Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет *Інфокомунікацій*
(повна назва)
Кафедра *Інформаційно-мережної інженерії*
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Використання веб-контенту для створення схованого каналу обміну даними
(тема)

Виконав:
студент _2_ курсу, групи *ІМІм-21-2*
Будянський В.
(прізвище, ініціали)

Спеціальність *172. Телекомунікації та радіотехніка*
(код і повна назва спеціальності)
Тип програми *освітньо-наукова*
(освітньо-професійна або освітньо-наукова)
Освітня програма *Інформаційно-мережна інженерія*
(повна назва освітньої програми)

Керівник *ст.викл. Твердохліб В.В.*
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ *Безрук В.М.*
(підпис) (прізвище, ініціали)

2023 р.

Не містить відомостей, заборонених
до відкритого публікування

Керівник _____ /*В.В.Твердохліб*

Студент _____ / *В. Будянський*

Харківський національний університет радіоелектроніки

Факультет *Інфокомунікацій*

Кафедра *Інформаційно-мережної інженерії*

Рівень вищої освіти другий (магістерський)

Спеціальність *172. Телекомунікації та радіотехніка*
(код і повна назва)

Тип програми *Освітньо-наукова*
(освітньо-професійна або освітньо-наукова)

Освітня програма *Інформаційно-мережна інженерія*
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«17» *березня* 20 23 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові *Будянському Вадиму Сергійовичу.*
(прізвище, ім'я, по батькові)

1. Тема роботи *Використання веб-контенту для створення схованого каналу обміну даними*

затверджена наказом університету від 17 березня 2023 р. № 275 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 25 травня 2023 р.

3. Вихідні дані до роботи *Дослідити існуючі напрямки побудови стеганографічних алгоритмів та особливості класичних методів побудови прихованих каналів інформаційної взаємодії. Виявити їхні переваги та недоліки. З уразуванням виявлених недоліків запропонувати шляхи удосконалення одного з методів приховування даних. Дослідити стеганографічні алгоритми, орієнтовані на носії текстового типу.*

4. Перелік питань, що потрібно опрацювати в роботі *Вступ*

1 . Базова парадигма побудови стегоалгоритмів без прив'язки до певного типу даних

2. Класичні методи побудови прихованих каналів інформаційної взаємодії.

3. Дослідження можливості модифікації класичних методів побудови стегосистем для збільшення їх ефективності

4. Текстова стеганографія

Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) _____
слайди презентації в форматі Power Point (назва та мета роботи, ключові засади побудови стегоалгоритмів, класичні методи реалізації прихованих каналів, модифікація методу нерівномірних інтервалів, алгоритми стеганографії на базі текстових носіїв, висновки)

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Вступ	18.03-22.03	виконано
2	Базова парадигма побудови стегоалгоритмів без прив'язки до певного типу даних	23.03-30.03	виконано
3	Класичні методи побудови прихованих каналів інформаційної взаємодії	31.03-09.04	виконано
4	Дослідження можливості модифікації класичних методів побудови стегосистем для збільшення їх ефективності	10.04-23.04	виконано
5	Текстова стеганографія	24.04-06.05	виконано
6	Висновки	07.05-09.05	виконано
7	Оформлення пояснювальної записки	10.05 – 20.05	виконано

Дата видачі завдання *17 березня* 2023 р.

Студент _____
(підпис)

Керівник роботи _____ *ст. викл. Твердохліб В.В.*
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 67 с., 12 рис., 23 джерела, 2 додатки

СТЕГАНОГРАФІЯ, JPEG, ASCII, ДКП, НАЙМЕНШ ЗНАЧИМИЙ БІТ, СТЕГОКОНТЕЙНЕР, РОБАСТНІСТЬ

Об'єкт дослідження – методи побудови каналів прихованої інформаційної взаємодії на базі модифікованих стеганографічних алгоритмів.

Мета роботи – дослідження існуючих підходів до побудови стеганографічних алгоритмів та їх модифікація виходячи з виявлених недоліків.

Виконується огляд проблематики, пов’язаної з побудовою прихованих каналів на базі стеганографічних алгоритмів. Досліджуються переваги та недоліки стандартизованих методів. Розкривається сутність підходів щодо удосконалення існуючих стегнографічних алгоритмів, виходячи з попередньо виявлених недоліків алгоритмів класичної групи. Розглядаються алгоритми прихованого передавання даних, що у якості носіїв розглядають текстову інформацію.

THE ABSTRACT

Explanatory note: 67 p., 12 fig., 23 sources, 2 apps.

STEGANOGRAPHY, JPEG, ASCII, DCT, LESS SIGNIFICANT BIT, STEGOCAINTER, ROBUSTNESS

The object of research is the methods of building channels of hidden information interaction based on modified steganographic algorithms.

The purpose of the work is to study the existing approaches to the construction of steganographic algorithms and their modification based on the identified shortcomings.

An overview of the issues related to the construction of hidden channels based on steganographic algorithms is performed. Advantages and disadvantages of standardized methods are investigated. The essence of the approaches to the improvement of the existing steganographic algorithms is revealed, based on the previously identified shortcomings of the algorithms of the classical group. Algorithms of covert data transmission, which consider textual information as carriers, are considered.

ЗМІСТ

С.

ПЕРЕЛІК СКОРОЧЕНЬ

SSH – (Secure Shell) — протокол захищеного звязку типу «point-to-point»;
IPSEC – (IP Security) — протокол захищеного зв’язку;
VoIP – (voice over IP) – спосіб передавання голосового трафіку через IP-мережу;
AoD – (Video on Demand) — сервіси «аудіо на запит»;
H.264/AVC — технологія високоефективного кодування відео;
H.265/HEVC — технологія високоефективного кодування відео;
BMP — графічний формат, який передбачає зберігання даних без попереднього ущільнення;
JPEG — графічний формат, який базується на дискретному косинусному перетворенні вихідної інформації та передбачає можливість внесення контрольованої похибки;
LSB — (least significant bit) — найменш значущий біт;
MIME — (Multipurpose Internet Mail Extensions) — стандарт, що описує передачу різних типів даних електронною поштою;
UUENCODE — метод представлення двійкових даних у текстовій формі, придатній для передачі засобами, призначеними лише для передачі текстів;
UTF-8 — стандарт кодування символів, що дозволяє більш компактно зберігати та передавать символи;
ASCII — стандарт кодування символів;
WYSIWYG — (What You See Is What You Get) — властивість прикладних програм або веб-інтерфейсів, в яких зміст відображається в процесі редагування і виглядає максимально близьким до кінцевої продукції, яка може бути друкованим документом, веб-сторінкою або презентацією.

ВСТУП

Глобальне інформаційне середовище, починаючи з 2008-2010 років, характеризується високим рівнем агресивності по відношенню потенційно до усіх учасників інформаційного обміну, як то:

- рядові користувачі мережевих сервісів;
- власники мережевих сервісів та афільовані з ними особи;
- хостери та інтернет-провайдери.

Така ситуація пояснюється впливом ряду ключових чинників, серед яких наступні [1, 2]:

1. Постійне зростання та структуризація інформаційного середовища у цілому.
2. Збільшення відсотку сервісів, що являють собою потенційні об’єкти інтересу зловмисників (платіжні та фінансові системи, сервіси, інформаційні системи підприємств, що можуть містити інформацію, яка може мати фінансову конвертованість тощо);
3. Розширення можливостей зловмисника, зокрема, за рахунок:

- розвитку хмарних сервісів, на базі яких може бути оперативно реалізовано процедури обробки великих масивів даних (наприклад, для атак brutforce, зламу шифрів та ін.);
- уніфікації алгоритмів шифрування даних найвищої критичності, на тлі застосування криптографічних схем обмеженої стійкості, та зростання обчислювальних можливостей зловмисників і виявлених недоліків існуючих шифросистем.

У результаті цього потенційно вразливим до зловмисного впливу є як кожен кінцевий/мережевий вузол, так і дані, що надсилаються загальним розподіленим середовищем.

У свою чергу, захист даних на рівні локального та/або мережевого вузла може забезпечуватися за участю [3]:

- мережевих екранів;
- IPS-систем;
- засобів TDS;
- шифрування ключових сегментів інформації.

Водночас, у ході передавання пакетів даних існує ризик того, що зловмисник отримає до них доступ, використовуючи для цього:

- сніфінг;
- імітацію вузла-приймача;
- попереднє інфікування мережевого вузла/вузлів.

За таких умов існує ймовірність того, що треті особи можуть отримати у своє розпорядження критичну інформацію одного чи кількох наступних типів:

- комерційна інформація;
- особисті дані;
- дані з грифом «ДСК», «таємно» чи «цілком таємно» тощо.

Тобто, у процесі трансляції загальним розподіленим середовищем дані є суттєво більш вразливими, так як:

- зловмисник має у розпорядженні значний обчислювальний потенціал;
- ряд існуючихї шифросистем мають недоліки, що відомо широкій аудиторії;
- протоколи безпеки (SSH, IPSEC тощо) також мають вразливі місця;
- зловмиснику відомий об’єкт атаки.

У зазначених умовах суттєво більш доцільним для створення умов безпечного передавання інформації загальним розподіленим середовищем є використання прихованих каналів на базі стеганографічних алгоритмів. Це є наслідком того, що:

- стегоканал може бути реалізовано на базі будь-кого з великої кількості алгоритмів маскування (або власного), відстежити які досить складно;
- об’єкт атаки для зловмисника є невідомим.

Таким чином, усі питання, що стосуються розробки, удосконалення та впровадження стеганографічних алгоритмів, на сьогодні є актуальними.

1. БАЗОВА ПАРАДИГМА ПОБУДОВИ СТЕГОАЛГОРИТМІВ БЕЗ ПРИВ'ЯЗКИ ДО ПЕВНОГО ТИПУ ДАНИХ

1.1 Сервіси та канали зв'язку, для яких є актуальним використання прихованих каналів

Беручи до уваги специфіку функціонування сьогоднішніх інформаційно-комунікаційних мереж, можемо зазначити, що у цілому наявність каналів прихованого обміну даними є актуальною, фактично, для будь-яких сервісів та їх користувачів.

Наприклад (рис.1.1) [4, 5]:

- системи електронного банкінгу (для обміну найбільш критичними даними або для повної реалізації транзакцій);
- месенджери – як такі, що використовуються для приватних комунікацій, так і корпоративні;
- системи корпоративної взаємодії (системи спільної роботи над проектами – наприклад, Jira та подібні;
- засоби підтримки віддаленої взаємодії типу «філія-головний офіс» або «віддалений користувач-офіс»; це, у свою чергу, покликано збільшити стійкість захищеного обміну даними, що реалізується на базі VPN чи SSH-каналів;
- у принципі, будь-який сервіс чи канал комунікації, що теоретично може бути об'єктом атаки.

При цьому, додатки та сервіси, зазначені вище, можуть функціонувати у таких режимах, як:

- повне маскування (коли 100% трафіку надсилається/приймається прихованим каналом);
- часткове приховування (частина трафіку транслюється у відкритому вигляді, частина – маскованими каналами); при цьому, рішення щодо маркування даних для передавання прихованим каналом приймає користувач.

Режим часткового приховування є перспективним підходом до побудови безпечного обміну даними загальним середовищем. Утім, на зараз йому властиві деякі недоліки, зокрема такі як:

- ручне маркування сесій користувачем вимагає від нього адекватної оцінки важливості сеансу зв’язку;

- необхідність маркування сесії створює для користувача ряд незручностей;

- побудова алгоритму автоматичного маркування сесій відноситься до класу нетривіальних завдань, що на сьогодні не мають загального рішення.

Таким чином, питання автоматичного маркування, відноситься до одного з пріоритетних, що потребують вирішення у майбутньому.

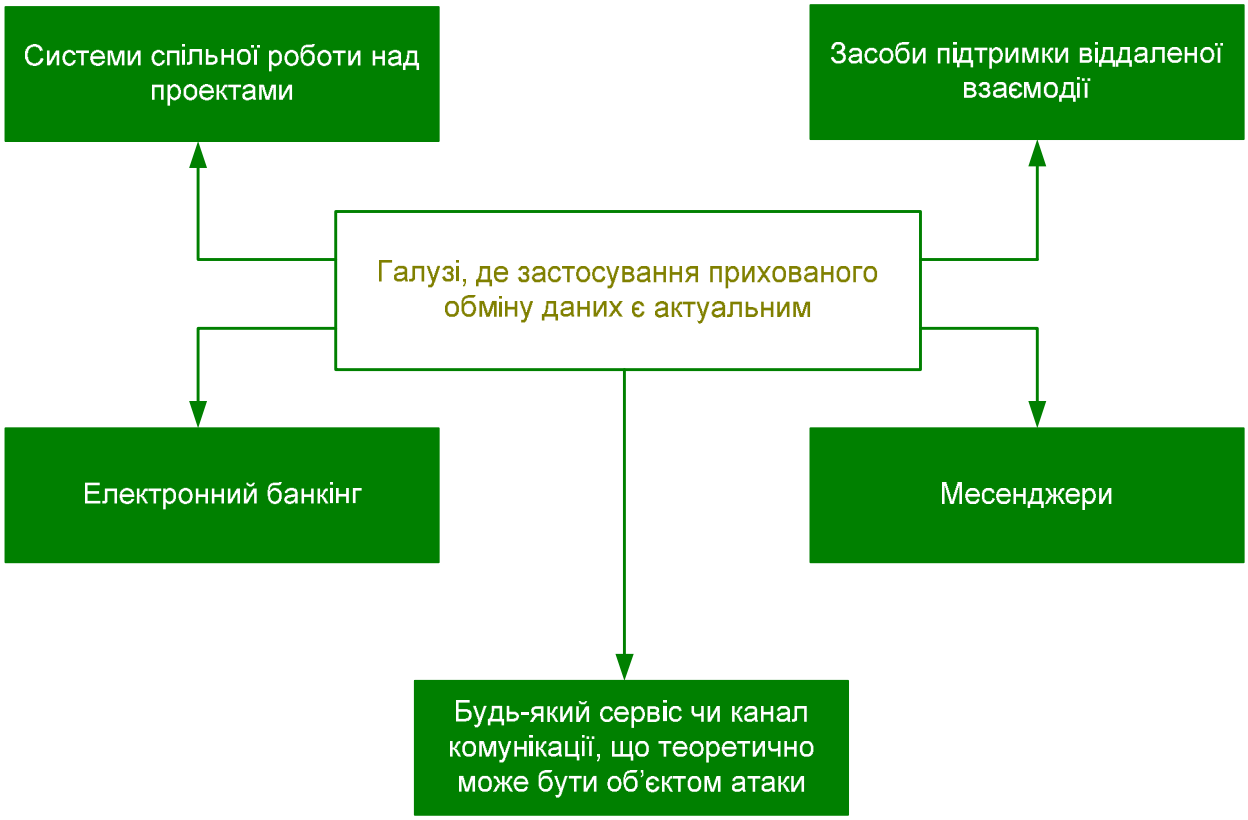

Рисунок 1.1 – Перелік сервісів та додатків, для яких актуальна наявність прихованих каналів обміну даними

1.2 Архітектура каналу прихованого передавання даних

Незважаючи на особливості реалізації, загальна схема каналу прихованого передавання даних буде такою, як зображено рис. 1.2. Це стосується як її ключових складників, так і зв’язків між ними [6].

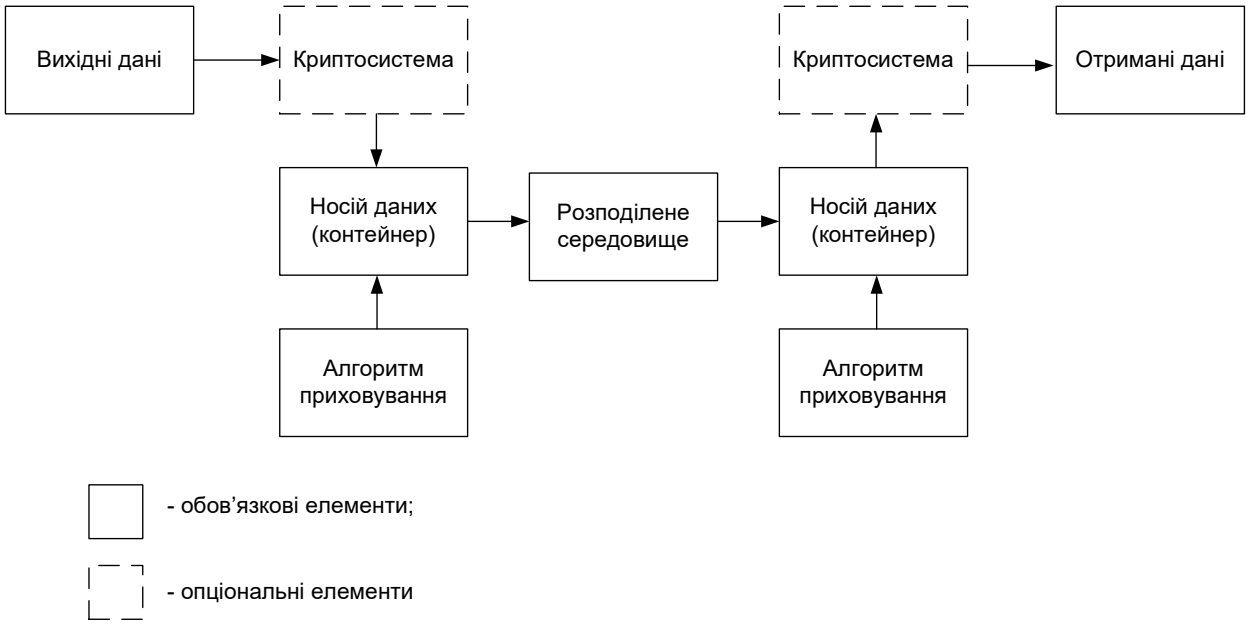

Рисунок 1.2 – Базова структура каналу прихованого передавання даних

На рисунку 1.2 зображена базова каналу прихованого передавання даних.

Тут Джерело даних генерує вихідну інформацію, яка далі попередньо шифрується з застосуванням тих чи інших криптосистем. Цей технологічний етап перетворення є опціональним.

Далі, шифрована (чи відкрита) інформація вбудовується у *носій даних,* або *контейнер.* Принцип такого вбудовування визначається застосовуваним *алгоритмом приховування,* що визначає порядок запису біт секретного повідомлення у носій.

Після того, як біти секретного повідомлення, чи його частини, вбудовані у носій, файл-носій надсилається загальним розподіленим середовищем.

У свою чергу, на прийомному боці виконується зворотній каскад перетворень, а саме:

- вилучаються дані з носія, беручи до уваги алгоритм вбудовування та його оції;

- відновлюється вихідний сегмент даних з криптограми (якщо вана мала місце);

- дані приводяться до вигляду, що можуть сприйматися відповідним додатком (перетворення у рамках моделі OSI).

У зазначеній схемі криптосистема являє собою додатковий механізм захисту, що покликаний утруднити інтерпретації даних зловмисником на випадок, якщо прихований канал ним було викрито.

1.3 Відмінності шифрованого та прихованого каналів

Уявімо, що у нашому розпорядженні є криптований канал, тобто, такий, що передбачає надсилання даних мережею у шифрованому вигляді (рис.1.3) [7].

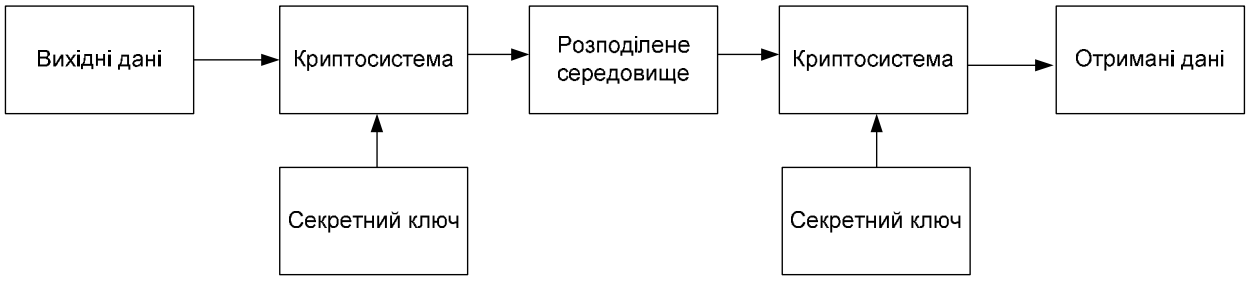

Рисунок 1.3 – Канал передавання даних на базі криптосистеми

У випадку, зображеному рис. 1.3, до мережі надсилатиметься шифрований потік даних (чи його частина).

Безумовно, застосування шифрів суттєво утруднює (а у ряді випадків - унеможливлює) доступ зловмисників до даних, що надсилаються. Разом з тим, порівнюючи схеми, зображені рис.1.2 та 1.3, можемо зробити висновки про наступне:

- переважна більшість шифросистем не гарантує захищеність даних (на сьогодні абсолютно стійкою є лише шифросистема на базі алгоритму «одноразових блокнотів»), тобто криптографічні алгоритми мають обмежену стійкість а відтак – можуть бути зламаними;

- у випадку застосування криптоканалів, зловмисник початково інформований про об’єкт атаки.

При цьому, на відміну від випадку криптоканалу, для стегоканалу певний об’єкт атаки зловмиснику апріорі не може бути відомим.

За рахунок цього забезпечується [8]:

- потенційна можливість передавання даних прихованим каналом з мінімальною ймовірністю викриття, так як факт його існування зловмиснику у загальному випадку не є відомим;

- ефективний захист даних, що надсилаються/приймаються прихованим каналом навіть в умовах, що зловмисник знає про сам факт його існування.

Останній фактор пояснюється тим, що:

- для викриття прихованого каналу зловмисник повинен сканувати увесь масив даних, що генерується тим чи іншим джерелом на постійній основі;

- зловмиснику, навіть за умови, що він може сканувати увесь масив вхідних/вихідних даних джерела, невідомо, який саме алгоритм/алгоритми та на базі яких саме носіїв використовується для створення прихованого каналу; це, у свою чергу, мінімізує ймовірність самого факту викриття прихованої передачі інформації.

Зазначене, безумовно, свідчить про більш високу ефективність використання прихованих каналів відносно каналів, утворених на базі криптосистем.

1.4 Принцип внесення даних приховуваного повідомлення до носія (контейнеру)

Нехай повідомлення M, яке має передаватися прихованим каналом, являє собою масив з $m = \overline{1,\ M}$ біт b_m.

У свою чергу, носій розглядається як масив $\psi = \overline{1,\ \Psi}$ біт b_ψ відповідно.

Якщо при цьому не брати до уваги співвідношення M та Ψ, спрощено процедуру вбудовування (інкапсуляції) біт b_m повідомлення у біти b_ψ носія можна зобразити у наступному вигляді [7, 9]:

$$\begin{cases} b_\psi := b_\psi \mid b_m = b_\psi; \\ b_\psi := b_m \mid b_m \neq b_\psi. \end{cases} \qquad (1.1)$$

З виразу (1.1) бачимо, що у випадку співпадання значень b_m та b_ψ, величина b_ψ ψ-го біту носія не змінюється, і навпаки.

При цьому, принцип, конкретний механізм та параметри інкапсуляції біт b_m повідомлення у біти b_ψ носія визначаються:

- характером та типом файлів, що розглядаються, як носії;

- базовим механізмом інкапсуляції на базі того чи іншого стеганографічного алгоритму;

- застосованим стегоключем.

1.5 Узагальнені вимоги до каналу обміну прихованими даними (стегоканалу)

Узагальнений принцип інкапсулювання біт повідомлення, яке необхідно приховано надіслати приймачеві за умови побітової заміни бінарних елементів b_ψ елементами b_m самого повідомлення, проілюстровано виразом (1.1).

При цьому, реалізація прихованого каналу обміну даними передбачає, що процес, поданий формулою (1.1) утілюється на базі того чи іншого стеганографічного алгоритму.

У даному разі для того, щоб стеганографічний канал вважався ефективним, він має відповідати вимогам стосовно [7-9]:

- рівня заповненості носія;

- безпосередньо самого алгоритму;

- особливостей носія.

Розглянемо далі означені вимоги більш детально.

1.5.1 Вимога щодо рівня заповненості носія прихованих даних

Будь-яка прихована передача інформації загальним розподіленим середовищем апріорі не може бути ефективною, якщо не забезпечується співвідношення об’єму V носія до об’єму V_{fill} заповненого носія у відповідності до наступного виразу:

$$\frac{V_{fill}}{V} \to 0. \qquad (1.2)$$

Тобто, з виразу (1.2) бачимо, що першочергово необхідно, щоб рівень заповненості носія був незначним на тлі його загального потенційно можливого об’єму.

Ця вимога пояснюється тим, що для викриття факту існування прихованого каналу на рівні одного носія можуть застосовуватися [10-13]:

- універсальні алгоритми виявлення ознак інкапсуляцій (що не орієнтуються на певний стегоалгоритм), базовані на зборі статистичних даних у межах файла, що теоретично може містити приховано інформацію; такі алгоритми є гарантовано ефективними тоді, коли носій заповнено не менш, ніж на 60% від його потенційно можливого об'єму;

- алгоритми, орієнтовані на викриття прихованого каналу, побудованого на базі того чи іншого конкретного алгоритму; за умови низької наповненості носія не гарантується виявлення характерних ознак інкапсуляції.

1.5.2 Вимога щодо рівня надмірності носія

Дана вимога також є однією з ключових для стегоканалу. У даному випадку рівень Y надмірності самого носія, має відповідати вимозі, як показано виразом:

$$Y \to \max, \tag{1.3}$$

а отже, рівень надмірності носія має бути максимально можливим.

Зазначена вимога є наслідком вимоги (1.2) та пояснюється тим, що для можливості передавання в одиницю часу якомога більшого обсягу приховуваних даних, контейнер повинен мати якомога вищий рівень надлишковості, тоді збільшення параметру V_{fill} у виразі (1.2) не порушить справедливість самого співвідношення.

1.5.3 Вимога відносно часу запису та зчитування приховуваних даних у контейнер

У будь-якому випадку, а, перш за все, коли мова йде про інтерактивний обмін даними, час інкапсуляції та час декапсуляції, повинен відповідати наступним вимогам:

$$t_{inc}, t_{dec} \to \min, \tag{1.4}$$

де t_{inc} та t_{dec} - час вбудовування та час виокремлення вбудованих даних у носій відповідно.

Забезпечення виконання вимоги (1.4) може досягатися як за рахунок самого стегоалгоритму, так і шляхом вибору відповідних носіїв.

1.5.4 Вимога відносно рівня розповсюдженості даних, що відповідає носієві

Для того, щоб носій не привертав уваги з боку систем стегоаналізу, необхідно, щоб значний показник Θ поширеності мали [8, 14]:

- тип даних, до яких належить носій;
- тип файлів, що відповідає носієві.

Це зумовлено тим, що трансляція пакетів даних, які не мають широкого розповсюдження у мережі та на локальних носіях як на рівні типу даних, так і на рівні типу файлів, сприймається як своєрідна аномалія, що незворотньо привертає до них увагу.

1.5.5 Вимога відносно робастності носія

Робастність R являє собою властивість носія зберігати цілісність у наслідок виконання процедури інкапсулювання приховуваних біт.

Очевидно, що ніякий стегоалгоритм не може вважатися ефективним, якщо вбудовування приховуваної інформації у носій того чи іншого типу веде до його часткового чи повного руйнування, що проявляється у [6, 9]:

- помітних викривленнях;
- падінні функціональності.

На сьогодні максимальний рівень робастності характерний даним відео та графічних типів.

Дещо поступаються за рівнем робастності аудіо файли.

Водночас, серед текстових файлів робастністю характеризуються фали htm (html), txt, css та, у принципі, будь-які неструктуровані текстові файли.

При цьому, якщо розглядати їх, як носії на рівні контенту, то робастними є також ряд структурованих текстових файлів (наприклад, doc).

У свою чергу, бінарні файли не є робастними.

1.6 Ключова характеристика будь-якого стегоалгоритму

З урахуванням усього зазначеного, одну з ключових характеристик, що свідчать про ефективність того чи іншого стегоалгоритму, а саме –

ймовірність P_{dm} викриття прихованого каналу, можемо подати у наступному вигляді [11]:

$$P_{dm} = f(Y; \frac{V_{fill}}{V}; \Theta), \quad (1.5)$$

тобто, даний показник є, у сутності, функціоналом від рівня Y надмірності носія, співвідношення між величинами V та V_{fill}, а також поширеністю R типу даних, які використовуються у ролі носія, мережею.

У свою чергу, з виразу (1.5) можемо сформулювати ознаку ефективності стегоалгоритма.

Отже, стегоалгоритм є ефективним, якщо, у загальному випадку, виконується наступна вимога:

$$P_{dm} \to 0, \quad (1.6)$$

тобто, ймовірність викриття стегоканалу теоретично виключається.

Водночас, умова (1.6) формально виконуватиметься, якщо буде забезпечено виконання умов:

$$P_{dm} \to 0 \,|\, (Y \to \max) \& \left(\frac{V_{fill}}{V} \to 0 \right) \& (\Theta \to \max) \& (R \to \max) \quad (1.7)$$

1.7 Вибір типу носія для побудови прихованого каналу обміну даними загальним розподіленим середовищем

Для того, щоб виявити, які типи даних (файлів) є найбільш прийнятними у ролі носіїв, розглянемо ряд найбільш поширених типів трафіку, що розповсюджуються мережею.

Спочатку розглянемо носії з точки зору відповідності критерію розповсюдженості.

Так, якщо розглянути статистику Cisco VNI Forecast останні 5 років, бачимо, що максимальна частка мережевого трафіку належить відео [15].

Тут слід відмітити, що зараз найбільш широко є розповсюдженими файли Mp4.

На тлі цього, приблизно, менше у 8-10 разів поширення при цьому має аудіоконтент, що найчастіше представлений звуковими потоками Mp3 у складі VoIP, AoD, інтернет-радіо тощо.

При цьому якщо розглядати дані статичного типу, то серед них найбільше поширення на сьогодні мають:

- графічні файли;

- текстові файли.

Водночас, серед графічних файлів за рівнем розповсюдженості на сьогодні є лідерами:

- jpeg, як найбільш поширений тип файлів, що містить растрову інформацію;

- png та gif, що містять або початково растрову інформацію, або дані, що мали векторне походження та надалі були конвертовані до растрового вигляду.

У свою чергу, з файлів текстового типу найширше поширеними є:

- html (htm) – файли, що завантажуються з веб-серверів та містять гіпертекстову розмітку;

- файли css, що є супутніми html (htm) файлам та виконують службові функції відносно до них;

- файли doc, що є базовим форматом офісних текстових структурованих документів;

- файли txt, що містять у собі неструктуровані текстові дані; на відміну від doc, файли цього типу містять лише текстову інформацію та не мають внутрішньої сегментації.

Це еквівалентно виразу:

$$\Theta_{vid} > (\Theta_{grp} \,\&\, \Theta_{txt}) > \Theta_{aud}, \qquad (1.8)$$

де Θ_{vid} - рівень розповсюдженості відеофайлів;

Θ_{grp} - розповсюдженість графічних файлів;

Θ_{aud} - розповсюдженість звукових файлів;

Θ_{txt} - ступінь розповсюдженості у мережі файлів текстового типу.

Разом з тим, якщо розглядати носії з позиції відповідності критерію 2, тоді отримуємо:

- надмірність відео є найбільшою з усіх інших типів даних;

- друге місце за рівнем надмірності займають растрові графічні файли, у першу чергу jpeg;

- надмірність аудіо даних є суттєво меншою, ніж графічних файлів, проте суттєво перевищує рівень надмірності, властивий текстовій інформації.

Вищезазначене може бути проілюстроване наступною нерівністю:

$$Y_{vid} > Y_{grp} > Y_{aud} > Y_{txt}, \quad (1.9)$$

де Y_{vid} - надмірність файлів відео;

Y_{grp} - ступінь надмірності файлів графічного типу;

Y_{aud} - показник надмірності аудіо файлів;

Y_{txt} - надмірність у мережі текстових файлів.

У підсумку можемо зазначити, що:

- ефективний стеганоалгоритм має працювати з носіями, що мають високу надмірність;

- носій має бути якомога більше розповсюдженим у мережі;

- заповненість контейнеру на тлі загального теоретично можливого обсягу має бути незначною.

1.8 Деталізація вимог до носія прихованої інформації для забезпечення ефективності стегоканалу

Створення умов для виконання вимоги (1.6), у загальному випадку, досягається за рахунок [6, 16]:

- реалізації продуктивного алгоритмічного та математичного апарату самого механізму приховування;

- вибору та використання носіїв, які відповідають усім вищезазначеним вимогам.

Якщо брати до уваги попередньо сформовані вимоги до контейнерів, то найбільш прийнятними для вбудовування даних у загальному випадку можемо вважати:

- файли відео mp4; сюди відносяться відео, кодовані у базисі як безпосередньо Mpeg-4, так і на основі рекомендацій H.264/AVC та/або H.265/HEVC;

- файли відео avi та mkv (тут використовуються кодеки, зазначені раніше);

- графічні файли jpeg;

- аудіо mp3.

2. КЛАСИЧНІ МЕТОДИ ПОБУДОВИ ПРИХОВАНИХ КАНАЛІВ ІНФОРМАЦІЙНОЇ ВЗАЄМОДІЇ

На сьогодні серед усіх існуючих напрямків до реалізації прихованої інформаційної взаємодії найбільш широкого застосування набули:

- напрямки, орієнтовані на графічні носії;
- напрямки, орієнтовані на контейнери аудіо.

Виконаємо оцінку ефективності кожного з напрямків окремо.

2.1 Аналіз методів побудови прихованих каналів, орієнтованих на використання носіїв графічного типу

Розглянемо принципи функціонування таких поширених стеганографічних методів, як:

- метод «останнього біту»;
- метод нерівномірних відрізків.

При цьому, у загальному випадку реалізація даних методів можлива для випадків BMP та JPEG –носіїв.

2.1.1 Принцип функціонування стеганографічного методу «останнього біту» на базі BMP-носія

Для випадку контейнеру BMP стегосистема, що базується на використанні класичного методу «останнього біту» є найпростішою у реалізації серед більшості стегосистем, що використовують графічні носії.

При цьому, метод «останнього біту», або LSB (less significant bit – найменш значущий біт) є прикладом побітового вбудовування секретного повідомлення у контейнер [6, 8, 17].

У рамках даного методу для реалізації інкапсулювання біт b_m, що належать приховуваному повідомленню, у графічний контейнер, передбачається виконання безпосередньої модифікації пікселів $p_{(\alpha,\beta)}$ вихідного зображення-носія.

У загальному випадку така модифікація реалізується відповідно принципу, зазначеного виразом (1.1).

Проте, на цей випадок вираз (1.1) доповнюється до вигляду:

$$\begin{cases} b_{\psi}^{(0)} := b_{\psi}^{(0)} \mid b_m = b_{\psi}^{(0)}; \\ b_{\psi}^{(0)} := b_m \mid b_m \neq b_{\psi}^{(0)}, \end{cases} \tag{2.1}$$

де $b_{\psi}^{(0)}$ - наймолодший біт (LSB) пікселя $p_{(m,n)}$.

Разом з тим, слід урахувати те, що зображення являє собою двовимірний масив $M \times N$ пікселів $p_{(m,n)}$, де $m \in [1, M]$, $n \in [1, N]$.

При цьому, слід брати до уваги той факт, що BMP являє собою формат, де паралельно існують 3 рівнозначних колірних канали. Інакше кажучи, у рамках BMP незалежно існують канали R (червоний), G (зелений) та B (синій), кожен з яких рівноймовірно може бути використано для інкапсуляції приховуваних даних. Відтак, необхідно внести відповідні корективи у вираз (2.1), тобто:

$$\begin{cases} b_{m,n}^{(0,clr)} := b_{\alpha,\beta}^{(0)} \mid b_m = b_{m,n}^{(0,clr)}; \\ b_{m,n}^{(0,clr)} := b_m \mid b_m \neq b_{m,n}^{(0,clr)}, \end{cases} \tag{2.2}$$

де $b_{m,n}^{(0,clr)}$ - біт нульового розряду пікселя $p_{(m,n)}$ на позиції (m,n) у колірному каналі clr.

Зазначену специфіку формату розглянемо для випадку реалізації алгоритму LSB на базі файлів даного типу.

Структуру пікселя у межах формату BMP зображено рис. 2.1.

У загальному випадку BMP дозволяє вбудувати до 3 біт (1 біт на колірний канал) секретного повідомлення лише на рівні одного пікселя $p_{(m,n)}$ зображення-носія. Це пояснюється тим, що теоретично на рівні повноколірного BMP молодший біт $b_{m,n}^{(0,clr)}$ може підлягати модифікації:

- у червоному каналі;
- у зеленому каналі;
- у синьому каналі.

Тобто, за певних умов може виконуватися співвідношення:

$$V_{fill}(p_{(m,n)}) \to 3 \tag{2.3}$$

У той же час, якщо носієм виступає зображення у відтінках сірого, тоді у межах одного пікселя може бути модифіковано один LSB-біт $b_{m,n}^{(0,clr)}$.

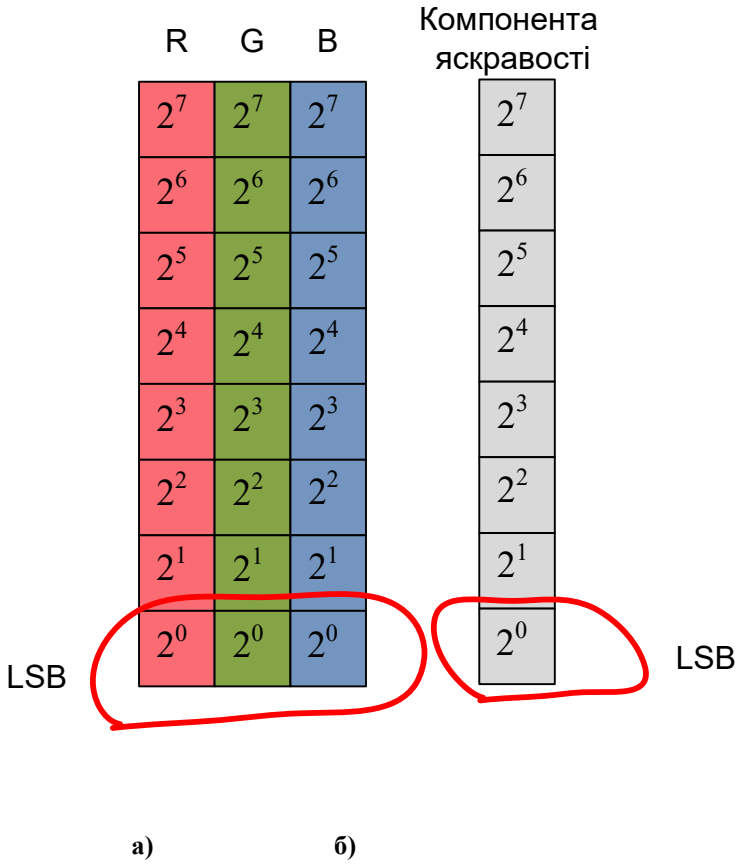

Рисунок 2.1 - Бітове представлення пікселя у палітрі RGB а) та у відтінках сірого б)

При цьому, також теоретично модифікації у межах того чи іншого пікселя $p_{(m,n)}$ може підлягати додатково біт $b_{\psi}^{(1)}$, тобто, такий, що належить 1-му розряду двійкового опису пікселя, (LSB+1).

Водночас, залежно від особливостей змісту носія, модифікація біту $b_{\psi}^{(0)}$ 1-го розряду пікселя $p_{(m,n)}$ може спричинювати появу візуально помітних викривлень носія, тому є недоцільною.

У рамках класичного методу LSB для випадку BMP-носія роздільної здатності $A \times B$ його реалізація виконується за такі кроки:

1. Встановлення стартової позиції $p(start)_{(m,n)}$, $m \in [1, M]$, $n \in [1, N]$, починаючи з якої буде виконуватися обхід LSB-простору того чи іншого колірного каналу.

2. Встановлення порядку обходу носія (зазвичай це або за напрямком рядків, або за стовпцями).

3. Конвертація кожного пікселя $p_{(m,n)}$ зображення у двійковий формат представлення, як показує наступний вираз:

$$p_{(\alpha,\beta)} = \left\langle 2^{(7,R)}_{(m,n)};..\ 2(R)^{(\nu,R)}_{(m,n)};..\ 2(R)^{(0,R)}_{(m,n)};\ 2(G)^{(7,G)}_{(m,n)};..\ 2(G)^{(\nu,G)}_{(m,n)};..\ 2(G)^{(0,G)}_{(m,n)};\ 2(B)^{(7,B)}_{(m,n)};..\ 2(B)^{(\nu,B)}_{(m,n)};..\ 2(B)^{(0,B)}_{(m,n)} \right\rangle, \quad (2.4)$$

$$\nu = \overline{1;7}$$

де R, G та B - ν-і біти, що належать червоному, зеленому та синьому каналам відповідно.

4. Побудова LSB-каналу, для чого виконується поєднання сукупності останніх біт з простору R, G або B, як це показує наступний вираз:

$$\Omega^{(0,\,clr)} = \bigcup_{m=1}^{M}\bigcup_{n=1}^{N} b^{(0,clr)}_{m,n}, \quad (2.5)$$

де $\Omega^{(0,\,clr)}$ - множина останніх біт (бітова площино нульового порядку) одного з колірних каналів - R, G чи B.

Подібним чином, за потреби, формується площина $\Omega^{(1,\,clr)}$ першого порядку на ін.

5. Модифікація біт $b^{(0,clr)}_{m,n}$ згідно з принципом, зазначеним виразом (2.1).

У цілому, процедура виконання LSB-модифікації у межах класичної реалізації методу, ураховуючи перелічені вище технологічні кроки, додатково вимагатиме лише вибору колірного каналу для розміщення біт приховуваного повідомлення.

Даний алгоритм для випадку, коли виконується інкапсуляція з координати (1,1), може бути подано у вигляді, який ілюструється рисунком 2.2.

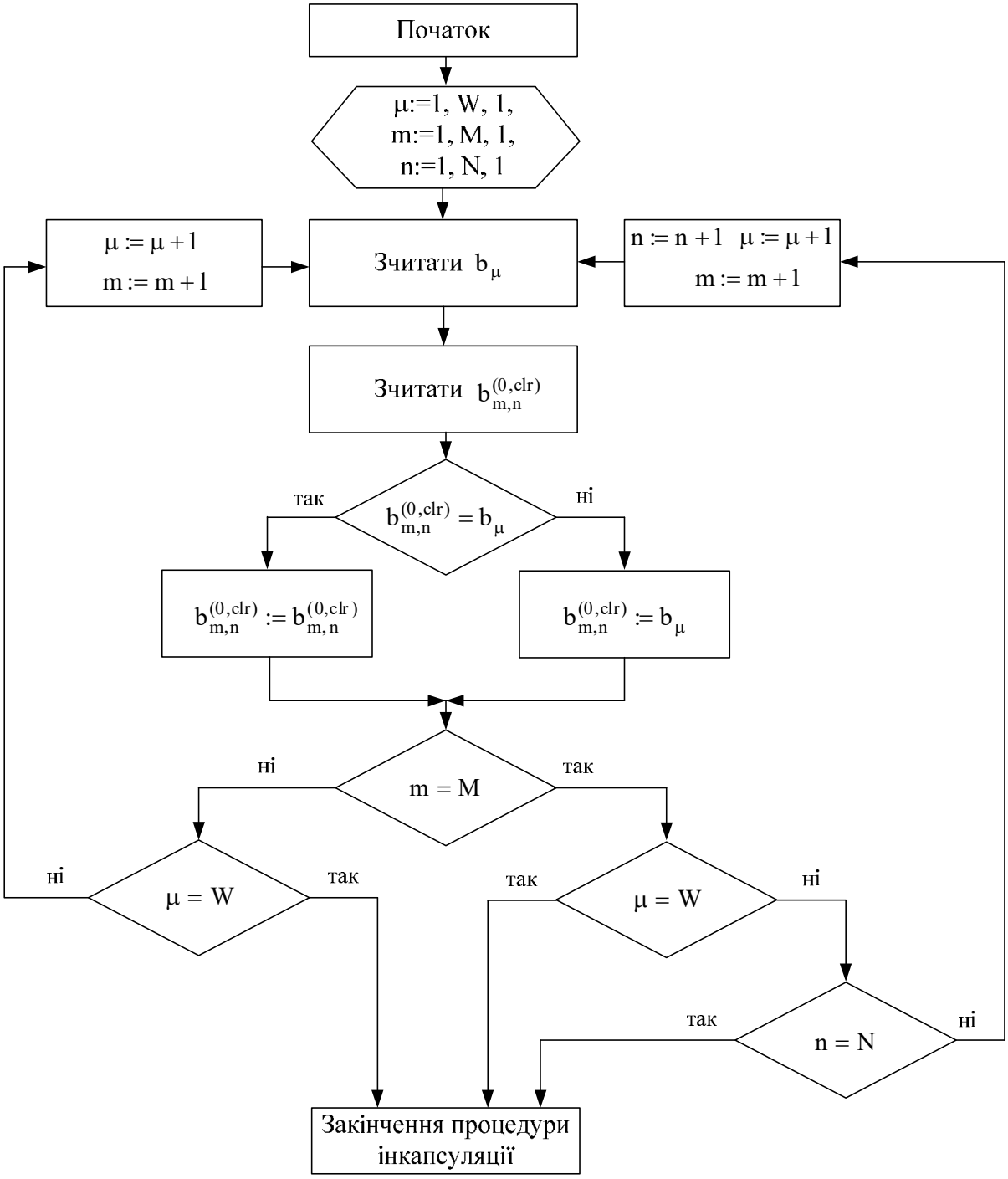

Рисунок 2.2 – Схематичне зображення алгоритму вбудовування даних секретного повідомлення у носій за класичним LSB-принципом

Згідно з алгоритмом, зображеним на рис. 2.2, у процесі інкапсуляції виконується послідовне зчитування кожного біта b_μ повідомлення та біта $b_{m,n}^{(0,clr)}$ носія відповідно до вказаного напрямку його обходу (за рядками/стовпцями).

Величини b_μ та $b_{m,n}^{(0,clr)}$ зчитуються та порівнюються між собою з наступною модифікацією значення $b_{m,n}^{(0,clr)}$ відповідно до виразу (2.2).

При цьому, якщо рядок/стовпець оброблено повністю тобто, виконується умова:

$$(m = M) \vee (n = N), \tag{2.6}$$

далі виконується перехід на наступну позицію у новому рядку/стовпецю, де, у свою чергу, зчитуються значення наступних біт b_μ та $b_{m,n}^{(0,clr)}$.

Алгоритм завершується у випадках, коли:

1. Задіяно усі біти носія у просторі $\Omega^{(0,clr)}$, а за умовами роботи використання простору $\Omega^{(1,clr)}$, чи перехід у площину $\Omega^{(0,clr+1)}$ або $\Omega^{(0,clr+2)}$ не передбачено.

2. Вбудовано усі W біт b_μ секретного повідомлення.

Розглянемо приклад модифікованого пікселя (рис. 2.3).

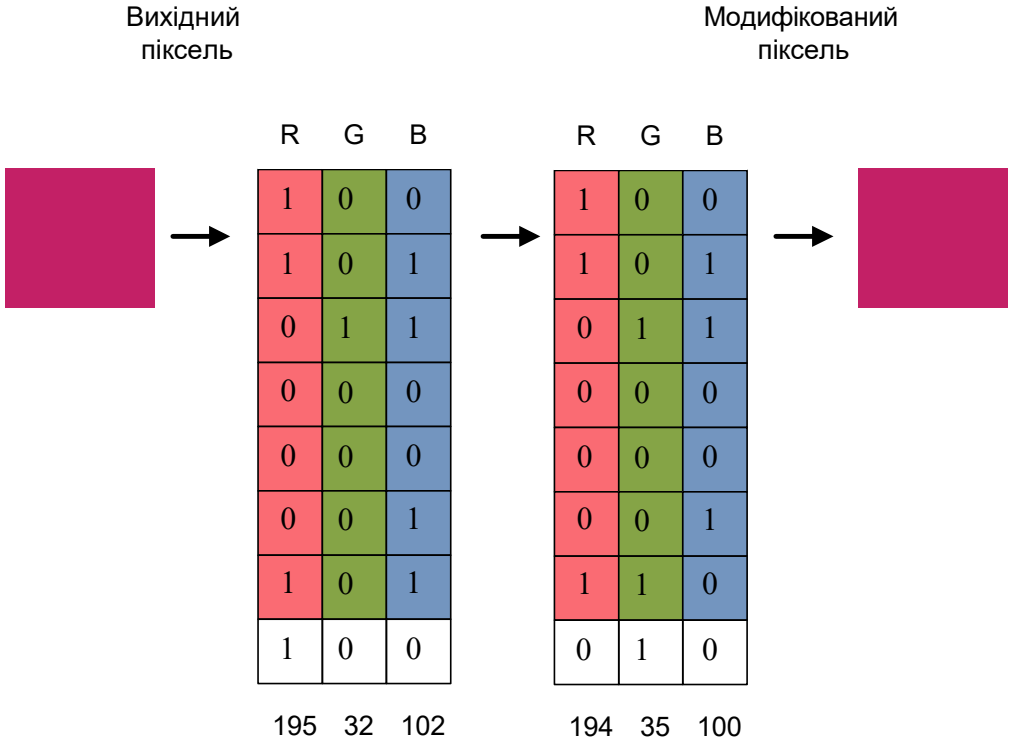

Рисунок 2.3 – Приклад вбудовування послідовності біт 101100 у вихідний піксель зображення

Тобто, судячи з візуального аналізу рис. 2.2, можемо констатувати повну відсутність візуально помітних відмінностей між вихідним та модифікованим пікселями носія.

Водночас, виходячи з особливостей представлення даних файлу BMP необхідно внести ряд уточнень, а сааме [7, 16, 17]:

1. Формат BMP не є поширеним у мережі з причини надмірної ваги файлів – так, порівняно з форматом JPG, вага файлу може бути більшою у 15-32 разів; відтак, його сам факт його використання може викликати підозру;

2. Говорячи про формат BMP, необхідно внести уточнення щодо того, які саме біти у даному випадку може бути використано у рамках концепції LSB.

У підсумку слід зазначити, що використовувати BMP-формату у «чистому» вигляді з тим, щоб факт його застосування не викликав підозр, необхідно не на пряму (фотоматеріали, діаграми тощо), а опосередковано, наприклад, у вигляді сервісних файлів, зокрема:

- на рівні файлів favicon.ico у веб-середовищі;

- як складових інтерфейсу веб-додатків з обмеженою роздільною здатністю і вагою.

В інших випадках використання BMP-формату, виходячи з його надмірної ваги (24 біта/піксель) не є виправданим.

2.1.2 Принцип функціонування стеганографічного методу «останнього біту» на базі JPEG -носія

У спосіб, подібний до розглянутого у п.2.1.1, реалізується процес інкапсуляції секретних біт до носія формату JPEG [6, 8, 17].

Відмінності при цьому полягають у наступному:

- якщо на випадок BMP (не стисненого формату) кожен піксель є доступним для модифікації безпосередньо, то модифікація LSB-біт носія JPEG потребує попереднього виконання ряду технологічних перетворень згідно з базисом JPEG (рис. 2.4);

- викривлення, внесені у результаті інкапсуляції біт b_m секретного повідомлення у носій JPEG, суттєво меншою мірою впливають на ступінь візуальної відмінності між вихідним та підсумковим носіями;

- замість пікселів $p_{(m,n)}$, на випадок JPEG модифікації підлягають останні біти $b_{m,n}^{(0,clr)}$ компонент $c_{m,n}^{(cnl)}$, у яких ідентифікатор cnl вказує на належність компоненти яскравісному Y чи одному з хроматичних каналів (Cb та Cr);

- обробка носія JPEG виконується на рівні сегментів $s_{x,y}^{(cnl)}$ розміром 8x8 компонент, тобто, на цей випадок $m \in [1,8]$, $n \in [1,8]$;

- потенційно доступна ємність V носіїв JPEG- типу у загальному випадку не є фіксованою, та залежить від застосованих опцій кодування, а саме – обрану модель проріджування хроматичних складових їх сегментів $s_{x,y}^{(cnl)}$.

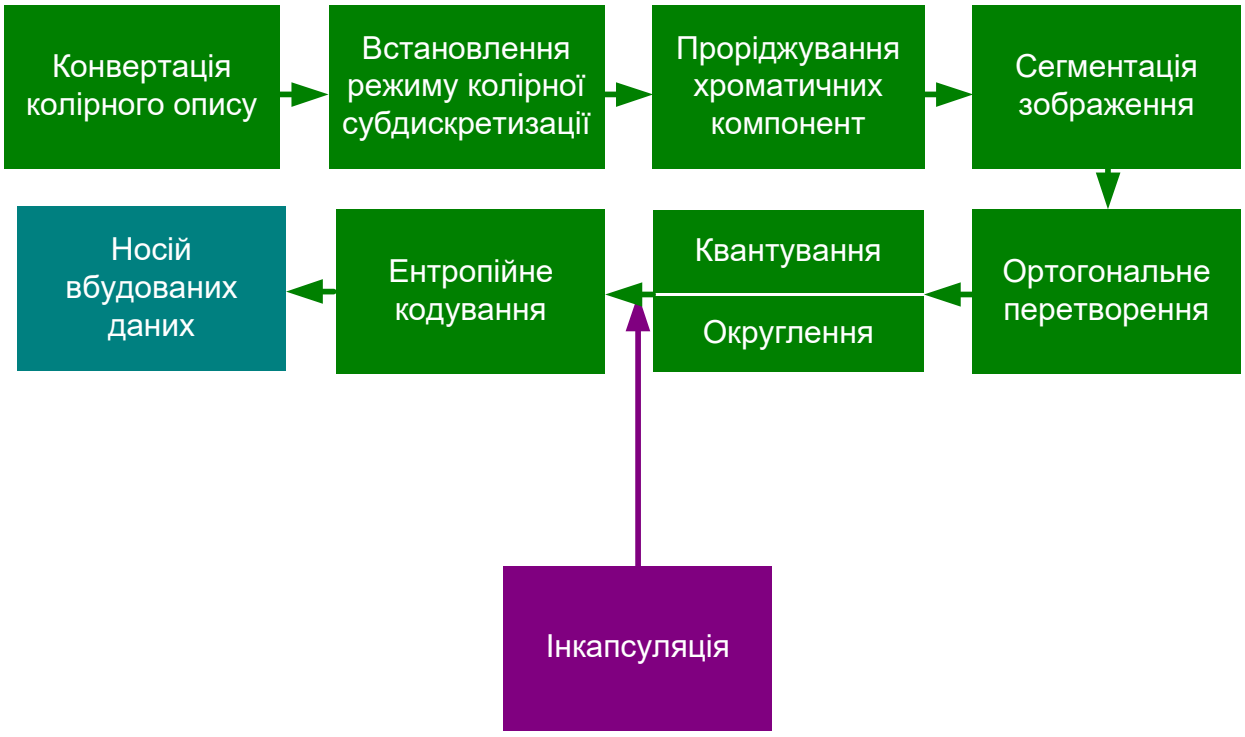

Рисунок 2.4 – Етапність вбудовування біт секретного повідомлення на засадах LSB для контейнеру типу JPEG

2.1.3 Аналіз переваг та недоліків стеганографічного методу «останнього біту»

Безумовно, як на випадок контейнерів BMP, так і JPEG, суттєвою перевагою можемо вважати простоту, тривіальність реалізації методу та його

низьку ресурсоємність. Тобто, такий метод може бути реалізовано сьогодні, практично, на будь-яких пристроях.

Водночас, суттєвим недоліком розглянутого методу є технологічна легкість виявлення факту присутності інкапсуляції [10, 11].

Для цього достатньо носій, що підлягає аналізу, піддати декомпозиції, та сформувати візуальний образ бітової площини нульового порядку згідо з виразом 2.4 з наступним його аналізом. Даний тип атаки на ймовірну стегосистему отримав назву «візуальної атаки».

При цьому у випадку, якщо файл-зображення міститиме вбудовану інформацію, про факт її наявності свідчитиме аномальний характер змісту однієї чи кількох бітових площин $\Omega^{(n,\,cnl)}$ (зазвичай $n \in [1;0]$), або локальних ділянок площин.

Приклади виявлених аномалій, утворених у наслідок виконання інкапсуляції біт b_m повідомлення у носій на рівні $\Omega^{(0,\,Cb)}$, демонструє рис. 2.5.

а) б)

Рисунок 2.5 – Вихідний файл-носій а) та зміст його LSB на рівні бітової площини $\Omega^{(0,\,Cb)}$ нульового порядку б)

Як бачимо з рисунку 2.5, факт наявності вбудованих даних у розглянутому випадку повністю підтверджується.

Таким чином, попри простоту самого алгоритму, та легкість його утілення, ключовим його недоліком у класичній реалізації є відносно низька захищеність P стеганограм.

При цьому, якщо для випадку BMP вбудовування даних може виконуватися напряму в той чи інший (або кілька одразу) колірний канал, то на випадок JPEG-носія процедура є більш складнішою. У свою чергу, це зумовлено тим, що, на відміну від BMP, інкапсуляція даних виконується у ході конвеєрних перетворень JPEG, як показано рисунком 2.4. Тобто, етап інкапсуляції необхідно інтегрувати у сам алгоритм JPEG, як додатковий ктор обробки.

2.2 Принцип функціонування стеганографічного методу нерівномірних відрізків

Недоліку, притаманного класичному LSB-методу, позбавлений методу нерівномірних відрізків. Означений метод також відноситься до групи LSB-орієнтованих.

Разом з тим, хоча для його випадку процес вбудовування біт секретного повідомлення також реалізується відповідно до виразу (2.1), водночас, класичний LSB-метод та метод нерівномірних відрізків мають ряд відмінностей, серед яких ключовими є [6]:

- принципово різні алгоритми обходу LSB-простору носія у процесі інкапсуляції;

- різні рівні теоретично можливої та фактичної величин ємності V_{fill} заповненого носія, який вважається прийнятним та не суперечить умовам (1.2).

Розглянемо сутність методу більш детально.

Отже, як і у випадку класичного методу LSB, біти повідомлення вбудовуються у площину $\Omega^{(0,\,cnl)}$ нульового порядку.

Проте, у даному випадку замість алгоритму, зображеному на рис. 2.5 (який є характерним класичному LSB), будуть мати місце ряд відмінностей, а саме:

1. Обробка виконується на рівні усього простору носія, відтак, у цьому разі $m \in [1,8]$, $n \in [1, 8]$.

2. Позиція біту $b_{m,n}^{(0,cnl)}$ обирається, виходячи з особливостей компоненти $c_{m,n}^{(cnl)}$, якій належить попередньо вбудований $b_{\mu-1}$-й біт секретного повідомлення. Тобто, якщо у межах класичного методу LSB

інкапсуляція біт виконується послідовно, керуючись обраном напрямком обходу згідно принципу, проілюстрованому рис.2.2, для методу нерівномірних відрізків позиція $b_{\mu+1}$-го біту визначається на базі наступного виразу:

$$\Delta b_{\mu+1} = \sum_{i=1}^{I} ((b(i)_{\mu} - 1) \,|\, b(i)_{\mu} = 1), \qquad (2.7)$$

де $\Delta b_{\mu+1}$ - зміщення позиції вбудовування $b_{\mu+1}$-го біту відносно вбудованого b_{μ} - го.

Тобто, біт $b_{\mu+1}$ буде розміщено відносно b_{μ} на відстань $\Delta b_{\mu+1}$, що визначається за кількістю одиничних біт (з 7 по 1 розряди), задіяних для представлення компоненти $c_{m,n}^{(cnl)}$, якій належить b_{μ}.

Таким чином, застосування методу візуальної атаки, як у випадку класичного LSB, не здатне виявити факт присутності ознак модифікації біт $b_{m,n}^{(0,cml)}$ простору $\Omega^{(0,cnl)}$.

Разом з тим, ключовий недолік методу полягає у тому, що заповнення носія за принципом (2.7) зазвичай передбачає обхід за рядками/стовпцями, починаючи зі стартового біту $b_{1,n}^{(0,cml)}$, $b_{n,1}^{(0,cml)}$ або взагалі $b_{1,1}^{(0,cml)}$. У таких умовах, виконуючи атаку за прямим зчитуванням, зловмисник має $(M + N - 1)$ варіантів сканування ймовірної інкапсуляції перебором з наступною її інтерпретацією. Тому, якщо вбудовану інформацію попередньо не було шифровано, зловмисник може її отримати.

3. ДОСЛІДЖЕННЯ МОЖЛИВОСТІ МОДИФІКАЦІЇ КЛАСИЧНИХ МЕТОДІВ ПОБУДОВИ СТЕГОСИСТЕМ ДЛЯ ЗБІЛЬШЕННЯ ЇХ ЕФЕКТИВНОСТІ

3.1 Базові засади модифікації класичних методів побудови стегосистем

У будь-якому випадку, концепція, на базі якої створюється та функціонує канал прихованого передавання даних, є поєднанням понять «носій», «стеганографічний алгоритм» та «стегоключ».

При цьому, ключ являє собою, у сутності, лише масив параметрів алгоритму приховування.

У свою чергу, на носії, як файли ряду типів, що є прийнятними для інкапсуляції біт приховуваного повідомлення, про що зазначалося у п. 1.8, розробник стегосистеми, за великим рахунком, не може чинити вплив.

Разом з тим, стегоалгоритм, як ключовий складник усієї концепції, по-перше, чинить суттєвий вплив на усю стегосистему у цілому а по-друге – може корегуватися розробниками.

Таким чином, модифікація класичних методів побудови стегосистем, фактично, зводиться до усунення недоліків, властивих даним алгоритмам.

У даному випадку розглянемо приклади модифікації алгоритму нерівномірних інтервалів.

3.2 Напрямки модифікації алгоритму нерівномірних інтервалів

Як було виявлено попередньо, ключовим недоліком зазначеного алгоритму є фіксована прив’язка стартової позиції, а саме:

- до початку координат у файлі (позиція (1,1), яка відповідає лівому верхньому куту зображення-носія);
- до першого стовпця у межах носія;
- до першого рядка у межах носія.

Додатковим недоліком можна також вважати орієнтованість на моноконтейнери у ході передавання даних.

Розглянемо ймовірні підходи до усунення означених вище недоліків.

3.2.1 Вибір підходів до усунення фіксованої прив'язки стартового біту інкапсуляції повідомлення

Для того, щоб забезпечити можливість нечіткої прив'язки початкового біту, необхідно сформувати правило первинного позиціонування алгоритму, за яким позиція (m,n) першого біту b_μ повідомлення обиратиметься динамічно.

При цьому, процедура вибору має відповідати таким вимогам:

- ознака, за якою здійснюється вибір стартової координати (m,n), має забезпечувати однозначність рішення. Тобто, ймовірність помилкового вибору координати у ході виокремлення прихованих даних повинна мінімізуватися;

- нетривіальність механізму вибору стартової координати, що у загальному випадку можна розглядати, як неможливість отримання її значення з множини ймовірних, одержаних шляхом лінійних перетворень відносно набору координат (1, 1), (1, n) та (n, 1) (наприклад, (0,25N, 1), (0,5N, 0,5M) тощо).

Відтак, за таких умов, використовуючи атаку прямим перебором зловмисник буде змушений здійснити розгляд $(M \times N)$ варіантів позиціонування стартової координати.

Виходячи з зазначених вище вимог, принцип вибору стартової координати у загальному випадку може бути описано наступною функціональною залежністю: $b_{m,n}^{(0,cnl)}$

$$b(start)_{m,n}^{(0,cnl)} = \varphi(\Xi), \tag{3.1}$$

де Ξ - множина параметрів носія, на базі яких виконується первинне позиціонування алгоритму;

φ - алгоритм обробки параметрів, який є відомим передавачу та приймачеві.

При цьому, якщо дану концепцію деталізувати, вираз (3.1) буде переписано до вигляду:

$$b(start)_{m,n}^{(0,cnl)} = f(K;Q), \tag{3.2}$$

де Q - множина особливостей змісту носія;

K - секретний ключ, що містить у собі перелік певних параметрів та специфіку інтерпретації кожного з них.

За таких умов, якщо навіть зловмисник отримає доступ до усієї сукупності Ξ параметрів, які використовуються для позиціонування, йому не буде відомим спосіб інтерпретації цих даних.

У даному випадку може бути розглянуто ряд підходів до реалізації зазначеного завдання, у рамках кожного з яких можуть бути використаними зовнішні чи внутрішні параметри, які при цьому однаково відомі як передавачу, так і приймачеві.

Так, до зовнішніх параметрів можемо віднести:

- час доби та день неділі;

- відомості щодо провайдерів інтернет-сервісів і тариф обслуговування передавача та приймача тощо.

У свою чергу, внутрішніми параметрами тут можемо вважати:

- сукупність статистичних характеристик ймовірного носія приховуваних даних;

- обсяг $d(t_n)$ даних, які було надіслано приймачеві відкритим каналом у деякому визначеному проміжку часу $[t_n; t_{n+1}]$.

- відомості про тип та версію ПЗ передавача, яке є базисним для побудови стегоканалу.

Отже, щонайменше 5 підходів може бути використано для створення динамічної прив’язки щодо інкапсуляції біт приховуваного повідомлення, в основі кожного з яких – вибір тих чи інших параметрів позиціонування.

Розглянемо далі специфіку використання кожного з підходів для виявлення найбільш ефективного з них.

3.2.2 Визначення найбільш ефективного підходу до усунення фіксованої прив’язки стартового біту інкапсуляції повідомлення

Очевидно, що найбільш ефективним можемо вважати той підхід до побудови динамічної прив’язки, який забезпечує найвищий рівень динаміки змін стартової позиції як для кожного нового повідомлення, так і у рамках єдиного повідомлення, яке транслюється приймачеві за кілька сеансів.

Виходячи з цього, відомості щодо провайдерів інтернет-сервісів і тариф обслуговування передавача та приймача не можуть використовуватися як фактори динамічної прив’язки стартового біту інкапсуляції приховуваного

повідомлення. Це зумовлено тим, що дані відносно провайдерів сервісу та конкретні тарифи є незмінною у часі величиною.

Далі виконаємо дослідження доцільності вибору відомостей щодо часу доби та дня неділі як параметрів алгоритму, використовуючи які можливо забезпечити динамічний вибір стартового біту для інкапсуляції повідомлення.

Це, у свою чергу, пояснюється тим, що згідно даного підходу, увесь імовірний час трансляції (умовно – з часової мітки t_{start} до мітки t_{stop}) попередньо підлягає розподілу на елементарні ділянки $(t_n; t_{n+1})$, як показано рис. 3.1.

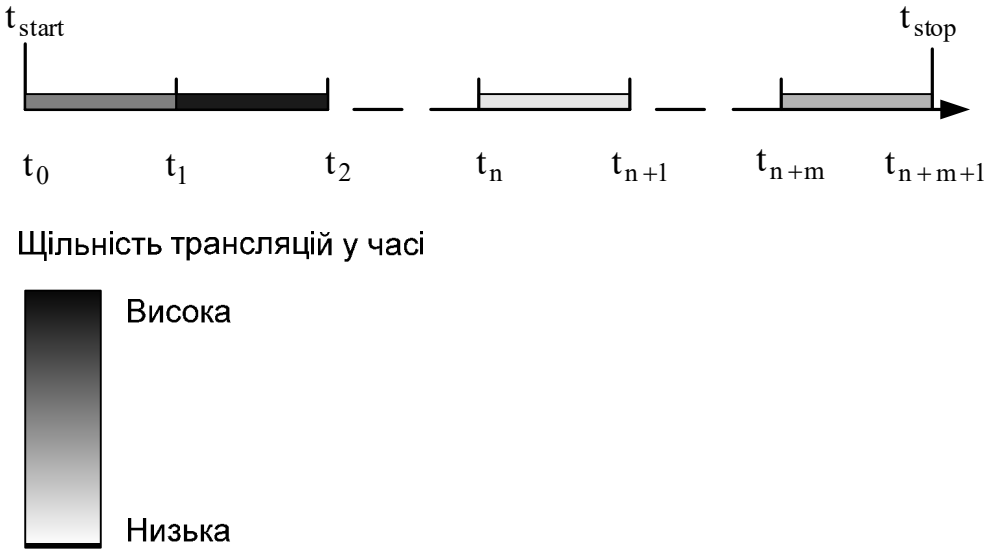

Рисунок 3.1 – Принцип розподілу загального часу ймовірних трансляцій на окремі ділянки та приклади щільності трансляцій

При цьому, у ході ймовірного періоду трансляції $[t_{start}; t_{stop}]$ загальна кількість сеансів прихованого передавання даних, по-перше, є випадковою величиною. По-друге, розподіл щільності трансляцій за період $[t_{start}; t_{stop}]$ також буде випадковим [18].

Відтак, для будь-якої ділянки $(t_n; t_{n+1})$ може спостерігатися велика кількість сеансів трансляцій, для кожного з яких буде обрано єдину множину $\Xi = \{\xi_1; \xi_1; \xi_1 \ldots; \xi_1\}$ параметрів.

Тобто, на ділянці часу $(t_n; t_{n+1})$ буде задіяно одну і ту ж позицію (m, n) першого біту повідомлення. Це потенційно веде до зменшення стійкості системи у цілому.

На цей випадок дану проблему частково може бути частково вирішено, користуючись залежністю між ймовірністю P_{dm} викриття стегосистеми та шириною елементарної ділянки часу трансляції. Така залежність може бути представлена наступним співвідношенням:

$$P_{dm} \propto \frac{1}{(t_{n+1} - t_n)}, \tag{3.4}$$

тобто, величини P_{dm} та P_{dm} - обернено пропорційні.

Отже, чим вужчим буде інтервал $(t_n; t_{n+1})$, тим нижчою буде ймовірність P_{dm} викриття стегосистеми.

Виходячи з цього, відрізок $(t_n; t_{n+1})$ доцільно скоротити до часу, мінімально необхідного для надсилання повідомлення у рамках прийнятої технології передавання даних.

Недоліком такого підходу є необхідність забезпечення побітної синхронізації для того, щоб приймач міг коректно зчитувати дані з носія. Тобто, такий підхід не можемо вважати ефективним [8].

Розглянемо тепер підхід, у рамках якого параметром, необхідним для встановлення позиції першого біту секретного повідомлення, прийматиметься обсяг $d(t_n)$ даних, які було надіслано приймачеві відкритим каналом у деякому визначеному проміжку часу $[t_n; t_{n+1}]$.

Такий підхід є більш ефективним, ніж попередньо розглянутий, так як при цьому, у загальному випадку, справедливою є залежність:

$$d(t_1; t_2) \neq d(t_2; t_3) \neq ... \neq d(t_n; t_1)... \neq d(t_N; t_N), \tag{3.5}$$

тобто, теоретично за кожен інтервал часу надсилається різна кількість біт, отже, позиція першого біту буде зміщуватися для кожного наступного повідомлення.

Разом з тим, головний недолік такого підходу полягає у необхідності існування каналу відкритої взаємодії між передавачем та приймачем.

Водночас, даний підхід може бути використано і тоді, якщо величину $d(t_n)$ вимірювати для каналу прихованої взаємодії.

Що стосовно підходу, у рамках якого параметрами позиціонування розглядається сукупність статистичних характеристик ймовірного носія приховуваних даних, можемо зазначити наступне [8]:

- даний підхід не потребує використання додаткових каналів для збору статистичної інформації;

- як у рамках одного контейнеру, так і різних контейнерів у цілому, властивим є широкий діапазон значень їх статистичних характеристик;

- у рамках контейнеру/його частини може бути зафіксовано велику кількість статистичних характеристик;

- з усієї ймовірної статистичних характеристик розробник може використати будь-яку їх кількість у будь-якій прив’язці одна до одної.

За таких умов, у свою чергу, може забезпечуватися найвища динаміка змін позиції першого біту секретного повідомлення.

У свою чергу, ефективність застосування відомостей щодо типу та версії ПЗ передавача, що є базисним для побудови стегоканалу, можемо вважати низькою, так як ці відомості є статичними.

Таким чином, за сукупністю можливих переваг, найбільш прийнятним для реалізації концепції динамічного зміщення позиції стартового біту стеганограми можемо вважати підхід, що базується на використанні сукупності статистичних характеристик ймовірного носія.

3.2.3 Характеристики контейнеру, які може бути використано у якості параметрів позиціонування стартового біту інкапсуляції

У раках даного підходу може бути використано:

- характеристики файлу-носія;

- характеристики окремих ділянок файлу-носія;

- певну комбінацію характеристик файлу-носія та його окремих ділянок.

При цьому, до можливої множини характеристик файлу-носія, які може бути обчислено без суттєвого навантаження на передавач/приймач, можемо віднести:

- кількість біт $d(g)_p$ корисного навантаження, яка може розраховуватися за виразом:

$$d(g)_p = d(g) - \sum_{q=1}^{E} d(seg_q), \tag{3.6}$$

де $d(g)$ - загальна вага файлу-носія;

$d(seg_q)$ - кількість біт для опису одного з E службових сегментів у файлі;

- найбільша (найменша, середня) величина компоненти за одним з каналів – Y, Cb чи Cr у носієві;

- роздільна здатність $M \times N$ носія тощо.

Очевидним недоліком такого підходу є тривіальність механізму вибору параметрів.

Отже, більш доцільним є використання саме комбінації статистичних характеристик файлу-носія та його окремих ділянок.

Означене рішення дає підставу поєднати та деталізувати вирази (3.1) та (3.2), а саме:

$$b(start)_{m,n}^{(0,cnl)} = \varphi(\Xi; \Phi) = f(K; \Xi; \Phi), \tag{3.7}$$

де Ф - множина параметрів локальних ділянок носія, на базі яких виконується первинне позиціонування алгоритму.

Виходячи з цього, має сенс застосувати такі параметри носія, які забезпечують:

- гнучку зміну закономірностей обробки носія без внесення корекцій до алгоритму, обмежуючись внесенням змін до секретного ключа К;

- утруднення процесу виявлення шляхом прямого перебору.

3.2.4 Алгоритм позиціонування стартового біту з урахуванням характеристик контейнерів

У загальному випадку, для вибору стартового біту $b(start)_{m,n}^{(0,cnl)}$ доцільним для використання може бути наступний принцип:

$$b(start)_{m,n}^{(0,cnl)} \in \{\xi c(max)_{m,n}^{(0,cnl)}; (\xi + \chi) c(max)_{m,n}^{(0,cnl)}\}, \tag{3.8}$$

де ξ та χ - параметричні коефіцієнти, який розглядаються як складова ключа К.

У свою чергу, якщо говорити про урахування параметрів ділянок контейнеру, тоді має сенс знову звернутися до поділу початкового носія на сегменти $s_{x,y}^{(cnl)}$.

При цьому, якщо розглядати окремі ділянки файлу-носія, слід попередньо визначитися з наступним:

- яка ділянка/ділянки носія будуть братися до уваги;
- яка кількість ділянок розглядатиметься.

З точки зору поєднання характеристик файлу-носія та характеристик окремих ділянок файлу-носія, як було зазначено раніше, доцільною для застосування буде концепція, яка передбачає, що:

1. Характеристики носія та його окремої ділянки спільно ураховуються.

2. На першому етапі вибору стартової координати, подібно до ідеології, зазначеної виразом (3.8), з усієї множини сегментів, $s_{x,y}^{(cnl)}$, які утворюють носій, обирається стартовий $s(start)_{x,y}^{(cnl)}$ сегмент, що відповідає наступним умовам:

$$s(start)_{x,y}^{(cnl)} = s_{x,y}^{(cnl)} \mid ((\Phi_1 \in [\Phi_1(max); \delta\Phi_1(max)]) \,\&\, \& (\Phi_3 \in [\Phi_2(max); \upsilon\Phi_2(max)]) \,\&\, ... \,\&\, (\Phi_n \in [\Phi_n(max); \gamma\Phi_n(max)])) \quad (3.9)$$

де Φ_1, Φ_2 та Φ_n - ряд характеристик сегментів;

δ, υ та γ - параметричні множники, які є відповідними полями стегоключа.

Тут у ролі характеристик Φ_1-Φ_n сегментів $s_{x,y}^{(cnl)}$ можуть застосовуватися, наприклад:

- кількість нульових компонент $c_{m,n}^{(cnl)}$ у межах сегменту $s_{x,y}^{(cnl)}$;
- потужність добутку значень компонент $c_{m,n}^{(cnl)}$ за однією чи кількома конкретними діагоналями тощо.

3. У решті решт на третьому етапі, маючи попередньо визначений сегмент $s(start)_{x,y}^{(cnl)}$, далі необхідно запровадити механізм для одностайного виявлення координат (x,y) стартового біту $b(start)_{m,n}^{(0,cnl)}$ у його межах.

У загальному випадку, для цього також може бути використано алгоритм, що дозволяє обирати координату стартового біту $b(start)_{m,n}^{(0,cnl)}$ за сукупністю характеристик сегменту $s(start)_{x,y}^{(cnl)}$.

При цьому, для мінімізації помилкового вибору координати (m,n) у ході зчитування прихованої послідовності приймачем, пропонується:

- координати визначати не за однією, а за сукупністю характеристик сегменту $s(start)_{x,y}^{(cnl)}$;

- виконувати оцінку тих чи інших характеристик сегменту у яскравісній та хроматичних областях – тобто, брати до уваги $s(start)_{x,y}^{(Y)}$, $s(start)_{x,y}^{(Cb)}$;

- обчислювати окремо координату m та окремо n.

Тоді, наприклад, координату m може бути визначено у наступний спосіб:

1. Визначається одна з характеристик H_1 сегментів $s(start)_{x,y}^{(cnl)}$ серед усього носія, беручи до уваги $s(start)_{x,y}^{(Cb)}$ та $s(start)_{x,y}^{(Cr)}$, :

$$H_1 = f(s(start)_{x,y}^{(Cb)}; s(start)_{x,y}^{(Cr)}) \tag{3.10}$$

Наприклад, це може бути максимальна сума/добуток ненульових хроматичних компонент на рівні носія, тобто:

$$H_1(max) = \sum_{m=1}^{M}\sum_{n=1}^{N}((c_{m,n}^{(Cb)} + c_{m,n}^{(Cr)}) \mid c_{m,n}^{(Cr)} \neq 0). \tag{3.11}$$

У нашому випадку розглянутий приклад є одним з найбільш простіших. Замість нього може бути використано такі характеристики, як, наприклад:

- сума різниць між найбільшою та найменшою компонентами у каналах Cb та Cr;

- добуток ненульових хроматичних компонент;

- добуток ненульових хроматичних компонент, що мають парні/непарні координати тощо.

2. Вимірюється величина H_1 для сегменту $s(start)_{x,y}^{(Cr)}$.

4. Встановлюється градація величин від одержаної і до 0. При цьому, відрізок $[H_1; 0]$ розподіляється на 8 рівномірних піддіапазонів, а саме:

- $[H_1(\max) ; 0{,}875\,H_1(\max)]$;
- $(0{,}875\,H_1(\max) ; 0{,}75\,H_1(\max)]$;
- $(0{,}75\,H_1(\max) ; 0{,}625\,H_1(\max)]$;
- $(0{,}625\,H_1(\max) ; 0{,}5\,H_1(\max)]$;
- $(0{,}5\,H_1(\max) ; 0{,}375\,H_1(\max)]$;
- $(0{,}375\,H_1(\max) ; 0{,}25\,H_1(\max)]$;
- $(0{,}25\,H_1(\max) ; 0{,}125\,H_1(\max)]$;
- $(0{,}125\,H_1(\max) ; 0]$.

Далі кожному піддіапазону ставиться у відповідність координата m відповідно до наступної системи виразів:

$$\begin{cases} H_1 \in [H_1(\max) ; 0{,}875\ H_1(\max)] \rightarrow m = 8; \\ H_1 \in (0{,}875\ H_1(\max); 0{,}75\ H_1(\max)] \rightarrow m = 7; \\ H_1 \in (0{,}75\ H_1(\max); 0{,}625 H_1(\max)] \rightarrow m = 6; \\ H_1 \in (0{,}625\ H_1(\max); 0{,}5\ H_1(\max)] \rightarrow m = 5; \\ H_1 \in (0{,}5\ H_1(\max); 0{,}375\ H_1(\max)] \rightarrow m = 4; \\ H_1 \in (0{,}375\ H_1(\max); 0{,}25 H_1(\max)] \rightarrow m = 3; \\ H_1 \in (0{,}25\ H_1(\max); 0{,}125\ H_1(\max)] \rightarrow m = 2; \\ H_1 \in (0{,}125 H_1(\max) ; 0] \rightarrow m = 1. \end{cases} \qquad (3.12)$$

Таким чином, якщо даний показник H_1 для сегменту $s(start)_{x,y}^{(Cr)}$ належить тому чи іншому діапазону, це визначає номер позиції координати у рядку.

3. Встановлюється циклічний зсув значень піддіапазонів

Даний технологічний етап виконується для збільшення захищеності стеганограми, та передбачає виконання зміщення діапазонів значень m у проміжку $\theta = \overline{1; 7}$. Дана процедура може бути подана наступним виразом:

$$m := sh(\theta, dir, m), \qquad (3.13)$$

де θ - кількість позицій, на які виконується зсув;

dir - напрямок зсуву, відповідно – вправо, чи вліво.

Отже, наприклад, при $\theta = 2$ та за умови виконання зсуву вліво, маємо:

$$\begin{cases} H_1 \in [H_1(\max)\,;\, 0{,}875\ H_1(\max)] \rightarrow m = 6; \\ H_1 \in (0{,}875\ H_1(\max);\, 0{,}75\ H_1(\max)] \rightarrow m = 5; \\ H_1 \in (0{,}75\ H_1(\max);\, 0{,}625 H_1(\max)\ ] \rightarrow m = 4; \\ H_1 \in (0{,}625\ H_1(\max);\, 0{,}5\ H_1(\max)] \rightarrow m = 3; \\ H_1 \in (0{,}5\ H_1(\max);\, 0{,}375\ H_1(\max)] \rightarrow m = 2; \\ H_1 \in (0{,}375\ H_1(\max);\, 0{,}25 H_1(\max)\ ] \rightarrow m = 1; \\ H_1 \in (0{,}25\ H_1(\max);\, 0,125\ H_1(\max)] \rightarrow m = 8; \\ H_1 \in (0{,}125 H_1(\max)\,;\, 0] \rightarrow m = 7. \end{cases} \qquad (3.14)$$

У свою чергу, координата n обчислюватиметься аналогічно способу, запропонованому для m. Відмінність полягатиме виключно в алгоритмі розрахунку характеристики H_2, величина якої, відповідно, задаватиме позицію стартового біту у стовпці. При цьому, наприклад, спочатку розраховуємо максимальну величину H_2 серед усіх сегментів носія за формулою:

$$H_2(\max) = \sum_{m=1}^{M}\sum_{n=1}^{N}(c_{m,n}^{(Y)} \mid c_{m,n}^{(Y)} > 0) - \sum_{m=1}^{M}\sum_{n=1}^{N}(c_{m,n}^{(Cr)} \mid c_{m,n}^{(Cr)} > 0) \qquad (3.15)$$

З урахуванням розглянутих алгоритмів вибору координати (m, n) стартового біту $b(\text{start})_{m,n}^{(0,\text{cnl})}$, стегоключ, що використовується у рамках доповненого методу нерівномірних відрізків, матиме наступний вигляд:

$$K = \{\chi; \delta; \gamma; \upsilon; \xi; \theta_1; \text{dir}_1; \theta_2; \text{dir}_2\}, \qquad (3.16)$$

де dir_1 та dir_2 - напрямки циклічного зсуву значень піддіапазонів у ході обчислення значень стартової координати за рядком та стовпцем відповідно;

θ_1 та θ_2 - кількість позицій, на які виконується зсув у ході обчислення значень стартової координати за рядком та стовпцем відповідно.

У свою чергу, до процедурної частини модифікованого алгоритму нерівномірних інтервалів належатимуть:

1. Сам механізм вбудовування, який реалізовано у складі класичного алгоритму.

2. Алгоритм вибору стартової координати для вбудовування, що містить:

- механізми оцінки характеристик Φ_1 - Φ_n на рівні носія;

- механізми оцінки характеристик H_1 та H_2 на рівні попередньо обраного стартового сегменту;

- механізми, що ставлять у відповідність розрахованим величинам H_1 та H_2 певні значення m та n.

4. ТЕКСТОВА СТЕГАНОГРАФІЯ

4.1 Чинники, що зумовлюють актуальність використання алгоритмів приховування даних на базі тексту

На сьогодні алгоритми побудови прихованих каналів, які базуються на використанні носіїв текстового типу, можуть скласти конкуренцію алгоритмам, які, у свою чергу, орієнтуються на використання відео, графічних та аудіо- контейнерів [7, 8].

При цьому, чинниками, що зумовлюють зазначене, є:

- надзвичайна поширеність текстового контенту мережею;
- робастність більшості форматів текстових файлів;
- гнучка можливість регулювати рівень наповненості носія у чіткій відповідності до вимоги (1.2);
- простота реалізації алгоритмів стеганографії, що використовують текстові носії, та їх низька ресурсоємкість, що відповідає вимозі (1.4);
- дуже обмежена кількість реалізованих методів виявлення текстових стеганограм.

Так, на сьогодні мережею щосекунди транслюється надзвичайно велика кількість, наприклад, веб-документів та файлів, що є їм супутніми. Це, у першу чергу, php, html, css, та файли зовнішніх скриптів різних форматів. За таких умов має сенс розглядати текстову стеганографію, як один з досить потужних інструментів побудови прихованих каналів обміну даними.

4.2 Загальні засади побудови стегоалгоритмів на базі текстових носіїв

Базисом переважної більшості стегоалгоритмів, що орієнтуються на текстові носії є специфіка розміщення символів у тексті, які не сприймаються людиною при читанні, та далеко не завжди беруться до уваги програмним забезпеченням, яке виконує відображення та аналіз текстової інформації.

У загальному випадку до такої специфіки розміщення символів у тексті можна віднести додатковий обсяг символів пробілу та знаків табуляції у тих чи інших локаціях рядку, чергування деяких службових символів, що не

ураховуються ПЗ-інтерпретатором, великих та рядкових літер, символів з різних алфавітів, що зовні однакові [19, 20].

Виходячи з цього, алгоритми текстової стеганографії реалізуються за такими напрямками, як [20, 21]:

- маніпуляція форматуванням тексту;
- зміна порядку слідування маркерів кінця рядку;
- т.з. «методи хвостових пробілів;
- методи, що використовують символи однакового написання;
- зміна коду символу пробіла тощо.

Попри велику кількість переваг даних підходів, зараз методи текстової стеганографії мають суттєво нижчий рівень розповсюдженості, ніж, наприклад, методи, орієнтовані на використання графічних носіїв.

Так, на сьогодні найширше застосовуються алгоритми, які орієнтовані на форматування тексту символами пробілу.

Для того, щоб детально проаналізувати існуючі стегоалгоритми на базі тексту, розглянемо попередньо існуючі особливості текстових дані.

4.3 Базові відомості про текстові дані

Дані текстового типу, зазвичай, є послідовності, які формуються на базі підмножини символів. При цьому, до такої підмножини символів входять лише символи, що друкуються (літери алфавіту, цифри, розділові знаки), а також ряд т.з. «управляючих знаків», - символи пробілу, табуляції, символи переведення рядку [22].

Отже, текстові дані — це інформація, яка подана у вигляді певного набору друкованих символів.

При цьому, згідно зі специфікацією MIME, таким даним відповідає тип text/plain.

Водночас, дані текстового типу нерідко протиставляються бінарному формату.

Ключовою відмінністю тут, що очевидно, є те, що бінарний формат не призначений на безпосереднє його сприйняття людиною.

На сьогодні існує ряд методів, наприклад, таких, як UUENCODE, які дають змогу представлення даних будь-якого формату, без виключення, у вигляді текстової інформації.

Важливим також є те, велика кількість мережевих протоколів початково орієнтовані на роботу виключно з даними с текстового типу, при цьому не можуть сприймати та далі виконувати інтерпретацію послідовностей бінарних символів.

4.3.1 Представлення текстових даних

Представлення текстової інформації в інформаційних системах у вигляді, з яким безпосередньо може взаємодіяти людина, потребує використання т.з. таблиць кодування [23].

Такі таблиці співставляють символи тексту з їхніми числовими інтерпретаторами (кодами), на рівні яких з ними взаємодіють обчислювальні системи.

На сьогодні у світі є чинними велика кількість таблиць кодування, серед яких одними з найбільш поширеними є:

- ASCII;
- Unicode;
- Windows 1250;
- Koi-8 тощо.

Так, таблиця ASCII (American standard code for information interchange,) містить у собі схему співставлення ряду широко розповсюджених друкованим та не друкованим символам відповідних числових кодів.

Сучасний варіант ASCII передбачає 8-бітний формат опису символа, тобто, на її базі може бути закодовано до 255 символів, на відміну від традиційної 7-бітної ASCII.

До таблиці входять:

- десяткові цифри;
- символи латинського алфавіту;
- символи національного алфавіту;
- розділові знаки;
- керуючі символи (не є друкованими).

Приклад кодів сучасної таблиці ASCII у прив’язці до QWERTY-клавіатури зображено на рис. 4.1.

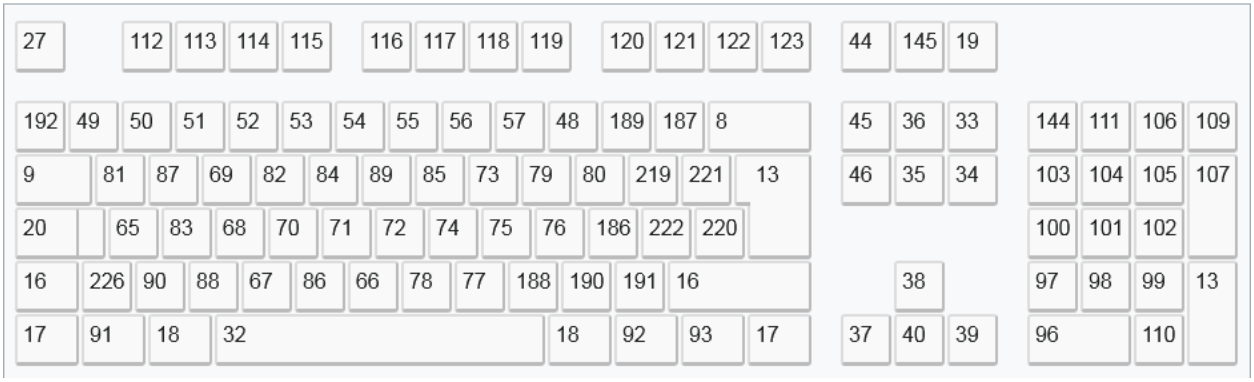

Рисунок 4.1 – Коди ASCII у прив’язці до QWERTY-клавіатури

У свою чергу, стандарт кодування символів Unicode, що у загальних рисах подібний як до ASCII, так і до інших подібних систем, містить у собі символи майже усіх мов світу, включаючи розділові знаки, цифри, спеціальні та керуючі символи. На сьогодні стандарт є домінуючим у Всесвітній мережі.

Так, за ідеологією Консорціуму розробників даного стандарту - Unicode — це унікальный код для будь-якого символу, незалежно від будь-яких умов - від програмної та апаратної платформи, від програмного засобу чи від мови [http://www.unicode.org/standard/translations/russian.html].

Однією з головних відмінностей Unicode від ASCII є, зокрема, суттєво більший простір кодових значень. Тому для більш компактного представлення символів Unicode розроблено і використовуються ряд стандартів їх зберігання.

Зокрема, найбільш широкого застосування набув зараз стандарт UTF-8 (Unicode Transformation Format, 8-bit) — 8-бітний формат перетворення Юникоду.

Так, у рамках UTF-8 передбачається:

- використання змінної кількості байт (від 1 до 4) для опису символу, за разунок чого забезпечується їх більш компактне зберігання та передача;

- повна сумісність з розширеною та традиційною 7-бітною таблицею ASCII.

Символи як UTF-8, так і Unicode взагалі, є крос-платформовими, тобто, однаково коректно інтерпретуються MAC OS, UNIX та Windows-середовищами.

Ще одним поширеним форматом кодування символів тексту, як зазначалося вище, є Windows-125x, на сьогодні у рамках цього формату розроблено стандарти, починаючи з 1250, по 1258.

Дане сімейство стандартів обмежене середовищем Windows, та застосовується для кодування символів мов ряду країн центральної та східної Європи.

Отже, таблиці кодування мають ряд спільних рис, серед яких у нашому випадку найбільш важливим є наявність керуючих символів, які використовуються у т.ч. для форматування тексту.

При цьому, як було зазначено раніше, на базі маніпулювання особливостями форматування тексту може бути створено стегоканал.

4.4 Приклали використання керуючих символів для форматування тексту

Одним з елементів форматування текстових даних є їх розбиття на рядки.

Так, у межах UNIX-подібних операційних системах розбиття текстового масиву на рядки кодується одним керуючим символом, якому відповідає код 10 з таблиці ASCII (це т.з. символ Line Feed, або LF).

У свою чергу, для випадків Windows та MS-DOS для відокремлення рядку необхідно задіяти 2 керуючих символа, що мають коди за таблицею ASCII, що дорівнюють 13 и 10 відповідно. Тут ця пара символів зветься Carriage Return та Line Feed, CR/LF.

При цьому, у середовищі Mac OS (виключаючи Mac OS X), відокремлення рядку кодується також, як для випадку UNIX, лише одним символом, якому у даному разі відповідає код 13.

Означені підходи до розбиття тексту на рядки одним чи двома управляючими знаками зумовлено наслідуванням принципу, властивого друкарським машинкам, у яких позиція вводу встановлювалася положенням барабану з папером. При цьому, для повертання барабану з послідуючим переходом до наступного рядку було необхідно натиснути одну або дві клавіші.

Аналогічним чином, символи розбиття тексту на рядки свого часу застосовувалися для керування механічним принтерами. У цьому випадку символ LF використовувався для повертання рулону з паперовою стрічкою. У свою чергу, завданням символу CR був виклих повернення друкувальної каретки до початку рядку. Це і зумовило назву символів —Line Feed (переведення рядку) та Carriage Return (повернення каретки).

Водночас, у межах ряду платформ процедуру розбиття тексту на рядки було реалізовано інакше. Так, текст на цей випадок розглядався як сукупність записів, кожен з яких був фіксованої довжини.

При цьому, рядки, що мали довжину меншу, ніж інші, доповнювалися необхідною кількістю пробілів. Даний принцип форматування даних, зокрема, властивий перфокартам.

Слід також додати, що текст нерідко може бути використано для представлення таких даних, які не можна вважати повністю текстовими. На цей випадок інші формати даних надбудовуються над текстом, для чого представлення їх керуючих конструкцій здійснювалося з використанням друкованих слів та розділових знаків. Такий підхід забезпечує зручність роботи з даними на двох рівнях.

Зокрема, текстові дані, розміщені у файлах HTML та XML можуть переглядатися а також - редагуватися, як у режимі форматування WYSIWYG (особливості форматування не ураховуються), так і з можливістю повного доступу до усіх можливостей мови розмітки. Наприклад, сьогодні більність існуючих мов програмування для створення вихідного коду програми використовує саме текстовий формат.

Окрім цього, конфігураційні файли багатьох програмних засобів мають у своєму складі щонайменше один сегмент текстового формату, навіть за тієї умови, що інший зміст файлу являє собою числові послідовності та бінарні перемикачі (так/ні).

Означений підхід, з одного боку, певним чином ускладнює відповідні програмні засоби з тієї причини, що виникає необхідність перетворення даних, поданих у текстовому форматі, у внутрішній формат ПЗ, а далі – вокнувати зворотне перетворення. Проте, з іншого боку, отримується можливість виконувати конфігурування програмних модулів вручну, не використовуючи при цьому засоби налаштування програмних продуктів. Така можливість, у свою чергу, здатна щонайменше заощадити час користувачам та/або розробникам.

4.5 Методи приховування даних на базі текстових носіїв

4.5.1 Алгоритм зміни порядку розміщення маркерів кінця рядку

В основі даного методу знаходиться несприйняття переважною більшістю засобів відображення текстових даних черговості розміщення

символів переведення рядку CR та повернення каретки LF, які обмежують кожен рядок у текстовому масиві. У рамках даного методу передбачається, що шляхом маніпулювання черговістю слідування символів CR та LF, може кодуватися той чи інший бінарний символ.

Наприклад, випадку використання традиційної черговості слідування символів, тобто, CR/LF відповідає вбудовування символу 0, а за умови використання інверсійної черговості, тобто, LF/CR - символ 1.

Для цього випадку функціонал зазаченого алгоритму зображено наступною схемою, як показано рис.4.2.

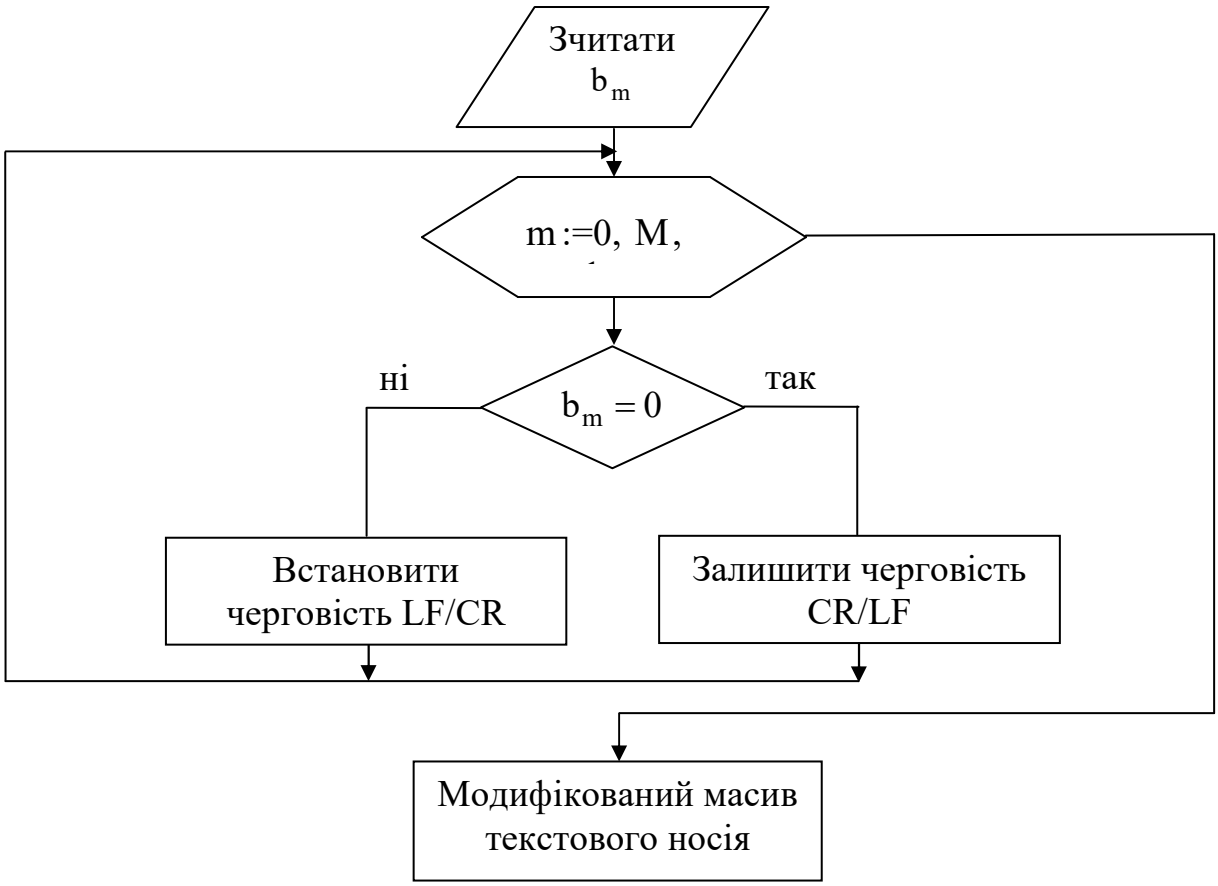

Рисунок 4.2 – Схема алгоритму зміни порядку розміщення маркерів кінця рядку

Таким чином, можемо зазначити, що до переваг алгоритму зміни порядку розміщення маркерів кінця рядку можемо віднести [20]:

- тривіальність та простоту реалізації;
- низьке обчислювальне навантаження на систему;

- незначна величина відношення $\frac{V_{fill}}{V}$, яка у відсотковому відношенні у середньому дорівнює 0,15%;

- велика кількість текстових файлів різних типів, які може бути використано у якості носія.

Водночас, недоліками алгоритму є:

- можливість його застосування лише у межах Windows/MS-DOS;

- неможливість реалізації методу на базі носіїв форматів doc/rtf, так як офісні пакети (WordPad, MS Word) некоректно опрацьовують змінену черговість слідування маркерів кінця рядку.

Отже, носіями у цьому випадку можуть бути файли cfg, html, css, php, txt, hlp тощо.

4.5.2 Алгоритм «хвостових пробілів»

Аналогічно передуючому алгоритму, даний алгоритм для інкапсулювання біт приховуваного повідомлення виконує маніпуляції з символами, що локалізуються у кінці рядку.

При цьому, у кінці рядку файлу-носія вписується додатковий символ пробілу на той випадок, коли необхідно кодувати символ 1 повідомлення, яке вбудовується.

При цьому, на випадок вбудовування нульового біту додатковий символ пробілу не вноситься.

Як і у випадку алгоритму зміни черговості розміщення маркерів кінця рядку, зчитування тексту виконується порядково.

У цьому випадку попередньо у кінці рядку попередньо видаляються символи пробілів та супутні, а саме – як пробіли, так і символи табуляції, символи повернення каретки та нового рядку.

Далі, залежно від того, який саме біт секретного повідомлення необхідно вбудувати у носій, приймається рішення або про пропуск кінця рядку, або про дописування символу пробілу у кінець рядку.

Після цього модифікований подібним чином рядок вписується до результуючого файлу.

Функціональний фрагмент програмної реалізації даного алгоритму демонструється далі:

```
if (encoding)
{
      // отримується старший біт поточного символу повідомлення
      if ((c & 1) == 1)
      // вбудовування одиничного біту
currentLine += "  " ; // подвійний пробіл у кінці рядку кодує 1
      else
      // вбудовування нульового біту
      currentLine += " " ; // одинарний пробіл у кінці рядку
відповідає 0

      // перехід до наступного біту повідомлення
      c = c >> 1;
}
```

Графічна інтерпретація алгоритму наводиться рис. 4.3.

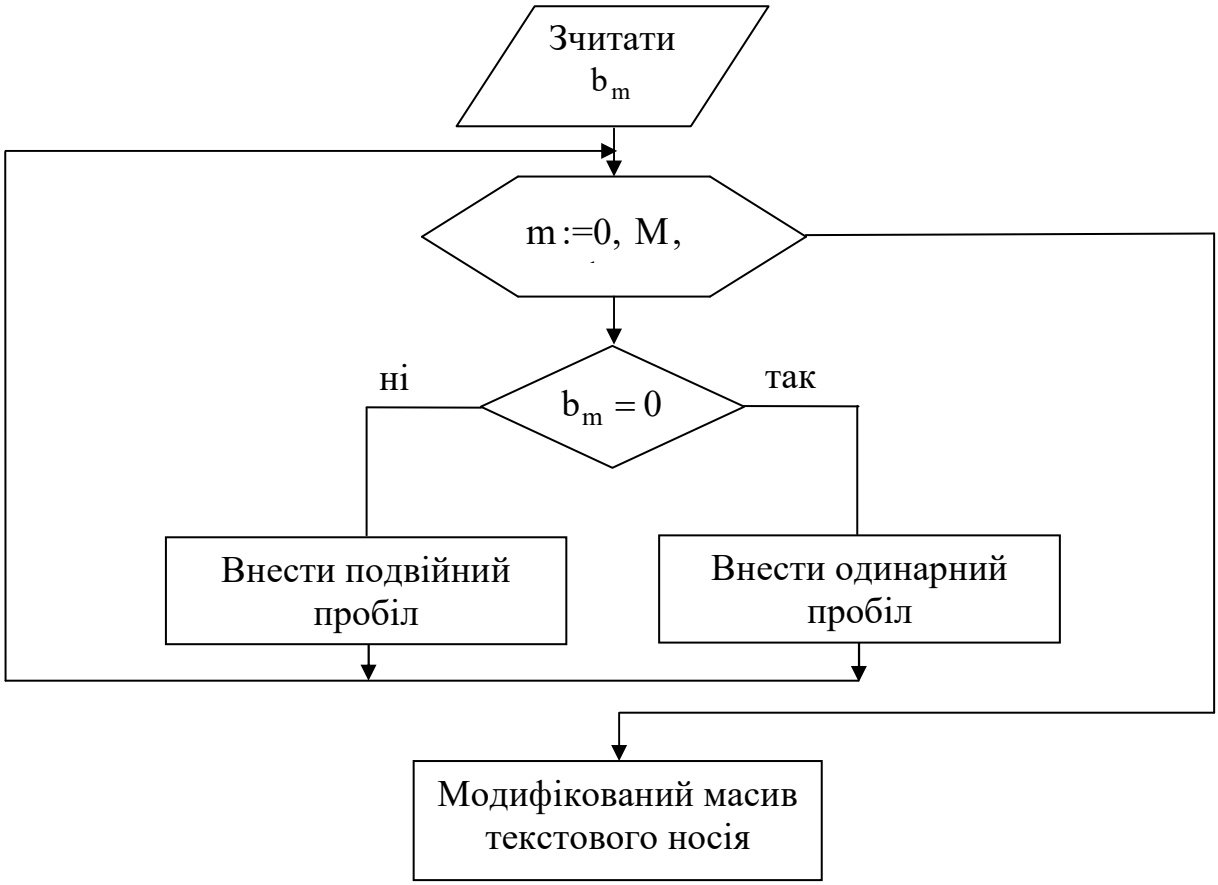

Рисунок 4.3 – Схема алгоритму «хвостових пробілів»

У свою чергу, у процесі виокремлення приховуваних даних зі стегоконтейнеру, зчитування файлу-носія також виконується порядково. При цьому, поточний біт вбудованого повідомлення буде відновлено залежно від

наявності, чи, навпаки, відсутності додаткового символу пробілу і кінці рядку.

При цьому, функціональний фрагмент програмної реалізації тієї частини алгоритму, що відповідає за відновлення інкапсульованих даних у точці прийому, демонструється далі:

```
// декодування біту
if (clLen >= 2) // перевірка ймовірності наявності кодованого
біту (а саме, одного чи двох символів пробілу)
{
     if (currentLine[clLen - 2] == ' ' && currentLine[clLen - 1]
== ' ' )
     {
     // вилучення одиниці
     c = c | 0x80; // 1000 0000

     decoding = true ;
     }
     else if (currentLine[clLen - 1] == ' ' )
     {
     // вилучення нуля
     c = c & 0x7F; // 0111 1111
     decoding = true ;
     }
     else
     decoding = false ;
}
```

До переваг алгоритму хвостових пробілів можемо віднести:

1. Простоту реалізації, та, відповідно, потенційно високу швидкодію.

2. Можливість застосування на базі великої кількості типів носіїв, у першу чергу – веб-орієнтовані текстові файли;

3. Відносно високу захищеність прихованого контенту за рахунок:

- відсутності у широкому застосунку відповідних алгоритмів викриття текстових стеганограм;

- відсутності чіткого зв’язку присутності «зайвих» пробілів у кінці рядку з наявністю заповненого контейнеру.

Останній фактор з зазначених у пункті 3 переваг потребує додаткового пояснення та уточнення.

Так, беручи до уваги специфіку верстки веб-документів, можемо зазначити наступне.

По-перше, з одного боку, ряд веб-документів зайвих пробілів не будуть мати апріорі. Це стосується:

- html та php-файлів, що генеруються динамічно, як результат обробки клієнтського запиту backend-обробником на рівні серверу; такі файли формально не мають зайвих пробілів у тексті (у т.ч. у кінці рядків);

- шаблони CMS.

З іншого боку, така закономірність є відсутньою для усіх файлів, що апріорі заповнюються верстальником, а саме:

- css-таблиці;

- контентні складові документів.

Наприклад, у рамках шаблонів CMS це відповідає (на прикладі будь-якої теми WordPress) файлам:

- single.php;

- page.php;

- header.php;

- footer.php;

- sidebar.php.

Тобто, алгоритм може бути реалізовано на базі даних файлів, розглядаючи їх у якості носіїв.

Разом з тим, до недоліків методу можемо віднести наступне:

- низький рівень робастності; це зумовлено можливістю доступу до файлів-контейнерів верстальника/адміністратора сайту, які можуть вносити зміни у файли;

- суттєва залежність потенційно можливої ємності контейнеру від структури текстового файлу, а саме – кількості рядків у цьому.

У середньому, як свідчить існуюча сьогодні статистика, ємність носія на випадок реалізації стегоалгоритма «хвостових пробілів» у середньому становить не більш, ніж 0.15%. Тобто, ємність є занизькою.

4.5.3 Алгоритм на базі використання візуально однакових символів

В основі даного алгоритму знаходиться той факт, що ряд кириличних символів, а також символів латиниці є візуально аналогічними.

При цьому, зрозуміло, що кожному з таких символів відповідають різні коди символьних таблиць.

Тобто, алгоритм на базі використання візуально однакових символів передбачає або виконання заміни (наприклад, для інкапсуляції одиничного

біту), або нехтування такою заміною (для вбудовування нульового біти) кириличних символів їх візуальними аналогами з латиниці. Або навпаки.

У свою чергу, реалізація даного методу потребує попереднього створення т.з. таблиці замін, тобто, попереднього встановлення відповідності між символами кирилиці та латиниці.

При цьому, така таблиця може застосовуватися як для заміни латинських символів символами кирилиці, так і для побудови зворотної заміни.

Приклад формування такої таблиці відповідності символів у коді Python для випадку, коли носієм є текстовий файл на латиниці, демонструється далі:

```
// формуємо таблицю замін з англійської мови на українську
exEngTable.Add( 'a' , 'а' );
exEngTable.Add( 'e' , 'е' );
exEngTable.Add( 'k' , 'к' );
exEngTable.Add( 'o' , 'о' );
exEngTable.Add( 'p' , 'р' );
exEngTable.Add( 'c' , 'с' );
exEngTable.Add( 'y' , 'у' );
exEngTable.Add( 'x' , 'х' );
exEngTable.Add( 'A' , 'А' );
exEngTable.Add( 'B' , 'В' );
exEngTable.Add( 'E' , 'Е' );
exEngTable.Add( 'K' , 'К' );
exEngTable.Add( 'M' , 'М' );
exEngTable.Add( 'O' , 'О' );
exEngTable.Add( 'P' , 'Р' );
exEngTable.Add( 'C' , 'С' );
exEngTable.Add( 'Y' , 'У' );
exEngTable.Add( 'X' , 'Х' );
```

У свою чергу, для реалізації процесу інкапсулювання секретного повідомлення, його зміст попередньо трансформується до двійкового формату представлення, після чого виконується посимвольне зчитування бінарних даних.

При цьому, розробник попередньо має встановити ***правило заміни символа текста-носія***. Наприклад, може бути реалізовано один з наведених далі підходів, а саме:

- заміна символа кирилиці/латиниці на символ, візуально аналогічний вихідному, відповідає інкапсуляції 1, якщо потрібно вбудувати 0 – заміна не виконується;

- заміна символа кирилиці/латиниці на символ, зовні ідентичний вихідному, відповідає випадку вбудовування 0, якщо ж необхідно закодувати 1 – заміна не здійснюється.

Фрагмент коду Python, який здійснює заміну символів, буде наступним:

```
// зчитування символу
currentChar = ( char )sr.Read();
if (exEngTable.ContainsKey(currentChar)) // можлива заміна
{
// якщо необхідно вбудувати символ повідомлення
if (k == 0)
{
encoding = false ;
…
if ( encoding )
{
if ((c & 1) == 1)
// інкапсулюємо "одиницю"
{
currentChar = ( char )exEngTable[currentChar];
}
else
// інкапсулюємо "нуль"
{
// зміна не виконується
}
// переходимо на наступного біту
c = c >> 1;
}
…
} // exEngTable.ContainsKey(currentChar)
```

Далі, у процесі виокремлення прихованого повідомлення, файл-носій зчитується посимвольно, але не на бінарному рівні, а у форматі представлення тексту (Ascii, Unicode тощо).

При цьому, для кожного символу тексту виконується перевірка:

- факту його належності до таблиці замін;

- якщо символ є присутнім у таблиці замін, чи було виконано процедуру заміни символу на візуально аналогічний, але з латиниці/кирилиці.

Відповідний фрагмент коду наводиться далі:

```
// зчитування поточного символу
currentChar = ( char )sr.Read();

if (exEngTable.ContainsKey(currentChar) ||
exEngTable.ContainsValue(currentChar))
{
// декапсуляція біту
if (exEngTable.ContainsValue(currentChar))
{
// декапсуляція одиничного біту
c = c | 0x80; // 1000 0000
decoding = true ;
}
else if (exEngTable.ContainsKey(currentChar))
{
// декапсуляція нульового біту
c = c & 0x7F; // 0111 1111
decoding = true ;
}
else
decoding = false ;
…
// перехід до наступного біту
c = c >> 1;
} // char.IsLetter
```

Попередній аналіз розглянутого алгоритму використання візуально однакових символів дозволяє зробити висновки про переваги та недоліки даного методу.

Так, до переваг методу можемо віднести:

- простоту реалізації та суттєву швидкодію алгоритму;

- відносно високу захищеність методу, як наслідок незначної поширеності методів стегоаналізу, орієнтованих на текстові носії;

- суттєво вищу, порівняно з алгоритмом «хвостових пробілів» ємність стегосистеми – більше, ніж у 20 разів (3,2% проти 0,15%).

Разом з тим, недоліками методу можемо вважати:

- можливість імплементації алгоритму лише у середовищах, де не виконується перевірка орфографії;

- суттєва залежність фактичної ємності стегосистеми від особливостей змісту тексту-носія.

Таким чином, найбільш прийнятними для використання у якості носія у даному випадку є текстові файли веб-середовища.

4.5.4 Алгоритм заміни коду символа пробіла

Даний алгоритм базується на тому факті, що, зокрема, символ пробілу може бути представлено у межах текстового файлу з використанням різних числових кодів.

Зокрема, наприклад, у рамках Windows-1251 передбачається, що для кодування символу пробілу можуть використовуватися числові коди 32 та 160.

Це, у свою чергу, дає змогу реалізації стегоканалу, використовуючи толй чи інший числовий код пробілу.

Наприклад, використання коду 32 може бути задіяно для інкапсуляції символу 0 секретного повідомлення.

Тоді функціональна частина коду, яка реалізує дану процедуру, буде наступною:

```
// зчитування символу
currentChar = ( char )sr.Read();
if (currentChar == ' ' )
{
…
if ( encoding )
{
// отримання біту повідомлення
if ((c & 1) == 1)
// інкапсулюється "одиниця"
currentChar = ( char )160; // пробіл з кодом 160 вбудовує 1
else
// інкапсулюється "нуль"
currentChar = ( char )32; // пробіл з кодом 32 вбудовує 0
// перехід до наступного біту
c = c >> 1;
}
…
} // currentChar == ' '
```

У свою чергу, процес виокремлення стеганограми з тексту-носія є досить простим та ілюструється наступним фрагментом коду:

```
// зчитується символ
currentChar = sr.Read();
if (currentChar == 32 || currentChar == 160)
{
// дакапсуляція біту
if (currentChar == 160) // пробел с кодом 160
{
// декапсуляція "одиниці"
c = c | 0 x 80; // 1000 0000
decoding = true ;
}
else if (currentChar == 32) // пробіл з кодом 32
{
// декапсуляція "нуля"
c = c & 0x7F; // 0111 1111
decoding = true ;
}
else
decoding = false ;


…
// перехід до наступного біту
c = c >> 1;
}
```

У свою чергу, до переваг розглянутого алгоритму можемо віднести:

- як і для попередньо розглянутих методів, на сьогодні – відносно великий рівень захищеності стеганограм;

- найвищий рівень ємності стегосистеми серед усіх розглянутих раніше – так, ймовірність появи пробілу для української мови становить близько 17%;

- для реалізації алгоритму може бути використано фактично будь-які файли текстового типу – txt, doc, rtf, css, php, html, ini, cfg, hlp тощо.

До недоліків методу можемо віднести відсутність гарантування вимогам робастності, оскільки для цього файл, що застосовується у межах стегоканалу, як носій, не повинен редагуватися до моменту зчитування з нього інкапсульованих даних.

4.6 Висновки за розділом

Виконано аналіз найбільш дієвих алгоритмів приховування секретних даних на базі текстових носіїв, а саме:

- алгоритму «хвостових пробілів»;

- алгоритму візуально однакових символів;

- алгоритму заміни коду символа пробіла;

- алгоритму зміни порядку розміщення маркерів кінця рядку.

Зазначені методи було проаналізовано з позиції їх відповідності таким показникам, як:

- рівень захищеності утвореного стегоканалу;

- ємність стегосистеми;

- діапазон файлів, які можуть бути використаними у якості носіїв.

За результатами виконаного аналізу виходить, що:

- найбільшу кількість типів файлів-носіїв може використовувати алгоритм заміни коду символу пробілу;

- алгоритму заміни коду символу пробілу відповідає також найвища ємність стегосистеми.

ВИСНОВКИ

У відповідності з технічним завданням було виконано:

- дослідження існуючих напрямків реалізації стеганографічних алгоритмів, а також специфіки класичних методів побудови прихованих каналів інформаційної взаємодії;
- аналіз класичних стеганографічних методів з метою виявлення їхніх переваг та недоліків;
- розробку шляхів удосконалення одного з методів приховування даних;
- дослідження стеганографічних алгоритмів, орієнтованих на використання носіїв текстового типу.

При цьому, у рамках виконання першого завдання було виявлено, що:

- у сьогоднішніх реаліях існує суттєвий відсоток мережевих сервісів систем та додатків, які є об’єктами інтересу для зловмисників а отже – потребують першочергового захисту;
- існуючі стеганографічні системи, незалежно від алгоритму приховування даних, на базі якого їх побудовано, мають однакову архітектуру;
- найбільш прийнятними у ролі носіїв є типи даних/файли, що характеризуються найвищими рівнями надмірності.

При цьому, так як найвищий рівень надмірності відповідає графічним та відео файлам, у рамках рішення другого завдання виконувався аналіз стегоалгоритмів, що використовують носії графічного типу. Зокрема, було досліджено метод «останнього біту» та метод нерівномірних відрізків.

За результатами виконаного дослідження можемо стверджувати, що:

- стійкість від викриття стеганограм, побудованих за класичними сценаріями реалізації з використанням досліджених методів стеганографічного приховування є залежними від способу обходу контейнеру (метод останнього біту) та алгоритму вибору стартової позиції (метод нерівномірних інтервалів);
- в поточній реалізації досліджені методи не можуть вважатися стійкими до алгоритмів стеганографічного аналізу.

У свою чергу, у рамках завдання з розробки шляхів удосконалення одного з методів приховування даних було виконано:

- побудову контентно-залежного алгоритму вибору стартового блоку для реалізації алгоритму нерівномірних інтервалів;

- розробку механізму вибору стартової позиції для вбудовування першого біту даних у межах попередньо обраного блоку.

Водночас, у ході дослідження стеганографічних алгоритмів, орієнтованих на використання носіїв текстового типу, було досліджено:

- метод хвостових пробілів;
- метод використання візуально подібних символів;
- метод зміни коду пробілу;
- метод маніпуляції черговістю розміщення символів кінця рядку.

За результатами дослідження можна стверджувати, що найбільш стійким до викриття є метод використання візуально подібних символів. Для даного методу, а такод методу хвостових пробілів побудовано кодову реалізацію.

Таким чином, усі завдання виконано у повній мірі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Microsoft security report [Електронний ресурс] – Режим доступу: https://microsoft.com/securityinsights.
2. Кибератаки – определение, виды, профилактика [Електронний ресурс] – Режим доступу: https://techarks.ru/category/security/ Кибератаки – определение, виды, профилактика.html.
3. Противодействие целевым атакам (APT) и угрозам 0 дня (Zero Day) – Octava.ua [Електронний ресурс] – Режим доступу: https://octava.ua/services/products-and-solutions/protivodejstvie-zero-day-atakam.
4. Can we test ATP defenses even if we can’t agree on how to define APTs? [Електронний ресурс] – Режим доступу: https://news.sophos.com/en-us/2015/10/23/can-we-test-apt-defenses-even-if-we-cant-agree-on-how-to-define-apts.
5. APT infographics [Електронний ресурс] – Режим доступу: https://sophos.files.wordpress.com/2014/04/apt-infographic_wr.pdf.
6. Конахович Г. Ф., Пузыренко А. Ю. Компьютерная стеганография. Теория и практика. - К.: МК-Пресс, 2006. - 288 с
7. Шаньгин В.Ф. Информационная безопасность и защита информации. ДМК-Пресс., 2017, 702 с.
8. Шелухин О.И., Канаев С.Д. Стеганография. Алгоритмы и программная реализация. Горячая линия – Телеком, научно-техническое издательство 2017, 592 с.
9. Рябко, Б.Я. Основы современной криптографии и стеганографии [Электронный ресурс] : [монография] / А.Н. Фионов, Б.Я. Рябко. — М. : Горячая линия – Телеком, 2010 .— 233 с.
10. Гизунов Д.С. Методика автоматизированного обнаружения скрытой информации в компьютерных файлах / Д.С. Гизунов, О.А. Демченко, Е.И. Никутин // Известия ТРТУ. – 2006. – Т. 71, № 16. – С. 49-53.
11. Provos N. Detecting steganographic content on the internet / N. Provos, P. Honeyman. // Technical Report CITI 01-1a, University of Michigan, 2001.
12. Юренский П.В. МЕТОДЫ СТАТИСТИЧЕСКОГО И НЕЙРОСЕТЕВОГО СТЕГОАНАЛИЗА СКРЫТЫХ КАНАЛОВ // Инновации в науке: научный журнал. – № 1(89). – Новосибирск., Изд. АНС «СибАК», 2019. – С. 11-13.

13. Голуб В.А. Комплексный подход для выявления стеганографического скрытия в JPEG-файлах / В.А. Голуб, М.А. Дрюченко // Инфокоммуникационные технологии. – 2009. – Т. 7, № 1. – С. 44-50
14. Грибунин В. Г., Оков И. Н., Туринцев И. В. Цифровая стеганография. М.: СОЛОН-Пресс, 2016, - 315 с.
15. VNI Forecast Highlights Tool [Електронний ресурс] – Режим доступу: https://www.cisco.com/c/m/en_us/solutions/service-provider/vni-forecast-highlights.html
16. Кустов В.Н., Параскевопуло А.Ю. Простые тайны стегоанализа / В.Н. Кустов, А.Ю. Параскевопуло // Защита информации, INSIDE. – 2005. – № 4. – С. 72-78.
17. Быков С. Ф. Алгоритм сжатия JPEG с позиции компьютерной стеганографии Защита информации. Конфидент. - СПб.: 2000, № 3
18. Bejtlich R. The Practice of Network Security Monitoring: Understanding Incident Detection and Response / Richard Bejtlich. – San Francisco: Search Press Inc, 2013. – 341 c.
19. Fridrich Y. Steganography in Digital Media: Principles, Algorithms and Applicaticks. Cambridge Press, 2010. 462 p.
20. S. Chaudhary, M. Dave and A. Sanghi, "Aggrandize text security and hiding data through text steganography," 2016 IEEE 7th Power India International Conference (PIICON), Bikaner, India, 2016, pp. 1-5, doi: 10.1109/POWERI.2016.8077346.
21. A.P. Singh. An Acquaintance to Text-Steganography and its Methods. 2021 J. Phys.: Conf. Ser. 1950 012005.
22. Maini, Anil Kumar. Digital Electronics: Principles, Devices and Applications. John Wiley and Sons. p. 28. ISBN 978-0-470-03214-5.
23. Vint Cerf. ASCII format for Network Interchange. IETF. doi:10.17487/RFC0020. RFC 20.