**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені Ігоря СІКОРСЬКОГО»**
**Навчально-науковий фізико-технічний інститут**
**Кафедра математичних методів захисту інформації**

«На правах рукопису»
УДК 003.26:629.7.05

«До захисту допущено»
Завідувач кафедри
_____ Сергій ЯКОВЛЄВ
«__» _____ 2025 р.

# Дипломна робота

**на здобуття ступеня бакалавра**
за освітньо-професійною програмою
«Математичні методи криптографічного захисту інформації»

зі спеціальності: 113 Прикладна математика
на тему: **«Розробка та порівняльний аналіз алгоритмів шифрування зображень у середовищі з сильно обмеженими ресурсами»**

Виконав:
студент IV курсу, групи ФІ-14
Недашківська Аріна Віталіївна _____

Керівник:
д.т.н., проф. каф. ММЗІ
Ковальчук Людмила Василівна _____

Рецензент:
д.ф., доц. каф. ММАД
Яйлимова Ганна Олексіївна _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

Київ — 2025

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені Ігоря СІКОРСЬКОГО»**
**Навчально-науковий фізико-технічний інститут**
**Кафедра математичних методів захисту інформації**

Рівень вищої освіти — перший (бакалаврський)
Спеціальність — 113 Прикладна математика,
ОПП «Математичні методи криптографічного захисту інформації»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій ЯКОВЛЄВ
«__» _____ 2025 р.

## ЗАВДАННЯ
### на дипломну роботу

Студент: Недашківська Аріна Віталіївна

1. Тема роботи: «*Розробка та порівняльний аналіз алгоритмів шифрування зображень у середовищі з сильно обмеженими ресурсами*», науковий керівник дисертації: д.т.н., проф. каф. ММЗІ Ковальчук Людмила Василівна,

затверджені наказом по університету №__ від «__» _____ 2025 р.

2. Термін подання студентом роботи: «__» _____ 2025 р.

3. Об'єкт дослідження: *процес шифрування та передачі зображення у БПЛА*

4. Предмет дослідження: *малоресурсні алгоритми шифрування зображень, які можуть використовуватись у безпілотних літальних апаратах.*

5. Перелік завдань:

– *провести огляд опублікованих джерел за тематикою дослідження*

– *провести аналіз малоресурсних алгоритмів шифрування*

– *запропонувати нові алгоритми шифрування зображень*

– *провести порівняльний аналіз для запропонованих алгоритмів*

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: *презентація доповіді*

7. Дата видачі завдання: 10 вересня 2024 р.

Календарний план

| № з/п | Назва етапів виконання магістерської дисертації | Термін виконання | Примітка |
|---|---|---|---|
| 1 | Узгодження теми роботи із науковим керівником | 01-15 вересня 2024 р. | Виконано |
| 2 | Огляд опублікованих джерел за тематикою дослідження | Вересень-жовтень 2024 р. | Виконано |
| 3 | Ознайомлення з малоресурсними алгоритмами шифрування | Жовтень-грудень 2024 р. | Виконано |
| 4 | Аналіз малоресурсних алгоритмів шифрування | Січень-лютий 2025 р. | Виконано |
| 5 | Формування вимог для шифрування | Березень-квітень 2025 р. | Виконано |
| 6 | Розробка алгоритмів шифрування | Квітень-травень 2025 р. | Виконано |
| 7 | Оформлення дипломної роботи | Травень-червень 2025 р. | Виконано |

Студент _____ Аріна НЕДАШКІВСЬКА

Керівник _____ Людмила КОВАЛЬЧУК

# РЕФЕРАТ

Кваліфікаційна робота містить: 58 стор., 3 рисунки, 10 таблиць, 9 джерел.

**Метою дослідження** є розробка і порівняльний аналіз алгоритмів шифрування зображень у безпілотних літальних апаратах.

**Об'єктом дослідження** є процес шифрування та передачі зображення у безпілотних літальних апаратах.

**Предметом дослідження** є малоресурсні алгоритми шифрування зображень, які можуть використовуватись у безпілотних літальних апаратах.

У ході проведення дослідження було запропоновано ряд алгоритмів, які можна використовувати для шифрування зображень у безпілотних літальних апаратах. Серед запропонованих алгоритмів є як повністю нові розробки, так і ті, що базуються на шифрі Вернама та на малоресурсних алгоритмах шифрування, таких як PRESENT, GIFT-64, ENOCORO. Крім того, для кожного запропонованого алгоритму проведено оцінку обсягу ключового простору та необхідної пам'яті для зберігання ключа шифрування та зашифрованого зображення.

МАЛОРЕСУРСНА КРИПТОГРАФІЯ, ШИФРУВАННЯ ЗОБРАЖЕНЬ, БПЛА, МАЛОРЕСУРСНІ КРИПТОГРАФІЧНІ АЛГОРИТМИ

# ABSTRACT

The qualification work contains: 58 pages, 3 figures, 10 tables, 9 citations.

**The aim of the study** is to develop and compare image encryption algorithms in unmanned aerial vehicles.

**The object of the study** is the process of encrypting and transmitting images in unmanned aerial vehicles.

**The subject of the study** is lightweight image encryption algorithms that can be used in unmanned aerial vehicles.

During the research, a number of algorithms were proposed that can be used to encrypt images in unmanned aerial vehicles. Among the proposed algorithms are both completely new developments and those based on the Vernam cipher and lightweight encryption algorithms such as PRESENT, GIFT-64, and ENOCORO. In addition, for each proposed algorithm, an assessment was made of the key space volume and the memory required to store the encryption key and the encrypted image.

LIGHTWEIGHT CRYPTOGRAPHY, IMAGE ENCRYPTION, UAVS, LIGHTWEIGHT CRYPTOGRAPHY ALGORITHMS

# ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ФТІ — Фізико-технічний інститут

SPN — мережа підстановки-перестановки

БПЛА — безпілотні літальні апарати

$\oplus$ — операція побітового додавання

$V_n$ — множина всіх бітових векторів довжини $n$

ЗСУ — Збройні Сили України

# ВСТУП

**Актуальність дослідження.** Актуальність даного дослідження полягає у зростаючій необхідності захисту інформації у безпілотних літальних апаратах під час обробки та передавання зображень. Сучасні безпілотні літальні апарати активно застосовуються у військовій, цивільній та промисловій сферах, де обробка та передача візуальної інформації є критично важливими для виконання завдань моніторингу, розвідки, картографування та інспекції.

Особливу актуальність має розробка та впровадження ефективних алгоритмів шифрування зображень у БПЛА з урахуванням їхніх апаратних обмежень — низької обчислювальної потужності, обмеженого обсягу пам'яті та обмеженого енергоспоживання. Існуючі алгоритми шифрування не завжди задовольняють вимоги безпеки та реального часу, що створює потребу у дослідженні, порівнянні та оптимізації криптографічних алгоритмів для таких систем.

Крім того, враховуючи швидке зростання кіберзагроз та активне використання дронів у різних сферах діяльності, забезпечення конфіденційності та цілісності візуальних даних стає пріоритетним завданням. Важливою специфікою алгоритмів шифрування для БПЛА є те, що вони не повинні розповсюджувати помилку, тобто помилки у зашифрованих або переданих даних не повинні призводити до руйнування великої частини зображення після дешифрування. Це забезпечує стійкість системи до випадкових або навмисних спотворень сигналу в умовах обмеженого каналу зв'язку та підвищує надійність передачі.

Таким чином, проведення дослідження з розробки та аналізу алгоритмів шифрування зображень для БПЛА є важливим для підвищення безпеки, надійності та стійкості функціонування цих систем у реальних умовах експлуатації.

**Метою дослідження** є розробка і порівняльний аналіз алгоритмів шифрування зображень у безпілотних літальних апаратах. Для досягнення мети необхідно вирішити такі **завдання дослідження:**

1) провести огляд опублікованих джерел за тематикою дослідження;

2) провести аналіз малоресурсних алгоритмів шифрування;

3) запропонувати нові алгоритми шифрування зображень;

4) провести порівняльний аналіз для запропонованих алгоритмів.

*Об'єктом дослідження* є процес шифрування та передачі зображення у БПЛА.

*Предметом дослідження* є малоресурсні алгоритми шифрування зображень, які можуть використовуватись у БПЛА.

При розв'язанні поставлених завдань використовувались такі *методи дослідження*: методи криптографії та криптоаналізу, алгебраїчні методи, комбінаторні методи.

**Наукова новизна** отриманих результатів полягає в тому, що вперше було проаналізовано алгоритми шифрування саме з точки зору специфіки їх використання у БПЛА. Вперше було запропоновано нові алгоритми шифрування, які розроблені спеціально для шифрування зображень з урахуванням специфіки подання зображення у БПЛА.

**Практичне значення** результатів полягає у можливості використання запропонованих алгоритмів шифрування зображень у безпілотних літальних апаратах з урахуванням їхніх апаратних обмежень та специфіки подання зображення у БПЛА. Завдяки проведеному аналізу пам'яті кожного з алгоритмів, користувачі можуть обирати найбільш оптимальний метод шифрування залежно від конкретних умов застосування та обмежень ресурсів.

# 1 КРИПТОГРАФІЯ В СЕРЕДОВИЩІ З ОБМЕЖЕНИМИ РЕСУРСАМИ

В першому розділі розглянуті основні поняття та загальні відомості про малоресурсну криптографію, її значення і специфіку застосування в умовах обмежених обчислювальних ресурсів, зокрема безпілотних літальних апаратів. Крім того, детально розглянуто шифр Вернама, а також основні приклади малоресурсних алгоритмів, таких як PRESENT, GIFT-64, ENOCORO, що будуть використовуватись у другому та третьому розділах.

## 1.1 Малоресурсна криптографія

Використання безпілотних літальних апаратів в цивільних та військових сферах стрімко зростає. БПЛА застосовуються для моніторингу, спостереження, картографування та інших завдань [1]. Однак апаратні обмеження, такі як обмежена обчислювальна потужність, обмежений обсяг пам'яті та обмежені енергетичні ресурси, створюють значні виклики для реалізації криптографічних алгоритмів на борту БПЛА. Багато криптографічних алгоритмів, спроектованих для настільних комп'ютерів або серверів, важко та практично неможливо реалізувати на БПЛА через обмежені ресурси. Навіть якщо їх можна реалізувати, їхня продуктивність може бути неприйнятною в умовах реального часу та обмеженої енергетичної ємності акумуляторів.

БПЛА функціонують у вкрай динамічних і часто ворожих середовищах, де швидка та надійна передача даних є критично важливою. Відповідно, криптографічні алгоритми, застосовані на борту, мають забезпечувати не лише високу ступінь захисту інформації, а й бути максимально ефективними за часом обробки, енергоспоживанням і

використанням пам'яті.

Тому малоресурсна криптографія стає оптимальним рішенням для БПЛА. Вона орієнтована на розробку криптографічних методів, які відповідають специфічним апаратним і операційним обмеженням безпілотних систем. Використання малоресурних алгоритмів дозволяє забезпечити надійний захист інформації без сильного навантаження на ресурси апаратного забезпечення.

**Означення 1.1.** Малоресурсна криптографія [2] – це підгалузь криптографії, яка розроблена для пристроїв з обмеженими ресурсами. Метою якої є створення алгоритмів і протоколів, що враховують обмеженння в продуктивності, пам'яті та енергоспоживанні характерних для невеликих пристроїв.

**Означення 1.2.** SPN [3] – це мережа, яка обробляє відкритий текст і ключі шляхом чергування раундів шарів підстановки (sBoxLayer) і перестановки (pLayers) для генерації зашифрованого тексту.

## 1.2 Основні малоресурні криптографічні алгоритми

Існує два типи симетричних алгоритмів шифрування [4]: блокові та потокові шифри. Обидва виконують функцію шифрування або дешифрування повідомлення. Основна відмінність між ними полягає в тому, що блокові шифри працюють з блоками фіксованого розміру і оперують бітами повідомлення і ключа, в той час як потокові шифри генерують псевдовипадковий вектор, який називається потоковим ключем, і шифрують повідомлення за допомогою операції XOR.

За останнє десятиліття було запропоновано низку криптографічних примітивів із низькими вимогами до ресурсів, які демонструють переваги в продуктивності порівняно з традиційними алгоритмами шифрування, включаючи як блокові, так і потокові шифри. Ці примітиви зазвичай не призначені для універсального застосування, а створені з урахуванням

специфічних потреб систем із обмеженими обчислювальними та пам'ятними ресурсами, наприклад, для використання на безпілотних літальних апаратах.

### 1.2.1 Алгоритм PRESENT

**PRESENT** [5] – це SPN-структурований шифр з 31-го повного раунду шифрування. Складається з 64-бітового розміру стану (тобто шістнадцяти 4-бітових слів) і розміру ключа 80 бітів. Також можлива версія з 128-бітовим ключем. Один раунд якого складається з шару підстановки (sBoxLayer), заданого таблицею 1.1 , і шару перестановки (pLayers), заданого таблицею 1.2.

Шифр описано в псевдокоді – рисунок 1.1.

```
generateRoundKeys()
for i = 1 to 31 do
    addRoundKey(STATE,K_i)
    sBoxLayer(STATE)
    pLayer(STATE)
end for
addRoundKey(STATE,K_32)
```

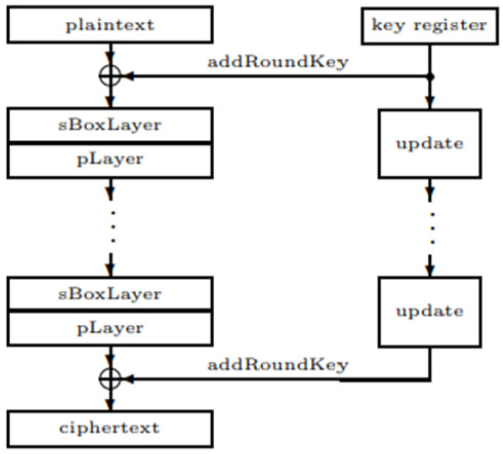



**Рисунок 1.1** – Алгоритмічний опис шифру PRESENT

**Таблиця 1.1** – S-блок шифру PRESENT

| $x$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S(x)$ | C | 5 | 6 | B | 9 | 0 | A | D | 3 | E | F | 8 | 4 | 7 | 1 | 2 |

**Таблиця 1.2** – Лінійна бітова перестановка шифру PRESENT

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $P(i)$ | 0 | 16 | 32 | 48 | 1 | 17 | 33 | 49 | 2 | 18 | 34 | 50 | 3 | 19 | 35 | 51 |
| $i$ | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| $P(i)$ | 4 | 20 | 36 | 52 | 5 | 21 | 37 | 53 | 6 | 22 | 38 | 54 | 7 | 23 | 39 | 55 |
| $i$ | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| $P(i)$ | 8 | 24 | 40 | 56 | 9 | 25 | 41 | 57 | 10 | 26 | 42 | 58 | 11 | 27 | 43 | 59 |
| $i$ | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
| $P(i)$ | 12 | 28 | 44 | 60 | 13 | 29 | 45 | 61 | 14 | 30 | 46 | 62 | 15 | 31 | 47 | 63 |

### 1.2.2 Алгоритм GIFT-64

**GIFT-64** [6] – це SPN-структурований шифр з 28 раундів шифрування. Складається з 64-бітового розміру стану (тобто шістнадцяти 4-бітових слів) і розміру ключа 128 бітів. Один раунд якого складається з шару підстановки (sBoxLayer), заданого таблицею 1.3 , і шару перестановки (pLayers), заданого таблицею 1.4.

Шифр описано в псевдокоді – рисунок 1.2.

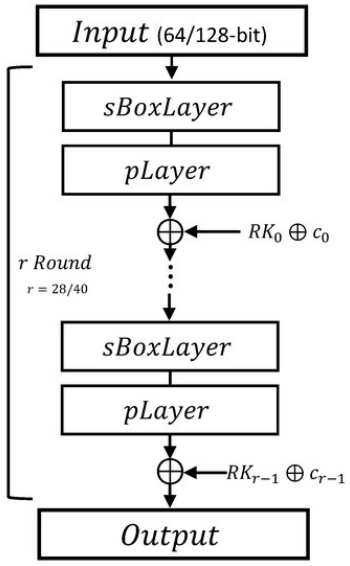



**Рисунок 1.2** – Алгоритмічний опис шифру GIFT-64

**Таблиця 1.3** – S-блок шифру GIFT-64

| $x$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $GS(x)$ | 1 | A | 4 | C | 6 | F | 3 | 9 | 2 | D | B | 7 | 5 | 0 | 8 | E |

**Таблиця 1.4** – Лінійна бітова перестановка шифру GIFT-64

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $P(i)$ | 0 | 17 | 34 | 51 | 48 | 1 | 18 | 35 | 32 | 49 | 2 | 19 | 16 | 33 | 50 | 3 |
| $i$ | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| $P(i)$ | 4 | 21 | 38 | 55 | 52 | 5 | 22 | 39 | 36 | 53 | 6 | 23 | 20 | 37 | 54 | 7 |
| $i$ | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| $P(i)$ | 8 | 25 | 42 | 59 | 56 | 9 | 26 | 43 | 40 | 57 | 10 | 27 | 24 | 41 | 58 | 11 |
| $i$ | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
| $P(i)$ | 12 | 29 | 46 | 63 | 60 | 13 | 30 | 47 | 44 | 61 | 14 | 31 | 28 | 45 | 62 | 15 |

### 1.2.3 Алгоритм ENOCORO

**ENOCORO** [7] – це малоресурсний потоковий алгоритм шифрування, що використовується в стандарті ISO/IEC 29192-3:2012 (Part 3). Який складається з трьох частин: блок генерації гами, блок накладання гами та ініціалізації стану.

1. Блок генерації гами - функціонує як генератор псевдовипадкових послідовностей, приймаючи на вхід 128-бітний ключ та 64-бітний ініціалізаційний вектор й видаючи 8 бітів гами за один такт роботи;

2. Блок накладання гами - здійснює побітове додавання (операцію XOR) між 8 бітами гами та 8 бітами відкритого тексту;

3. Ініціалізація стану - перед початком роботи алгоритму для кожного нового повідомлення виконується ініціалізація стану алгоритму.

Алгоритм оперує елементами поля $GF(2^8)$, визначеного поліномом $\psi(x) = x^8 + x^4 + x^3 + x^2 + 1$. Використовує матрицю:

$$L = \begin{pmatrix} 1 & 1 \\ & \\ 1 & d \end{pmatrix}, d = 0\texttt{x}02.$$

Та блок $s_8 : V_8 \to V_8$ - перестановка чисел від 0 до 255.

Також використовуються константи $C_0, \ldots, C_9$:

$C_0 = 0\texttt{x}66, C_1 = 0\texttt{xe9}, C_2 = 0\texttt{x4b},$

$C_3 = 0\texttt{xd4}, C_4 = 0\texttt{xef}, C_5 = 0\texttt{x8a},$

$C_6 = 0\texttt{x2c}, C_7 = 0\texttt{x3b}, C_8 = 0\texttt{x88}, C_9 = 0\texttt{x4c}$

## 1.3 Шифр Вернама та потокове шифрування

Шифр Вернама є прикладом потокового шифрування[8]. Його принцип дії полягає у побітовому додаванню по модулю 2 (операції XOR) відкритого тексту з випадковим ключем тієї ж довжини, який використовується лише один раз:

$$C_i = M_i \oplus K_i,$$

де $C_i$ – біт шифротексту, $M_i$ – біт відкритого тексту, $K_i$ – біт ключа

Шифр Вернама має абсолютну криптостійкість згідно з критерієм Шеннона. Це означає, що навіть маючи необмежену обчислювальну потужність, зловмисник не зможе зменшити ентропію повідомлення без знання ключа, ймовірність відгадати правильне повідомлення дорівнює ймовірності будь-якого іншого варіанту того ж розміру.

Проте, незважаючи на абсолютну стійкість, шифр Вернама має серйозні практичні обмеження: необхідність генерації великого об'єму дійсно випадкових ключів, проблеми з безпечним розповсюдженням та збереженням ключового матеріалу, а також неможливість повторного використання ключа. Ці обмеження унеможливлюють використання шифру Вернама у малоресурсних системах як БПЛА.

Тому використовуються потокові шифри, які побудовані на основі шифру Вернама, але з меншими вимогами до ключів. Вони також грунтуються на операції XOR:

$$C_i = M_i \oplus R_i,$$

де $R_i$ – біт з псевдовипадкової послідовності, згенерованої на основі короткого початкового ключа.

Це дозволяє значно зменшити вимоги до зберігання і передавання ключів. Хоча ентропія ключа у потоковому шифруванні суттєво нижча, ніж у шифрі Вернама, вона все одно може забезпечити достатню практичну стійкість, якщо алгоритм реалізовано правильно, а ключовий матеріал оновлюється вчасно.

## 1.4 Необхідність шифрування зображень у БПЛА

Зображення, отримані з безпілотних літальних апаратів (БПЛА), часто містять критично важливу тактичну інформацію: координати, розташування підрозділів, інженерні споруди, техніку тощо. Потрапляння таких даних до супротивника може призвести до втрат особового складу, зриву операцій або цілеспрямованого вогню. У зв'язку з цим шифрування переданих зображень є обов'язковою умовою інформаційної безпеки під час бойових дій.

«У 99% випадків витік інформації та прослуховування переговорів відбувається не від злому шифрування рації через ефір, а від її втрати. Якщо є хоч найменша підозра, що одна радіостанція могла потрапити до супротивника, потрібно змінювати шифри.[9]» Це твердження прямо вказує на важливість ключової дисципліни та періодичної зміни ключів, згідно з настановами ЗСУ. Для радіорозвідки противника пріоритет мають ключі з малим обсягом (наприклад, 40 біт), однак навіть вони можуть не бути зламані вчасно, якщо застосовуються короткотермінові

ключі й ретельно дотримуються інструкції зі зміни криптографічних параметрів.

З цього випливає, що навіть використання простіших потокових шифрів, які мають меншу ентропію, але є достатньо ефективними при правильному використанні, може на практиці забезпечити реальний захист даних, у тому числі відео- і фотоінформації з БПЛА.

Таким чином, шифрування зображень є не тільки доцільним, а й життєво необхідним елементом сучасного цифрового поля бою. Його відсутність або неправильне застосування шифрування — це прямий ризик для життя військових.

## Висновки до розділу 1

В ході цього розділу було розглянуто основні поняття малоресурсної криптографії, її особливості та значення для захисту інформації в системах із обмеженими обчислювальними ресурсами, для проведення подальшої роботи. Розглянуто алгоритми шифрування PRESENT, GIFT-64, ENOCORO та шифр Вернама. Як видно, тема є актуальною, оскільки за останні роки опубліковано багато наукових статей, які присвячені проблемам ефективного і безпечного шифрування.

# 2 ОЦІНКА СТІЙКОСТІ ТА ПОРІВНЯЛЬНИЙ АНАЛІЗ АЛГОРИТМІВ ШИФРУВАННЯ

У другому розділі буде проведено детальний аналіз стійкості зменшених версій криптографічних малоресурсних алгоритмів PRESENT та GIFT-64 до основних видів атак: лінійного, диференціального та алгебраїчного криптоаналізу. Окрім того, буде виконано порівняльний аналіз розглянутих алгоритмів шифрування та буде визначенно основні вимоги до криптографічних алгоритмів, які застосовуються в умовах обмежених ресурсів.

## 2.1 Стійкість зменшених версій алгоритму PRESENT до атак

В роботі[5] `A.Bogdanov, L.R.Knudsen та інші` зазначити та довели чотири додаткові умови для S-блоків:

Коефіцієнт Фур'є для S позначається через

$$S_b^W(a) = \sum_{x \in \mathbb{F}_2^4} (-1)^{\langle b, S(x) \rangle + \langle a, x \rangle}.$$

1. Для будь-якої ненульової фіксованої вихідної різниці $\Delta_O \in \mathbb{F}_2^4$ та будь-якої ненульової фіксованої вхідної різниці $\Delta_I \in \mathbb{F}_2^4$:

$$\#\{x \in \mathbb{F}_2^4 \mid S(x) + S(x + \Delta_I) = \Delta_O\} \leqslant 4. \tag{2.1}$$

2. Для будь-якої ненульової фіксованої вихідної різниці $\Delta_O \in \mathbb{F}_2^4$ та будь-якої ненульової фіксованої вхідної різниці $\Delta_I \in \mathbb{F}_2^4$ таких, що $wt(\Delta_O) = wt(\Delta_I) = 1$:

$$\{x \in \mathbb{F}_2^4 \mid S(x) + S(x + \Delta_I) = \Delta_O\} = \varnothing. \tag{2.2}$$

3. Для всіх ненульових $b \in \mathbb{F}_4$ і всіх ненульових $a \in \mathbb{F}_2^4$:

$$\left|S_b^W(a)\right| \leqslant 8 \tag{2.3}$$

4. Для всіх ненульових $b \in \mathbb{F}_4$ і всіх $a \in \mathbb{F}_2^4$ таких, що $wt(b) = wt(a) = 1$

$$S_b^W(a) = \pm 4 \tag{2.4}$$

### 2.1.1 Лінійний криптоаналіз

За побудови шару перестановки Perm 1.2 та вимог (2.3) і (2.4), автори `A.Bogdanov, L.R.Knudsen та інші`, вивели та довели теорему 2.1 в роботі [5].

**Теорема 2.1.** *$\epsilon_{4R}$ – максимальна величина відхилення лінійної апроксимації для чотирьох раундів PRESENT, не перевищує $\frac{1}{2^7}$: $\epsilon_{4R} \leqslant \frac{1}{2^7}$.*

Використовуючи теорему 2.1 отримуємо верхню оцінку відхилення 4–раундової апроксимації для PRESENT:

$$(2^{-7})^{\frac{4}{4}} = 2^{-7}$$

Отже, для атаки з відновленням ключа потрібно $2^{14}$ пар відкритого тексту та відповідних шифротекстів. Така кількість даних не перевищує відкритий текст, тобто алгоритм є вразливим до такої атаки. Час, необхідний для її проведення, буде не меншим за час, потрібний для виконання $2^{14}$ зашифрування.

В таблиці 2.1 наведено значення, отримані із застосуванням теореми 2.1 для алгоритму PRESENT з різною кількістю раундів.

**Таблиця 2.1** – Результати лінійного аналізу для різних конфігурацій PRESENT

| **Кількість раундів** | **Шифротекст** | **Лінійна ймовірність** |
|---|---|---|
| 4-7 | $2^{14}$ | $2^{-7}$ |
| 8-11 | $2^{28}$ | $2^{-14}$ |
| 12-15 | $2^{42}$ | $2^{-21}$ |
| 16-23 | $2^{56}$ | $2^{-28}$ |
| 24-27 | $2^{84}$ | $2^{-42}$ |
| 28-31 | $2^{98}$ | $2^{-49}$ |

### 2.1.2 Диференціальний криптоаналіз

За побудови шару перестановки Perm 1.2 та вимоги (2.2), автори `A.Bogdanov, L.R.Knudsen та інші`, вивели та довели теорему 2.2 в роботі [5].

**Теорема 2.2.** *Будь-яка п'ятираундова диференціальна характеристика алгоритму PRESENT має не менші ніж 10 активних S-блоків.*

Використовуючи теорему 2.2 і вимогу (2.1) отримуємо верхню оцінку ймовірності диференціальної характеристи для 7-раундового PRESENT:

$$\left(2^{-2}\right)^{10\cdot\frac{5}{5}+2} = 2^{-24}$$

Отже, для атаки з відновленням ключа потрібно $2^{24}$ пар відкритого тексту та відповідних шифротекстів. Така кількість даних не перевищує відкритий текст, тобто алгоритм є вразливим до такої атаки. Час, необхідний для її проведення, буде не меншим за час, потрібний для виконання $2^{24}$ зашифрування.

В таблиці 2.2 наведено значення, отримані із застосуванням теореми 2.2 для алгоритму PRESENT з різною кількістю раундів.

**Таблиця 2.2** – Результати диференціального аналізу для різних конфігурацій PRESENT

| **Кількість раундів** | **Шифротекст** | **Диференціальна ймовірність** |
|---|---|---|
| 5 | $2^{20}$ | $2^{-20}$ |
| 6 | $2^{22}$ | $2^{-22}$ |
| 7 | $2^{24}$ | $2^{-24}$ |
| 8 | $2^{26}$ | $2^{-26}$ |
| 9 | $2^{28}$ | $2^{-28}$ |
| 10 | $2^{40}$ | $2^{-40}$ |
| 11 | $2^{42}$ | $2^{-42}$ |
| 12 | $2^{44}$ | $2^{-44}$ |
| 13 | $2^{46}$ | $2^{-46}$ |
| 14 | $2^{48}$ | $2^{-48}$ |
| 15 | $2^{60}$ | $2^{-60}$ |
| 16 | $2^{62}$ | $2^{-62}$ |
| 20 | $2^{80}$ | $2^{-80}$ |
| 24 | $2^{88}$ | $2^{-88}$ |
| 25 | $2^{100}$ | $2^{-100}$ |
| 27 | $2^{104}$ | $2^{-104}$ |
| 31 | $2^{122}$ | $2^{-122}$ |

### 2.1.3 Алгебраїчний криптоаналіз

В роботі [5] було доведено, що S-блок в алгоритмі PRESENT описується двадцять одним квадратним рівнянням з 8 вхідними/вихідними бітовими змінними над полем $F_2$. Тому, для того щоб описати роботу усього алгоритму потрібно $21 \cdot n$ квадратичних рівнянь з $8 \cdot n$ змінними. Де n – кількість S-блоків задіяних в алгоритмі шифрування та в процесі розкладу ключів.

Використовуючи дане дослідження, у випадку 8-раундового PRESENT маємо

$$n = 8 \cdot 16 + 8 = 136,$$

тобто формулюється система з 2856 квадратичних рівнянь і 1088 змінних.

Така задача,задача розв'язку нелінійної системи квадратичних рівнянь, є NP-складною. На сьогодні не існує ефективних практичних

результатів алгебраїчного криптоаналізу, оскільки всі відомі методи стискаються з обмеженнями обчислювального часі та обсягу пам'яті.

В таблиці 2.3 наведено значення, отримані із застосуванням дослідження в роботі [5] для алгоритму PRESENT з різною кількістю раундів.

**Таблиця 2.3** – Результати алгебраїчного аналізу для різних конфігурацій PRESENT

| **Кількість раундів** | **Задіяні S-блоки** | **Квадратичні рівняння** | **Змінні** |
|---|---|---|---|
| 4 | 68 | 1428 | 544 |
| 5 | 85 | 1785 | 680 |
| 6 | 102 | 2142 | 816 |
| 7 | 119 | 2499 | 952 |
| 8 | 136 | 2856 | 1088 |
| 9 | 153 | 3213 | 1224 |
| 10 | 170 | 3570 | 1360 |
| 11 | 187 | 3927 | 1496 |
| 12 | 204 | 4284 | 1632 |
| 13 | 221 | 4641 | 1768 |
| 14 | 238 | 4998 | 1904 |
| 15 | 255 | 5355 | 2040 |
| 16 | 272 | 5712 | 2176 |
| 20 | 340 | 7140 | 2720 |
| 24 | 408 | 8568 | 3264 |
| 25 | 425 | 8925 | 3400 |
| 27 | 459 | 9639 | 3672 |
| 31 | 527 | 11067 | 4216 |

## 2.2 Стійкість зменшених версій алгоритму GIFT-64 до атак

### 2.2.1 Диференціальний криптоаналіз

В роботі [6] автори `Subhadeep Banik, Sumit Kumar Pandey, Thomas Peyrin` та інші довели, що диференціальна ймовірність для 9-раундового GIFT-64 становить $2^{-44.415}$. А середнє значення ймовірності за раунд становить $2^{-4.935}$.

Використовуючи дане доведення, отримуємо верхню оцінку диференціальної характеристики для 7 –раундового GIFT-64:

$$(2^{-4.935})^7 = 2^{-34.545}$$

Отже, для атаки з відновленням ключа потрібно $2^{35}$ пар відкритого тексту та відповідних шифротекстів. Така кількість даних не перевищує відкритий текст, тобто алгоритм є вразливим до такої атаки. Час, необхідний для її проведення, буде не меншим за час, потрібний для виконання $2^{35}$ зашифрування.

В таблиці 2.4 наведено значення, отримані із застосуванням довення для 9-раундового GIFT-64 та середнього значення ймовірності за раунд в роботі [6] для алгоритму з різною кількістю раундів.

**Таблиця 2.4** – Результати диференціального аналізу для різних конфігурацій GIFT-64

| **Кількість раундів** | **Шифротекст** | **Диференціальна ймовірність** |
|---|---|---|
| 5 | $2^{25}$ | $2^{-24.675}$ |
| 6 | $2^{30}$ | $2^{-29.610}$ |
| 7 | $2^{35}$ | $2^{-34.545}$ |
| 8 | $2^{40}$ | $2^{-39.48}$ |
| 9 | $2^{45}$ | $2^{-44.415}$ |
| 10 | $2^{50}$ | $2^{-49.35}$ |
| 11 | $2^{55}$ | $2^{-54.285}$ |
| 12 | $2^{60}$ | $2^{-59.22}$ |
| 13 | $2^{65}$ | $2^{-64.155}$ |
| 14 | $2^{70}$ | $2^{-69.09}$ |
| 15 | $2^{75}$ | $2^{-74.025}$ |
| 16 | $2^{79}$ | $2^{-78.96}$ |
| 20 | $2^{99}$ | $2^{-98.7}$ |
| 24 | $2^{119}$ | $2^{-118.440}$ |
| 25 | $2^{124}$ | $2^{-123.375}$ |
| 27 | $2^{134}$ | $2^{-133.245}$ |
| 28 | $2^{139}$ | $2^{-138.18}$ |

### 2.2.2 Лінійний криптоаналіз

В роботі [6] автори Subhadeep Banik, Sumit Kumar Pandey, Thomas Peyrin та інші довели, що лінійна ймовірність для 9-раундового GIFT-64 становить $2^{-49.997}$. А середнє значення ймовірності за раунд становить $2^{-5.555}$.

Аналогічно до диференціального аналізу, використовуючи дане доведення, отримуємо верхню оцінку лінійної характеристики для 7 –раундового GIFT-64:

$$(2^{-5.555})^7 = 2^{-38.886}$$

Отже, для атаки з відновленням ключа потрібно $2^{39}$ пар відкритого тексту та відповідних шифротекстів. Така кількість даних не перевищує відкритий текст, тобто алгоритм є вразливим до такої атаки. Час, необхідний для її проведення, буде не меншим за час, потрібний для виконання $2^{39}$ зашифрування.

В таблиці 2.5 наведено значення, отримані із застосуванням довення для 9-раундового GIFT-64 та середнього значення ймовірності за раунд в роботі [6] для алгоритму з різною кількістю раундів.

### 2.2.3 Алгебраїчний криптоаналіз

В роботі [6] було доведено, що S-блок в алгоритмі GIFT-64 описується двадцять одним квадратним рівнянням з 8 вхідними/вихідними бітовими змінними над полем $F_2$. Тому, для того щоб описати роботу усього алгоритму потрібно $21 \cdot n$ квадратичних рівнять з $8 \cdot n$ змінними. Де n – кількість S-блоків задіяних в алгоритмі шифрування.

Використовуючи дане дослідження, у випадку 7-раундового GIFT-64 маємо

$$n = 7 \cdot 16 = 272,$$

**Таблиця 2.5** – Результати лінійного аналізу для різних конфігурацій GIFT-64

| **Кількість раундів** | **Шифротекст** | **Лінійна ймовірність** |
|---|---|---|
| 5 | $2^{28}$ | $2^{-27.776}$ |
| 6 | $2^{34}$ | $2^{-33.331}$ |
| 7 | $2^{39}$ | $2^{-38.887}$ |
| 8 | $2^{45}$ | $2^{-44.442}$ |
| 9 | $2^{50}$ | $2^{-49.995}$ |
| 10 | $2^{56}$ | $2^{-55.55}$ |
| 11 | $2^{62}$ | $2^{-61.105}$ |
| 12 | $2^{67}$ | $2^{-66.66}$ |
| 13 | $2^{73}$ | $2^{-72.215}$ |
| 14 | $2^{78}$ | $2^{-77.77}$ |
| 15 | $2^{84}$ | $2^{-83.325}$ |
| 16 | $2^{89}$ | $2^{-88.88}$ |
| 20 | $2^{112}$ | $2^{-111.1}$ |
| 24 | $2^{134}$ | $2^{-133.325}$ |
| 25 | $2^{139}$ | $2^{-138.881}$ |
| 27 | $2^{150}$ | $2^{-149.991}$ |
| 28 | $2^{156}$ | $2^{-155.54}$ |

тобто формулюється система з 5712 квадратичних рівнянь і 2176 змінних.

Аналогічно, як в алгебраїчному криптоаналізу PRESENT, така задача,задача розв'язку нелінійної системи квадратичних рівнянь, є NP-складною. На сьогодні не існує ефективних практичних результатів алгебраїчного криптоаналізу, оскільки всі відомі методи стискаються з обмеженнями обчислювального часі та обсягу пам'яті.

В таблиці 2.6 наведено значення, отримані із застосуванням дослідження в роботі [6] для алгоритму GIFT-64 з різною кількістю раундів.

## 2.3 Порівняльний аналіз алгоритмів шифрування для шифрування зображень у БПЛА

При розгляді криптографічних алгоритмів основна увага приділяється їхній криптостійкості, швидкодії та вимогам до ресурсів.

**Таблиця 2.6** – Результати алгебраїчного аналізу для різних конфігурацій GIFT-64

| **Кількість раундів** | **Задіяні S-блоки** | **Квадратичні рівняння** | **Змінні** |
|---|---|---|---|
| 4 | 64 | 1344 | 512 |
| 5 | 80 | 1785 | 680 |
| 6 | 96 | 2142 | 816 |
| 7 | 112 | 2499 | 952 |
| 8 | 128 | 2688 | 1024 |
| 9 | 144 | 3024 | 1152 |
| 10 | 160 | 3360 | 1280 |
| 11 | 176 | 3696 | 1408 |
| 12 | 192 | 4032 | 1536 |
| 13 | 208 | 1785 | 1664 |
| 14 | 224 | 4368 | 1792 |
| 15 | 240 | 5040 | 1920 |
| 16 | 256 | 5376 | 2048 |
| 20 | 320 | 6720 | 2560 |
| 24 | 384 | 8064 | 3072 |
| 25 | 400 | 8400 | 3200 |
| 27 | 432 | 9072 | 3456 |
| 28 | 448 | 9408 | 3584 |

Наприклад, блокові шифри, такі як PRESENT та GIFT-64, є типовими представниками малоресурсних криптографічних алгоритмів. Вони призначені для середовищ з обмеженими обчислювальними можливостями, таких як сенсори або вбудовані системи на борту БПЛА. Ці алгоритми характеризуються компактною реалізацією, низьким енергоспоживанням та стійкістю до відомих типів атак. Наприклад, шифр PRESENT (версія з 80-бітовим ключем) має 64-бітовий стан, як і GIFT-64, проте останній використовує ключ довжиною 128 біт, що впливає на вимоги до пам'яті.

У випадку передачі зображень ситуація ускладнюється. Передача зображення передбачає конвертацію його у цифрову форму, після чого дані шифруються, передаються і дешифруються на стороні приймача. При цьому важливо враховувати, що канал зв'язку БПЛА часто є нестабільним, тобто спотворення під час передачі шифртексту є неминучими. В таких умовах виникає ефект поширення помилки,

характерний для більшості блокових алгоритмів. Це явище полягає у тому, що навіть зміна одного біта у шифртексті призводить до значних змін у розшифрованому повідомленні. Іншими словами, помилка в одному біті після дешифрування може викликати спотворення цілої області зображення, що ускладнює подальший аналіз або візуалізацію. Наприклад, у шифрі PRESENT зміна одного біта вхідного повідомлення або шифртексту в одному раунді шифрування поширюється на багато бітів у наступних раундах, що є наслідком SPN-структури алгоритму, де шари SBoxLayer та PLayer змінюють бітову структуру стану. Аналогічна ситуація спостерігається і в GIFT-64, де складна перестановка pLayer сприяє ще сильнішому поширенню помилки у розшифрованому повідомленні.

Потокові шифри, наприклад ENOCORO, використовують побітове шифрування на основі генерації ключового потоку та виконання XOR-операцій над даними. Це дозволяє частково зменшити ефект поширення помилки, проте у ENOCORO існує особливість: при кожному новому повідомленні необхідна повна ініціалізація стану, що займає 96 тактів. В умовах обмежених обчислювальних ресурсів та потреби в обробці великої кількості окремих фрагментів зображення, це може впливати на затримки під час обробки.

Також можна розглянути алгоритм шифрування, який є аналогом шифру Вернама. У такому підході шифрування здійснюється шляхом XOR-операції між відкритим повідомленням та ключовим потоком. На відміну від класичного шифру Вернама, у цьому підході ключовий потік генерується на основі короткого ключа за допомогою генератора. Перевагою цього підходу є локалізований вплив помилки: зміна одного біта шифртексту призводить до зміни лише одного біта у розшифрованому повідомленні. Це особливо важливо у випадку передачі зображень, оскільки пошкодження одного або кількох бітів у шифртексті призводить до локальних спотворень, які незначно впливають на загальний вигляд зображення.

## 2.4 Вимоги до малоресурсних криптографічних алгоритмів

Отже, можемо вивести основні вимоги до алгоритмів для застосування у БПЛА:

1) Толерантність до помилок при передачі шифртексту: алгоритм повинен забезпечувати локальність помилки — тобто спотворення одного біта в шифртексті не повинно призводити до критичних змін у всьому розшифрованому повідомленні. Це особливо важливо при роботі з візуальними даними.

2) Низьке споживання ресурсів: алгоритм має бути придатним для реалізації у середовищах з обмеженими обчислювальними ресурсами, тобто мінімальне використання оперативної пам'яті, проста реалізація.

3) Достатній рівень криптостійкості: попри спрощення, алгоритм повинен бути захищеним від відомих типів атак та гарантувати тимчасову стійкість (до 2-3 годин) даних при передачі в реальному часі.

4) Висока швидкодія для обробки потоків зображень у реальному часі: шифрування та розшифрування має виконуватися з мінімальними затримками, щоб не впливати на швидкість обробки та передачі зображень, особливо при потоковій передачі з камери БПЛА.

## Висновки до розділу 2

В ході цього розділу проведений криптоаналіз показав, що зменшені версії алгоритмів PRESENT та GIFT-64 вразливі до лінійного та диференціального криптоаналізу, проте все ж забезпечують достатній рівень криптостійкості, необхідний для гарантування тимчасової безпеки.

Також було проведено порівняльний аналіз малоресурсних криптографічних алгоритмі, таких як PRESENT, GIFT-64, ENOCORO, а також шифру Вернама і його аналогу, з точки зору їх застосування для шифрування зображень у безпілотних літальних апаратах. Хоча блокові

шифри, такі як PRESENT та GIFT-64, є ефективними та криптографічно стійкими рішеннями для вбудованих систем з обмеженими ресурсами, вони мають суттєвий недолік при використанні в умовах нестабільного зв'язку — сильне розповсюдження помилок у разі спотворення шифртексту. Це критично при передачі зображень, де навіть одна бітова помилка може суттєво зруйнувати візуальну інформацію. Тому, у таких умовах більш доцільним є застосування потокових шифрів, зокрема ENOCORO чи аналогу шифру Вернама, які забезпечують локалізацію помилок і зберігає придатність зображень до подальшої обробки.

У зв'язку з цим, доцільним є розгляд різних підходів до шифрування, які дозволяють поєднати переваги блокових і потокових алгоритмів та враховують особливості передачі візуальної інформації через нестабільні канали. У подальшому буде проаналізовано кілька варіантів, серед яких — адаптовані модифікації блокового шифрування та застосування потокових шифрів, таких як ENOCORO та аналог шифру Вернама.

# 3 ШИФРУВАННЯ ЗОБРАЖЕНЬ У БПЛА

У третьому розділі буде запропоновано декілька алгоритмів шифрування зображень із застосуванням розглянутих малоресурсних шифрів, а також алгоритм на основі перестановок та інверсії. Для кожного алгоритму буде проведено оцінку обсягу ключового простору, а також визначено обсяг пам'яті, необхідний для зберігання ключа та зашифрованого зображення.

## 3.1 Подання зображення

Спочатку формалізуємо подання даних зображення для того щоб потім використовувати це подання при зашифруванні.

Нехай маємо зображення розміру $n \times m$, яке складається з $n$ рядків та $m$ стовпців пікселів. Кожен піксель задається значенням яскравості в градаціях сірого, тобто належить до множини $V_L = \{0,1\}^L$, де $L = 8$ – кількість бітів, необхідних для представлення одного пікселя. Також можливе використання кольорового формату зображення. У цьому випадку для кожного пікселя необхідно використовувати $L = 24$ біти.

Тоді зображення можна подати у вигляді матриці:

$$A := (a_{ij})_{i=1,j=1}^{n,m}, \quad a_{ij} \in V_L, \tag{3.1}$$

де $a_{ij}$— це $L$–бітове представлення значення пікселя в $i$–му рядку і $j$–му стовпцю.

Для деяких типів шифрування ми будемо перетворювати матрицю $A$ у бітовий рядок наступним чином: всі елементи $a_{ij}$ зчитуються у певному фіксованому порядку – по рядках зліва направо. Таким чином отримаємо бітову послідовність довжини $N = n \cdot m \cdot L$:

$$M = (a_{11} \parallel a_{12} \parallel \cdots \parallel a_{1m} \parallel a_{21} \parallel \cdots \parallel a_{nm}) \in V_N, \tag{3.2}$$

де $\parallel$ – конкатенація бітових векторів.

## 3.2 Шифрування за допомогою зменшеної версії одного з блокових шифрів

Використання зменшеної версію одного із блокових шифрів, зокрема 8-16 раундів алгоритму PRESENT, має низку переваг. Цей алгоритм має компактну реалізацію, мале споживання пам'яті та обчислювальних ресурсів, що важливо для вбудованих систем, зокрема в умовах використання у безпілотних літальних апаратів. Крім того, шифр демонструє достатній рівень криптографічної стійкості, до 2 – 3 годин, для задач обробки та передачі зображень.

Але блокові шифри має ефект лавиноподібного поширення помилки. Це означає, що одне бітове викревлення у зашифрованих даних може призвести до масової деформації результату при розшифруванню. Замість частково пошкодженого зображення ми отримаємо повністю зіпсовану картинку.

Саме тому пропоную використовувати зменшену версію алгоритму PRESENT у режимі лічильника – CTR(Counter Mode) для генерації послідовності бітів, гамми. Цей режим шифрування перетворює блоковий шифр на потоковий, де шифрування кожного блоку здійснюється незалежно від інших. Такий підхід забезпечує стійкість до часткових викривлень: якщо окремі біти або навіть цілі блоки шифротексту будуть пошкоджені або змінені, це призведе лише до локального спотворення відповідного фрагменту зображення при розшифруванні, не впливаючи на інші частини. Тобто інформація не знищується, а лише частково спотворюється, що є критично важливим для цілісності візуального сприйняття при передаванні зображень у реальному часі.

Необхідно також розглянути два варіанти генерації гами, на землі та на борту БПЛА.

### 3.2.1 Генерація гамми на борту

Цей варіант доцільний у випадках, коли обчислювальних ресурсів БПЛА достатньо для реалізації алгоритму.

Нехай $E_k^{(8)} : V_{64} \to V_{64}$ – зменшена версія шифру PRESENT з 8 раундами, що приймає на вхід 64–бітовий блок та 80–бітовиий ключ $k \in V_{80}$. Вхідна бітова послідовність зображення $M$ (3.2) розбивається на блоки довжиною 64 біти:

$$M = M_1 \parallel M_2 \parallel \cdots \parallel M_t, \quad M_i \in V_{64},$$

де до останнього блоку застосовується паддінг(доповнення нулями) за необхідністю.

Далі формуються блоки лічильники $T_i$ за правилом:

$$T_i = \mathrm{IV} \parallel \langle i \rangle,$$

$\langle i \rangle \in V_r$ – двійкове представлення номеру блоку $i$, де $r = \lceil \log_2 t \rceil$ – довжина, тобто кількість бітів, необхідна для кодування всіх номерів блоків від 0 до $t - 1$. $\mathrm{IV} \in V_{64-r}$ – ініціалізуючий або початковий вектор, що задається випадково або відомим значенням.

І обчислюється ключова гамма:

$$\begin{aligned} &s_i = E_k^{(8)}(T_i), \quad i = 1, 2, \ldots, t, \quad s_i \in V_{64}, \\ &S = s_1 \parallel s_2 \parallel \cdots \parallel s_t \end{aligned} \tag{3.3}$$

Далі шифрування відбувається за формулою:

$$C_i = M_i \oplus s_i, \quad i = 1, 2, \ldots, t \tag{3.4}$$

В результаті утворюється послідовність зашифрованих блоків $C_i \in V_{64}$, які об'єднуються в один шифротекст:

$$C = C_1 \parallel C_2 \parallel \cdots \parallel C_t \in V_{t \cdot 64} \tag{3.5}$$

Розшифрування відбувається аналогічно:

$$T_i = \mathrm{IV} \parallel \langle i \rangle,$$

$$S_i = E_k^{(8)}(T_i),$$

$$M_i = C_i \oplus S_i, \quad i = 1, 2, \ldots, t$$

В результаті отриману послідовність $M = M_1 \parallel M_2 \parallel \ldots \parallel M_t$ перетворюємо назад у матрицю зображення, послідовно зчитуючи по 8–бітів на піксель. Перед цим, у разі застосування паддінгу під час шифрування, необхідно відкинути зайві нульові біти.

### 3.2.2 Генерація гамми завчасно на землі

Цей варіант використовується у випадках, коли пам'ять або обчислювальні ресурси БПЛА обмежені та не дозволяють повноцінну реалізацію зменшеного алгоритму PRESENT на борту, через необхідність зберігання таблиць підстановок та перестановок.

Ключова гамма $S$ (3.3) завчасно генерується на землі та попередньо завантажується у пам'ять БПЛА.

Далі шифрування та розшифрування зображення відбувається аналогічно до випадку, коли гама генерується на борту.

Вхідна бітова послідовність зображення $M$ (3.2) розбивається на блоки довжиною 64 біти:

$$M = M_1 \parallel M_2 \parallel \cdots \parallel M_t, \quad M_i \in V_{64},$$

де до останнього блоку застосовується паддінг(доповнення нулями) за необхідністю.

Шифрування відбувається за формулою:

$$C_i = M_i \oplus s_i, \quad i = 1, 2, \ldots, t \tag{3.6}$$

В результаті утворюється послідовність зашифрованих блоків $C_i \in V_{64}$, які об'єднуються в один шифротекст:

$$C = C_1 \parallel C_2 \parallel \cdots \parallel C_t \in V_{t \cdot 64} \tag{3.7}$$

Розшифрування:

$$T_i = \text{IV} \parallel \langle i \rangle,$$

$$S_i = E_k^{(8)}(T_i),$$

$$M_i = C_i \oplus S_i, \quad i = 1, 2, \ldots, t$$

В результаті отриману послідовність $M = M_1 \parallel M_2 \parallel \ldots \parallel M_t$ перетворюємо назад у матрицю зображення, послідовно зчитуючи по 8–бітів на піксель. Перед цим, у разі застосування паддінгу під час шифрування, необхідно відкинути зайві нульові біти.

## 3.3 Шифрування за допомогою потокових алгоритмів

Для забезпечення стійкого до спотворень шифрування зображень, які передаються БПЛА в умовах перешкод у каналі зв'язку, пропоную використовувати потоковий підхід на основі аналога шифру Вернама. У цьому випадку шифрування виконується за допомогою побітової операції XOR між байтами зображення та псевдовипадковою гамою, що генерується криптографічно стійким генератором ENOCORO.

Завдяки властивостям аналога шифру Вернама, навіть у разі спотворення одного або кількох бітів у шифротексті, після дешифрування

буде спотворено лише відповідні біти повідомлення, без руйнування всієї структури.

Генератор ENOCORO є потоковим шифром, здатним генерувати криптографічно стійку послідовність байтів для використання як гамми. Попри складну початкову ініціалізацію (96 тактів), він забезпечує високу продуктивність після її завершення. Тому пропоную здійснювати ініціалізацію генератора на землі, формуючи заздалегідь ключову гамму або набір ключових гам, які БПЛА використовує під час польоту з періодичним оновленням. Та попередньо завантажується у пам'ять БПЛА.

Нехай $E_k$ – генератор ENOCORO, що приймає на вхід 128-бітний ключ $k \in V_{128}$ та 64-бітний ініціалізаційний вектор IV $\in V_{64}$. І нехай $A_0 \in V_{32 \cdot 8}$ – стартове заповнення стану і $B_0 \in V_{2 \cdot 8}$ – стартове заповнення буфера.

Ініціалізація генератора виконується на землі шляхом прокрутки ENOCORO:

$$S_{96} = E_k^{(96)}(IV),$$

де $E_k^{(96)}(IV)$ — результат роботи генератора після 96 тактів з використанням ключа $k$ та вектора ініціалізації $IV$.

З цього результату знаходяться початкові значення:

$$B_0 = buf(S_{96}), \quad A_0 = st(S_{96}), \tag{3.8}$$

де функції $buf$ та $st$ повертають заповнення буфера і стану відповідно.

### 3.3.1 Використання єдиної гамми

На основі $B_0$ та $A_0$ (3.8) ENOCORO генерує потік бітів, який буде використовуватись як ключова гамма, видаючи 8 бітів гамми за один такт роботи. Ця послідовність генерується на землі до вильоту БПЛА і

завантажується в його пам'ять:

$$\gamma = \gamma_1 \parallel \gamma_2 \parallel \cdots \parallel \gamma_{n \cdot m} \quad \gamma_i \in V_8, \tag{3.9}$$

Якщо вся ключова гамма поміщається у пам'ять БПЛА, то шифрування всіх зображень під час польоту відбувається з використанням цієї єдиної гамми без її оновлення, що забезпечує простоту реалізації і мінімальні обчислювальні затрати.

Вхідна бітова послідовність зображення $M$ (3.2) розбивається на блоки довжиною 8 біти:

$$M = M_1 \parallel M_2 \parallel \cdots \parallel M_{n \cdot m}, \quad M_i \in V_8$$

Шифрування виконується так:

$$C_i = M_i \oplus \gamma_i, \quad i = 1, 2, \ldots, n \cdot m \tag{3.10}$$

В результаті утворюється послідовність зашифрованих блоків $C_i \in V_8$, які об'єднуються в один шифротекст:

$$C = C_1 \parallel C_2 \parallel \cdots \parallel C_{n \cdot m} \tag{3.11}$$

Розшифрування виконується аналогічно:

$$M_i = C_i \oplus \gamma_i, \quad i = 1, 2, \ldots, n \cdot m.$$

В результаті отриману послідовність $M = M_1 \parallel M_2 \parallel \ldots \parallel M_{n \cdot m}$ перетворюємо назад у матрицю зображення, послідовно зчитуючи по 8–бітів на піксель.

### 3.3.2 Використання кількох унікальних гамм

Якщо пам'ять БПЛА дозволяє зберігати кілька гамм, можна заздалегідь на землі згенерувати набір гамм.

На основі $B_0$ та $A_0$ (3.8) ENOCORO генерує потік бітів, який буде використовуватись як ключова гамма, видаючи 8 бітів гамми за один такт роботи:

$$\gamma = \gamma_1 \parallel \gamma_2 \parallel \cdots \parallel \gamma_{n \cdot m} \parallel \gamma_{n \cdot m+1} \parallel \cdots \parallel \gamma_{k \cdot n \cdot m}, \quad \gamma_i \in V_8, \tag{3.12}$$

де $k$ – кількість підгамм, тобто максимальна кількість зображень, які можуть бути зашифровані унікальними гамами перед повторенням. Ця послідовність генерується на землі до вильоту БПЛА і завантажується в його пам'ять.

Далі ця послідовність розбивається на $k$ підпослідовностей:

$$\gamma = \gamma^{(1)} \parallel \gamma^{(2)} \parallel \cdots \parallel \gamma^{(k)}, \quad \gamma^{(j)} \in V_{8 \cdot n \cdot m}$$

де кожна підпослідовність має вигляд:

$$\gamma^{(j)} = \gamma_1^{(j)} \parallel \gamma_2^{(j)} \parallel \cdots \parallel \gamma_{n \cdot m}^{(j)}, \quad \gamma_i^{(j)} \in V_8$$

Вхідна бітова послідовність зображення $M$ (3.2) розбивається на блоки довжиною 8 біти:

$$M = M_1 \parallel M_2 \parallel \cdots \parallel M_{n \cdot m}, \quad M_i \in V_8$$

Під час польоту БПЛА гамма змінюється кожні 5 секунд, тобто кожне нове зображення шифрується новою підгаммою:

$$\gamma^{(1)} \to \gamma^{(2)} \to \cdots \to \gamma^{(k)} \to \gamma^{(1)} \to \ldots.$$

Якщо політ триває довше, ніж вистачає унікальних підгамм, відбувається

циклічне повторне використання гамм. Тобто для $j$-го зображення використовується гамма з індексом $j \bmod k$.

Шифрування здійснюється за формулою:

$$C_i = M_i \oplus \gamma_i^{j \bmod k}, \quad i = 1, 2, \ldots, n \cdot m. \tag{3.13}$$

В результаті утворюється послідовність зашифрованих блоків $C_i \in V_8$, які об'єднуються в один шифротекст:

$$C = C_1 \parallel C_2 \parallel \cdots \parallel C_{n \cdot m}, \tag{3.14}$$

Розшифрування здійснюється аналогічним чином:

$$M_i = C_i \oplus \gamma_i^{j \bmod k}, \quad i = 1, 2, \ldots, n \cdot m.$$

В результаті отриману послідовність $M = M_1 \parallel M_2 \parallel \ldots \parallel M_{n \cdot m}$ перетворюємо назад у матрицю зображення, послідовно зчитуючи по 8 бітів на піксель.

Отже, для кожного наступного зображення у процесі польоту БПЛА використовується нова підгамма відповідно до визначеного циклу, забезпечуючи оновлення ключового потоку та підвищення криптостійкості системи при передачі послідовних зображень.

### 3.3.3 Використання часткової гамми

У разі якщо навіть одна повна гамма не поміщається в пам'ять БПЛА, використовується укорочена ключова гамма довжиною $8 \cdot r \cdot m$ бітів:

$$\gamma = \gamma_1 \parallel \gamma_2 \parallel \cdots \parallel \gamma_{r \cdot m}, \quad \gamma_i \in V_8, \quad r < n \tag{3.15}$$

де $r$ – кількість рядків зображення, для яких генерується гамма. Як і в попередніх варіантах, ця гамма генерується на землі генератором ENOCORO на основі значень $B_0$ та $A_0$ (3.8). І завантажується у пам'ять

БПЛА.

У цьому випадку шифрування зображення виконується блоками, кожен з яких обробляється окремо, але шифруються тією самою гаммою.

Вхідна бітова послідовність зображення $M$ (3.2) розбивається на блоки довжиною $8 \cdot r \cdot m$ бітів:

$$M = B_1 \parallel B_2 \parallel \cdots \parallel B_{\frac{n}{r}}, \quad B_j \in V_{8 \cdot r \cdot m}, \quad j = 1, \ldots, \frac{n}{r}$$

Далі кожен блок $B_j$ розбивається на блоки довжиною 8 біт:

$$B_j = b_{j_1} \parallel b_{j_2} \parallel \cdots \parallel b_{j_{r \cdot m}}, \quad b_{j_i} \in V_8$$

Для кожного блоку $B_j$ виконується шифрування:

$$c_{j_i} = b_{j_i} \oplus \gamma_i, \quad i = 1, \ldots, r \cdot m, \quad j = 1, \ldots, \frac{n}{r}. \tag{3.16}$$

Отримуємо шифротекст для кожного $B_j$ довжиною $8 \cdot r \cdot m$ бітів:

$$C_j = c_{j_1} \parallel c_{j_2} \parallel \cdots \parallel c_{j_{r \cdot m}}, \quad j = 1, \ldots, \frac{n}{r}.$$

В результаті фінальний шифротекст для зашифрованого зображення має вигляд:

$$C = C_1 \parallel C_2 \parallel \cdots \parallel C_{\frac{n}{r}}, \quad C_j \in V_{8 \cdot r \cdot m} \tag{3.17}$$

Розшифровування в цьому випадку також відбувається аналогічно до шифрування. Шифротекст зашифрованого зображення розбивається на блоки:

$$C = C_1 \parallel C_2 \parallel \cdots \parallel C_{\frac{n}{r}}, \quad C_j \in V_{8 \cdot r \cdot m}$$

Далі кожен блок $C_j$ розбивається на блоки довжиною 8 біт:

$$C_j = c_{j_1} \parallel c_{j_2} \parallel \cdots \parallel c_{j_{r \cdot m}}, \quad c_{j_i} \in V_8$$

Для кожного блоку $C_j$ виконується шифрування:

$$b_{j_i} = c_{j_i} \oplus \gamma_i, \quad i = 1, \ldots, r \cdot m, \quad j = 1, \ldots, \frac{n}{r}.$$

Отримуємо шифротекст для кожного $C_j$ довжиною $8 \cdot r \cdot m$ бітів:

$$B_j = b_{j_1} \parallel b_{j_2} \parallel \cdots \parallel b_{j_{r \cdot m}}, \quad j = 1, \ldots, \frac{n}{r}.$$

В результаті відновлюємо послідовність $M = B_1 \parallel B_2 \parallel \cdots \parallel B_{\frac{n}{r}}$ і перетворюємо назад у матрицю зображення, послідовно зчитуючи по 8 бітів на піксель.

### 3.4 Шифрування за допомогою перестановок

Наступний алгоритм шифрування зображень полягає у простому, але ефективному способі захисту зображення без перетворення його в бітову послідовність. Це є особливо актуальним для передавання зображень із безпілотних літальних апаратів у складних умовах каналу зв'язку з перешкодами.

Для шифрування будемо використовувати дві перестановки: $\sigma_1$ – для рядків і $\sigma_2$ – для стовпців. Обидві перестановки є бієктивними відображеннями:

$$\sigma_1 : \{1, \ldots, n\} \to \{1, \ldots, n\}, \quad \sigma_2 : \{1, \ldots, m\} \to \{1, \ldots, m\} \tag{3.18}$$

Ці перестановки визначають новий порядок рядків та стовпців у матриці зображення, забезпечуючи певний рівень перемішування пікселів.

Проте, якщо використовувати лише перестановки, існує ризик, що деякі рядки або стовпці, які є однотонними або мають схожу структуру, можуть зберігати свою візуальну впізнаваність навіть після шифрування. Це особливо проблематично для зображень із великими одноколірними областями або послідовностями пікселів зі схожими значеннями. У такому

випадку перестановка не зможе забезпечити достатній рівень перемішування.

Для захисту вхідного зображення пропоную застосовувати перестановки рядків та стовпців, а також інверсію, змінення порядок елементів на протилежний для деяких рядків та стовпців. Такий підхід забезпечує ефективне перемішування пікселів і унеможливлює відновлення візуальної структури без знання ключа. Навіть у випадку часткового спотворення чи втрати даних, якщо ключ збережено, зображення можна відновити з мінімальними спотвореннями.

### 3.4.1 Використання лінійної перестановки та інверсії

Нехай маємо вхідне зображення розміру $n \times m$ у вигляді матриці $A := (a_{ij})_{i=1,j=1}^{n,m}$ (3.1).

Як частковий випадок, для реалізації перестановок $\sigma_1$ та $\sigma_2$, можна застосувати лінійну перестановку:

$$\begin{aligned} \sigma_1(i) &= (u_1 \cdot i + v_1) \bmod n, \qquad \gcd(u_1, n) = 1, \\ &\quad u_1, v_1 \in \{0, \ldots, n-1\}, \\ \sigma_2(j) &= (u_2 \cdot j + v_2) \bmod m, \qquad \gcd(u_2, m) = 1, \\ &\quad u_2, v_2 \in \{0, \ldots, m-1\}. \end{aligned} \tag{3.19}$$

А вектори для інверсії $R$ і $B$ задаються випадково, мають довжину $n$ і $m$ відповідно та визначають, чи потрібно змінити порядок елементів рядка чи стовпця на протилежний після перестановки:

$$\begin{aligned} R &= (r_i)_{i=0}^{n-1}, \quad r_i \in \{0{,}1\} \\ B &= (b_j)_{j=0}^{m-1}, \quad b_j \in \{0{,}1\} \end{aligned} \tag{3.20}$$

Ключем шифрування буде $k = (u_1, v_1, u_2, v_2, R, B)$

Тоді шифрування виконується так:

1. Перестановка рядків:

$$A^{(1)} = (a_{ij}^{(1)})_{i=1,j=1}^{n,m}, \quad a_{ij}^{(1)} = a_{\sigma_1(i),j}$$

2. Перестановка стовпців:

$$A^{(2)} = (a_{ij}^{(2)})_{i=1,j=1}^{n,m}, \quad a_{ij}^{(2)} = a_{i,\sigma_2(j)}^{(1)}$$

3. Інверсія рядків:

Для кожного рядка $i$, якщо $r_i = 1$ змінюємо порядок елементів на протилежний:

$$a_{i,j}^{(3)} = \begin{cases} a_{i,j}^{(2)}, & \text{якщо } r_i = 0, \\ a_{i,m-1-j}^{(2)}, & \text{якщо } r_i = 1. \end{cases}$$

4. Інверсія стовпців:

Для кожного стовпця $j$, якщо $b_j = 1$ змінюємо порядок елементів на протилежний:

$$c_{i,j} = \begin{cases} a_{i,j}^{(3)}, & \text{якщо } b_j = 0, \\ a_{n-1-i,j}^{(3)}, & \text{якщо } b_j = 1. \end{cases} \tag{3.21}$$

Отримана матриця $C = (c_{ij})_{i=1,j=1}^{n,m}$ – зашифрована матриця зображення.

Для розшифрування необхідно мати ключ – $k$ та матрицю зашифрованого зображення. Розшифрування виконується так:

1. Зворотня інверсія стовпців:

$$c_{i,j}^{(1)} = \begin{cases} c_{i,j}, & \text{якщо } b_j = 0, \\ c_{n-1-i,j}, & \text{якщо } b_j = 1. \end{cases}$$

2. Зворотня інверсія рядків:

$$c_{i,j}^{(2)} = \begin{cases} c_{i,j}^{(1)}, & \text{якщо } r_i = 0, \\ c_{i,m-1-j}^{(1)}, & \text{якщо } r_i = 1. \end{cases}$$

3. Зворотня перестановка стовпців:

$$A^{(1)} = (c_{ij}^{(3)})_{i=1,j=1}^{n,m}, \quad c_{ij}^{(3)} = c_{i,\sigma_2^{-1}(j)}^{(2)}$$

4. Зворотня перестановка рядків:

$$A = (a_{ij})_{i=1,j=1}^{n,m}, \quad a_{ij} = c_{\sigma_1^{-1}(i),j}^{(3)}$$

В результаті отримуємо оригінальну матрицю зображення. Яку перетворюємо у зображення.

## 3.5 Аналіз запропонованих алгоритмів шифрування зображення

Наразі у відкритих джерелах немає достовірної інформації про точний обсяг оперативної пам'яті, доступної на борту конкретного безпілотного літального апарата. Проте очевидно, що використання пам'яті для алгоритмів шифрування має бути мінімальним, щоб залишити ресурси для інших критичних завдань.

Усі запропоновані алгоритми вимагають зберігання в оперативній пам'яті ключа шифрування та зашифрованого зображення. Ключ шифрування необхідно зберігати в оперативній пам'яті щонайменше до моменту завершення шифрування та передачі даних на землю. А зашифроване зображення, залежно від способу передачі, може зберігатися в оперативній пам'яті повністю або частково — до моменту завершення передачі.

Таким чином, доцільно оцінити обсяг ключового простору, а також визначити пам'ять, необхідну для зберігання ключа та зашифрованого зображення для кожного алгоритму.

### 3.5.1 Зменшена версія алгоритму PRESENT у режимі CTR

Нехай маємо зображення розміру $1280 \times 720$.

**Генерація гами на борту**

Алгоритм використовує зменшену версію шифру PRESENT з довжиною ключа 80 біт. Таким чином потужність ключового простору становить:

$$2^{80} = 1.21 \times 10^{24}$$

Цей обсяг є достатньо великим для забезпечення стійкості до атаки повного перебору ключів.

**1. Необхідна пам'ять для зберігання ключових параметрів**

При використанні зменшеної версії алгоритму PRESENT, 8 раундів, для генерації гамми (3.3) на борту БПЛА, не потрібно зберігати повну гамму.

Необхідно зберігати та передавати по відкритому каналу оператору:

1) Основний ключ PRESENT – $k$: 80 біт або 10 байт

2) Ініціалізуючий вектор – $IV$: 47 біт або 6 байт

3) Таблицю підстановки: $16 \cdot 8 = 128$ біт або 16 байт

4) Таблицю перестановки: $64 \cdot 8 = 512$ біт або 64 байти

Це пояснюється тим, що гамма —- це послідовність бітів, яка генерується на основі ключа, ініціалізуючого вектора та таблиць підстановки і перестановки. На основі цих параметрів оператор на землі може повторно згенерувати гамму та дешифрувати отриманий шифротекст зашифрованого зображення.

**2. Необхідна пам'ять для зберігання шифротексту зашифрованого зображення**

*Випадок 1:* Якщо зв'язок нестабільний і є достатньо пам'яті для зберігання, БПЛА передає по відкритому каналу оператору шифротекст зашифрованого зображення пакетно, тобто після повного

шифрування (3.5). У цьому випадку необхідно зберігати весь шифротекст: $n \cdot m \cdot 8 = 1280 \cdot 720 \cdot 8 = 7372800$ біт або 921600 байт.

*Випадок 2:* Якщо зв'язок стабільний і не вистачає пам'яті для зберігання повного шифротексту, передача здійснюється потоково по блоках (3.4), одному або декількох. Для збереження одного блоку достатньо виділити 64 біти або 8 байт оперативної пам'яті. Після передачі блока він одразу затирається, а наступний блок завантажується.

**Генерація гамми завчасно на землі**

**1. Необхідна пам'ять для зберігання ключової гамми**

Згенеровану гамму (3.3) попередньо вшивають у пам'ять БПЛА. Тому необхідно виділити пам'ять для зберігання повної гамми, що використовується для шифрування всього зображення: $n \cdot m \cdot 8 = 1280 \cdot 720 \cdot 8 = 7372800$ біт або 921600 байт.

**2. Необхідна пам'ять для зберігання шифротексту зашифрованого зображення**

*Випадок 1:* Якщо зв'язок нестабільний і є достатньо пам'яті для зберігання, БПЛА передає по відкритому каналу оператору шифротекст зашифрованого зображення пакетно, тобто після повного шифрування (3.7). У цьому випадку необхідно зберігати весь шифротекст: $n \cdot m \cdot 8 = 1280 \cdot 720 \cdot 8 = 7372800$ біт або 921600 байт.

*Випадок 2:* Якщо зв'язок стабільний і не вистачає пам'яті для повного зберігання шифротексту, передача здійснюється потоково по блоках (3.6), одному або декількох. Для збереження одного блоку достатньо виділити 64 біти або 8 байт оперативної пам'яті. Після передачі блока він одразу затирається, а наступний блок завантажується.

### 3.5.2 Потоковий алгоритм

В розробленому алгоритмі шифрування зображень генерація ключової гамми або набору ключових гамм виконується завчасно на

земній станції за допомогою генератора ENOCORO. Після чого заздалегідь завантажується у пам'ять БПЛА.

Нехай маємо зображення розміру $1280 \times 720$.

### Використання єдиної гамми

**1. Необхідна пам'ять для зберігання ключової гамми**

Згенеровану гамму (3.9) попередньо вшивають у пам'ять БПЛА. Тому необхідно виділити пам'ять для зберігання повної гамми, що використовується для шифрування всього зображення: $n \cdot m \cdot 8 = 1280 \cdot 720 \cdot 8 = 7372800$ біт або 921600 байт

**2. Необхідна пам'ять для зберігання шифротексту зашифрованого зображення**

*Випадок 1:* Якщо зв'язок нестабільний і є достатньо пам'яті для зберігання, БПЛА передає по відкритому каналу оператору шифротекст зашифрованого зображення пакетно, тобто після повного шифрування (3.11). У цьому випадку необхідно зберігати весь шифротекст: $n \cdot m \cdot 8 = 1280 \cdot 720 \cdot 8 = 7372800$ біт або 921600 байт

*Випадок 2:* Якщо зв'язок стабільний і не вистачає пам'яті для повного зберігання шифротексту, передача здійснюється потоково по блоках (3.10), одному або декількох. Для збереження одного блоку достатньо виділити 8 біт або 1 байт оперативної пам'яті. Після передачі блока він одразу затирається, а наступний блок завантажується.

### Використання кількох унікальних гамм

Нехай $k = 10$ – максимальна кількість зображень, які можуть бути зашифровані унікальними гамами перед повторенням.

**1. Необхідна пам'ять для зберігання ключової гамми**

Згенеровану гамму (3.12) попередньо вшивають у пам'ять БПЛА. Тому необхідно виділити пам'ять для зберігання повної гами, що використовується для шифрування зображень: $k \cdot n \cdot m \cdot 8 = 10 \cdot 1280 \cdot 720 \cdot 8 = 73728000$ біт або 9216000 байт.

**2. Необхідна пам'ять для зберігання шифротексту зашифрованого зображення**

*Випадок 1:* Якщо зв'язок нестабільний і є достатньо пам'яті для зберігання, БПЛА передає по відкритому каналу оператору шифротекст зашифрованого зображення пакетно, тобто після повного шифрування (3.14). У цьому випадку необхідно зберігати весь шифротекст: $n \cdot m \cdot 8 = 1280 \cdot 720 \cdot 8 = 7372800$ біт або 921600 байт

*Випадок 2:* Якщо зв'язок стабільний і не вистачає пам'яті для зберігання повного шифротексту, передача здійснюється потоково по блоках (3.13), одному або декількох. Для збереження одного блоку достатньо виділити 8 біт або 1 байт оперативної пам'яті. Після передачі блока він одразу затирається, а наступний блок завантажується.

### Використання часткової гамми

Нехай $r = 2$ – максимальна кількість рядків зображення, для яких генерується гамма.

**1. Необхідна пам'ять для зберігання ключової гамми**

Згенеровану гамму (3.15) попередньо вшивають у пам'ять БПЛА. Тому необхідно виділити пам'ять для зберігання повної згенерованої гами, що використовується для шифрування $r$ рядків зображення: $r \cdot m \cdot 8 = 2 \cdot 720 \cdot 8 = 11520$ біт або 1440 байт.

**2. Необхідна пам'ять для зберігання шифротексту зашифрованого зображення**

*Випадок 1:* Якщо зв'язок нестабільний і є достатньо пам'яті для зберігання, БПЛА передає по відкритому каналу оператору шифротекст зашифрованого зображення пакетно, тобто після повного шифрування (3.17). У цьому випадку необхідно зберігати весь шифротекст: $n \cdot m \cdot 8 = 1280 \cdot 720 \cdot 8 = 7372800$ біт або 921600 байт

*Випадок 2:* Якщо зв'язок стабільний і не вистачає пам'яті для зберігання повного шифротексту, передача здійснюється потоково по блоках (3.16), одному або декількох. Для збереження одного блоку

достатньо виділити $r \cdot m \cdot 8 = 2 \cdot 720 \cdot 8 = 11520$ біт або 1440 байт оперативної пам'яті. Після передачі блока він одразу затирається, а наступний блок завантажується.

### 3.5.3 Перестановки та інверсії

Нехай маємо зображення розміру $1280 \times 720$.

У розробленому алгоритмі ключ шифрування складається з таких компонентів:

$$(u_1, v_1, u_2, v_2, R, B)$$

Для перестановок $\sigma_1$ та $\sigma_2$ (3.19): $u_1$ має $\varphi(n)$ можливих значень, для $v_1$ – $n$ можливих значень. Відповідно для $u_2$ — $\varphi(m)$ можливих значень, для $v_2$ — $m$.

Для векторів інверсії (3.20): $R$ має $2^n$ можливих значень, а $B$ — $2^m$.

Таким чином потужність ключового простору визначається як:

$$|k| = \varphi(n) \cdot n \cdot \varphi(m) \cdot m \cdot 2^{n+m} = 512 \cdot 1280 \cdot 192 \cdot 720 \cdot 2^{1280+720} = 1.04 \cdot 10^{613}$$

Цей обсяг ключового простору є надзвичайно великим і гарантує надійний захист від атаки повного перебору навіть для найсучасніших обчислювальних потужностей.

**1. Необхідна пам'ять для зберігання ключових параметрів**

Незважаючи на величезний обсяг ключового простору, практичні вимоги до зберігання ключових параметрів досить помірні. Для передачі та зберігання достатньо зберігати випадкові параметри:

1) $u_1, v_1$ для перестановки рядків: $2 \cdot \lceil \log_2 n \rceil = 22$ біт або 4 байти

2) $u_1, v_2$ для перестановки стовпців: $2 \cdot \lceil \log_2 m \rceil = 20$ біт або 4 байти

3) Вектор інверсії рядків: $n = 1280$ біт або 160 байт

4) Вектор інверсії стовпців: $m = 720$ біт або 90 байт

Таким чином, загальний обсяг пам'яті для зберігання всіх ключових параметрів не перевищує кількох сотень байт. Це дозволяє ефективно

застосовувати алгоритм шифрування. Збережені параметри дозволяють оператору наземного центру безпечно відтворити ключ шифрування та виконати дешифрування зображення після його отримання.

**2. Необхідна пам'ять для зберігання матриці зашифрованого зображення**

*Випадок 1:* Якщо зв'язок нестабільний і є достатньо пам'яті для зберігання, БПЛА передає по відкритому каналу оператору матрицю зашифрованого зображення пакетно, тобто після повного шифрування. У цьому випадку необхідно зберігати всю зашифровану матрицю: $n \cdot m \cdot 8 = 1280 \cdot 720 \cdot 8 = 7372800$ біт або 921600 байт

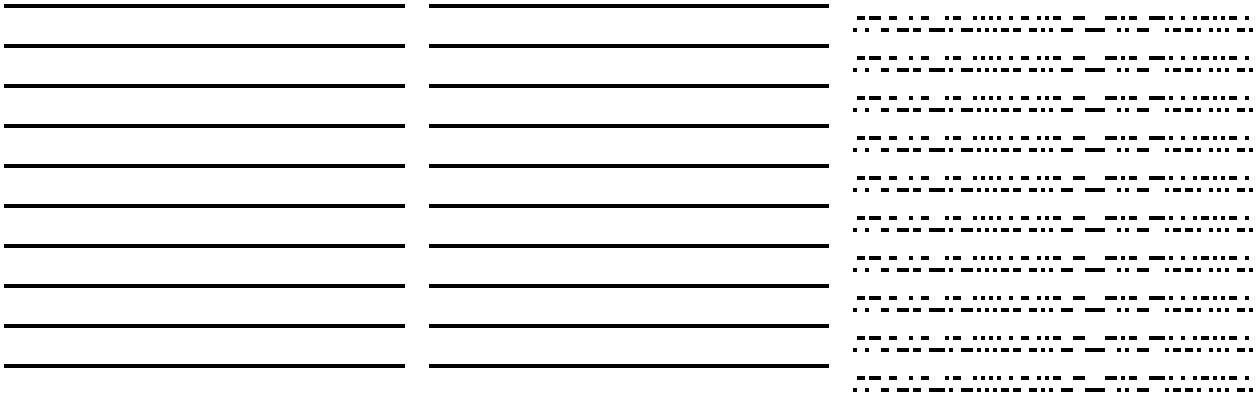
*Випадок 2:* Якщо зв'язок стабільний і не вистачає пам'яті для повного зберігання повної матриці, передача здійснюється потоково по пікселях (3.21), одному або декількох. У цьому випадку кожен піксель передається одразу після обробки шифрування. Для збереження одного пікселя достатньо виділити 8 біт або 1 байт оперативної пам'яті. Після передачі пікселя він одразу затирається, а наступний блок завантажується.

## Демонстрація результату шифрування зображення

Щоб продемонструвати ефективність розробленого алгоритму шифрування, проведено експеримент із використанням простого зображення.

*На рисунку* (3.1а)*:* оригінальне зображення представлено у відтінках сірого кольору. Це зображення демонструє початковий вигляд даних до шифрування.

*На рисунку* (3.1б)*:* результат шифрування із застосуванням лише перестановок рядків та стовпців . Для цього використовується запропонована перестановка $\sigma_1$ для рядків та перестановка $\sigma_2$ для стовпців (3.19). Цей підхід не змінюється вигляд початкового зображення. Це пов'язано з тим, що перестановки просто міняють місцями рядки або стовпці, не змінюючи самі пікселі.



**(а)** Оригінальне зображення

**(б)** Шифрування перестановками

**(в)** Шифрування перестановками та інверсіями

**Рисунок 3.1** – Приклади шифрування зображень

*На рисунку* (3.1в)*:* результат розробленого методу шифрування із застосуванням як перестановок $\sigma_1$ і $\sigma_2$ (3.19), так і інверсій рядків $R$ та стовпців $B$ (3.20). Цей варіант забезпечує додаткове ускладнення структури зображення, оскільки після інверсій пікселі змінюють порядок у межах рядків і стовпців, що ефективно порушує регулярність і періодичність. В результаті зашифроване зображення набуває хаотичного вигляду.

## Висновки до розділу 3

У третьому розділі було запропоновано декілька алгоритмів шифрування зображень на основі розглянутих малоресурсних алгоритмів, а також алгоритм шифрування з використанням перестановок та інверсії. Для кожного алгоритму проведено оцінку обсягу ключового простору, визначено необхідний обсяг пам'яті для зберігання ключа та зашифрованого зображення. Також було наведено приклад шифрування зображення з використанням лише перестановок та запропонованого алгоритму, що поєднує перестановки та інверсії.

# ВИСНОВКИ

При виконанні даної роботи було проведено огляд опублікованих джерел за тематикою використання алгоритмів для шифрування зображень у середовищах із сильно обмеженими ресурсами, зокрема в БПЛА. Аналіз літератури показав, що існує широкий спектр алгоритмів, які потенційно можуть бути застосовані в БПЛА. Однак, по-перше, не всі вони враховують специфіку подання та обробки зображень у таких системах, а по-друге, відсутня достатня інформація щодо вимог до цих алгоритмів стосовно їхньої практичної реалізації в БПЛА.

Проведено детальний аналіз малоресурсних алгоритмів шифрування, що дозволив виділити їхні переваги та недоліки з точки зору обчислювальної потужності та особливості передачі візуальної інформації через нестабільні канали зв'язку. Виявлено, що не всі алгоритми можуть ефективно застосовуватись у системах із обмеженнями. У зв'язку з цим, доцільним є розгляд різних підходів до шифрування: адаптованих модифікацій блокових шифрів, використання потокових шифрів, а також розробка нових алгоритмів спрямованих безпосередньо на шифрування зображення з урахуванням їх подання подання у БПЛА.

Запропоновано алгоритми шифрування зображення із застосуванням низки малоресурсних алгоритмів шифрування, які були додатково модифіковані для підвищення їхньої ефективності у специфічних умовах. Зокрема, для алгоритму PRESENT запропоновано використання зменшеної версії з 8 раундами у режимі лічильника. Такий підхід забезпечує стійкість до часткових викривлень, коли локальні пошкодження шифротексту призводять лише до локального спотворення відповідних фрагментів зображення, не впливаючи на інші його частини. Це критично важливо для збереження цілісності візуального сприйняття при передачі зображень у реальному часі.

Для підвищення стійкості шифрування до перешкод у каналі зв'язку запропоновано застосування потокового шифрування на основі аналога шифру Вернама з використанням криптографічно стійкого генератора псевдовипадкової гамми ENOCORO. При цьому генерація ключової гамми здійснюється на землі через високі обчислювальні вимоги, що зменшує навантаження на апаратні ресурси БПЛА.

Також окрім модифікованого PRESENT та потокового шифрування, запропоновано алгоритм на основі перестановок та інверсій рядків і стовпців зображення. Цей метод, простий за реалізацією, забезпечує ефективне перемішування пікселів та суттєво знижує можливість збереження візуальної структури зображення без знання ключа, що робить його придатним для використання в умовах перешкод зв'язку.

Всі три запропоновані алгоритми оцінено з точки зору обсягу ключового простору, необхідної пам'яті для зберігання ключа та зашифрованого зображення. Але через відсутність у відкритих джерелах достовірної інформації про точний обсяг оперативної пам'яті та обчислювальних ресурсів, доступної на борту безпілотного літального апарата, неможливо визначити точні вимоги того алгоритму який може там використовуватись. Проте проведений аналіз дозволив зробити висновок, що запропоновані алгоритми є перспективними для використання у БПЛА. Найбільш перспективним видаються алгоритми на основі перестановок і інверсій, оскільки вони забезпечуєть якісне шифрування з мінімальними обчислювальними затратами і високою стійкістю до втрат або пошкоджень даних.

Для подальшого розвитку теми доцільно зосередитися на отриманні більш точної інформації щодо апаратних характеристик безпілотних літальних апаратів, що дозволить оптимізувати малоресурсні алгоритми шифрування під реальні умови експлуатації.

# ПЕРЕЛІК ПОСИЛАНЬ


[1] Hazim Shakhatreh та ін. «Unmanned Aerial Vehicles (UAVs): A Survey on Civil Applications and Key Research Challenges». В: *IEEE Access* 7 (2019), 48572–48634. ISSN: 2169-3536. DOI: 10.1109/access.2019.2909530. URL: http://dx.doi.org/10.1109/access.2019.2909530.

[2] Thomas Eisenbarth та ін. «A Survey of Lightweight-Cryptography Implementations». В: *IEEE Design amp; Test of Computers* 24.6 (листоп. 2007), 522–533. ISSN: 0740-7475. DOI: 10.1109/mdt.2007.178. URL: http://dx.doi.org/10.1109/MDT.2007.178.

[3] Jia Hao Kong, Li-Minn Ang та Kah Phooi Seng. «A comprehensive survey of modern symmetric cryptographic solutions for resource constrained environments». В: *Journal of Network and Computer Applications* 49 (бер. 2015), 15–50. ISSN: 1084-8045. DOI: 10.1016/j.jnca.2014.09.006. URL: http://dx.doi.org/10.1016/j.jnca.2014.09.006.

[4] Jesús Soto-Cruz та ін. «A Survey of Efficient Lightweight Cryptography for Power-Constrained Microcontrollers». В: *Technologies* 13.1 (груд. 2024), с. 3. ISSN: 2227-7080. DOI: 10.3390/technologies13010003. URL: http://dx.doi.org/10.3390/technologies13010003.

[5] A. Bogdanov та ін. «PRESENT: An Ultra-Lightweight Block Cipher». В: *Cryptographic Hardware and Embedded Systems - CHES 2007*. Springer Berlin Heidelberg, 450–466. ISBN: 9783540747352. DOI: 10.1007/978-3-540-74735-2_31. URL: http://dx.doi.org/10.1007/978-3-540-74735-2_31.

[6] Subhadeep Banik та ін. *GIFT: A Small Present*. Cryptology ePrint Archive, Paper 2017/622. 2017. URL: https://eprint.iacr.org/2017/622.

[7] *ISO/IEC 29192-3:2012*. Edition 1, 2012. URL: https://www.iso.org/standard/56426.html.

[8] Friedrich L. Bauer. «Vernam Cipher». В: *Encyclopedia of Cryptography and Security*. Springer US, 647–647. ISBN: 9780387234731. DOI: 10.1007/0-387-23483-7_453. URL: http://dx.doi.org/10.1007/0-387-23483-7_453.

[9] Сергей Флейш. *Шифрування радіо зв'язку в ЗСУ*. URL: https://www.facebook.com/photo.php?fbid=5634878876580450&id=100001751811701&set=a.265677330167325.

# ДОДАТОК А ТЕКСТИ ПРОГРАМ

## А.1 Програма 1

Програмний код на мові Python для алгоритму шифрування та розшифрування зображень: з використанням лінійної перестановки, а також з використанням лінійної перестановки та інверсії.

```
def find_coprime_number(length):
    while True:
        u = random.randint(0, length-1)
        if gcd(u, length) == 1:
            return u

def generate_uv(length):
    u = find_coprime_number(length)
    v = random.randint(0, length-1)
    return u, v

def permutation(length, u, v):
    return [(u * i + v) % length for i in range(length)]

def generate_key(img_shape):
    n, m = img_shape
    u1, v1 = generate_uv(n)
    u2, v2 = generate_uv(m)
    flip_rows = [random.randint(0, 1) for _ in range(n)]
    flip_cols = [random.randint(0, 1) for _ in range(m)]
    return u1, v1, u2, v2, flip_rows, flip_cols


def encrypt_image_only_with_perm(img_array, u1, v1, u2, v2):
    n, m = img_array.shape
    row_perm = permutation(n, u1, v1)
    col_perm = permutation(m, u2, v2)

    permuted = img_array[row_perm, :]
    permuted = permuted[:, col_perm]

    return permuted


def encrypt_image_with_perm_inver(img_array, u1, v1, u2, v2, flip_rows, flip_cols):
    n, m = img_array.shape
```

```
    row_perm = permutation(n, u1, v1)
    col_perm = permutation(m, u2, v2)

    permuted = img_array[row_perm, :]
    permuted = permuted[:, col_perm]

    for i in range(n):
        if flip_rows[i] == 1:
            permuted[i, :] = permuted[i,::-1]

    for j in range(m):
        if flip_cols[j] == 1:
            permuted[:, j] = permuted[::-1, j]

    return permuted

def decrypt_image_only_with_perm(enc_array, u1, v1, u2, v2):
    n, m = enc_array.shape

    row_perm = permutation(n, u1, v1)
    col_perm = permutation(m, u2, v2)
    inv_row_perm = np.argsort(row_perm)
    inv_col_perm = np.argsort(col_perm)

    temp = enc_array[:, inv_col_perm]
    decrypted = temp[inv_row_perm, :]

    return decrypted

def decrypt_image_with_perm_inver(enc_array, u1, v1, u2, v2, flip_rows, flip_cols):
    n, m = enc_array.shape

    for j in range(m):
        if flip_cols[j] == 1:
            enc_array[:, j] = enc_array[::-1, j]

    for i in range(n):
        if flip_rows[i] == 1:
            enc_array[i, :] = enc_array[i, ::-1]

    row_perm = permutation(n, u1, v1)
    col_perm = permutation(m, u2, v2)
    inv_row_perm = np.argsort(row_perm)
    inv_col_perm = np.argsort(col_perm)

```

```
82      temp = enc_array[:, inv_col_perm]
83      decrypted = temp[inv_row_perm, :]
84
85      return decrypted
```