуву (LFSR) у потокових шифрах?  
LFSR (Linear Feedback Shift Register) — це послідовність бітів, яка оновлюється за лінійним законом. Біт зворотного зв’язку генерується як XOR деяких позицій регістру.  
У потокових шифрах використовується для генерації ключового потоку (псевдовипадкової послідовності).

3. У чому полягає основний принцип роботи шифру RC4?  
RC4 використовує псевдовипадкову генерацію ключового потоку, який потім XOR-иться з відкритим текстом.  
Ключовий потік створюється з допомогою перестановки байтів у масиві розміром 256 елементів, які циклічно перемішуються залежно від ключа.

4. Які переваги має ChaCha20 у порівнянні з RC4?

1. Безпечніший – RC4 має численні вразливості, ChaCha20 вважається криптографічно стійким.

2. Висока швидкість і ефективність на програмному рівні.

3. Підтримує 256-бітні ключі та 96-бітні nonce.

4. Стійкий до атаки на початкові біти ключового потоку, яка характерна для RC4.

5. Які основні компоненти та принципи роботи шифру "Струмок"?  
**"Струмок" — український потоковий шифр:**

1. Основа — регістри зсуву з лінійним і нелінійним зворотним зв’язком.

2. Використовується комбінація кількох LFSR та нелінійної функції для генерації ключового потоку.

**Принцип:** генерація ключового потоку та XOR з відкритим текстом.

**Блокові шифри**

1. У чому різниця між потоковими та блоковими шифрами?

**1. Потоковий:** шифрує побітово/побайтово; використовує псевдовипадковий потік.

**2. Блоковий:** шифрує фіксовані блоки (наприклад, 128 біт); застосовує ті самі операції до кожного блоку.

2. Які властивості повинен мати блоковий шифр, щоб вважатися криптографічно стійким?

1. Велика кількість раундів.

2. Стійкість до лінійного та диференційного криптоаналізу.

3. Великий простір ключів (щонайменше 2¹²⁸).

4. Відсутність статистичних залежностей між відкритим текстом та шифртекстом.

5. Використання підстановок (S-box), перестановок, ключового змішування.

3. У чому полягає відмінність між структурою SPN (Substitution-Permutation Network) та Фейстеля?

**1. SPN:** повна обробка всього блоку — підстановки, перестановки, XOR з ключем.

**2. Фейстель:** блок ділиться на дві частини; одна змінює іншу через криптографічну функцію; легко реалізується зворотне перетворення навіть для нефункціонально оберненої криптографічної функції.

4. Які особливості архітектури AES, що забезпечують його безпеку?

Структура SPN із 10–14 раундами.

Операції: SubBytes (S-box), ShiftRows, MixColumns, AddRoundKey.

Великий розмір ключа (128/192/256 біт).

Висока стійкість до сучасних атак (в т.ч. диференційного та лінійного криптоаналізу).

5. Які параметри підтримує шифр "Калина" (розмір блоку, ключа, кількість раундів)?

Блок: 128 біт.

Ключ: 128, 256, або 512 біт.

Кількість раундів:

10 (128-бітний ключ),

14 (256-бітний ключ),

18 (512-бітний ключ).

**Атаки на симетричні шифри**

1. Які основні типи атак на симетричні шифри?

1. Атака грубої сили (Brute Force).

2. Частотний аналіз.

3. Атака з вибраним відкритим текстом (CPA).

4. Атака з вибраним шифртекстом (CCA).

5. Атаки на побудову (наприклад, лінійний чи диференційний криптоаналіз).

2. Що таке атака "грубої сили" (Brute Force), і як її можна уникнути?  
**Спроба перебрати всі можливі ключі.  
Захист:**

1. Використання довгих ключів (128 біт і більше).

2. Обмеження кількості спроб (наприклад, ліміти на спроби входу).

3. Як працює частотний аналіз, і які шифри є вразливими до нього?  
Використовується для класичних шифрів (наприклад, Цезаря) — аналізуються частоти символів у шифртексті та порівнюються зі статистикою мови.  
Сучасні шифри використовують дифузію, щоб протистояти частотному аналізу.

4. У чому полягає суть атаки з вибраним відкритим текстом (Chosen Plaintext Attack, CPA)?  
Атакувальник може вибрати відкритий текст і отримати відповідний шифртекст – це дозволяє вивчити поведінку шифру та потенційно відновити ключ або структуру.

5. Як атака з вибраним шифртекстом (CCA) може бути використана проти слабких шифрів?  
Атакувальник отримує доступ до дешифрування певного шифртексту.  
Може використовуватись для витоку інформації про відкритий текст або ключ.  
Слабкі шифри або погані реалізації можуть повертати підказки (наприклад, помилки, які залежать від вмісту).

6. Чому повторне використання одного і того ж ключа або вектора ініціалізації (IV) є небезпечним?

Дає змогу атакувальнику виявити шаблони у шифротекстах.

У потокових шифрах — повторне використання IV дозволяє обчислити XOR двох відкритих текстів.

У CBC — однакові IV дають однакові шифротексти для однакових блоків.