https://doi.org/10.31891/2219-9365-2025-82-20
УДК 004.056:621.397.3:004.942

КЛЬОЦ Юрій
Хмельницький національний університет
https://orcid.org/0000-0002-3914-0989
e-mail: klots@khmnu.edu.ua
ДЖУЛІЙ Володимир
Хмельницький національний університет
https://orcid.org/0000-0003-1878-4301
e-mail: dzhuliivm@khmnu.edu.ua
КОРОБКО Роман
Хмельницький національний університет
e-mail: korobkord@khmnu.edu.ua

АЛГОРИТМ СИМЕТРИЧНОГО ШИФРУВАННЯ ПІДВИЩЕНОЇ КРИПТОГРАФІЧНОЇ СТІЙКОСТІ

Розглянуто теоретичні основи мережі Фейстеля та її важливість у сучасній криптографії. Наведено порівняльний аналіз запропонованого алгоритму шифрування з існуючими сучасними методами, що дозволить оцінити його переваги та недоліки в різних сценаріях використання.

Здійснено формальну постановку задачі, визначено мету, проведено аналіз об'єкта та предмета дослідження. Основною метою дослідження є демонстрація того, як нові криптографічні методи можуть покращити стійкість шифрів до сучасних атак, а також забезпечити більшу швидкість обробки даних без втрати надійності. Проведене дослідження не тільки сприяє подальшому розвитку теорії шифрування, але й має практичне значення для розробки нових, більш безпечних систем захисту інформації.

Запропоновано алгоритм симетричного шифрування підвищеної криптографічної стійкості, який базується на мережі Фейстеля та враховує сучасні вимоги до криптостійкості. Алгоритм використовує блоки розміром 119 біт та секретний ключ довжиною 112 біт, забезпечуючи процес шифрування через 10 раундів. Ключовим аспектом алгоритму шифрування є те, що при відомій структурі алгоритму, криптостійкість забезпечується лише за рахунок секретного ключа.

Ключові слова: криптографічна стійкість, інформаційна безпека, алгоритм шифрування, симетричне шифрування, секретний ключ.

KLOTS Yuriy, DZHULIY Volodymyr, KOROBKO Roman
Khmelnytskyi National University

SYMMETRIC ENCRYPTION ALGORITHM OF INCREASED CRYPTOGRAPHIC RESISTANCE

This article proposes a symmetric encryption algorithm of increased cryptographic stability, which is based on the Feistel network, but includes improved solutions to increase the level of security reliability and efficiency. The main goal of the research is to demonstrate how new cryptographic methods can improve the resistance of ciphers to modern attacks, as well as provide greater speed of data processing without loss of reliability.

The algorithm uses 119-bit blocks and a 112-bit key, ensuring an encryption process of 10 rounds. The key aspect of the encryption algorithm is that with the known structure of the algorithm, crypto-resistance is ensured only at the expense of the secret key.

Special attention is paid to the need to use longer encryption blocks (119 bits) than the standard 64-bit blocks in symmetric encryption algorithms, to protect against possible attacks in view of the growth of computing power. In addition, the peculiarity of the symmetric encryption algorithm of increased cryptographic stability is that when using standard encodings (ASCII, UTF-8, etc.), no block will contain a whole number of logical symbols. This makes it difficult to carry out cryptographic analysis, since parts of symbols can be on the border of different blocks, which creates additional difficulties for attackers.

The proposed symmetric encryption algorithm of increased cryptographic stability has potential for use in modern encryption systems and provides a high level of information protection.

The conducted research not only contributes to the further development of encryption theory, but also has practical significance for the development of new, more secure information protection systems.

Keywords: cryptographic stability, information security, encryption algorithm, symmetric encryption, secret key.

Стаття надійшла до редакції / Received 12.04.2025
Прийнята до друку / Accepted 03.05.2025

ПОСТАНОВКА ПРОБЛЕМИ У ЗАГАЛЬНОМУ ВИГЛЯДІ ТА ЇЇ ЗВ'ЯЗОК ІЗ ВАЖЛИВИМИ НАУКОВИМИ ЧИ ПРАКТИЧНИМИ ЗАВДАННЯМИ

У сучасному світі захист даних став однією з головних проблем, що потребують уваги. Доступність освіти у сфері високих технологій та розвиток інформаційних технологій визначають широке застосування систем обробки, зберігання, передачі даних та, як наслідок, загрози інформаційної безпеки. У сучасній організації бізнес процеси неможливі без застосування корпоративних мереж передачі даних та

перспективних інформаційних систем. Якщо механізми захисту даних від зовнішніх загроз досягли відповідних гарантованих рівнів, то способи та методи протидії внутрішньому порушнику слабо розвинені, в більшості документів, що регламентують політику безпеки кофіденційним даним компанії, містяться постулати про відсутність внутрішнього порушника, що тягне за собою зростання ймовірності порушення інформаційної безпеки даних, що захищаються [1,3].

В умовах постійно зростаючих загроз інформаційної безпеки, особливо важливою є розробка нових, більш надійних методів шифрування.

Одним із широко вивчених підходів до алгоритмів симетричного шифрування є мережа Фейстеля, яка використовується в багатьох відомих алгоритмах, таких як DES та його похідних [3].

У даній статті пропонується новий алгоритм симетричного шифрування, який базується на мережі Фейстеля, але включає вдосконалені рішення для підвищення рівня надійності захищеності та ефективності. Основною метою дослідження є демонстрація того, як нові криптографічні методи можуть покращити стійкість шифрів до сучасних атак, а також забезпечити більшу швидкість обробки даних без втрати надійності.

Розглянуто теоретичні основи мережі Фейстеля та її важливість у сучасній криптографії. Представлено детальний опис алгоритму симетричного шифрування підвищеної криптографічної стійкості на основі мережі Фейстеля, включаючи специфікації кожного етапу шифрування/дешифрування. У заключній частині роботи наведено порівняльний аналіз запропонованого алгоритму шифрування з існуючими сучасними методами, що дозволить оцінити його переваги та недоліки в різних сценаріях використання.

Таким чином, проведене дослідження не тільки сприяє подальшому розвитку теорії шифрування, але й має практичне значення для розробки нових, більш безпечних систем захисту інформації.

ПОСТАНОВКА ЗАДАЧІ

При розробці алгоритму шифрування підвищеної криптографічної стійкості, не залежно від способу реалізації будемо керуватися наступним набором вимог:

1. Відомості про те, як влаштований алгоритм симетричного шифрування підвищеної криптографічної стійкості не повинно впливати на криптостійкість шифру [9].

2. Крипттостійкість алгоритму шифрування повина залежати лише від секретного ключа [9].

3. Довжина блоку в 64 біти важжається безпечною на сьогоднішній день, але при подальшому стрімкому зростанні можливостей обчислювальної техніки, у найближчому майбутньому цього може не вистачати, і зробити алгорим вразливим до деяких видів атак. Таким чином, довжину блока шифрування симетричного алгоритму шифрування необхідно збільшити [9].

У сучасних алгоритмах симетричного шифрування використовується мережа Фейстеля з двома гілками яка використовується в запропонованому підході [5]. Мережа Фейстеля використовує два блоки які позначаються як *L* та *R*. На кожному раунді обчислюється наступний вираз (1):

$$\begin{aligned} L_i &= R_{i-1} XOR\left(F\left(L_{i-1}, k_{i-1}\right)\right), \\ R_i &= L_{i-1} \end{aligned} \tag{1}$$

де F - деяка функція, а k_{i-1} - ключ *i*-го раунду. Результатом виконання *n* раундів є $\left(L_n, R_n\right)$. Але зазвичай в *n*-му раунді перестановка L_n і R_n не виконуються, що дозволяє використовувати ту ж процедуру і для дешифрування, просто інвертувавши порядок використання раундової ключової інформації (2):

$$\begin{aligned} L_{i-1} &= R_i XOR\left(F\left(L_i, k_{i-1}\right)\right) \\ R_{i-1} &= L_i \end{aligned} \tag{2}$$

Невеликі зміни дозволяють досягнути повної ідентичності процедур шифрування та дешифрування. Одною із переваг такої моделі є застосовність алгоритму незалежно від функції F, і вона може бути довільної складності.

Після виконання певної кількості раундів *n* у результаті отримуємо L_n та R_n, які і будуть кінцевими зашифрованими блоками вхідного тексту. На рис. 1 наведено схему перетворення *i*- раунду в мережі Фейстеля з двома гілками [5].

ISSN 2219-9365

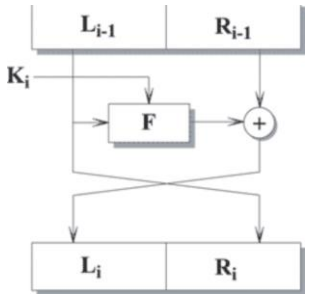

Рис. 1. Схема перетворення і- раунду в мережі Фейстеля з двома гілками

ВИКЛАД ОСНОВНОГО МАТЕРІАЛУ

При проектуванні алгоритму симетричного шифрування підвищеної криптографічної стійкості вибрано довжину блоку в 119 біт та ключ шифрування/дешифрування довжиною 112 біт. Ширування двох гілок буде проводитися у десять раундів. Алгоритм шифрування підвищеної криптографічної стійкості наведено на рис.2.

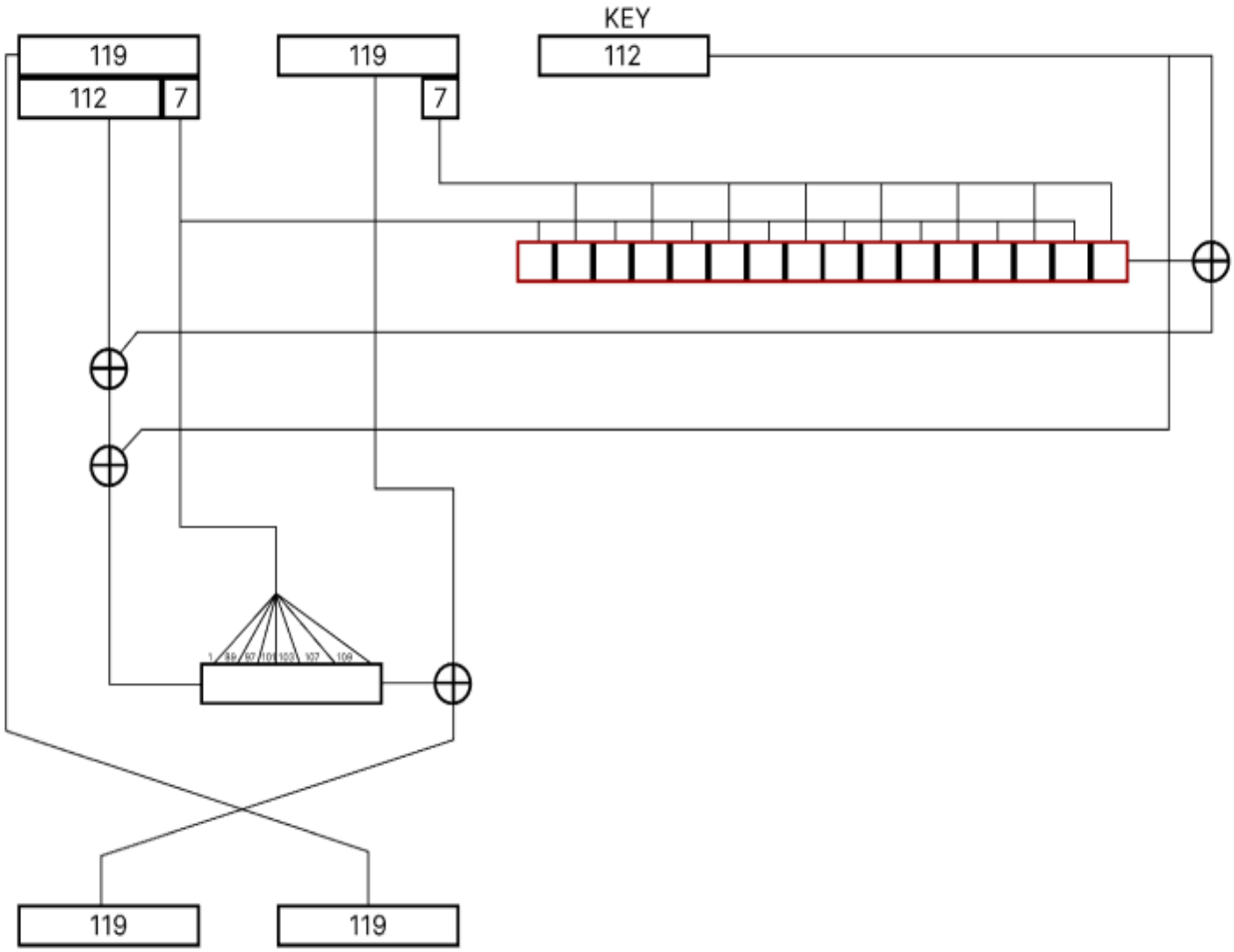

Рис. 2. Алгоритм симетричного шифрування підвищеної криптографічної стійкості

Перед початком роботи алгоритму симетричного шифрування підвищеної криптографічної стійкості необхідно претворити вхідний текст на послідовність біт. Враховуючи особливості роботи алгоритму довжина послідовності біт повина бути кратною значенню 952, тому необхідно додати визначену кількість нулів в кінець послідовності, також в якості вектора ініціалізації будемо використовувати довжину відкритого тексту.

Наступним кроком роботи алгоритму симетричного шифрування підвищеної криптографічної стійкості - розбиття послідоності біт на блоки довжиною в 119 біт. Далі блоки вхідної послідовності об’єднуємо по парам. Пари цих блоків і будуть приймати участь у раундах шифрування/дешифрування вхідного тексту.

Розглянемо роботу одного раунду алгоритму симетричного шифрування підвищеної криптографічної стійкості для вхідної послідовності бітів з двох блоків. На першому етапі виділяємо по сім біт з кінців лівого та правого блоків вхідної послідовності, в результаті їх об'єднання, отримаємо послідовність довжиною в чотирнадцять біт, продублюємо отриману послідовність бітів вісім разів, як результат отримаємо послідовність з 112 біт. Над отриманою послідовністю в 112 бітів та заздалегідь зготовленим секретними ключем, довжина якого також складає 112 біт, виконаємо операцію *XOR*, в результаті виконання даної операції отримаємо підключ для поточного раунду та даних блоків. Наступним етапом є виділення з лівого блоку послідовності довжиною в 112 біт та почергово здійснюємо операцію *XOR* з виділеною послідовністю та підключем поточного раунду, операцію *XOR* отриманого результату з секретним ключем. На наступному етапі виконується операція *crossing* або переплетення, крайні сім біт лівого блоку послідовності запишемо у 1, 89, 97, 101, 103, 107, 109 позиції послідовності бітів отриманої на попередньому етапі, вжливо замітити, що при вставці бітів необхідно зсувати біти, що залишилися на одну позицію вправо. Таким чином отримаємо послідовність довжиною в 119 біт, над якою виконуємо операцію *XOR* з правим блоком даних. Результатом проведеної операції є послідовніcь бітів, яка і буде зашифрованим блоком. На наступному кроці необхідно переставити зашифрований блок на місце лівого блоку, а на місце правого блоку незашифрований лівий блок. Розглянуті операції необхідно повторити десять раз для всіх пар блоків вхідної послідовності. На останньому раунді для кожної пари необхідно переставити лівий і правий блоки місцями. На наступному етапі необхідно відкинути лишні біти до тієї кількості бітів, яка була на початку роботи алгоритму, дану кількість бітів можна уточнити з вектора ініціалізаії. У результаті буде отримно кінцевий зашифрований текст. Дешифруванян буде відбуватися аналогічним чином.

Однією з особливостей алгоритму симетричного шифрування підвищеної криптографічної стійкості є те що при шифруванні вхідної інформації стандартних кодувань, таких як ASCI, ASCII, UTF-8, UTF-16, UTF-32 і їм подібним, ніколи не вдасться вмістити цілу кількись логічних символів в один блок. Наприклад для ASCII один символ якого займає 8 біт, вдасться умістити в одному блоці 14 символів та ще 7 біт які будуть належати символу що стоїть на розмежуванні блоків. В залежності від розтащування блоку ці 7 біт можуть бути розподілені у різній кількості, як на початку блоку так і в кінці. Така особливість роботи алшоритму має значно ускладнити методи сучасного криптоаналізу.

TIME IT TAKES A HACKER TO BRUTE FORCE YOUR PASSWORD IN 2022

Number of Characters	Numbers Only	Lowercase Letters	Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters, Symbols
4	Instantly	Instantly	Instantly	Instantly	Instantly
5	Instantly	Instantly	Instantly	Instantly	Instantly
6	Instantly	Instantly	Instantly	Instantly	Instantly
7	Instantly	Instantly	2 secs	7 secs	31 secs
8	Instantly	Instantly	2 mins	7 mins	39 mins
9	Instantly	10 secs	1 hour	7 hours	2 days
10	Instantly	4 mins	3 days	3 weeks	5 months
11	Instantly	2 hours	5 months	3 years	34 years
12	2 secs	2 days	24 years	200 years	3k years
13	19 secs	2 months	1k years	12k years	202k years
14	3 mins	4 years	64k years	750k years	16m years
15	32 mins	100 years	3m years	46m years	1bn years
16	5 hours	3k years	173m years	3bn years	92bn years
17	2 days	69k years	9bn years	179bn years	7tn years
18	3 weeks	2m years	467bn years	11tn years	438tn years

Рис. 3. Таблиця часу необхідного для повного перебору паролю заданої довжини

Алгоритм симетричного шифрування підвищеної криптографічної стійкості має криптографічну стійкість на рівні з AES. Не зважаючи на те, що у запропонованому алгоритмі шифрування використовується

ключ в 112 біт а в алгоритмі AES 128 біт, алгоритм має більшу стійкість до критографічного аналізу, хоча криптографічна стійкість до методу грубого підбору менша. Краща стійкість до сучасних методів криптоаналізу забезпечується тим що за основу взято структуру з семи бітів. В порінянні з алгоримом AES суттю якого є максимальне уникнення співпадінь, уникнення однакових послідовностей, унеможливення взлому за допомогою таблиць частотності, якщо прорівнювати за цими критеріями то запропонований алгоритм шифрування є більш криптостійким. Це випливає з того що оскільки вхідниі дані діляться на групи по 119 біт після чого шифруються, будь яка спроба провести відповідність між вхідними та вихідними даними закінчиться невдачею.

Повертаючися до методу грубого перебору, можна оцінити алгоритми за часом який знадобитися на повний перебір ключа. Для агоритму AES що має довжину ключа 128 біт яккий можна представити як 16 символів у кодуванні ASCII потрібно затратити 3 мільярди років, для ключа з літерами вверхнього і нижнього регістру та цифрами. Для повного перебору ключа у запропонованому алгоритмі що має довжину 112 біт та можна представити як 14 символів у кодуванні ASCII потрібно затратити 750 тисяч років згідно інформації наданої “hivesystems” за 2022 рік [13] (рис.3).

Реалізація алгоритм симетричного шифрування підвищеної криптографічної стійкості

Розглянемо реалізацію алгоритму симетричного шифрування підвищеної криптографічної стійкості на мові програмування C#. Створимо окремий клас для шифрування та реалізуємо в ньому наступні методи:

1. Run – точка входу в алгоритм, це повинен бути публічний метод який буде приймати вхідний текст та вектор ініціалізації (рис.4).

```
2 references
public static string Run(string input, int lenght)
{
    List<BitArray> output = new List<BitArray>();

    byte[] byteArray = Encoding.ASCII.GetBytes(input);

    BitArray allBits = new BitArray(byteArray);

    List<BitArray> blocks = GetBlocksFormBits(allBits);

    for (int i = 0; i <= blocks.Count - 2; i += 2)
        output.AddRange(Rounds(blocks[i], blocks[i + 1], new BitArray(112)));

    return ToString(output, lenght);
}
```

Рис. 4. Точка входу в алгоритм шифрування

2. GetBlocksFromBits – метод розбиває вхідну послідовність біт на блоки довжиною 119 біт (рис.5).

```
1 reference
private static List<BitArray> GetBlocksFormBits(BitArray bits)
{
    int blocksCount = (bits.Length / 952 + 1) * 8;
    List<BitArray> blocks = new List<BitArray>();
    for (int i = 0; i < blocksCount; i++)
    {
        BitArray block = new BitArray(119);
        for (int j = 0; j < 119; j++)
            try
            {
                block[j] = bits[i * 119 + j];
            }
            catch
            {
                block[j] = false;
            }
        blocks.Add(block);
    }
    return blocks;
}
```

Рис. 5. Метод GetBlocksFromBits

3. Rounds – метод що виконує *i*-раунд для пари блоків вхідної послідовності (рис.6).

```
1 reference
45      private static List<BitArray> Rounds(BitArray block1, BitArray block2, BitArray key, int rounds = 10)
46      {
47          List<BitArray> result = new List<BitArray>() { block1, block2 };
48          for (int i = 0; i < rounds; i++)
49              result = Round(result[0], result[1], key);
50          return new List<BitArray>() { result[1], result[0] };
51      }
52
```

Рис.6. Метод Rounds

4. Round – механізм роботи одного раунду (рис.7).

```
1 reference
53      private static List<BitArray> Round(BitArray block1, BitArray block2, BitArray key)
54      {
55          BitArray last71 = Get7Bit(block1);
56          BitArray last72 = Get7Bit(block2);
57
58          BitArray keyMask = CreateKeyMask(last71, last72);
59          BitArray subkey = keyMask.Xor(key);
60
61          BitArray block1112 = new BitArray(112);
62          for (int i = 0; i < 112; i++)
63              block1112[i] = block1[i];
64
65          block1112.Xor(key).Xor(subkey);
66          BitArray resultBlock1 = Crossing(block1112, last71).Xor(block2);
67
68          return new List<BitArray>() { resultBlock1, block1 };
69      }
70
```

Рис.7. Механізм роботи одного раунду

5. Get7Bit – допоміжний метод що повертає останні 7 біт з блоку який йому передано (рис.8).

```
2 references
71      private static BitArray Get7Bit(BitArray block)
72      {
73          BitArray last7 = new BitArray(7);
74          for (int i = 0; i < 7; i++)
75              last7[i] = block[111 + i];
76          return last7;
77      }
78
```

Рис.8. Метод Get7Bit

6. GreateKeyMask – метод що приймає даві послідовності по 7 біт та дублює їх до тих пір поки не оримає послідовність з 112 біт (рис.9).

```
1 reference
79      private static BitArray CreateKeyMask(BitArray bit71, BitArray bit72)
80      {
81          BitArray mask = new BitArray(112);
82
83          for (int i = 0; i < 8; i++)
84          {
85              mask.LeftShift(7);
86              for (int k = 0; k < 7; k++)
87                  mask[0] = bit71[k];
88              mask.LeftShift(7);
89              for (int k = 0; k < 7; k++)
90                  mask[0] = bit72[k];
91          }
92          return mask;
93      }
94
```

Рис.9. Метод GreateKeyMask

7. Crossing – метод у якому проводитися переплетеня вже частково закодованої частини блоку з частиною незакодованого блоку (рис.10).

```
1 reference
private static BitArray Crossing(BitArray block, BitArray bits)
{
    BitArray crossBlock = new BitArray(119);
    int k = 0;
    for (int i = 0; i < 119; i++)
    {
        switch (i)
        {
            case 1:
                crossBlock[i] = bits[0];
                i++;
                k++;
                break;
            case 89:
                crossBlock[i] = bits[1];
                i++;
                k++;
                break;
            case 97:
                crossBlock[i] = bits[2];
                i++;
                k++;
                break;
            case 101:
                crossBlock[i] = bits[3];
                i++;
                k++;
                break;
            case 103:
                crossBlock[i] = bits[4];
                i++;
                k++;
                break;
            case 107:
                crossBlock[i] = bits[5];
                i++;
                k++;
                break;
            case 109:
                crossBlock[i] = bits[6];
                i++;
                k++;
                break;
            default:
                break;
        }
        crossBlock[i] = block[i - k];
    }
    return crossBlock;
}
```

Рис.10. Метод Crossing

8. ToString – метод поверне читабельний рядок зашифрованим/ ltшифрованим текстом (рис.11).

```
1 reference
public static string ToString(List<BitArray> bits, int size)
{
    BitArray bitArray = new BitArray(119 * bits.Count);
    string result = "";
    for (int i = 0; i < bits.Count(); i++)
        for (int j = 0; j < bits[0].Length; j++)
            bitArray[119 * i + j] = bits[i][j];

    byte[] resultByteArray = new byte[(bitArray.Length + 7) / 8];
    bitArray.CopyTo(resultByteArray, 0);
    result = Encoding.ASCII.GetString(resultByteArray);

    return result;
}
}
}
```

Рис.11. Метод ToString

Висновки з даного дослідження і перспективи подальших розвідок у даному напрямі

Запропоновано алгоритм симетричного шифрування підвищеної криптографічної стійкості, який базується на мережі Фейстеля та враховує сучасні вимоги до криптостійкості. Алгоритм використовує блоки розміром 119 біт та секретний ключ довжиною 112 біт, забезпечуючи процес шифрування через 10 раундів. Ключовим аспектом алгоритму шифрування є те, що при відомій структурі алгоритму, криптостійкість забезпечується лише за рахунок секретного ключа.

Особлива увага приділена необхідності використання блоків шифрування більшої довжини (119 біт), ніж стандартні 64 бітні блоки в алгоритмах симетричного шифрування, для захисту від можливих атак з огляду в подальшому на зростання обчислювальних потужностей. Окрім того, особливість алгоритму

симетричного шифрування підвищеної криптографічної стійкості, полягає в тому, що при використанні стандартних кодувань (ASCII, UTF-8 тощо) жоден блок не міститиме цілої кількості логічних символів. Це ускладнює проведення криптографічного аналізу, оскільки частини символів можуть знаходитися на межі різних блоків, що створює додаткові труднощі для зловмисників.

Таким чином, запропонований алгоритм симетричного шифрування підвищеної криптографічної стійкості має потенціал для використання у сучасних системах шифрування та забезпечує високий рівень захисту інформації.

Література

1. Захист інформації від несанкціонованого доступу шляхом шифрування даних URL: https://dspace.kntu.kr.ua/server/api/core/bitstreams/f8c43f70-4720-4daa-9561-7811118de649/content

2. ПРОГРАМНИЙ МОДУЛЬ ШИФРУВАННЯ ІНФОРМАЦІЙНИХ ПОТОКІВ URL: https://er.nau.edu.ua/bitstream/NAU/50665/1/%D0%A2%D0%BE%D0%BC%D0%BD%D1%8E%D0%BA%20%D0%94.%D0%9D.%D0%9C..pdf

3. Блочні алгоритми URL: https://studfile.net/preview/6047915/page:6

4. DES URL: https://wikipedia.org/wiki/DES

5. Стандарт шифрования данных DES (Data Encryption Standard) URL: https://nmetau.edu.ua/file/info_defend_lab_3.pdf

6. AES (стандарт шифрования) URL: https://uk.wikipedia.org/wiki/Advanced_Encryption_Standard

7. Сучасні методи аналізу і синтезу криптографічних алгоритмів та протоколів URL: https://report.kpi.ua/uk/0113U000944

8. ПОКАЗНИКИ ОЦІНКИ ЕФЕКТИВНОСТІ АЛГОРИТМІВ ШИФРУВАННЯ НА ЕЛІПТИЧНИХ КРИВИХ URL: https://core.ac.uk/download/pdf/47231196.pdf

9. Алгоритм шифрування даних із використанням модульного експоненціювання для симетричних криптосистем / Data encryption algorithm using modular exposure for symmetric cryptosystems URL: http://dspace.wunu.edu.ua/bitstream/316497/40521/1/%D0%A1%D1%83%D1%81%D0%BBi%D0%BD_%D0%9C%D0%A0_2020.pdf

10. МЕТОД ВИБОРУ ОПТИМАЛЬНОГО АЛГОРИТМУ КРИПТОГРАФІЧНОГО ЗАХИСТУ ІНФОРМАЦІЇ URL: https://science.lpnu.ua/sites/default/files/journal-paper/2019/apr/16006/vis673ism-220-232.pdf

11. ДОСЛІДЖЕННЯ ХАРАКТЕРИСТИК КРИПТОСТІЙКОСТІ АЛГОРИТМУ СИМЕТРИЧНОГО ШИФРУВАННЯ DES URL: https://vlp.com.ua/files/09_21.pdf

12. ДОСЛІДЖЕННЯ МЕТОДІВ ПІДВИЩЕННЯ КРИПТОГРАФІЧНОЇ СТІЙКОСТІ URL: https://journals.indexcopernicus.com/api/file/viewByFileId/568392.pdf

13. Are Your Passwords in the Green? URL: https://www.hivesystems.com/blog/are-your-passwords-in-the-green

14. Остапов С. Е. Технології захисту інформації: навчальний посібник / С.Е. Остапов, С.П. Євсеєв, О.Г. Король. – Харків : Вид-во ХНЕУ, 2016. – 476 с

References

1. Zakhyst informatsii vid nesanktsionovanoho dostupu shliakhom shyfruvannia danykh URL: https://dspace.kntu.kr.ua/server/api/core/bitstreams/f8c43f70-4720-4daa-9561-7811118de649/content

2. PROHRAMNYI MODUL ShYFRUVANNIa INFORMATsIINYKh POTOKIV URL: https://er.nau.edu.ua/bitstream/NAU/50665/1/%D0%A2%D0%BE%D0%BC%D0%BD%D1%8E%D0%BA%20%D0%94.%D0%9D.%D0%9C..pdf

3. Blochni alhorytmy URL: https://studfile.net/preview/6047915/page:6

4. DES URL: https://wikipedia.org/wiki/DES

5. Standart shyfrovanyia dannыkh DES (Data Encryption Standard) URL: https://nmetau.edu.ua/file/info_defend_lab_3.pdf

6. AES (standart shyfrovanyia) URL: https://uk.wikipedia.org/wiki/Advanced_Encryption_Standard

7. Suchasni metody analizu i syntezu kryptohrafichnykh alhorytmiv ta protokoliv URL: https://report.kpi.ua/uk/0113U000944

8. POKAZNYKY OTsINKY EFEKTYVNOSTI ALHORYTMIV ShYFRUVANNIa NA ELIPTYChNYKh KRYVYKh URL: https://core.ac.uk/download/pdf/47231196.pdf

9. Alhorytm shyfruvannia danykh iz vykorystanniam modulnoho eksponentsiiuvannia dlia symetrychnykh kryptosystem / Data encryption algorithm using modular exposure for symmetric cryptosystems URL: http://dspace.wunu.edu.ua/bitstream/316497/40521/1/%D0%A1%D1%83%D1%81%D0%BBi%D0%BD_%D0%9C%D0%A0_2020.pdf

10. METOD VYBORU OPTYMALNOHO ALHORYTMU KRYPTOHRAFIChNOHO ZAKhYSTU INFORMATsII URL: https://science.lpnu.ua/sites/default/files/journal-paper/2019/apr/16006/vis673ism-220-232.pdf

11. DOSLIDZhENNIa KhARAKTERYSTYK KRYPTOSTIIKOSTI ALHORYTMU SYMETRYChNOHO ShYFRUVANNIa DES URL: https://vlp.com.ua/files/09_21.pdf

12. DOSLIDZhENNIa METODIV PIDVYShchENNIa KRYPTOHRAFIChNOI STIIKOSTI URL: https://journals.indexcopernicus.com/api/file/viewByFileId/568392.pdf

13. Are Your Passwords in the Green? URL: https://www.hivesystems.com/blog/are-your-passwords-in-the-green

14. Tekhnolohii zakhystu informatsii: navchalnyi posibnyk / S.E. Ostapov, S.P. Yevseiev, O.H. Korol–Kharkiv : Vyd-vo KhNEU, 2016. – 476 s.