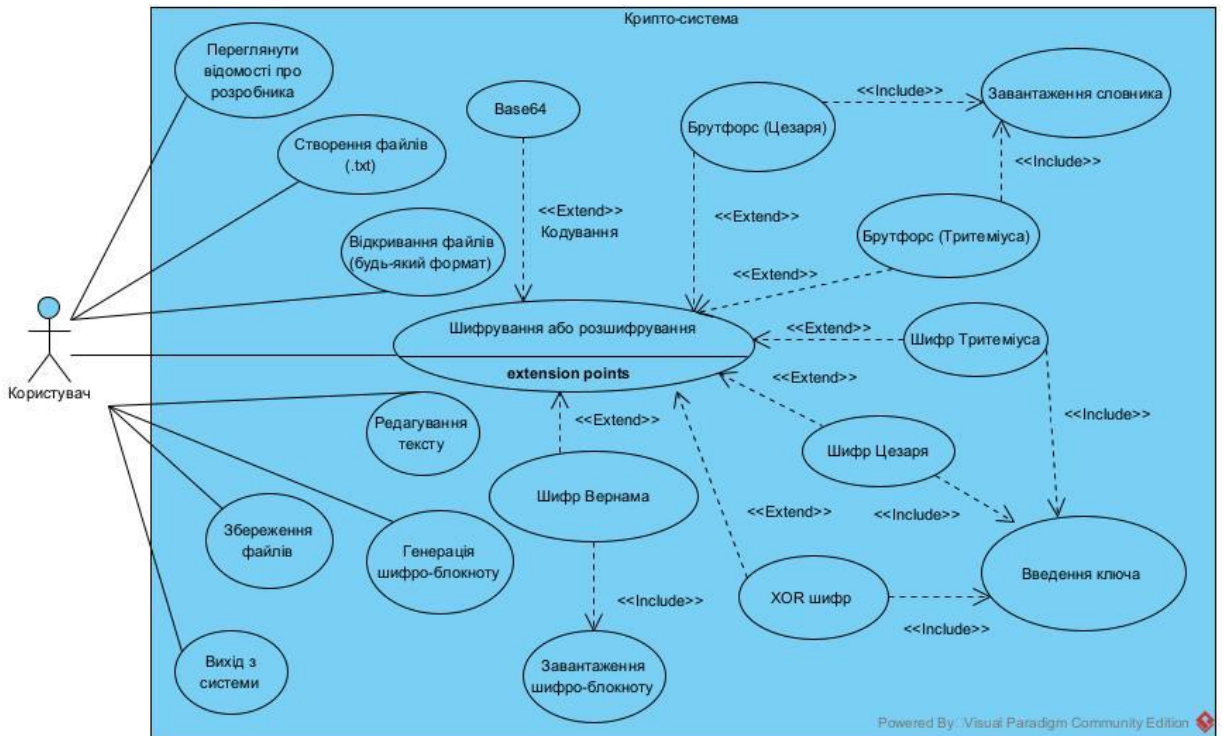


Міністерство освіти і науки України
НТУУ «КПІ ім. Ігоря Сікорського»
Навчально-науковий інститут атомної та теплової енергетики
Кафедра цифрових технологій в енергетиці

Лабораторна робота №3
з дисципліни «Безпека інформаційних систем»
«Шифр гамування»
Варіант № 22

Виконав: Студент групи ТР-12
Ковальов Олександр
Перевірів: доцент, к.ф.-м.н.
Тарнавський Ю. А.

Мета роботи. Розробити криптосистему на основі шифру гамування.
Діаграма прецедентів.

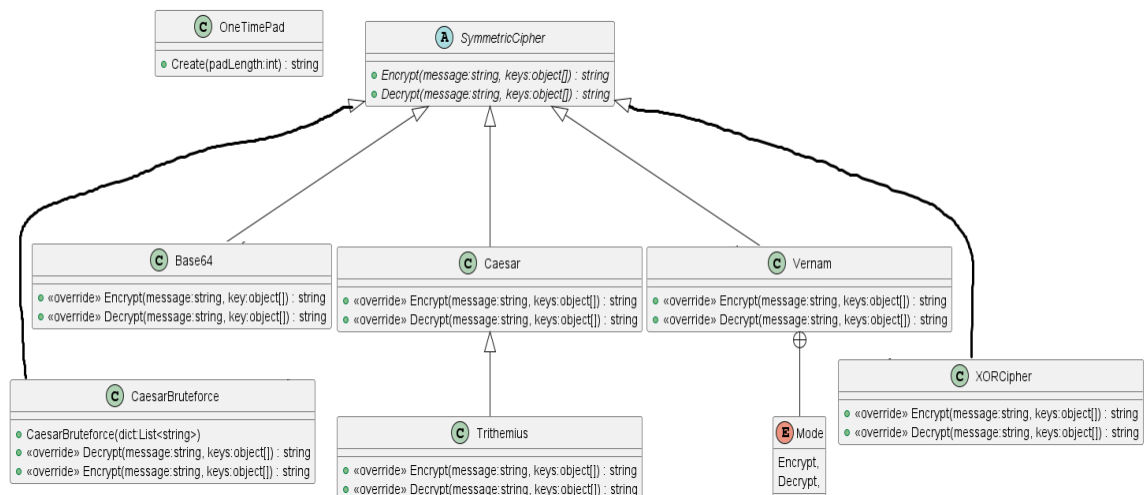


Діаграма класів.

В просторі імен Cryptography знаходяться всі шифри, які наслідуються від класу SymmetricCipher. Також там є перелік CipherEnum.

В класі XORCipher знаходяться основні методи для шифрування та розшифрування даних цим методом. Клас наслідується від класу SymmetricCipher. Це означає, що API класу складається з двох основних методів – Encrypt та Decrypt. В них викликаються приватні методи.

Методи приймають аргументи: повідомлення типу String, масив типу object[] keys. Перший аргумент – повідомлення, яке треба зашифрувати або розшифрувати. Друге – масив, помічений ключовим словом params. Це означає, що методу можна передавати будь-яку кількість аргументів. Вони автоматично запакуються в масив.



Фрагмент коду з реалізацією алгоритму шифрування/розшифрування.

```
public class XORCipher : SymmetricCipher
{
    public override string Encrypt(string message, params object[] keys)
    {
        return ValidateAndReturn(message, keys);
    }

    public override string Decrypt(string message, params object[] keys)
    {
        return ValidateAndReturn(message, keys);
    }

    private string CipherSeed(string message, int seed)
    {
        var random = new Random(seed);

        var sb = new StringBuilder();
        foreach (var c in message)
        {
            var gamma = (char) random.Next(1, UnicodeCardinal + 1);
            sb.Append((char) (c ^ gamma));
        }

        return sb.ToString();
    }

    private string CipherGamma(string message, string gamma)
    {
        if (gamma.Length != message.Length)
            throw new Exception("Message and pad lengths must be equal");

        var sb = new StringBuilder();

        for (var i = 0; i < message.Length; i++)
        {
            sb.Append((char) ((message[i] ^ gamma[i]) % UnicodeCardinal));
        }

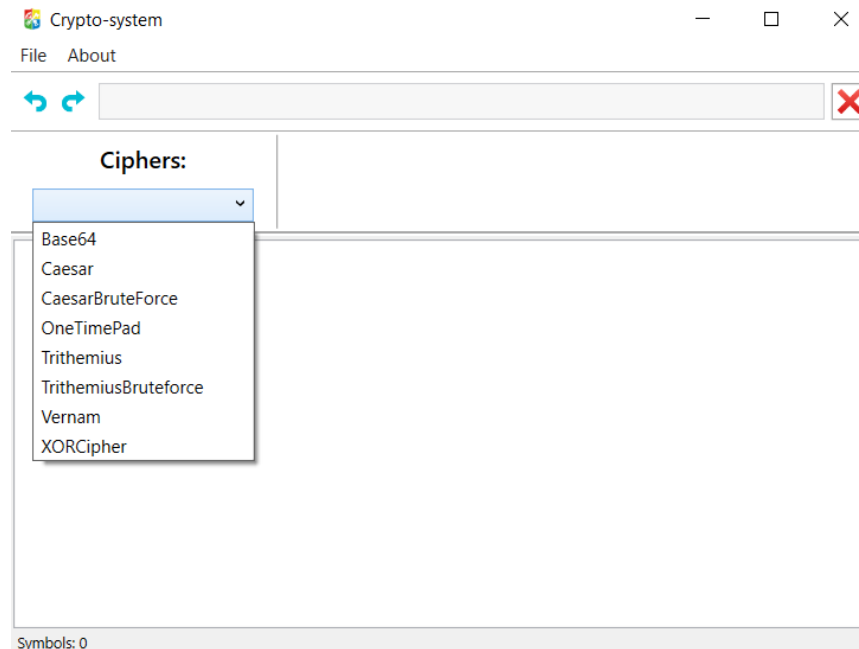
        return sb.ToString();
    }

    private string ValidateAndReturn(string message, params object[] keys)
    {
        if (keys.Length != 1) throw new Exception("Wrong args");

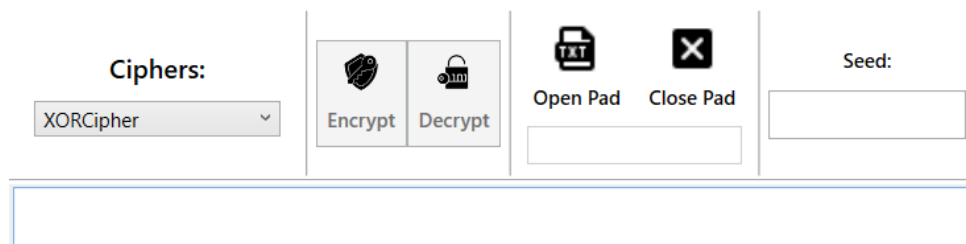
        return keys[0] switch
        {
            int seed => CipherSeed(message, seed),
            string gamma => CipherGamma(message, gamma),
            _ => throw new Exception("Wrong key type")
        };
    }
}
```

Скріншоти програми.

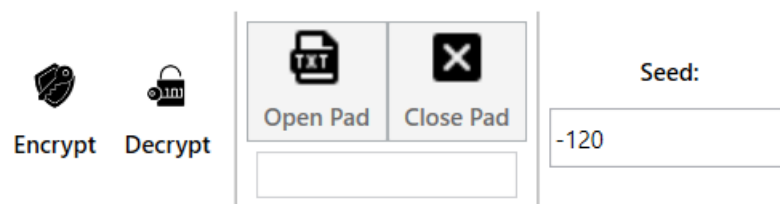
Головне вікно:



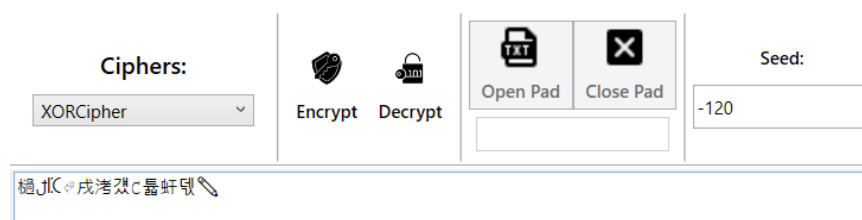
XOR шифрування. Є дві кнопки – шифрування та розшифрування. Також, можна або використати ключ, або завантажити шифро-блокнот:







Можна обрати лише один варіант ключа – при виборі одного блокується введення іншого, та розблоковуються кнопки шифрування (за умови, що текст вже є в робочій області):



Результат шифрування:



Результат розшифрування:

Ciphers: <div>XORCipher</div>	<div> Encrypt</div> <div> Decrypt</div>	<div> Open Pad</div> <div> Close Pad</div> <div></div>	Seed: <div>-120</div>
<div>Hello World!</div>			

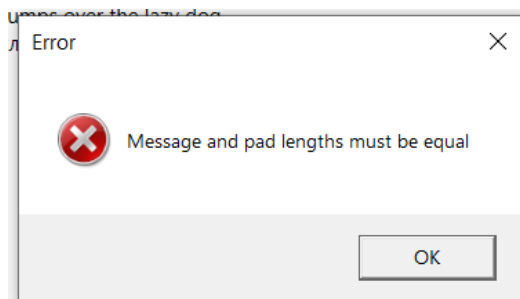
Шифро-блокнот генерується в цьому ж вікні – треба обрати вкладку OneTimePad:

Ciphers:		Length:
OneTimePad ▾	Generate	

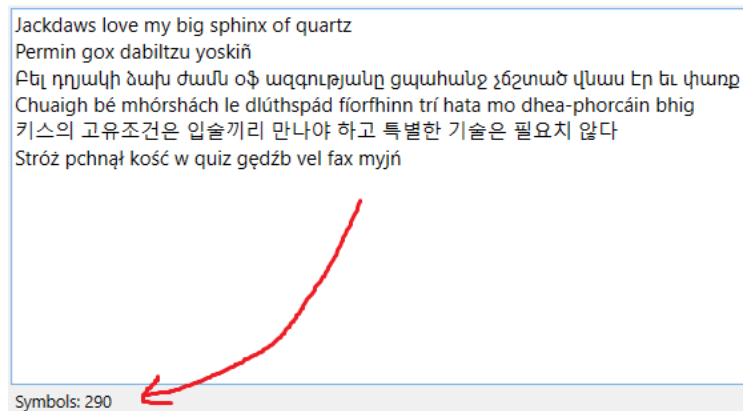
Результатом є абсолютно випадкова послідовність вказаної довжини.

[illegible]

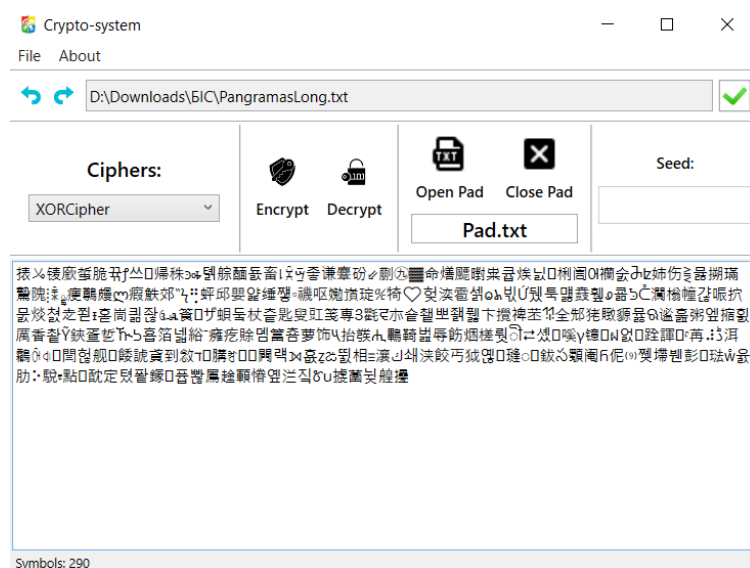
Для того, щоб збільшити криптостійкість, на XOR шифр поставлене обмеження – блокнот має бути такої ж довжини як і повідомлення:



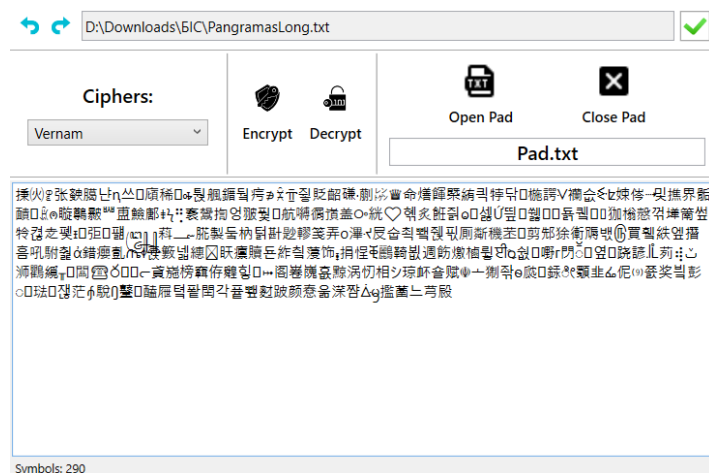
Довжину повідомлення можна дізнатись зі status bar знизу:



Приклад шифрування з шифро-блоком:



Також, є можливість зашифрувати текст шифром Вернама. Це абсолютно криптостійкий шифр, якщо в ньому виконуються три умови. Одна з умов – ключ повинен співпадати за розміром з текстом. Це встановлено в коді. Інші дві – залежать від користувача. Ключ повинен бути істинно випадковим та використовуватись один раз. Істинно випадковий ключ можна згенерувати на вкладці “OneTimePad”.



Режим СВС.

Перетворення:

$$C_i = E_{ki}(p_i \oplus C_{i-1})$$
$$p_i = C_{i-1} \oplus D_{ki}(C_i)$$

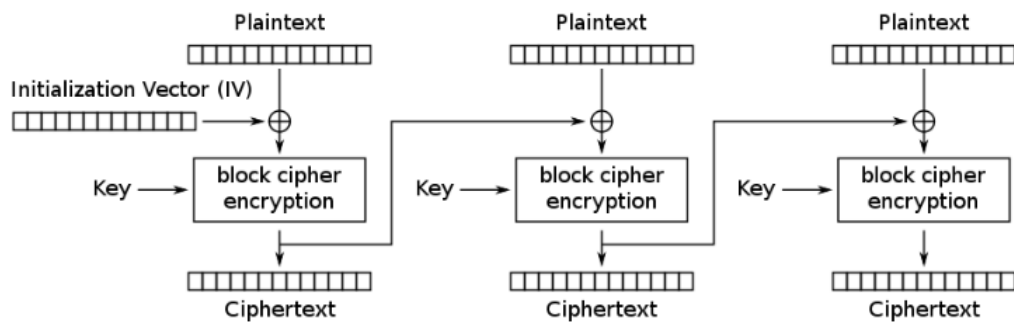
Доведення:

(Commutative)
(Abelian)

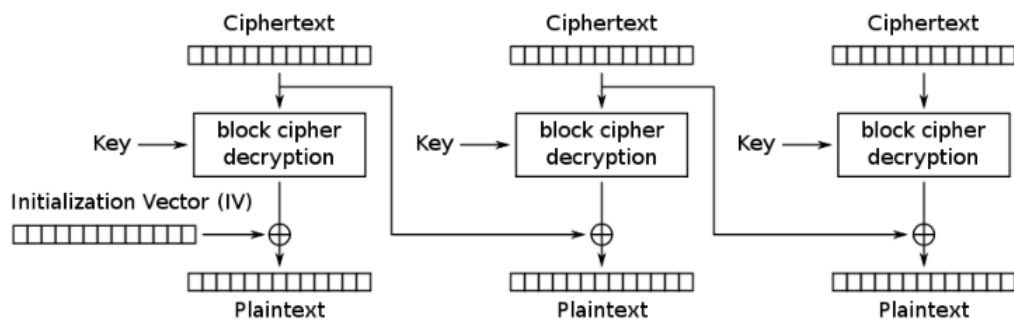
$$p_i = C_{i-1} \oplus D_{ki}(E_{ki}(p_i \oplus C_{i-1})) =$$
$$p_i = C_{i-1} \oplus p_i \oplus C_{i-1} =$$
$$p_i = C_{i-1} \oplus C_{i-1} \oplus p_i =$$
$$p_i = 0 \oplus p_i = p_i$$

Доведено

Схема:



Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption

Режим CFB.

Перетворення:

$$C_i = E_{ki}(C_{i-1}) \oplus p_i$$

$$p_i = E_{ki}(C_{i-1}) \oplus C_i$$

Доведення:

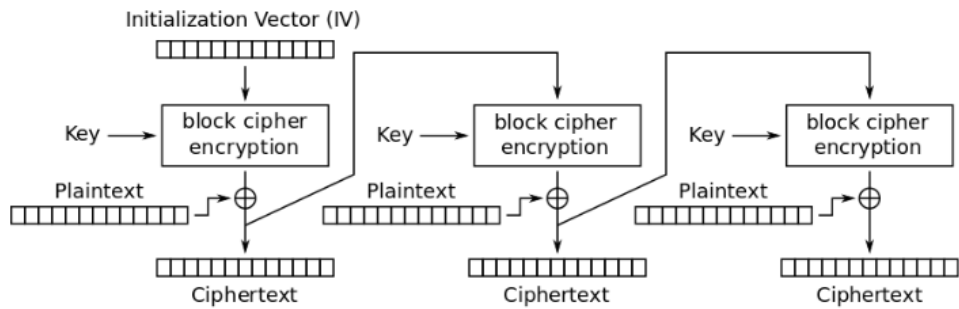
(Abelian)

$$p_i = E_{ki}(C_{i-1}) \oplus E_{ki}(C_{i-1}) \oplus p_i$$

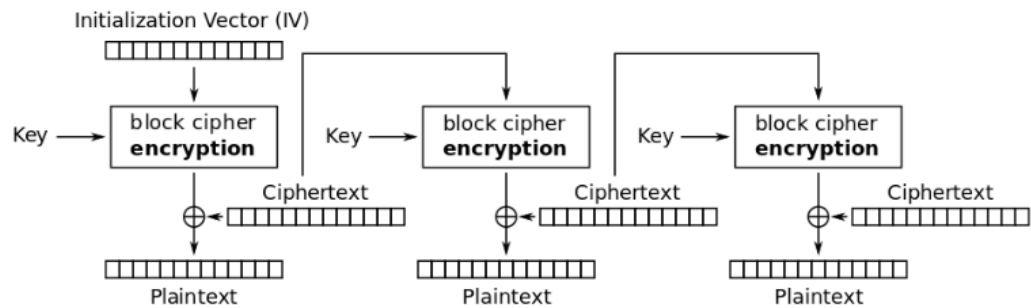
$$p_i = 0 \oplus p_i = p_i$$

Доведено

Схема:



Cipher Feedback (CFB) mode encryption



Cipher Feedback (CFB) mode decryption

Режим OFB.

Перетворення:

$$C_i = p_i \oplus O_i$$

$$p_i = C_i \oplus O_i$$

$$O_i = E_{ki}(O_{i-1})$$

$$O_1 = E_{ki}(C_0)$$

Доведення:

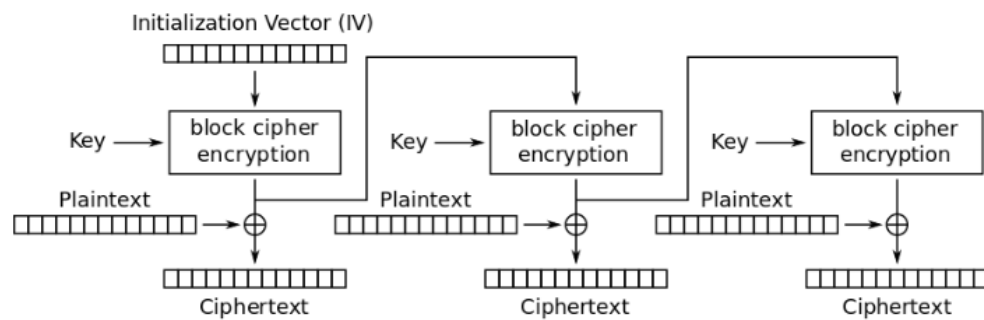
(Abelian)

$$p_i = p_i \oplus O_i \oplus O_i$$

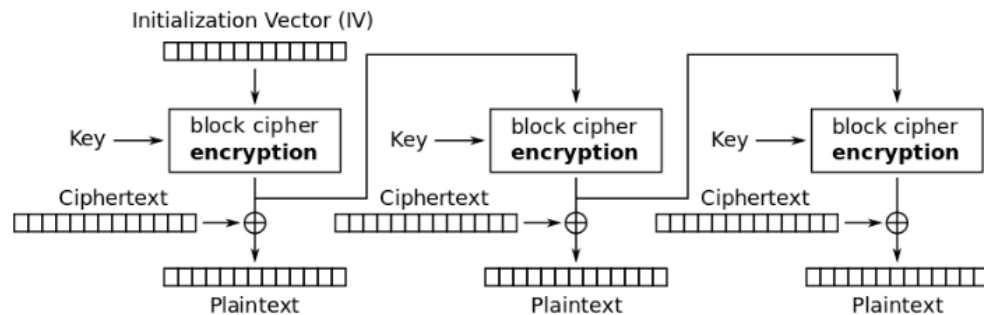
$$p_i = p_i \oplus 0 = p_i$$

Доведено

Схема:



Output Feedback (OFB) mode encryption



Output Feedback (OFB) mode decryption

Режим CTR.

Перетворення:

$$C_i = p_i \oplus O_i$$

$$p_i = C_i \oplus O_i$$

$$O_i = E_{ki}(T_i)$$

Доведення:

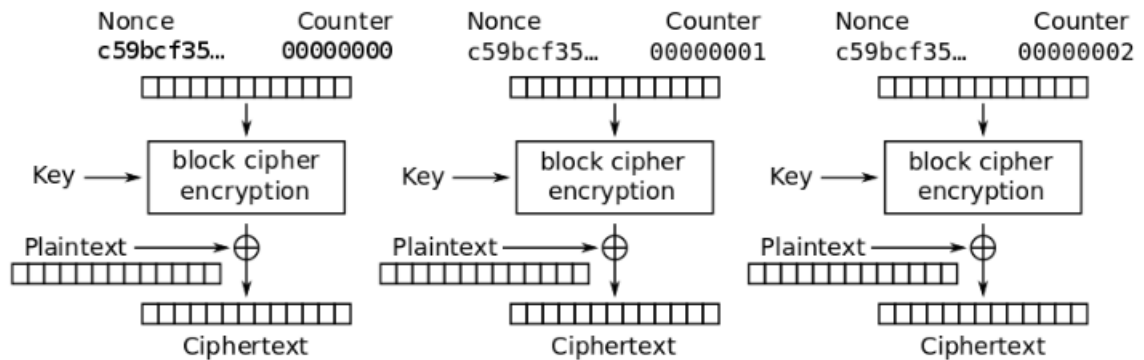
(Abelian)

$$p_i = p_i \oplus O_i \oplus O_i$$

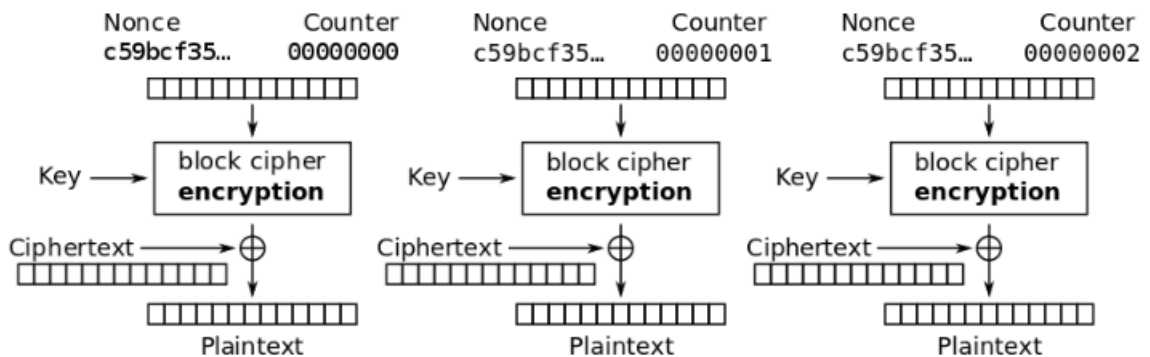
$$p_i = p_i \oplus 0 = p_i$$

Доведено

Схема:



Counter (CTR) mode encryption



Counter (CTR) mode decryption

Висновок: за результатами виконання цієї лабораторної роботи було ознайомлено з принципом роботи блокових шифрів та шифрів гамування. В крипто-систему було імплементовано такі шифри як XOR та Вернама. Також, був розроблений генератор одноразових шифро-блоктотів. При розробці в основі була концепція абсолютної криптостійкості.