МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

На правах рукопису
УДК 003.26:004.056.5

КІНЗЕРЯВИЙ ОЛЕКСІЙ МИКОЛАЙОВИЧ

СТЕГАНОГРАФІЧНІ МЕТОДИ ПРИХОВУВАННЯ ДАНИХ У ВЕКТОРНІ ЗОБРАЖЕННЯ, СТІЙКІ ДО АКТИВНИХ АТАК НА ОСНОВІ АФІННИХ ПЕРЕТВОРЕНЬ

Спеціальність 05.13.21 – Системи захисту інформації

Дисертація на здобуття наукового ступеня
кандидата технічних наук

Науковий керівник:
Ковтун Владислав Юрійович
кандидат технічних наук,
доцент кафедри безпеки
інформаційних технологій ІПДС НАУ

Київ – 2015

ЗМІСТ

ВСТУП...……..	5
Розділ 1. СТЕГАНОГРАФІЧНІ МЕТОДИ ЗАХИСТУ ІНФОРМАЦІЇ……...	11
1.1. Організація стеганографічного захисту інформації резервних каналів зв’язку…….	11
1.2. Аналіз стеганографічних методів приховування інформації у векторні зображення……	17
1.3. Аналіз активних атак на стеганосистеми з векторними зображеннями……..…….	27
1.4. Постановка задачі дослідження.....…….	34
1.5. Висновки до першого розділу…….	39
Розділ 2. МЕТОДИ ПРИХОВУВАННЯ ІНФОРМАЦІЇ У ВЕКТОРНІ ЗОБРАЖЕННЯ……	41
2.1. Принцип вбудовування інформації у векторні зображення……...	41
2.2. Параметри приховування інформації у векторні зображення……...	44
2.3. Метод побітового приховування інформації у векторні зображення……...	48
2.4. Метод шаблонного приховування інформації у векторні зображення……...	51

2.5. Розробка структурної моделі процесу прихованої передачі інформації резервним каналом зв’язку……..	55
2.5. Висновки до другого розділу……...	61
Розділ 3. АЛГОРИТМИ ПРИХОВУВАННЯ ІНФОРМАЦІЇ У КРИВІ БЕЗ’Є ВЕКТОРНИХ ЗОБРАЖЕНЬ…….	63
3.1. Дослідження типів кривих векторної графіки…….....	63
3.2. Процес вбудовування даних у криві Без’є за побітовим методом приховування інформації ……...……	67
3.3. Процес вбудовування даних у криві Без’є за шаблонним методом приховування інформації …….……..	78
3.4. Висновки до третього розділу…….	87
Розділ 4. ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ РОЗРОБЛЕНИХ РІШЕНЬ…….	89
4.1. Методика проведення експериментального дослідження……	89
4.2. Розробка програмного забезпечення для проведення експериментів……...	92
4.3. Дослідження швидкісних характеристик, стійкості до афінних перетворень, коефіцієнту візуального спотворення та розмірів стеганоконтейнерів запропонованих алгоритмів	98
4.4. Висновки до четвертого розділу……	137
ВИСНОВКИ……	139
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ……	141

Додаток А. Документи, що підтверджують впровадження результатів дисертації……....	153
Додаток Б. Результати експериментального дослідження стійкості алгоритмів StegoBIT і StegoTEMPL до афінних перетворень	156
Додаток В. Лістинги програмних засобів……..	186

ВСТУП

Актуальність. Розвиток глобальної мережі Інтернет та поширення її використання серед населення планети, сприяє збільшенню обсягів інформації, що передається, обробляється, зберігається та знищується. Використовуючи можливості і ресурси мережі Інтернет можна організувати резервний канал зв'язку, наприклад, з дипломатичними установами, що знаходяться на території іноземних держав. Скритність передачі інформації по такому каналу буде забезпечуватися стенографічними засобами захисту [1, 9, 30, 36, 38, 42, 46, 58, 60, 62, 64, 66-69, 79, 88]. Основна відміна стеганографії від інших методів захисту інформації, полягає саме у прихованні факту існування секретного повідомлення в іншому, не привертаючому уваги об'єкті – контейнері [1, 9, 30, 34, 36, 38, 46, 56, 66-69, 79, 88], використовуючи для цього структурні особливості побудови самого контейнера та властивості органів сприйняття людини.

Значний вклад в розвиток стеганографії в Україні й на пострадянському просторі внесли А.В. Аграновський, В.Г. Грибунін, В.К. Задірака, Є.А. Золотовкін, А.А. Кобозєва, Г.Ф. Конахович, О.О. Кузнецов, В.В. Лукічов, І.І. Маракова, О.А. Смірнов, В.О. Хорошко, М.Є. Шелест, Ю.Є. Яремчук та інші. Серед закордонних науковців варто згадати I. Cox, Y. Li, N. Nikolaidis, R. Ohbuchi, I. Pitas, V. Solachidis, M. Voigt та інших.

Більшість сучасних стеганографічних досліджень та методів присвячені приховуванню інформації в графічних зображеннях [12, 30, 36, 41, 45, 60, 62]. Враховуючи той факт, що шлюзи доступу в мережу Інтернет знаходяться під контролем спецслужб розвинутих країн, то при передачі зображень можуть застосовуватися активні фільтри, які візуально непомітно модифікують їх і таким чином знищують приховане повідомлення. До таких атак слід віднести різного роду трансформації на основі афінних перетворень [3, 16, 36, 96]. Забезпечити стійкість до афінних перетворень можливо завдяки використанню в якості контейнерів векторних зображень, які за своїми властивостями і

принципами побудови дозволяють будувати зображення з досить високою якістю [37, 43].

Проведений аналіз сучасних стеганографічних методів приховування інформації у векторні зображення показав слабку стійкість їх до афінних перетворень. Крім того, існуючі методи базуються на просторових та частотних перетвореннях векторних зображень, що внаслідок зміни їх контурів чи положень координат точок призводять до погіршення якості самого зображення та утворення видимих відхилень ліній векторних об’єктів.

Таким чином, *актуальним науковим завданням*, що має теоретичне і практичне значення, є розробка нових та удосконалення існуючих стеганографічних методів приховування інформації у векторні зображення, з метою підвищення стійкості до активних атак на основі афінних перетворень.

Зв’язок роботи з науковими програмами, планами, темами. Дисертаційні дослідження виконувалися в рамках «Основних наукових напрямів та найважливіших проблем фундаментальних досліджень у галузі природничих, технічних і гуманітарних наук на 2009–2013 роки» (затверджених наказом МОН України та НАН України № 1066/609 від 26.11.2009), держбюджетної науково-дослідної роботи «Організація систем захисту інформації від кібератак» (№ 0111U000171), «Методи та моделі стеганографічного захисту інформації від кібератак» (№ 101/14.01.06), «Методи забезпечення конфіденційності державних інформаційних ресурсів в інформаційно-комунікаційних системах» (№ 61/09.01.08), що проводились за планами НДР Національного авіаційного університету.

Мета і завдання дослідження. Метою роботи є підвищення стійкості стеганографічного захисту інформації на основі застосування нових методів приховування даних у векторні зображення, стійких до активних атак на основі афінних перетворень.

Для досягнення даної мети необхідно розв’язати такі *основні завдання*:

– провести аналіз сучасних стеганографічних методів приховування інформації у векторні зображення;

– формалізувати вимоги до вибору контейнера та визначити параметри приховування інформації у векторні зображення;

– розробити метод побітового приховування інформації у точково-задані криві векторних зображень, що забезпечує стійкість до активних атак на основі афінних перетворень;

– розробити метод шаблонного приховування інформації з визначеною таблицею співвідношень значень елементів шаблону різним крокам побудови точково-заданих кривих векторних зображень, що забезпечує стійкість до активних атак на основі афінних перетворень;

– розробити структурну модель процесу прихованої передачі інформації резервним каналом зв'язку;

– розробити алгоритми приховування інформації у криві Без'є третього ступеня векторних зображень;

– на основі запропонованих алгоритмів розробити програмне забезпечення (ПЗ) з метою їх верифікації.

Об'єктом дослідження є процес приховування інформації у векторні зображення.

Предметом дослідження є стеганографічні методи приховування інформації у точково-задані криві векторних зображень, що забезпечують стійкість до активних атак на основі афінних перетворень.

Методи дослідження. Проведені дослідження базуються на теоретико-множинному підході до розробки структурної моделі процесу прихованої передачі інформації резервним каналом зв'язку; для оцінки якісних показників методів приховування інформації у точково-задані криві векторних зображень використовуються чисельні методи, методи обчислювальної лінійної алгебри, матричного і статистичного аналізу; об'єктно-орієнтованого програмування (розробка ПЗ для експериментального дослідження) тощо.

Наукова новизна одержаних результатів полягає в такому:

– *вперше* визначено множину параметрів приховування даних у векторні зображення, які, за рахунок врахування особливостей побудови

векторних зображень (ступеня точково-заданих кривих, їх допустимої довжини відносно опорних точок) та стеганографічних перетворень (точність координат опорних точок, допустимої похибки округлення при вилученні даних та кількості інформації, що приховується в одну криву), дозволяють формалізувати вимоги до вибору контейнерів та впливати на процес приховування інформації у точково-задані криві;

– *вперше* розроблено метод побітового приховування інформації у точково-задані криві векторних зображень, який, за рахунок впливу послідовності даних на процес сегментації кривих з фіксованим кроком зміни параметра побудови заданих кривих (розбиття кривих на сегменти відбувається лише при вбудовуванні нульового/одиничного біта приховуваної послідовності даних), забезпечує високу швидкодію приховування, вилучення секретного повідомлення та підвищує стійкість до активних атак на основі афінних перетворень;

– *вперше* розроблено метод шаблонного приховування інформації у точково-задані криві векторних зображень, який, за рахунок впливу послідовності даних на процес сегментації кривих згідно визначеної таблиці співвідношень значень елементів шаблону різним крокам зміни параметра побудови заданих кривих (при розбитті кривої на два сегменти вбудовується блок даних), на відміну від побітового методу, дозволяє зменшити розміри стеганоконтейнерів, підвищити швидкість вбудовування та стійкість до активних атак на основі афінних перетворень.

Практичне значення одержаних результатів. Практична цінність роботи полягає в наступному:

– розробці структурної моделі процесу прихованої передачі інформації резервним каналом зв’язку, що дозволяє формувати множину стеганоконтейнерів, стійких до афінних перетворень.

– розробці двох нових стеганографічних алгоритмів приховування інформації у криві Без’є третього ступеня, що можуть бути використані для підвищення стійкості до активних атак на основі афінних перетворень;

– розробці методики проведення експериментального дослідження, що дозволяє оцінити ефективність приховування секретного повідомлення запропонованими методами;

– розробці програмних засобів для проведення експериментальних досліджень запропонованих рішень.

Результати дисертаційної роботи впроваджено у навчальному процесі кафедри безпеки інформаційних технологій Національного авіаційного університету (від 30.06.2015 р.) та у науково-технічних розробках ТОВ «Сайфер ЛТД» (від 23.06.2015 р.), «Каскад Груп Україна» (від 19.02.2015 р.), що підтверджено відповідними актами впровадження.

Особистий внесок здобувача. Всі результати, які складають основний зміст дисертаційної роботи, отримано здобувачем самостійно. У роботах, написаних у співавторстві, автору належать: [26, 87] – розробка методу шаблонного приховування інформації з визначеною таблицею співвідношень значень елементів шаблону різним крокам побудови точково-заданих кривих; [29, 63] – розробка методу побітового приховування інформації у точково-задані криві векторних зображень; [20-22, 24] – експериментальне дослідження стійкості побітового та шаблонного методу до атак на основі афінних перетворень, визначення вимоги до вибору контейнера, основних етапів та параметрів приховування інформації у векторні зображення; [30] – аналіз сучасних стеганографічних методів захисту інформації; [4, 27] – розробка принципів побудови нових блокових шифрів для їх використання в процесі вбудовування інформації.

Апробація результатів дисертації. Результати дисертаційної роботи доповідались та обговорювались більш ніж на 15 науково-технічних конференціях, серед яких: науково-технічна конференція «Безпека інформаційних технологій (ITSEC)» (Київ, 2012-2014 р); міжнародна науково-практична конференція «Інтегровані інтелектуальні робототехнічні комплекси (ІРТК)» (Київ, 2012-2014 р.); міжнародна науково-практична конференція «Інфокомунікації - сучасність та майбутнє» (Одеса, 2013 р., 2014 р.); науково-

практична конференція «Механізми управління безпекою підприємств в сучасних умовах господарювання» (Київ, 2013 р.); міжнародна науково-практична конференція «Захист інформації і безпека інформаційних систем» (Львів, 2014 р); міжнародний форум «Радиоэлектроника и молодежь в XXI веке» (Харків, 2014 р.); всесвітній конгрес «Безпека в авіації та космічні технології» (Київ, 2014 р.); науково-практична конференція «Актуальні питання забезпечення кібернетичної безпеки та захисту інформації» (Київ, 2015 р.); міжнародна науково-практична конференція «Політ. Сучасні проблеми науки» (Київ, 2015 р.); міжнародна науково-технічна конференція «Авіа-2015» (Київ, 2015 р.); міжнародна науково-практична конференція «Безпека інформації у інформаційно-телекомунікаційних системах» (Київ, 2015 р.); науково-методичні семінари кафедри безпеки інформаційних технологій Національного авіаційного університету.

Публікації. Основні положення дисертації опубліковано у 14 наукових працях, у тому числі – 7 статтях у фахових виданнях України (6 з яких входять до міжнародних наукометричних баз даних), а також 7 тезах доповідей на конференціях.

Структура роботи та її обсяг. Дисертація складається із вступу, чотирьох розділів, загальних висновків, додатків, списку використаних джерел, і має 140 сторінок основного тексту, 74 рисунків, 29 таблиць, 87 сторінок додатків. Список використаних джерел містить 109 найменувань і займає 12 сторінок. Загальний обсяг роботи − 239 сторінок.

РОЗДІЛ 1

СТЕГАНОГРАФІЧНІ МЕТОДИ ЗАХИСТУ ІНФОРМАЦІЇ

1.1. Організація стеганографічного захисту інформації резервних каналів зв'язку

В останнє десятиліття намітилася тенденція до зміни традиційного світоустрою, установленому протягом півстоліття. Перехід від однополярного, до двополярного і, в перспективі, до багатополярного світопорядку, веде до зростання напруженості в різних куточках планети. Незважаючи на зв'язані з цим локальні збройні конфлікти, велике значення приділяється їх мирному врегулюванню, яке лягає на плечі дипломатичних служб держав-посередників.

Враховуючи той факт, що питання зовнішньої політики більшості держав, знаходяться в компетенції президента або прем'єр-міністра, виникає необхідність в оперативному зв'язку дипломатичних представництв з міністерством закордонних справ.

На поточний момент, завдання забезпечення провідного і радіозв'язку дипломатичних установ лежить на підрозділах урядового зв'язку Держспецзв'язку України у взаємодії з відповідними підрозділами Міністерства закордонних справ [51-55]. На рис. 1.1, 1.2 приведенні приклади забезпечення зв'язку з різними дипломатичними установами, що знаходяться на території іноземних держав.

Для організації гарантованого захисту каналів зв'язку використовуються стаціонарні апаратні засоби криптографічного захисту інформації [60, 72, 73].

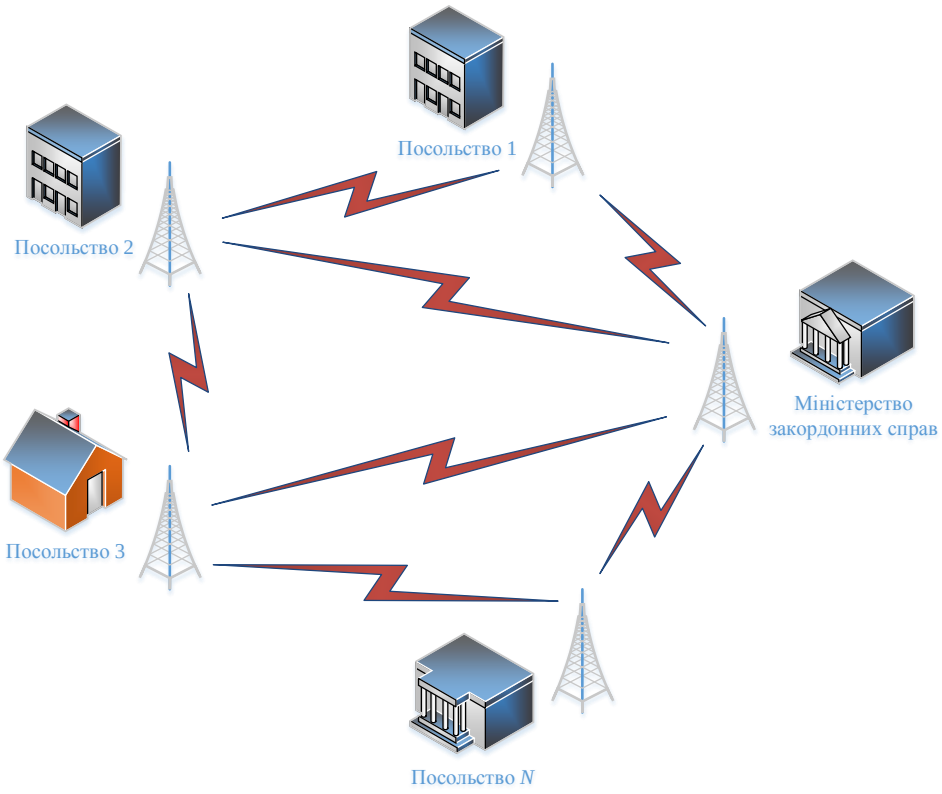


Рис. 1.1. Забезпечення радіозв’язку з дипломатичними установами на території іноземних держав

Історія ряду конфліктів (протистоянь) у країнах арабського світу, показала, що використання провідних засобів зв’язку з дипломатичними установами не завжди можливе, так як учасники конфлікту блокують роботу державних вузлів зв’язку, що забезпечують підтримку обміну інформації з дипломатичними установами [70, 71]. У таких випадках, виправданим є використання супутникового та радіоканального засобу зв’язку [60]. Однак, робота і цих засобів може бути порушена, так як антенно-фідерні пристрої можуть бути виведені з ладу фізично (дистанційно), з використанням стрілецького озброєння.

Враховуючи, що конфлікти найчастіше відбуваються за підтримки спецслужб низки розвинених країн світу, слід враховувати можливість подавлення державних засобів супутникового і радіоканального зв’язку за допомогою сучасних засобів радіоелектронної протидії. Таким чином, дипломатичне представництво країни може опинитися в повній інформаційній ізоляції.

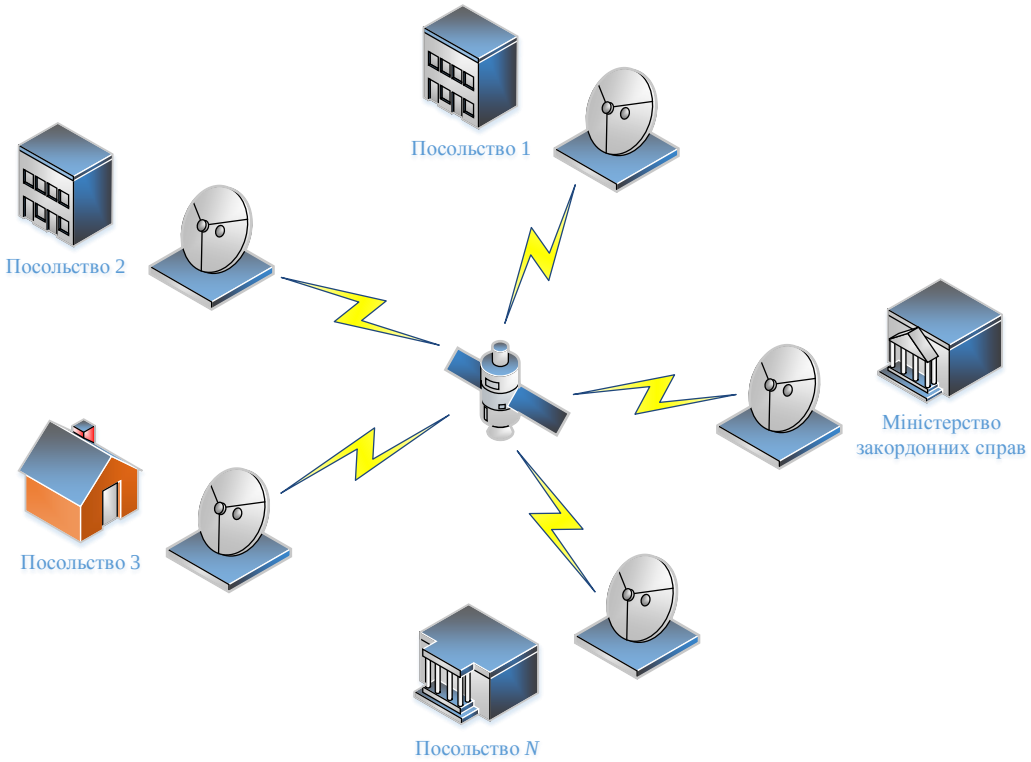


Рис. 1.2. Забезпечення супутникового зв’язку з дипломатичними установами на території іноземних держав

З іншого боку, аналіз подібного роду конфліктів показує активне використання учасниками та організаторами протистояння глобальної мережі Інтернет [47, 70, 71]: різних програмних додатків для швидкого обміну повідомлень, а також популярних соціальних мереж для координації своїх дій. Слід зазначити, що навіть при спробі влади, на території якій відбувається протистояння, закрити чи обмежити доступ до мережі Інтернет, виявлялося, що це практично неможливо [70, 71]. Це ставало можливим, в тому числі і завдяки розгорнутим рухомим базовим станціям мобільного зв’язку, за допомогою спецслужб розвинених країн, залучених до організації конфлікту.

Проведений аналіз показує, що при організації та здійсненні конфлікту на території деякої держави, доступ до глобальної мережі Інтернет є можливим за допомогою популярних мобільних пристроїв [6, 50, 60, 100]. Таким чином, для організації резервного каналу зв’язку з дипломатичним

представництвом, можна використовувати доступ до глобальної мережі Інтернет через мобільні пристрої рис. 1.3.

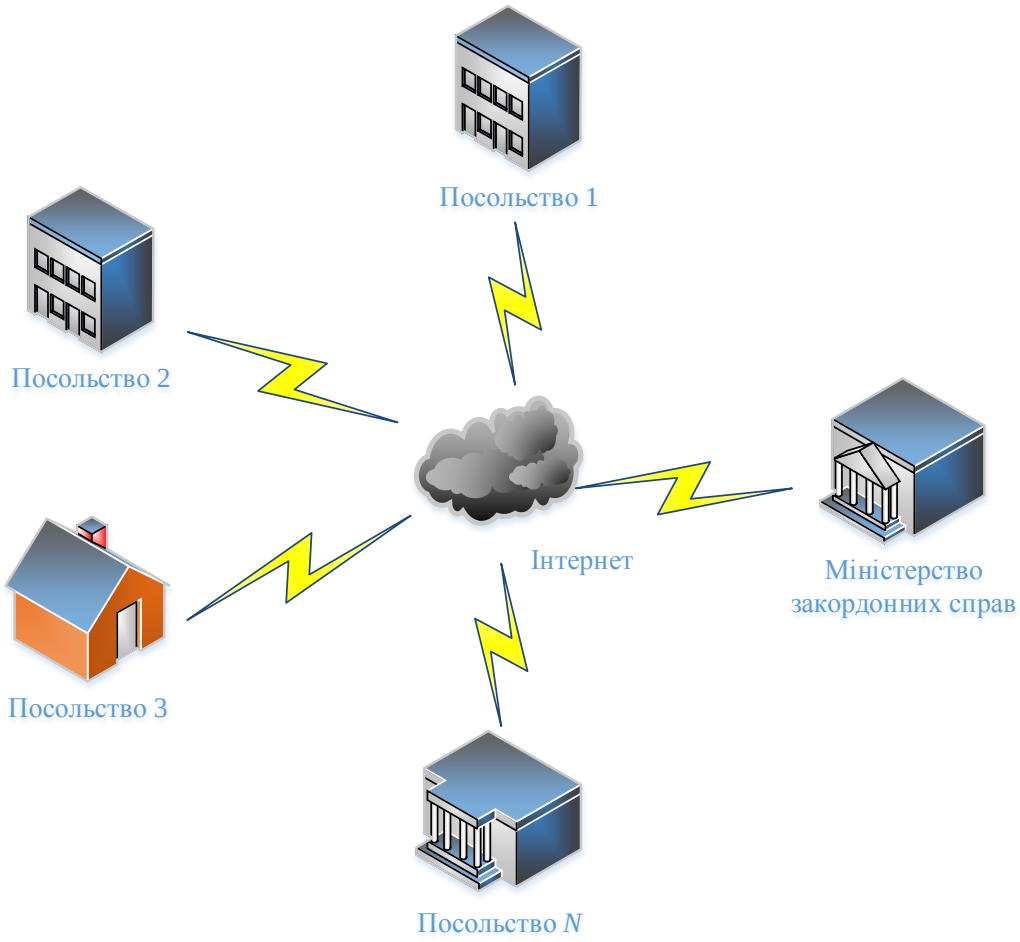


Рис. 1.3. Резервний канал зв’язку через мережу Інтернет з дипломатичними установами на території іноземних держав

Враховуючи, що організація доступу в мережу Інтернет здійснюється через підконтрольні спецслужбам шлюзи, слід звернути увагу на забезпечення не тільки конфіденційності, цілісності та авторства переданої інформації, а й забезпечення скритності передачі інформації, що ускладнить процес перехоплення повідомлень із зарубіжних дипломатичних установ.

Формалізуємо умови необхідні для організації резервного (прихованого та захищеного) каналу зв’язку з дипломатичними установами [6, 36, 47, 50, 100]:

1. Використання загальнодоступних мереж передачі інформації, таких як глобальна мережа Інтернет, можливо з відкритим доступом до деяких соціальних мереж і сервісів обміну миттєвими повідомленнями.

2. Обмеження на пропускну спроможність підключення. Наприклад, обмежившись мобільними мережами другого покоління 2G за стандартами GPRS та EDGE, що мають низьку швидкість передачі даних.

3. Використання загальнодоступних мобільних (портативних) точок доступу до мережі Інтернет, наприклад смартфон або планшет, з низькою продуктивністю.

4. Можливість моніторингу переданого трафіку, з боку учасників та спецслужб-організаторів протистояння, так і влади.

5. Можливість модифікації/втрати підозрілого вмісту, при проходженні через шлюзи доступу. Іншими словами, відбувається модифікація підозрілого вмісту повідомлення або його знищення.

При заданих умовах організації резервного каналу, скритність передачі інформації здійснюється стенографічними засобами захисту [1, 9, 30,36, 38, 42, 46, 58, 60, 62, 64, 66-69, 79, 88]. Тобто, захищене повідомлення за допомогою криптографічних засобів, вбудовується в не непримітний об’єкт – контейнер [1, 9, 30, 34, 36, 38, 46, 56, 66-69, 79, 88], який потім відкрито транспортується адресату. Одним із найпоширеніших та використовуваних контейнерів є зображення [12, 30, 36, 41, 45, 60, 62]. Це пов’язано з тим, що вони здатні приховувати досить великі обсяги даних, при цьому характер змін у зображенні може бути непомітним для людського зору [9, 36].

На рис. 1.4 наведений приклад реалізації резервного каналу зв’язку з дипломатичним представництвом через мережу Інтернет.

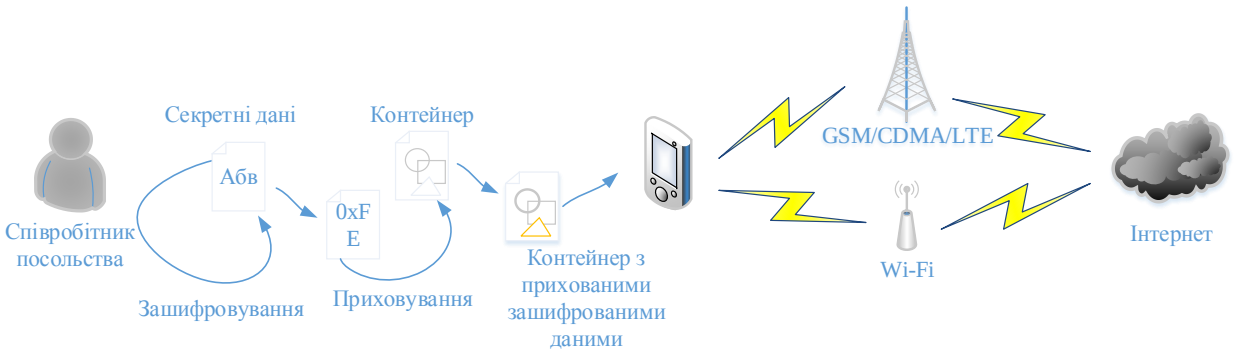


Рис. 1.4. Резервний канал передачі інформації з дипломатичним представництвом через мережу Інтернет

Однак, враховуючи той факт, що шлюзи доступу в Інтернет знаходяться під контролем спецслужб розвинутих країн, то можуть бути застосовані активні фільтри при передачі контейнерів-зображень. Такі атаки можуть візуально непомітно модифікувати саме зображення, і безповоротно знищити приховане повідомлення [3, 16, 22, 36, 96]. Наприклад, така процедура застосовується для ряду соціальних сервісів, коли, при завантаженні зображення оброблюється згідно вимог сервісу.

У зв’язку з цим, слід підібрати такі стеганографічні методи і відповідні контейнери, які б дозволили успішно протистояти більшості активних атак.

Враховуючи той факт, що обсяги інформації, які проходять через шлюзи досить значні, то швидше за все, з боку спецслужб будуть здійснюватися активні атаки, які не вимагають значних обчислювальних ресурсів.

В якості контейнерів, слід використовувати невеликі файли в популярних форматах, які не викличуть зайвих підозр щодо їх використання (наприклад, зображення з фотографіями місцевих визначних пам’яток). Файли невеликого розміру можуть легко передаватися по мережах з невисокою швидкістю передачі даних, навіть у разі їх високої завантаженості.

Проведений огляд популярності графічних форматів в мережі Інтернет [83], показав, що поряд з растровими зображеннями, починають активно використовуватися і векторні зображення.

Проведені дослідження [2, 59], показують, що стеганографічні методи, які використовують растрові, фрактальні зображення в якості контейнерів, погано протистоять активним атакам, що не дозволяє їх використовувати для вирішення поставленого завдання. У той час як векторні зображення, успішно протистоять таким атакам.

Враховуючи зростаючу популярність векторних зображень, надалі, в роботі будуть розглядатися тільки векторні зображення.

1.2. Аналіз стеганографічних методів приховування інформації у векторні зображення

Векторні зображення використовуються при побудові повнокольорових ілюстрацій, складних креслень, логотипів, емблем тощо, де потрібне використання зображень з досить високою точністю (якістю). Вони мають ряд властивостей, а саме [56, 63]: складаються з об’єктів, що описуються аналітично; володіють великою роздільною здатністю, тобто зміна масштабу відбудеться без втрати якості, тому, що положення точок зберігається, але при цьому змінюються їх координати; векторні об’єкти легко трансформуються (існує можливість редагування кожного елементу окремо). Векторні зображення будуються за допомогою графічних примітивів (елементарних фігур), що задаються своїми характерними параметрами [43, 56]. Основним елементом в векторній графіці є лінія, якій притаманні певні особливості: форма, товщина, колір тощо [37, 43, 63]. Будь-яка лінія задається координатами точок, де точка – це об’єкт на площині чи в просторі, що відображається за допомогою проективних координат даної точки [49, 61, 63]. Векторні об’єкти завжди мають шлях, що визначає маршрут за яким з’єднується початкова та кінцева точка. Якщо шлях є

замкненим, де кінцева точка співпадає з початковою, то об’єкт має внутрішню ділянку яку можна заповнити кольором або іншими об’єктами [37].

На особливостях побудови векторних зображень базуються ряд стеганографічних методів, що дозволяють приховувати інформацію в їх структуру. За принципом вбудовування інформації дані методи поділяються на два класи: просторові та частотні (рис. 1.5) [14, 15, 17, 18, 30, 78, 84-86, 89, 91, 92, 94-96, 98, 101, 103, 104].

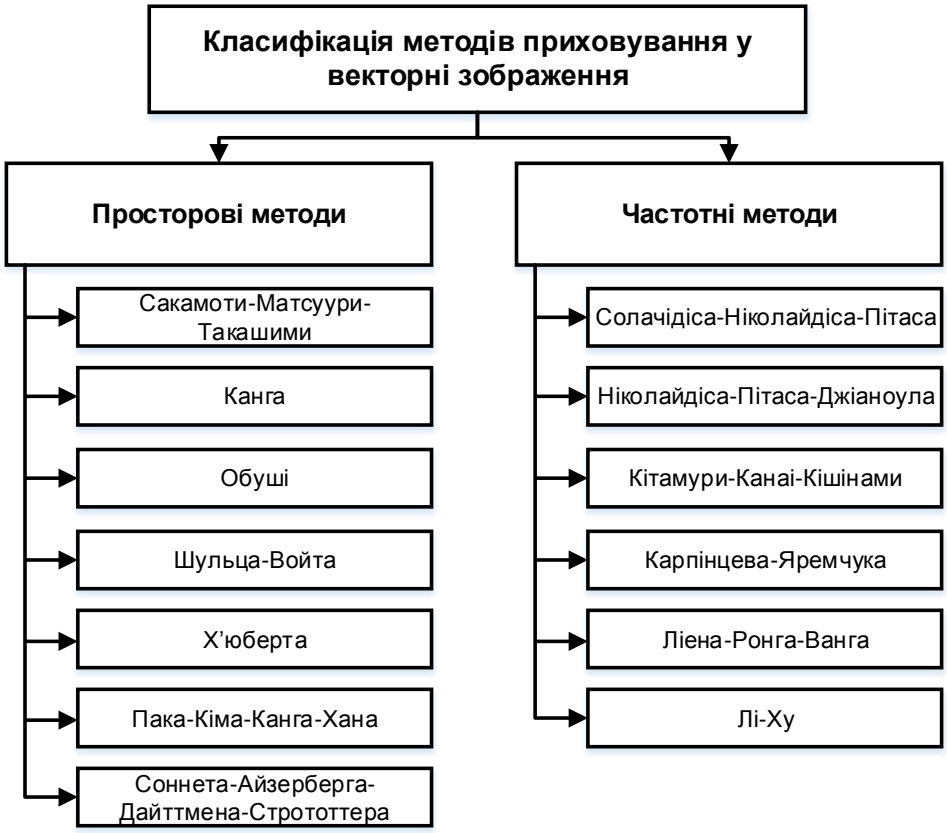


Рис. 1.5. Класифікація методів приховування інформації у векторні зображення

Просторові методи – вбудовують секретну інформацію шляхом незначної зміни абсолютних значень координат точок (зміна відстані, добавляння додаткових точок тощо) векторних зображень [18, 96]. До даного класу відносяться:

1. Метод Сакамоти-Матсуури-Такашими [101], який здійснює приховування інформації за допомогою зміни положень відстаней між

точками в зображенні. За даним методом, зображення поділяється на блоки і для кожного блоку визначається маска певного стандартного розміру. Вбудовування інформації відбувається шляхом зміни положень точок у масці при встановленні їх за відповідним зразком.

2. Метод Канга [85], який являєтеся модифікацією метода Сакамоти-Матсуури-Такашими. Відповідно даному методу зображення ділиться на блоки. Кожному блокові задається маска довільного розміру. Кожна маска поділяється на два трикутника розділених побічною діагоналлю, що іде з південного-заходу на північний-схід маски. Вбудовування біта «1» відбувається шляхом переміщення точок з нижнього трикутника у відповідні дзеркальні їх положення відносно діагоналі у верхньому трикутнику в масці, а для вбудовування біта «0» виконується протилежна послідовність дій.

3. Метод Обуші [98], який здійснює приховування інформації у зображення шляхом поділу його на прямокутні блоки з певною кількістю точок. Поділ зображення може відбуватися за допомогою: рівномірного розподілу; дерева квадрантів; модифікованого дерева квадрантів. Потім, відбувається перетворення прихованої послідовності в біполярну послідовність b_i, $i=\overline{1,N}$, де N – кількість біт. Для вбудовування даних також генерується псевдовипадкова послідовність p_i. Вбудовування одного біта b_i відбувається в один блок зображення за наступною формулою:

$$V_i^{'} = V_i + b_i \cdot p_i \cdot \alpha,$$

де V_i – початкові координати точки i блоку, $i=\overline{1,N}$, α – коефіцієнт амплітуди.

4. Метод Шульца-Войта [103], який вбудовує інформацію у векторні зображення за допомогою зміни місця розташування точок в певних смугах. За даним методом зображення ділиться на послідовність смуг висотою в $\frac{4}{3}\cdot\frac{\tau}{\sqrt{2}}$, де τ використовується для зменшення максимального

спотворення зображення. В кожній такій смузі будують дві лінії довжиною $1/3$ допустимого відхилення і позначаються відповідно, як області 0 і 1. Вбудовування даних відбувається шляхом переміщення точок в смузі з однієї області 0 чи 1 в іншу. Наприклад, якщо потрібно вбудувати біт «1», то переміщуємо точки з області 0 симетрично відносно діагоналі до області 1, а для вбудовування біта «0» виконуються протилежна послідовність дій.

5. Методи Х'юберта, Пака-Кіма-Канга-Хана, Соннета-Айзерберга-Дайттмена-Стрототтера [78, 84, 96], що дозволяють приховувати інформацію шляхом добавляння нових точок до зображення. Вбудовування інформації за методом Х'юберта відбувається шляхом вибору в декількох місцях двох суміжних точок в зображенні. Далі, між цими двома точками додаються додаткові вершини при одній з двох зафіксованих відстаней між ними. Якщо деякі точки мають одну певну відстань від попередньої, то ця відстань вказує на вбудовування біта «1», а при іншій відстані біта «0».

Основним недоліком просторових методів є недостатній рівень стійкості до активних атак застосовуваними над векторними зображеннями. Це пов'язано з низьким рівнем кореляції між сусідніми точками, що були змінені при вбудовуванні інформації, та помітністю спотворення самого зображення [18].

Частотні методи – вбудовування інформації шляхом представлення векторного зображення у певній формі за допомогою деякого аналітичного перетворення (дискретного перетворення Фур'є, дискретно косинусного перетворення та дискретного вейвлет-перетворення) над координатами його точок [18, 96]. До даного класу відносяться:

1. Метод Солачідіса-Ніколайдіса-Пітаса [94, 104], який здійснює приховування інформації у зображення за допомогою зміни частотних коефіцієнтів дискретного перетворення Фур'є. Приховування за даним методом здійснюється в ламані лінії, які складаються з N точок. Координати обраної лінії представляються у вигляді масиву точок $V = \{V_k\}$, $V_k = (X_k, Y_k)$, $k = \overline{1, N}$, де N – кількість точок ламаної лінії, X_k, Y_k – значення однієї точки.

Далі відбувається перетворення кожної пари точок з масиву V у комплексну послідовність даних (комплексний сигнал) за наступною формулою:

$$z(k) = X_k + i \cdot Y_k,$$

де i – уявна одиниця.

Над послідовністю $z(k)$ здійснюється дискретне перетворення Фур'є, в результаті чого одержується послідовність коефіцієнтів $Z(k)$, за наступною формулою:

$$Z(k) = \sum_{n=0}^{N-1} z(k)\left(\cos\left(\frac{2\pi \cdot n \cdot k}{N}\right) - i\sin\left(\frac{2\pi \cdot n \cdot k}{N}\right)\right),\ k = \overline{1, N}.$$

Стеганоключ використовується для генерації біполярної послідовності довжиною N псевдовипадкових чисел $W_0(k) \in \{-1, +1\}$, $k = \overline{1, N}$, що представляє один біт прихованої послідовності. Далі, використовуючи параметри a, b, $(0 < a < b < 1)$, які використовуються для вибору діапазону спектра при вбудовуванні даних, генерується сигнал $W(k)$ прихованого повідомлення:

$$W(k) = \begin{cases} W_0(k), a \cdot N < k < b \cdot N \vee (1-b)N < k < (1-a)N; \\ 0. \end{cases}$$

За допомогою параметрів a, b визначається число ненульових елементів сигналу $W(k)$ та число коефіцієнтів $Z(k)$, які будуть змінені.

Вбудовування одного елемента $W(k)$ відбувається на змінні $|Z(k)|$ коефіцієнтів Фур'є з використанням закону добавки або мультиплікативного вкладення:

$$|Z'(k)| = |Z(k)| + pW(k) \text{ або } |Z'(k)| = |Z(k)| + p|Z(k)|W(k),$$

де $k = \overline{1, N}$, p – показник потужності прихованого повідомлення.

Існує декілька модифікацій розглядуваного методу. Так, в роботі [95] запропоновано покращений алгоритм вбудовування інформації у зображення, який вже не потребує для вилучення даних контейнера-оригінала. Це досягається шляхом надмірної зміни частотних коефіцієнтів

дискретного перетворення Фур'є, що в деяких випадках може суттєво впливає на якість векторного зображення. В роботі [89] представлено вдосконалену процедуру вилучення даних з стеганоконтейнера, завдяки кращого рівня розпізнавання приховуваних біт, однак для вилучення даних потрібна наявність контейнера-оригінала.

2. Метод Карпінцева-Яремчука [14, 15, 17, 18], який здійснює приховування інформації у зображення за рахунок зміни частотних коефіцієнтів двовимірного дискретно косинусного перетворення з розмірністю матриці 8×8. За даним методом векторне зображення представляється у вигляді одновимірного масиву точок $V=\{V_i\}$, $V_i=(X_i,Y_i)$, $i=\overline{1,N}$, де N – кількість точок зображення, X_i, Y_i – координати однієї точки. Далі, з кожної послідовності координат за віссю абсцис або ординат 64-х точок з масиву V формуються двовимірні матриці $C_j(x,y)$, $j=\overline{1,k}$, $k=\dfrac{N}{64}$, де k – кількість сформованих двовимірних матриць, x, y – позиції координат в матриці, розмірністю 8x8. Для кожної матриці $C_j(x,y)$ проводиться пряме двовимірне дискретно косинусне перетворення, в результаті чого одержуються матриці коефіцієнтів $F_j(u,v)$, $j=\overline{1,k}$, де u, v – позиції коефіцієнтів в матриці, за наступною формулою:

$$F_j(u,v)=\frac{c(u)\cdot c(v)}{\sqrt{2n}}\cdot\sum_{x=0}^{n-1}\sum_{y=0}^{n-1}C_j(x,y)\cdot\cos\left(\frac{\pi\cdot u\cdot(2x+1)}{2n}\right)\cdot\cos\left(\frac{\pi\cdot v\cdot(2y+1)}{2n}\right),$$

де n – розмірність матриці, $c(u)=\begin{cases}\dfrac{1}{\sqrt{2}}, \text{якщо } u=0;\\ 1, \text{якщо } u>0.\end{cases}$, $c(v)=\begin{cases}\dfrac{1}{\sqrt{2}}, \text{якщо } v=0;\\ 1, \text{якщо } v>0.\end{cases}$.

Вбудовування одного біта a_m з прихованого повідомлення $a=\{a_m\}$, $a_m\in\{0,1\}$, $m=\overline{1,M}$, де M – кількість біт приховуваного повідомлення, відбувається в високочастотні коефіцієнти кожної матриці $F_j(u_1,v_1)$ в залежності від двох додатково обраних коефіцієнтів $F_j(u_2,v_2)$ і $F_j(u_3,v_3)$.

Вбудовування кожного біта $a_m = 0,\ m = \overline{1,M}$ виконується перевіркою умови:

$$F_j(u_1, v_1) < \frac{F_j(u_2, v_2) + F_j(u_3, v_3)}{2}.$$

Якщо дана умова виконується то значення коефіцієнта $F_j(u_1, v_1)$ лишається без змін, інакше значення коефіцієнта $F_j^{'}(u_1, v_1)$ обраховується за наступною формулою:

$$F_j^{'}(u_1, v_1) = \frac{F_j(u_2, v_2) + F_j(u_3, v_3)}{2} - P,$$

де P – деяка величина, що забезпечує чіткість вилучення прихованих даних.

Вбудовування кожного біта $a_m = 1,\ m = \overline{1,M}$ виконується перевіркою умови:

$$F_j(u_1, v_1) > \frac{F_j(u_2, v_2) + F_j(u_3, v_3)}{2}.$$

Якщо дана умова виконується то значення коефіцієнта $F_j(u_1, v_1)$ лишається без змін, інакше значення коефіцієнта $F_j^{'}(u_1, v_1)$ обраховується за наступною формулою:

$$F_j^{'}(u_1, v_1) = \frac{F_j(u_2, v_2) + F_j(u_3, v_3)}{2} + P.$$

Для зменшення відхилень точок векторного зображення при накладанні двовимірного дискретно косинусного перетворення авторами даного методу запропоновано спосіб відбору придатних для вбудовування інформації матриць $F_j(u, v),\ j = \overline{1,k}$. Визначення придатності матриці для вбудовування інформації відбувається при використанні граничного значення P_h, що визначає максимальне значення зміни коефіцієнтів $F_j(u_1, v_1)$ кожної j-ої матриці $F_j(u, v)$, наступним чином:

$$\begin{cases} \left|F_j(u_1,v_1) - F_j(u_2,v_2)\right| \le P_h \\ \left|F_j(u_1,v_1) - F_j(u_3,v_3)\right| \le P_h \end{cases}.$$

Якщо хоч одна з нерівностей не виконується, то розглядувана матриця $F_j(u,v)$ є не придатною для вбудовування біта інформації.

3. Метод Ліена-Ронга-Ванга [92], який приховує інформацію шляхом зміни середніх частотних коефіцієнти дискретно косинусного перетворення. Для приховування інформації зображення поділяється на прямокутні частини однакового розміру, які утворюють клітинки прямокутної сітки. Приховування інформації відбувається в непорожні клітинки прямокутної сітки $A = \{a_i(n_i)\}$, де n_i – кількість вершин в a_i клітинці, i – кількість клітинок в сітці. Далі, застосовується унітарне дискретне косинусне перетворення над масивом сітки A, $i = \overline{1,N}$, при якому одержується послідовність коефіцієнтів $y(k)$ в які буде вбудовуватися інформація, за наступною формулою,:

$$y(k) = w(k)\sum_{i=1}^{N} a_i(n_i) \cdot \cos\left(\frac{\pi(2i-1)(k-1)}{2N}\right),$$

де k – кількість коефіцієнтів, $k = \overline{1,N}$, значення послідовності $w(k)$ обраховується за наступним рівнянням:

$$w(k) = \begin{cases} \frac{1}{\sqrt{N}}, k = 1; \\ \sqrt{\frac{2}{N}}, 2 \le k \le N. \end{cases}$$

Вбудовування інформації відбувається в середньо частотні коефіцієнти $y(k)$, $k \in [S,E]$, де S, E – параметри контролюючі вибір середніх коефіцієнтів дискретно косинусного перетворення, $E = S + m$, m – кількість біт в прихованому повідомленні $\mathrm{w}(k)$, $0 < S < E < N$, дискретно косинусного перетворення. Вбудовування прихованого повідом відбувається за наступною формулою:

$$y(S+1)' = \begin{cases} |y(S+i)|, \mathrm{w}(i)=1; \\ -|y(S+i)|, w(i)=0. \end{cases}, 1 \le i \le m.$$

Відповідно при вбудовуванні біта $\mathrm{w}(i)=1$ знак коефіцієнта змінюється на додатній, в іншому випадку $\mathrm{w}(i)=0$ на від'ємний.

4. Метод Лі-Ху [91], який приховує інформацію у векторні зображення за рахунок зміни коефіцієнтів дискретного вейвлет перетворення. За даним методом векторне зображення представляється у вигляді одновимірного масиву точок $V=\{V_i\}$, $V_i=(X_i,Y_i)$, $i=\overline{1,N}$, де N – кількість точок зображення, X_i, Y_i – значення однієї точки. Далі відбувається перетворення кожної пари точок з масиву V у комплексну послідовність даних (комплексний сигнал) за наступною формулою:

$$z(k)=X_k+i\cdot Y_k,$$

де i – уявна одиниця.

Над одержаною послідовністю $z(k)$ здійснюється трьохрівневе вейвлет-перетворення, в результаті чого одержуються чотири набори вейвлет-коефіцієнтів: HH1, HH2, HH3, LL3. Враховуючи точність відхилення карт приховування інформації може здійснюватися в коефіцієнти HH2, HH3.

Для вбудовування інформації у вейвлет-коефіцієнти HH2 представляються у вигляді послідовності Z_i, $i=\overline{0,N-1}$. Вбудовування кожного біта a_m з прихованого повідомлення $a=\{a_m\}$, $a_m\in\{0,1\}$, $m=\overline{1,M}$, де M – кількість біт приховуваного повідомлення, відбуватиметься у непарні коефіцієнти послідовності Z_i. Далі, для кожного непарного коефіцієнта обраховується опорне значення R за наступною формулою:

$$R_m=\left(\frac{|Z_i|+|Z_{i+2}|}{2}\right)\cdot\alpha,$$

де $m=\overline{1,M}$, $i=\overline{0,N-1}$, α – коефіцієнт амплітуди. Після чого, за допомогою непарних коефіцієнтів вейвлет-послідовності Z_i та опорних значень R_m обраховуються показники управління вставкою K бітів прихованої послідовності a за наступною формулою:

$$K_m = round\left(\frac{|Z_i|}{R_m}\right),$$

де $m=\overline{1,M}$, i – непарні числа починаючи від 1 до $N-1$.

При вбудовуванні біта $a_m=0$ коефіцієнт $|Z_i|$ змінюють таким чином, щоб відповідний показник K_m став непарним, інакше, при вбудовуванні біта $a_m=1$ коефіцієнт $|Z_i|$ змінюють таким чином, щоб відповідний показник K_m став парним.

Частотні методи забезпечують кращу стійкість до активних атак, однак внаслідок зміни контурів векторних об’єктів чи положень координат їх точок, призводить до погіршення якості самого зображення та утворення видимих відхилень ліній векторних об’єктів [17, 18].

Провівши аналіз методів приховування інформації у векторні зображення було встановлено наступне:

1) всі розглядувані вище методи використовуються для вбудовування цифрових водяних знаків в зображення;

2) всі вони є вузько направленими і вбудовують інформацію тільки в топографічні векторні карти;

3) просторові методи характеризуються слабкою стійкістю до активних атак над стеганоконтейнерами, а частотні методи – забезпечують кращу стійкість. Однак, дані просторові та частотні методи можуть погіршувати якість та утворювати видимі відхилення ліній векторних об’єктів зображення;

4) для методів Кітамури-Канаі-Кішінами та Обуші потрібна наявність контейнера-оригінала для вилучення прихованої інформації з стеганоконтейнера.

Дана група методів не може використовуватися для вирішення поставленої задачі дисертаційної роботи. Таким чином, вникає необхідність у розробці нових стеганографічних методів приховування інформації у векторні зображення, які можуть використовувати поширені векторні формати зображень, та протидіяти активним атакам. З цією метою проведемо огляд та дослідження відомих активних атак на стеганографічні методи приховування у векторні зображення.

1.3. Аналіз активних атак на стеганосистеми з векторними зображеннями

При використанні стеганографічних методів досить важливим показником є здатність протидіяти різним видам атак [16, 36]. Адже, стеганосистема вважається стійкою, якщо порушнику не вдалося довести існування присутності прихованого повідомлення в стеганоконтейнері [36]. В залежності від типу обраного контейнера можуть змінюватися і рівні стійкості того чи іншого стеганографічного методу.

Усі види атак на стеганографічні системи можна поділити на *пасивні атаки* – спрямовані на виявлення присутності вмісту приховуваного повідомлення в стеганоконтейнері та *активні атаки* – спрямовані на знищення або пошкодження приховуваного повідомлення в стеганоконтейнері шляхом їх певної обробки [12, 16, 36, 38, 41].

До активних атак, що можуть застосовуватися над векторними зображеннями, відносяться:

1) *точкові атаки* – спрямовані на унеможливлення вилучення приховуваного повідомлення з стеганоконтейнера шляхом зміни кількості точок в зображенні, що досягається за рахунок вставки певної кількості нових точок або їх видаленням [16, 86, 96, 97, 99];

2) *атака перетворення формату* – базуються на програмній залежності та структурних принципів задання різних типів векторних

зображень [96, 97]. Кожний графічний пакет зберігає дані в певному форматі за своїми правилами. При перетворенні з одного формату в інший, дані відповідно змінюються до вимог іншого формату, що призводить до деякої видозміни (втрати) початкових даних;

3) *перестановка об'єктів* – базуються на зміні положень векторних об'єктів або координат точок заданого об'єкта в структурі стеганоконтейнера [96, 97]. Використовуються для утворення нових зображень з зміненою послідовністю побудови векторних об'єктів, що може виконуватись без погіршення якості, для унеможливлення вилучення прихованої інформації з стеганоконтейнера;

4) *спотворення шумом* – виконується для знищення прихованого повідомлення в стеганоконтейнері шляхом додавання додаткового шуму до координат точок зображення [16, 86, 96, 99, 109]. При надмірному добавлянні шуму до координат може спричинити до утворення видимих спотворень векторних об'єктів;

5) *атака злиттям* – передбачає об'єднання двох зображень з різних джерел, які відрізняються точністю розмірів та деталізацією [109]. Виконуючи таку атаку зловмисник обрізає певні частини різних зображень і поєднувати їх разом для утворення нового зображення, при цьому відкидаючи певні частини стеганоконтейнера де може міститися прихована інформація. Такі об'єднання можуть призводити до утворення видимих неточностей утворених зображень в місцях їх поєднання;

6) *атака відсікання* – базуються на обрізанні певного відсотка масиву даних у декількох місцях стеганоконтейнера, що призводить до втрати деякої частини точок оригінальних даних [16, 99, 109]. Такі атаки унеможливлюють вилучення приховуваної інформації з стеганоконтейнера у тих місцях де відбулося обрізання даних;

7) *атаки ущільненням* – використовуються для видалення деяких даних з стеганоконтейнера [16]. До даних атак відносяться: видалення надлишковості з зображення (наприклад, об'єднання декілька точок кривих в

одну криву без втрати якості зображення); ущільнення точок інтервалу, які полягають в збереженні першої та останньої точки кривої, але видаляють кожну K точку між ними; атака ущільнення кута або вертикальної відстані, для проведення атаки спочатку обчислюється кут або відстань між двома точками, потім одержане значення порівнюється з деяким граничним параметром. При не відповідності одержаного значення до встановленого граничного параметра одна з точок видаляється, в іншому випадку залишається;

8) *афінні перетворення* – використовуються для відображення об'єкта з однієї афінної системи координат в іншу, при якому координати будь-якої точки в першій системі координат збігаються з координатами її образу в другій системі координат [3, 16, 22-24, 29, 63, 86, 90, 96, 97, 99, 105]. Дані атаки призводять то зміни значень координат точок та відстаней між ними у стеганоконтейнері, що призводить до втрати прихованої інформації.

Зміни утворенні атаками перестановки об'єктів, зміни кількості точок, злиття, ущільнення та відсікання призводять до помітних видимих візуальних спотворень векторного зображення та можуть потребувати досить великих обчислювальних ресурсів на стороні шлюзу. Атака перетворення формату явно модифікує формат представлення даних, а також вносить похибку в координати векторних об'єктів в процесі перетворення.

Проведений аналіз активних атак застосовуваних над векторними зображеннями, показав, що згідно сформульованим умовам, найбільш ймовірними буде застосування противником атаки на основі афінних перетворень та спотворенням шумом.

У сучасній комп'ютерній графіці досить широко використовується подання зображень векторного типів через системи координат. Крім цього, координати використовуються для опису розміщення об'єктів та для створення зображень шляхом перетворень з однієї системи координат в іншу. На практиці широко використовуються двовимірні (2D) та тривимірні (3D) системи координат [8, 43]. Завдяки такому представленню зображень можна

з легкістю здійснювати над ними різноманітні трансформації на основі афінних перетворень.

Під афінними перетвореннями [43] розуміється – відображення об'єкта, площини в самих себе, при якому: 1) N-вимірний об'єкт відображається в N-вимірний, точка – в точку, лінія – в лінію, поверхня – в поверхню; 2) зберігається паралельність ліній і площин; 3) зберігаються пропорції паралельних об'єктів.

Афінне перетворення для N-вимірного простору в однорідних координатах задається за допомогою наступного матричного перетворення [22]:

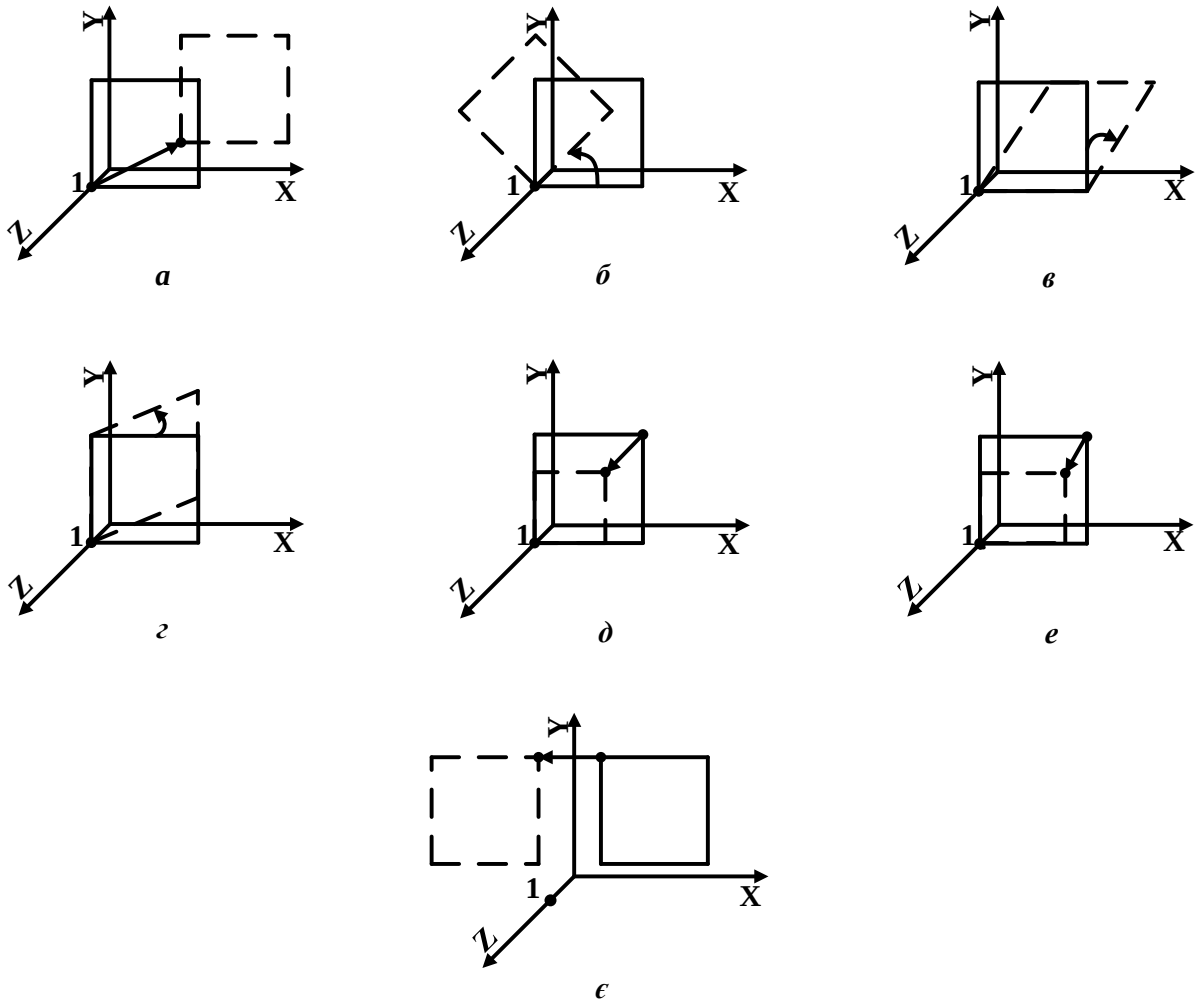
$$\begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & \dots & \alpha_{1,N} & \beta_1 \\ \alpha_{2,1} & \alpha_{2,2} & \dots & \alpha_{2,N} & \beta_2 \\ \dots & \dots & \dots & \dots & \dots \\ \alpha_{N,1} & \alpha_{N,2} & \dots & \alpha_{N,N} & \beta_N \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X_1 \\ X_2 \\ \dots \\ X_N \\ 1 \end{bmatrix} = \begin{bmatrix} X_1^* \\ X_2^* \\ \dots \\ X_N^* \\ 1 \end{bmatrix}, \qquad (1.1)$$

де X_i – початкові координати заданої точки, $\alpha_{i,j}$ – довільні дійсні числа, що визначають операції повороту, зсуву, масштабування, $\alpha_{i,j} \in R$, β_i – довільні дійсні числа, що визначають операції перенесення, $\beta_i \in R$, $i = \overline{1,N}$, $j = \overline{1,N}$.

Частковий випадок афінних перетворень для 2D простору ($N = 2$) у системі координат XYZ, де координата $Z = 1$, розглядається як наступне матричне перетворення [7, 22-24, 29, 43]:

$$\begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & \beta_1 \\ \alpha_{2,1} & \alpha_{2,2} & \beta_2 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} X^* \\ Y^* \\ 1 \end{bmatrix}. \qquad (1.2)$$

На рис.1.6 зображені типи афінних перетворень для 2D простору [3, 22, 43, 63, 74].



а *б* *в* *г* *д* *е* *є*

Рис. 1.6. Типи афінних перетворень для 2D простору: *а* – перенесення; *б* – поворот; *в* – зсув за віссю абсцис; г – зсув за віссю ординат; *д* – пропорційне масштабування; *е* – непропорційне масштабування; *є* – дзеркальне відображення

Розглянемо кожне афінне перетворення представлене за формулою (1.2) більш детально.

Перенесення об’єкта здійснюється шляхом зміни всіх координат заданого об’єкту на довільне стале значення β_1 за віссю абсцис та β_2 за віссю ординат за наступною формулою [3, 22, 24, 29, 43]:

$$\begin{bmatrix} 1 & 0 & \beta_1 \\ 0 & 1 & \beta_2 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} X^* \\ Y^* \\ 1 \end{bmatrix}. \qquad (1.3)$$

Поворот об’єкта здійснюється шляхом зміни всіх координат заданого об’єкту на кут θ навколо центра з координатами $(0,0,1)$ за наступною формулою [3, 22, 24, 29, 43, 74]:

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} X^* \\ Y^* \\ 1 \end{bmatrix}. \qquad (1.4)$$

Зсув об’єкта здійснюється шляхом зміщення координат заданого об’єкта за віссю абсцис на певне відсоткове значення $\alpha_{1,2}$ (за віссю ординат на певне відсоткове значення $\alpha_{2,1}$), а координати за віссю ординат (відповідно за віссю абсцис) залишаються без змін. Перетворення зсуву обраховується за однією з наступних формул [22, 24, 74]:

а) зсув за віссю абсцис –
$$\begin{bmatrix} 1 & \alpha_{1,2} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} X^* \\ Y^* \\ 1 \end{bmatrix}; \qquad (1.5)$$

б) зсув за віссю ординат –
$$\begin{bmatrix} 1 & 0 & 0 \\ \alpha_{2,1} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} X^* \\ Y^* \\ 1 \end{bmatrix}. \qquad (1.6)$$

Масштабування об’єкту здійснюється шляхом розтягування/стискання заданого об’єкта вздовж координатних осей при довільних значеннях $\alpha_{1,2}$ та $\alpha_{2,1}$ за наступною формулою [3, 22, 29, 43, 74]:

$$\begin{bmatrix} \alpha_{1,1} & 0 & 0 \\ 0 & \alpha_{2,2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} X^* \\ Y^* \\ 1 \end{bmatrix}. \qquad (1.7)$$

Якщо $\alpha_{1,1} = \alpha_{2,2}$, то відбувається пропорційне масштабування об’єкту, що при $\alpha_{1,1} = \alpha_{2,2} > 1$ рівномірно розтягується (збільшується), а при $\alpha_{1,1} = \alpha_{2,2} < 1$ рівномірно стискається (зменшується). Якщо $\alpha_{1,1} \neq \alpha_{2,2}$, то відбуватиметься непропорційне масштабування. Окремим випадком масштабування є перетворення дзеркального відображення. Яке полягає в

відбиванні об’єкта на інших квадрантах координатної площини при різних значеннях $\alpha_{1,1} \in \{-1,1\}$, $\alpha_{2,2} \in \{-1,1\}$.

Також, слід розглянути випадки, коли при накладанні афінних перетворень, можуть вноситися незначні похибки при обчисленні нових координат точок векторного зображення. При реалізації таких афінних перетворень (надалі майже афінні перетворення) будуть одержуватися зображення подібні до оригіналу, але з деяким відхиленням. Дані афінні перетворення можуть використовуватися для внесення додаткового шуму в стеганоконтейнер, які, для 2D простору, можна задати наступним чином:

$$\begin{bmatrix} \alpha_{1,1}+\varepsilon_1 & \alpha_{1,2}+\varepsilon_2 & \beta_1 \\ \alpha_{2,1}+\varepsilon_3 & \alpha_{2,2}+\varepsilon_4 & \beta_2 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X+\varepsilon_5 \\ Y+\varepsilon_6 \\ 1 \end{bmatrix} = \begin{bmatrix} X^* \\ Y^* \\ 1 \end{bmatrix}, \qquad (1.8)$$

де ε_i – довільні досить малі дійсні числа, $i = \overline{1,6}$.

В табл. 1.1 знаходяться результати порівняння стійкості стеганографічних методів приховування інформації у векторні зображення до різноманітних афінних перетворень.

Таблиця 1.1

Порівняння стійкості методів приховування інформації у векторні зображення

Метод	Необх. контейнера-оригінала	Стійкість до афінних перетворень				
		Перенес.	Поворот	Зсув	Масштаб.	Майже афін. перетв.
Сакамоти-Матсуури-Такашими	+	-	-	-	-	-
Канга	-	-	-	-	-	-
Обуші	+	+/-	+/-	-	+/-	-
Шульца-Войта	-	-	-	-	-	-
Х’юберта	+	+/-	+/-	-	+/-	-
Солачідіса-Ніколайдіса-Пітаса	-	+/-	+/-	-	+/-	-
Кітамури-Канаі-Кішінами	+	+/-	+/-	-	+/-	-
Карпінцева-Яремчука	-	+	+	-	+/-	-
Ліена-Ронга-Ванга	-	+/-	+/-	-	+/-	-
Лі-Ху	-	+/-	+/-	-	+/-	-

Провівши порівняння стійкості існуючих стеганографічних методів приховування інформації у векторні зображення до афінних перетворень

встановлено, що методи Обуші, Х'юберта, Солачідіса-Ніколайдіса-Пітаса, Кітамури-Канаі-Кішінами, Карпінцева-Яремчука, Ліена-Ронга-Ванга, Лі-Ху забезпечують стійкість до афінних перетворень [14, 15, 17-19, 30, 78, 84-86, 89, 91, 92, 94-96, 98, 101, 103, 104]. Однак, в відкритих джерелах розглянуті вище методи досліджують стійкість шляхом одноразово накладання афінного перетворення до початкового стеганоконтейнера, при якому вже відбувається втрата прихованих даних. Так, для метода Карпінцева-Яремчука при накладанні однієї атаки пропорційного масштабування може відбутися втрата даних до 60% [19]. При накладанні декількох таких послідовних перетворень над обробленим стеганоконтейнером відбудиться повна втрата прихованої інформації. Також, слід відзначити, що розглядувані методи не досліджувалися на стійкість до перетворення зсуву, непропорційного масштабування та майже афінних перетворень.

1.4. Постановка задачі дослідження

Стеганографічний захист інформації використовуваний для реалізації резервного каналу зв'язку через мережу Інтернет повинен володіти поліпшеними характеристиками забезпечуваними сучасними стеганографічними системами. До яких відносяться [36, 59]:

1. Здатність забезпечувати непомітність вбудовування інформації в структуру контейнера. Тобто, спотворення, що вносяться у контейнери, повинні бути нижче деякої критичної межі (наприклад, рівня межі чутливості людини до спотворення контейнерів-зображень).

2. Здатність забезпечувати високу стійкість до активних атак здійснюваними над стеганоконтейнерами при їх передачі. Ймовірність помилкового вилучення прихованої інформації, після застосування даних атак, на приймальній стороні не повинна перевершувати наперед заданої величини.

3. Здатність забезпечувати високу пропускну спроможність передачі інформації застосовуваними стеганоканалами. Тобто, кількість вбдовуваних даних в контейнери повинна бути максимізована.

На рис. 1.7 представлена залежність між непомітністю вбудовування, стійкістю до активних атак та кількістю вбудовуваних даних [16, 36, 56, 65].

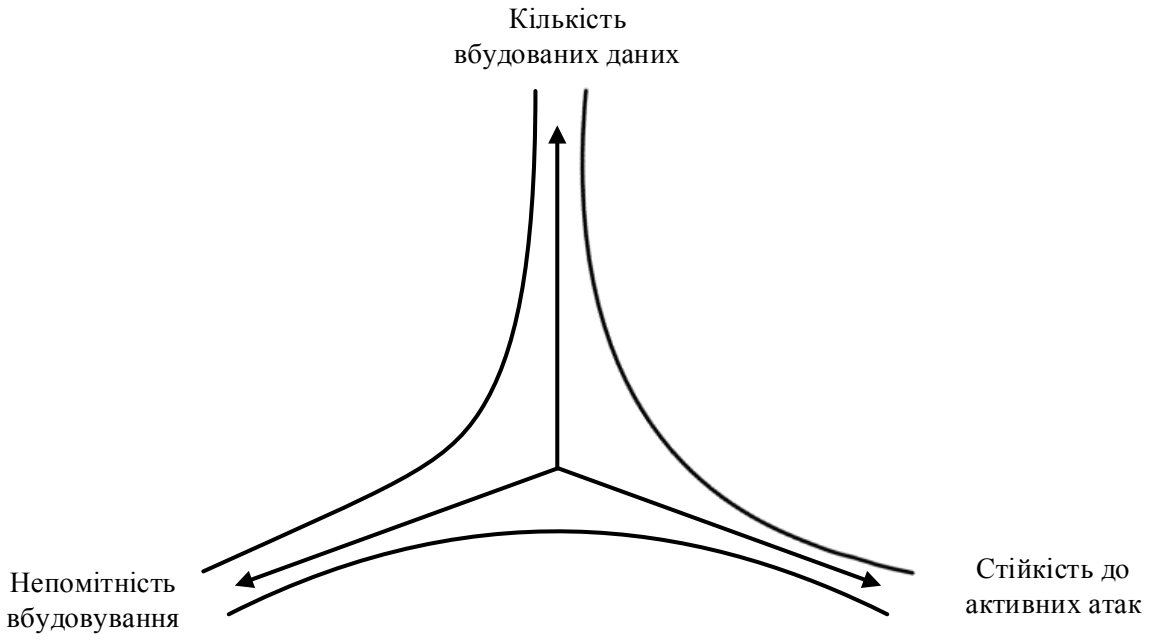


Рис. 1.7. Взаємозв'язок між непомітністю вбудовування, стійкістю до активних атак та кількістю вбудованих даних

Описати використання максимального/мінімального значення кожного з даних показників можна за допомогою системи цільових функцій за використанням деякої множини допустимих параметрів:

$$\begin{cases} f(x_1, x_2) \to \max; \\ g(x_2, x_3, x_4) \to \min; \\ h(x_1, x_2, x_3, x_4, x_5) \to \min. \end{cases}, \qquad (1.9)$$

де $f(x_1, x_2) \to \max$ – максимізувати кількість вбудовуваних біт K ($K \in N$) в контейнер, $g(x_2, x_3, x_4) \to \min$ – мінімізація коефіцієнта спотворення в межах $[0,1]$ одержаного стеганоконтейнера відносно початкового контейнера, $h(x_1, x_2, x_3, x_4, x_5) \to \min$ – мінімізація кількості втрачених біт в межах $[0, K]$,

K – кількість вбудовуваних біт, в результаті застосування активних атак, x_1 – пропускна здатність каналу зв'язку, x_2 – розмір контейнера/ стеганоконтейнера, x_3 – кількість вбудовуваних біт в контейнер, x_4 – множина активних атак на стеганоконтейнер, x_5 – коефіцієнт візуального спотворення зображення.

Розв'язок системи (1.9), при якій знаходилися оптимальні значення для кожної цільової функції, є неможливою задачею. На практиці пошук оптимального рішення знаходиться шляхом застосування компромісу між даними показниками [36].

Оскільки, значення шукані в цільових функціях $f(x_1, x_2)$ та $g(x_2, x_3, x_4)$ входять як окремі параметри до цільової функції $h(x_1, x_2, x_3, x_4, x_5)$ $(x_3 = f(x_1, x_2), \quad x_5 = g(x_2, x_3, x_4))$, то зафіксувавши їх певними сталими величинами (значеннями) і перейдем до розгляду тільки однієї цільової функції $h(x_1, x_2, x_3, x_4, x_5)$.

Опишемо множину допустимих параметрів x_i, $i = \overline{1,5}$, для цільової функції $h(x_1, x_2, x_3, x_4, x_5)$ та введемо обмеження для їх використання.

Пропускна здатність каналу зв'язку. Можливості передачі інформації по безпровідним система мобільного зв'язку невпинно зростають. В табл. 1.2 представлені покоління розвитку мобільних мереж та їх швидкісні характеристики [6, 93].

Таблиця 1.2

Порівняння швидкісних характеристик різних поколінь безпровідних систем мобільного зв'язку

Покоління безпровідних систем	Стандарт	Швидкість передачі
1G	AMPS, TACS, NMT	до 1,9 Кбіт/с
2G	TDMA, CDMA, GSM, PDC	до 14,4 Кбіт/с
2,5G	GPRS	до 160 Кбіт/с
	1xRTT	до 153,6 Кбіт/с
	EDGE (2,75G)	до 473,6 Кбіт/с
3G	WCDMA, CDMA2000, UMTS	до 2 Мбіт/с
3,5G	HSDPA, HSUPA, HSPA, HSPA+	до 14 Мбіт/с
4G	LTE-Advanced, WirelessMAN-Advanced	до 1 Гбіт/с

Оскілки, впровадження нових технологій 3G та 4G відбувається досить повільно, то на даний момент досить поширеною є технологія 2G за стандартами GPRS та EDGE [107]. Тому, при використанні мобільних мереж, для реалізації резервного каналу зв'язку через мережу Інтернет, слід обмежитися пропускною швидкістю передачі інформації за стандартами GPRS та EDGE. Пропускна швидкість передачі інформації за даними стандартами змінюється від кількості використовуваних одночасно тайм-слотів та застосовуваної схеми кодування певної мобільної мережі, що приведені в табл. 1.3 [6, 50, 100].

Таблиця 1.3

Швидкості передачі інформації за стандартами GPRS та EDGE

Стандарт	Схема кодування	Залежність швидкості передачі інформації Кбіт/с від кількості використовуваних тайм-слотів							
		1	2	3	4	5	6	7	8
GPRS	CS1	8,0	16	24	32	40	48	56	64
	CS2	12,0	24	36	48	60	72	84	96
	CS3	14,4	28,8	43,2	57,6	72	86,4	100,8	115,2
	CS4	20,0	40	60	80	100	120	140	160
EDGE	MCS1	8,4	16,8	25,2	33,6	42	50,4	58,8	67,2
	MCS2	11,2	22,4	33,6	44,8	56	67,2	78,4	89,6
	MCS3	14,8	29,6	44,4	59,2	74	88,8	103,6	118,4
	MCS4	17,6	35,2	52,8	70,4	88	105,6	123,2	140,8
	MCS5	22,4	44,8	67,2	89,6	112	134,4	156,8	179,2
	MCS6	29,6	59,2	88,8	118,4	148	177,6	207,2	236,8
	MCS7	44,8	89,6	134,4	179,2	224	268,8	313,6	358,4
	MCS8	54,4	108,8	163,2	217,6	272	326,4	380,8	435,2
	MCS9	59,2	118,4	177,6	236,8	296	355,2	414,4	473,6

Зважаючи на зміни швидкості передачі інформації, встановимо обмеження $8 \le x_1 \le 473{,}6$ Кбіт/с на пропускну здатність резервного каналу зв'язку за використання мобільних мереж 2G (GPRS та EDGE).

Розмір контейнера/стеганоконтейнера. Передача інформації через резервний канал зв'язку потребує використання контейнера невеликого розміру, який зможе швидко передаватися через мобільні мережі за стандартами GPRS та EDGE.

Проведений аналіз більше 13 тис. векторних зображень різних форматів взятих з декількох графічних банків даних [77, 81, 108] представлений в табл. 1.4, показав, що середньо статистичний розмір

контейнера становить $x_2 = 264,94$ Кб, що задовольняє пропускну можливість визначених мобільних мереж.

Таблиця 1.4

Визначення середньо статистичного розміру векторного зображення

Тип формату	AI	SVG	EPS	CDR
Кількість	2156	767	9354	795
Розмір, Кб	406890,52	44384,44	2910602,96	101399,66
Середній розмір одного зображення, Кб	188,72	57,87	311,16	127,55
Загальна кількість	13072			
Загальний розмір, Кб	3463277,58			
Загальний середній розмір одного зображення, Кб	264,94			

Слід зазначити, що використовуваний контейнер повинен бути досить поширеним та доступним, для запобігання зайвих підозр щодо його використання (наприклад, векторний формат SVG, що в останні роки набуває все більшого використання та підтримки рис.1.8 [10, 76, 82, 83, 106]).

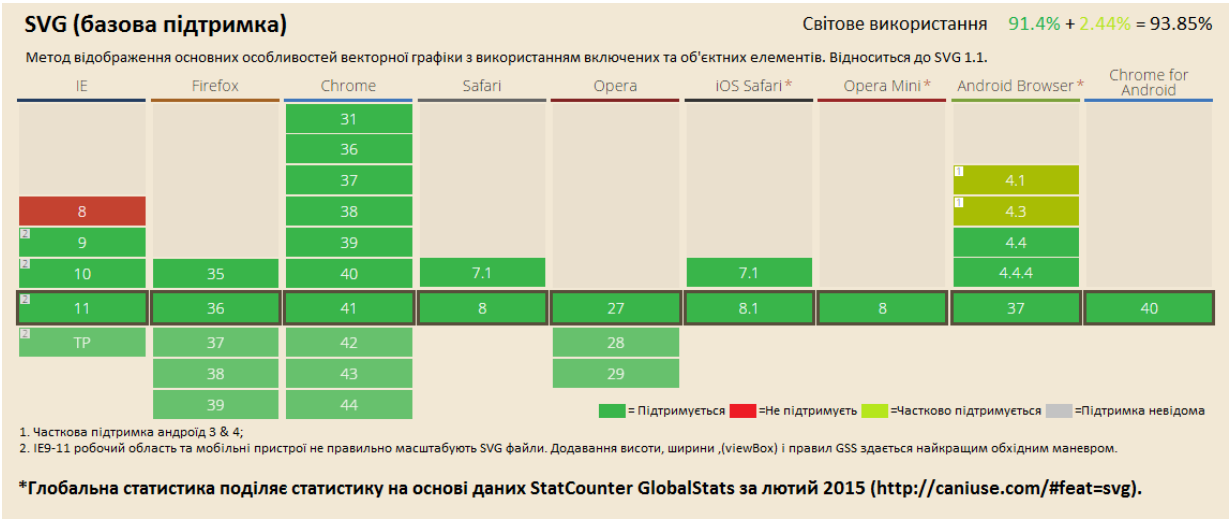


Рис. 1.8. Підтримка SVG формату

Вбудовування інформації у векторні зображення може призводити до збільшення розмірів одержуваних стеганоконтейнерів. Тому, при вбудовуванні інформації в обраний контейнер повинен фіксуватися розмір одержаного стеганоконтейнера, так щоб задовільнити допустиму швидкість передачі інформації через резервний канал зв'язку. Визначатися допустимий розмір стеганоконтейнера x_2 буде наступним чином:

$$x_2 = q(x_1, x_3),$$

де $q(x_1, x_3)$ – функція визначення допустимого розміру стеганоконтейнера, x_1 – пропускна здатність каналу зв'язку, x_3 – кількість вбудовуваних біт в контейнер.

Кількість вбудовуваної інформації в контейнер. Для визначення максимальної кількості вбудовуваної інформації в один контейнер, будемо орієнтуватися на середньо статистичну кількість символів розташовуваній на одній сторінці А4 (210×297 мм) текстового редактора Microsoft Word з стандартними полями (верхнє – 20 мм, нижнє – 20 мм, ліве – 30 мм, праве – 15 мм), розміром 14 пт, шрифтом Times New Roman та полуторним міжрядковим інтервалом, що становить приблизно 2340 друкованих знаків з пробілами [48]. З огляду на це, встановимо обмеження на кількість вбудовуваної інформації в контейнер – $x_3 \le 2340 \cdot 8 = 18720$ біт.

Для передання великих обсягів інформації через мережу Інтернет може застосовуватися попереднє стиснення приховуваного повідомлення програмами архіваторами, що дозволить зменшити розмір приховуваних даних.

Множина активних атак на стеганоконтейнер. Дана множина визначатиметься атаками на основі афінних та майже афінних перетворень $x_4 = \{\text{перенесення, поворот, зсув, масштабування}\}$ (див. підрозділ 1.2).

Коефіцієнт візуального спотворення зображення. Рівень спотворення зображення при вбудовуванні даних в його структуру не повинен перевищувати такої межі $x_5 \le 3\%$ [9, 36, 38].

1.5. Висновки до першого розділу

У першому розділі дисертаційної роботи були отримані наступні результати:

1. Визначено сферу використання стеганографічного захисту інформації, як спосіб організації резервного каналу зв'язку з

дипломатичними установами, що знаходяться на території іноземних держав, через загальнодоступну мережу Інтернет.

2. Проведено аналіз сучасних методів приховування інформації, що використовують у якості контейнера векторні зображення.

3. Проведено аналіз активних атак над векторними зображеннями. Виділені атаки на основі афінних перетворень. Проведено порівняння стійкості методів приховування інформації у векторні зображення до афінних перетворень, яке показало слабку стійкість розглядуваних стеганографічних методів відносно даних атак. Тому, існуючі методи приховування інформації у векторні зображення не підходять для реалізації резервного каналу зв'язку.

4. Визначена цільова функція зменшення кількості втрати інформації, при її передачі резервним каналом зв'язку через глобальну мережу Інтернет з підтримкою мобільних мереж та забезпеченням стійкості до атак на основ афінних перетворень. Встановленні обмеження до основних параметрів даної цільової функції.

5. Для розв'язку заданої цільової функції існує необхідність у проведенні дослідження та розробки нових стеганографічних методів приховування інформації у векторні зображення стійких до афінних перетворень, характер змін яких не призводитиме до помітних відхилень об'єктів в контейнері, що дасть можливість підвищити ефективність стеганографічного захисту в цілому.

РОЗДІЛ 2

МЕТОДИ ПРИХОВУВАННЯ ІНФОРМАЦІЇ У ВЕКТОРНІ ЗОБРАЖЕННЯ

2.1. Принцип вбудовування інформації у векторні зображення

Основним елементом в векторній графіці є лінія, якій притаманні деякі властивості: форма, колір, товщина тощо [37, 43, 63]. Задається кожна лінія за допомогою аналітичної формули з певним числом параметрів необхідних для її представлення [5, 37]. Будь-який об’єкт (прямокутник, еліпс, парабола, текст, пряма лінія тощо) сприймається і подається як криві лінії (надалі криві) [5, 43, 63], винятком можуть служити тільки імпортовані растрові об’єкти. Досить великий клас кривих, які відрізняються різним ступенем гладкості, можна побудувати за сукупністю точок. Такі криві називають точково-заданими, а до їх складу відносяться ламана лінія та різноманітні сплайнові криві (криві Без’є, B-сплайни, інтерполяційні криві Ерміта та інші) [5, 43]. Деякий ряд кривих даного класу володіє властивістю афінно-інваріантності (незмінності). Тобто, над такими кривими можна здійснити афінні перетворення при якому N-вимірний об’єкт відобразиться в N-вимірний, точка – в точку, лінія – в лінію, поверхня – в поверхню [43]. За допомогою такої властивості можна забезпечити стійкість до атак на основі афінних перетворень (див. підрозділ 1.3).

Задаються задані криві за допомогою множини точок (вершин) $\mathbf{P} = \{P_i\}$, $i = \overline{0,n}$, n – ступінь кривої, $n+1$ – кількість точок, що при послідовному з’єднані утворюють ламану, яку називають опорною, а самі її вершини – опорними [43]. Така ламана вказує, як буде будуватися шукана крива. Побудувати такі криві можна за наступною формулою [43]:

$$R(t) = \sum_{i=0}^{n} f_i(t) P_i , \qquad (2.1)$$

де $f_i(t)$ – деякі функціональні коефіцієнти, t – параметр побудови кривої, $t \in [a,b]$.

Векторним об’єктам, які описуються через криві, завжди притаманний шлях, що визначає маршрут за яким з’єднується початкова та кінцева точка кривої [43, 63]. Кожен шлях складається з сукупності сегментів та вузлів, де сегмент – це окрема частина шляху, що може бути кривою або прямою лінією, а вузол – це початкова або кінцева точка сегмента [43, 63]. Параметричні рівняння кожного сегмента кривої знаходяться за формулою (2.1) при використанні лише частини опорних точок з множини **P** необхідних для його представлення [43]. На рис. 2.1 наведений приклад подання векторного об’єкта через сукупність сегментів та вузлів.

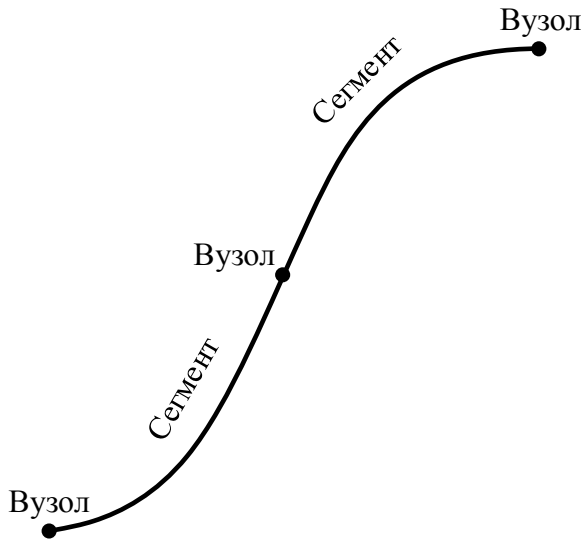
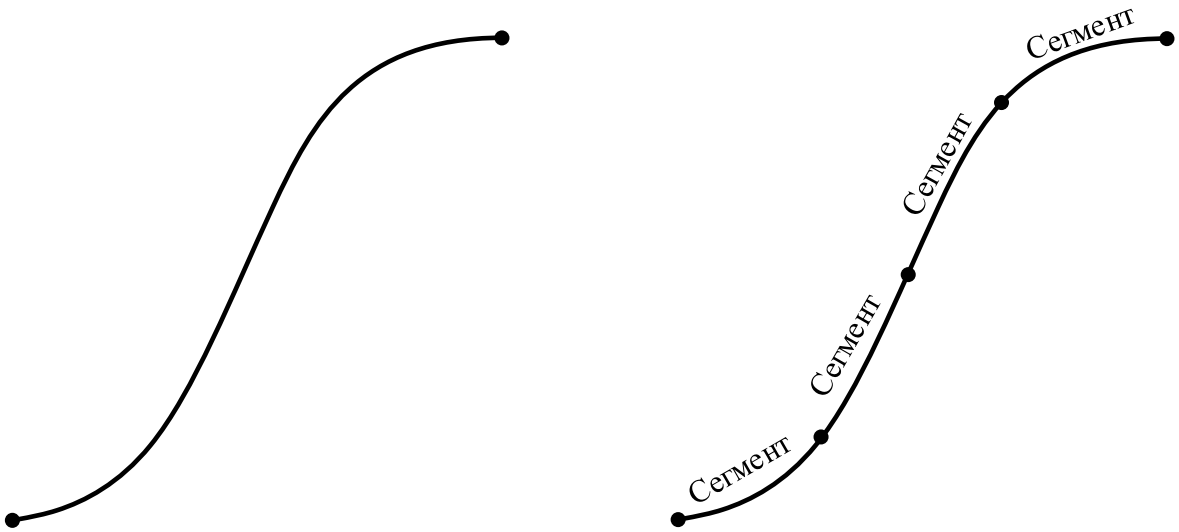


Рис. 2.1. Подання векторного об’єкта через сукупність сегментів та вузлів

Кількість використовуваних сегментів для задання векторного об’єкта може бути довільним. Також, існує можливість збільшення кількості таких сегментів шляхом добавляння додаткового вузла до кривої, що призводить до розбиття деякої частини кривої на два сегмента [43]. Одержана послідовність сегментів виглядатиме візуально однаково відносно початкової кривої.

Дана властивість може бути використана для приховування секретної інформації у векторні зображення шляхом поступового поділу та представлення початкових кривих через сукупності сегментів. На рис. 2.2

наведена схематична ілюстрація представлення початкової кривої та її копії поділеної на сегменти, що візуально виглядають однаково.



а ***б***

Рис. 2.2. *а* – початкова крива, *б* – крива розбита на сегменти

Побудова будь-якої кривої здійснюється за використанням параметра t, $t \in [a,b]$, кожне значення якого строго визначає положення точок на кривій [5, 26, 43, 63]. Якщо зафіксувавши певний крок Δt зміни параметра t можна здійснювати перехід між його значеннями $t_{i+1} = t_i + \Delta t$ в межах заданого проміжку $[a,b]$. Використовуючи значення t_i можна здійснювати приховування інформації у криві шляхом їх розбиття на сукупності сегментів в визначених точках, що потім замінятимуть початкові криві у векторному зображенні. Вилучення прихованої інформації може відбуватися шляхом поступового відтворення початкових кривих з одержаних сукупностей сегментів при різних значеннях параметра t з проміжку $[a,b]$ при наперед визначеному кроці Δt.

Отже, в даному підрозділі запропоновано принцип вбудовування інформації в структуру векторного зображення шляхом поділу та представлення кривих на сукупності сегментів у наперед визначених точках кривої, за рахунок фіксування кроку Δt зміни параметра побудови кривих t. За

допомогою властивості афінно-інваріантності, якою володіє ряд кривих даного класу, буде забезпечуватися стійкість до атак на основі афінних перетворень.

2.2. Параметри приховування інформації у векторні зображення

Різноманітні криві векторних зображень володіють властивостями за якими можна здійснити приховування інформації шляхом їх розбиття на довільні сукупності сегментів. Однак, векторні зображення задаються аналітично і на практиці завжди присутня похибка округлення координат точок її елементів.

Для того щоб ефективно здійснювати приховування інформації у криві векторного зображення введено множину параметрів $\mathbf{V}$, що впливатиме на вибір контейнера та на процес вбудовування/вилучення інформації [20-26].

Виділимо та опишемо кожен з параметрів, що відноситиметься до даної множини $\mathbf{V} = \{V_j\}$, $j \in \overline{1,5}$. Умовно дані параметри можна поділити на дві групи:

1) параметри, які впливатимуть на вибір контейнера;

2) параметри, які впливають на процес вбудовування/вилучення інформації.

До параметрів, які впливатимуть на вибір контейнера відносяться [20-26]:

– *Параметр* V_1 – визначає ступінь кривої в які буде вбудовуватися інформація. Відомо, що криві визначаються ступенем порядку, а саме кількістю опорних точок необхідних для їх представлення. Кількість таких точок для представлення кривої може бути довільною. Однак, на практиці більше використовують криві третього та четвертого ступеня. Таке використання пов’язане з тим, що криві вищих ступенів потребують більшу кількість опорних точок для їх представлення, тим самим підвищуючи обсяг обчислень при їх обробці і збільшують розмір зображення [43]. Тому, для досягнення найбільшої ефективності при роботі з кривими, накладемо обмеження на використовувану

ступінь кривої $V_1 \in \{1,2,3,4\}$. На рис. 2.3 наведені схематичні ілюстрації побудови кривих Без'є різного ступеня.

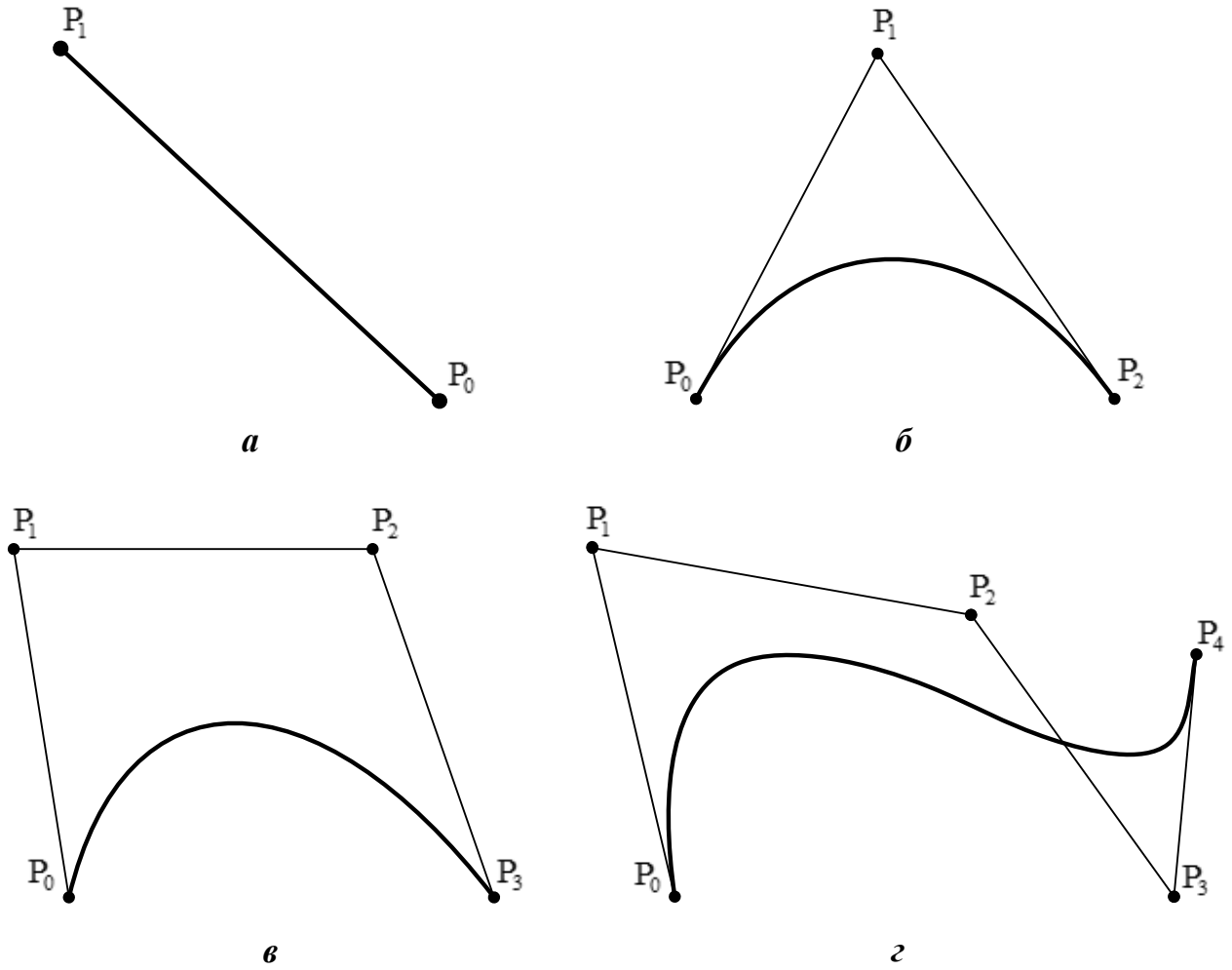


Рис. 2.3. Приклади кривих Без'є різного ступеня: *а* – лінійна крива; *б* – квадратична крива; *в* – кубічна крива; *г* – квадратична крива

– *Параметр* V_2 – мінімальна допустима відстань між опорними точками кривої. Розбиття досить малих кривих призводить до отримання опорних точок сформованих сегментів з координатами, що практично не відрізняються одна від одної. Використання таких кривих ускладнює процес приховування/вилучення інформації. Тому, введено обмеження на мінімальну допустиму відстань між опорними точками кривої $V_2 \geq 1$.

Для встановлення відповідності обраної кривої до встановленої відстані V_2, потрібно брати кожну точку P_i, $i = \overline{0,n}$, $n+1$ – кількість опорних точок необхідних для представлення даної кривої, з множини **P** та перевіряти її на

відстань відносно інших опорних точок, так щоб виконувалась наступна нерівність (рис. 2.4):

$$\sqrt{\left(X_i - X_j\right)^2 + \left(Y_i - Y_j\right)^2} \geq V_2,$$

де X_i, Y_i – координати P_i точки, X_j, Y_j – координати інших опорних точок з множини $\mathbf{P}$, $i = j = \overline{0,n}$, але $i \neq j$.

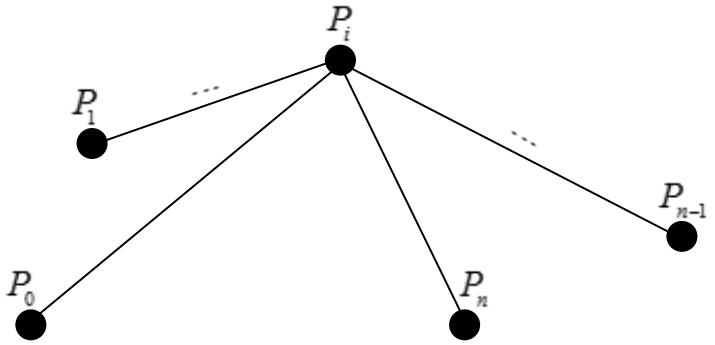


Рис. 2.4. Перевірка точки на відстань відносно інших опорних точок кривої

Параметри V_1 та V_2 слід використовувати для проведення аналізу векторних об’єктів контейнера на можливість приховування в них секретної інформації, а саме визначити допустимі криві, які можна використати для приховування даних. Однак, не всі векторні зображення можуть задаватися за допомогою кривих, то в такому випадку дані зображення слід перетворити та подати через криві лінії. В разі невідповідності допустимих відстаней між опорними точками кривих слід накладати перетворення масштабування, що призведе до зміни даних відстаней. Накладання таких змін потребує додаткових затрат на оброку контейнера.

До параметрів, які впливають на процес вбудовування/вилучення інформації відносяться [20-26]:

– *Параметр* V_3 – визначає максимальну кількість вбудовуваної інформації в одну криву. Приховування всього повідомлення в одну криву векторного зображення призведе до отримання великої кількості сегментів та координат опорних точок необхідних для їх представлення. Таке вбудовування може бути помітним при аналізі або обробці контейнера, тому введемо

обмеження стосовно кількості приховуваних даних в одну криву $x_3 \geq V_3 > 0$, де x_3 – кількість вбудовуваних біт в контейнер (див. підрозділ 1.4).

– *Параметр* V_4 – точність координат опорних точок утворених сегментів. Векторні зображення задаються координатами точок, які можуть мати довільну кількість розрядів дробової частини. Чим менший розряд тим менше він впливає на положення чи форму векторних об'єктів. Відкидання певної кількості розрядів до певної точності не вплине на якість зображення і тим самим зменшить розміри контейнера/стеганоконтейнера. Однак, відкидання досить великої кількості таких розрядів може унеможливити вилучення прихованої інформації з стеганоконтейнера. Тому, введемо обмеження на мінімально допустиму кількість таких розрядів $V_4 \geq 4$.

Слід зауважити, що параметр V_4 може бути збільшений для покращення ефективності та кількості вбудованої інформації у контейнер, але з врахуванням допустимого розміру одержаного стеганоконтейнера.

– *Параметр* V_5 – максимально допустима похибка при відтворенні початкової кривої, що впливає на коректність вилучення інформації з стеганоконтейнера. При розбитті/відтворенні початкових кривих з накладеною точністю координат V_4 буде присутня похибка округлення, яка стосуватиметься останніх розрядів дробової частини координат опорних точок. Однак, встановлення досить великої похибки, враховуваної при відтворенні початкової кривої з сукупності сегментів, може суттєво вплинути на коректність вилучення приховуваної інформації. Тому, встановимо обмеження на максимально допустиму похибку $V_5 \leq 5 \cdot 10^{1-V_4}$.

Отже, у даному підрозділі визначено множину параметрів приховування інформації у векторні зображення, що дозволяють впливати на вибір контейнерів та на процес приховування даних у криві векторного зображення.

2.3. Метод побітового приховування інформації у векторні зображення

На особливостях побудови різноманітних кривих розроблено стеганографічний *метод побітового приховування інформації* у векторні зображення [21, 22, 24, 28, 29, 31, 63]. За яким приховування інформації відбувається наступним чином:

Крок 1. Секретне повідомлення $a=\{a_i\}$, $a_i \in \{0,1\}$, $i=\overline{1,h}$, a_i – біт секретного повідомлення, h – кількість біт повідомлення a, ділиться на частини $a=\left\{a_1^1,...,a_{V_3}^1,a_1^2,...,a_{V_3}^2,...,a_1^j,...,a_{V_3}^j\right\}$, $a^j=\left\{a_1^j,...,a_{V_3}^j\right\}$, $a_i^j \in \{0,1\}$, $i=\overline{1,V_3}$, $j=\overline{1,m}$, $m=h/V_3$, де m – кількість кривих ступеня V_1 з мінімальною допустиму відстанню між опорними точками V_2 із сукупності векторних зображень в які приховуються частини повідомлення. Повідомлення a попередньо може бути стиснуте, зашифроване та до нього можуть бути застосовані методи завадостійкого кодування і перевірки цілісності.

Крок 2. Для кожної послідовності a^j визначається стеганоключ Δt^j, $j=\overline{1,m}$ кроку зміни параметра t, де кожний $\Delta t^j < 1/V_3$.

Крок 3. Приховування кожної послідовності a_i^j, $i=\overline{1,V_3}$ в криву D_j, $j=\overline{1,m}$ виконується шляхом розбиття її на послідовність сегментів $D_j^* = D_{V_1}^0 \cup D_{V_1}^1 \cup ... \cup D_{V_1}^w$, де w – індекс послідовності сегментів кривої $D_j{}^*$, $w \in N$ (перед приховуванням $w=0$ і $D_{V_1}^0 = D_j$). Розбиття виконується при заданому параметрі t_i ($t_i = t_{i-1} + \Delta t^j$, де t_0 – довільне початкове значення, $0 \le t_0 < 1 - V_3 \cdot \Delta t^j$):

3.1 При приховуванні біта $a_i^j = 0$ ($a_i^j = 1$ – в залежності від вибору значення біта при якому відбувається поділ кривих) в заданій точці розбиття t_i крива не ділиться, а відбувається перехід до наступного біта a_{i+1}^j.

3.2 При приховуванні біта $a_i^j = 1$ ($a_i^j = 0$ – в залежності від вибору значення біта при якому відбувається поділ кривих) в заданій точці розбиття t_i виконується поділ кривої $D_{V_1}^w$ на два сегмента $D_{V_1}^w$ і $D_{V_1}^{w+1}$. Координати опорних точок отриманих сегментів розраховуються із обраною точністю V_4. Подальше внесення наступного біта a_{i+1}^j відбувається при наступному значенні t_{i+1} в отриманий сегмент $D_{V_1}^{w+1}$. Кожний поділ кривої призводить до збільшення кількості послідовності сегментів на один ($w = w+1$).

3.3. Після приховування послідовності a^j у D_j криву, отримана послідовність сегментів D_j^* записується до стеганоконтейнера замість D_j кривої.

Після приховування інформації одержаний стеганоконтейнер передається резервним каналом передачі інформації через мережу Інтернет, а закритим каналом передаються кроки Δt^j, $j = \overline{1,m}$, зміни параметра t. Для більшої ефективності обробки даних через закритий канал можуть передаватися відомості про положення сукупностей сегментів з прихованою інформацією в контейнері, останні значення параметра t_{i+1} для кожної послідовності та параметри з множини $\mathbf{V}$. Проте, дані параметри можуть бути заздалегідь зафіксовані.

На рис. 2.5 представлений приклад приховування 8 біт (наприклад, 10101010_2) в криву векторного зображення за побітовим методом, розбивши її на 4-ри сегмента.

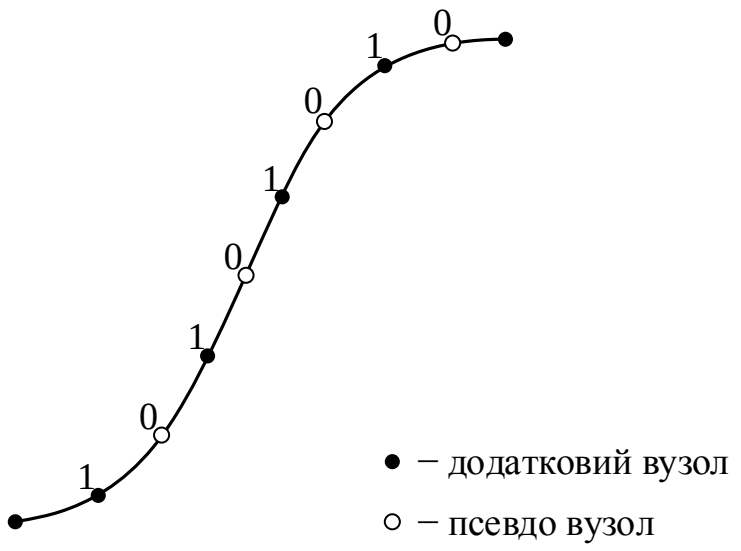


Рис. 2.5. Приклад приховування інформації за побітовим методом

Вилучення секретного повідомлення a^j ($a^j = \left\{a_1^j, ..., a_{V_3}^j\right\}$) за методом побітового приховування інформації виконується шляхом відтворення початкових кривих D_j з сукупностей сегментів D_j^*, $j = \overline{1,m}$, що здійснюється в процесі поступового об'єднання двох останніх кривих $D_{V_1}^{w-1}$ та $D_{V_1}^w$ в одну криву. Вилучення кожного біта a_i^j, $i = \overline{V_3,1}$, з послідовності сегментів D_j^*, $j = \overline{1,m}$, виконується наступним чином:

Крок 1. Визначається значення параметра $t_i = t_{i+1} - \Delta t^j$, $i = \overline{V_3,1}$, де початкове значення t_{V_3+1}, для кожної послідовності сегментів, визначається за формулою $t_{V_3+1} = \Delta t^j \cdot (V_3 + 1)$ або передається по захищеному каналі.

Крок 2. За отриманим значенням параметра t_i та опорними точками двох останніх кривих $D_{V_1}^{w-1}$ та $D_{V_1}^w$ обчислюються координати деякої опорної точки P_k. В результаті чого одержуються координати P_{k_1} та P_{k_2}.

Крок 3. В залежності від виконання нерівності $\left|P_{k_1} - P_{k_2}\right| \le V_5$, відбувається наступне:

3.1 При виконанні заданої нерівності виконується об'єднання двох останніх кривих послідовності сегментів в одну криву, при цьому кількість

кривих з сукупності сегментів зменшується на один ($w = w - 1$). У такому випадку $a_i^j = 0$ ($a_i^j = 1$ – в залежності від вибору значення біта при якому відбувається поділ кривих).

3.2 Якщо не виконується задана вище нерівність, то у такому випадку $a_i^j = 1$ ($a_i^j = 0$ – в залежності від вибору значення біта при якому відбувається поділ кривих), а кількість кривих не зменшується.

В даному підрозділі запропоновано стеганографічний метод побітового приховування інформації в структуру векторного зображення, який, за рахунок фіксування кроків Δt^j зміни параметра t, дозволяє вбудовувати інформацію в криві шляхом поступового поділу їх на сегменти в заданих точках розбиття.

2.4. Метод шаблонного приховування інформації у векторні зображення

Запропоновано стеганографічний *метод шаблонного приховування даних* у векторні зображення, який оперуватиме вже не бітами інформації, а цілими блоками (серіями) біт [20, 23, 25, 26, 32, 87]. Основною відмінністю шаблонного методу є можливість визначати наперед різні кроки зміни параметра t відповідно до кожного значення елемента шаблону (блоку біт, що буде приховуватись). В свою, чергу це дозволить приховувати цілий блок бітів лише за одне розбиття кривої. Таблиця значень шаблону буде відігравати роль стеганоключа, необхідного для приховування та відтворення даних з стеганоконтейнера [26].

Значення елементів шаблону, де кожному її елементу ставитиметься у відповідність свій крок зміни параметра t, будуть задаватися наступним співвідношенням [20, 23, 26]:

$$TV_l^k \to T\Delta t^k,$$

де k − індекс значень елементів шаблону, $k=\overline{1,2^l}$, TV_l^k − значення одного елементу шаблону, l − кількість біт одного значення елементу шаблону, $T\Delta t^k$ − відповідний крок зміни параметра t для приховування TV_l^k.

В табл. 2.1 наведений приклад задання таблиці співвідношення значень елементів шаблону з різними кроками зміни параметра побудови кривої t, при кількості біт в одному блоці $l=2$ та кількості елементів шаблону $k=4$.

Таблиця 2.1

Приклад співвідношення значень елементів шаблону з різними кроками побудови кривої

Індекс k	Шаблон TV_l^k, біт	Крок зміни $T\Delta t^k$
1	00	0,002
2	01	0,004
3	10	0,006
4	11	0,008

Приховування інформації за шаблонним методом відбувається наступним чином [20, 23, 25, 26, 32, 87]:

Крок 1. Секретне повідомлення $a=\{a_i\}$, $a_i\in\{0,1\}$, $i=\overline{1,h}$, a_i − біт секретного повідомлення, h − кількість біт повідомлення a, ділиться на частини $a=\{a_1^1,...,a_{V_3}^1,a_1^2,...,a_{V_3}^2,...,a_1^j,...,a_{V_3}^j\}$, $a^j=\{a_1^j,...,a_{V_3}^j\}$, $a_i^j\in\{0,1\}$, $i=\overline{1,V_3}$, $j=\overline{1,m}$, $m=h/V_3$ де m − кількість допустимих кривих ступеня V_1 з мінімальною допустиму відстанню між опорними точками V_2 із сукупності векторних зображень в які приховуються частини повідомлення. Після чого, кожна a^j послідовність ділиться на блоки $a^j=\{a_1^l,a_2^l,...,a_i^l...,a_z^l\}$, $z=V_3/l$, де a_i^l − i-та частина блоку a^j довжиною l біт, $i=\overline{1,z}$. Кожному a_i^l відповідає свій елемент з таблиці шаблонів TV_l^k, $k=\overline{1,2^l}$. Повідомлення a попередньо може бути стиснуте, зашифроване та до нього можуть бути застосовані методи завадостійкого кодування і перевірки цілісності.

Крок 2. Визначається максимально допустиме значення $\max\Delta t=l/V_3$ та встановлюється кожному елементу TV_l^k шаблону власний крок $T\Delta t^k$, $k=\overline{1,2^l}$,

зміни параметра побудови кривої t, де кроки $T\Delta t^k$ не повторюються та $T\Delta t^k \le \max \Delta t$.

Крок 3. Виконується приховування кожного a^j секретного повідомлення в криву D_j, $j=\overline{1,m}$, шляхом розбиття її на послідовність сегментів $D_j^* = D_{V_1}^0 \cup D_{V_1}^1 \cup ... \cup D_{V_1}^w$, де w – індекс послідовності створених сегментів кривої D_j^*, $w \in N$ (перед приховуванням $w=0$ і $D_{V_1}^0 = D_j$), при різних значеннях параметра t:

3.1. Приховування кожного елементу a_i^l ($a^j = \left\{a_1^l, a_2^l, ..., a_i^l ..., a_z^l\right\}$, $i=\overline{1,\mathrm{z}}$) послідовності a^j при певному значенні t_i ($t_i = t_{i-1} + T\Delta t^k$, де t_0 – довільне початкове значення, $0 \le t_0 < 1 - z \cdot \max \Delta t$, $T\Delta t^k$ відповідає кроку зміни параметра t для приховування a_i^l) відбувається шляхом розбиття кривої $D_{V_1}^w$ на два сегмента $D_{V_1}^w$ і $D_{V_1}^{w+1}$ в точці t_i. Координати опорних точок отриманих сегментів розраховуються із обраною точністю V_4. Подальше внесення наступних елементів a_{i+1}^l шаблону з послідовності a^j буде відбуватися при наступному значенні точки розбиття t_{i+1} в отриманий другий сегмент $D_{V_1}^{w+1}$ Кожне розбиття кривої збільшує кількості кривих в послідовності сегментів на один ($w = w+1$).

3.2. Після приховування послідовності a^j у D_j криву, отримана послідовність сегментів $D_j^* = D_{V_1}^0 \cup D_{V_1}^1 \cup ... \cup D_{V_1}^w$ записується до стеганоконтейнера замість D_j кривої.

Після приховування інформації одержаний стеганоконтейнер передається резервним каналом передачі інформації через мережу Інтернет, а закритим каналом передається таблиця співвідношень значень елементів шаблону TV_l^k з різними кроками $T\Delta t^k$, $k=\overline{1,2^l}$, зміни параметра побудови кривої t, останні значення параметра t_i для кожної послідовності сегментів та початкове значення t_0. Для більшої ефективності обробки даних через закритий

канал можуть передаватися відомості про положення сукупностей сегментів з прихованою інформацією в контейнері та параметри з множини **V**. Проте, таблиця співвідношень значень елементів шаблону, відомості про положення сукупностей сегментів та параметри з множини **V** можуть бути заздалегідь зафіксовані.

На рис. 2.6 представлений приклад приховування 8 біт (наприклад, 10101010_2) в криву векторного зображення за шаблонним методом, розбивши її на 4-и сегмента та приховавши за одне розбиття по 2 біта.

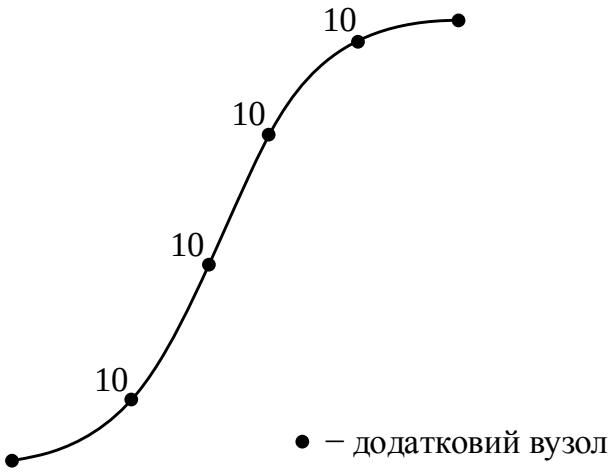


Рис. 2.6. Приклад приховування інформації за шаблонним методом

Вилучення секретного повідомлення a^j ($a^j = \{a_1^l, a_2^l, ..., a_i^l ..., a_z^l\}$) за методом шаблонного приховування інформації виконується шляхом відтворення початкових кривих D_j з сукупностей сегментів D_j^*, $j = \overline{1,m}$, що здійснюється в процесі поступового об'єднання двох останніх кривих $D_{V_1}^{w-1}$ та $D_{V_1}^w$ в одну криву. Вилучення кожного блоку a_i^l, $i = \overline{V_3 / l, 1}$, з послідовності сегментів D_j^*, $j = \overline{1,m}$, виконується наступним чином:

Крок. 1. За отриманим значенням параметра t_i (який передається закритим каналом) та опорними точками двох останніх сегментів $D_{V_1}^{w-1}$ та $D_{V_1}^w$ виконується їх об'єднання в одну криву.

Крок 2. Перебираються усі кроки $T\Delta t^k$, $k = \overline{1,2^l}$, зміни параметра побудови кривої t, для яких виконується наступна послідовність дій:

2.1. Визначається значення параметра $t_i = t_{i+1} - T\Delta t^k$, $i = \overline{V_3 / l - 1,1}$.

2.2. За отриманим значенням параметра t_i та опорними точками двох останніх кривих $D_{V_1}^{w-1}$ та $D_{V_1}^{w}$ обчислюються координати деякої опорної точки P_k. В результаті чого одержуються координати P_{k_1} та P_{k_2}.

2.3. В залежності від виконання нерівності $\left|P_{k_1} - P_{k_2}\right| \le V_5$, відбувається наступне:

2.3.1. При виконанні заданої нерівності вилучається відповідний прихований блок $a_i^l = TV_l^k$, який відповідає кроку зміни $T\Delta t^k$ використовуваного при обчисленні значенням параметра t_i, та виконується об'єднання двох останніх кривих послідовності сегментів в одну криву зменшуючи кількості кривих в сукупності сегментів на один ($w = w - 1$).

2.3.2. Якщо не виконується задана нерівність, то виконується перехід до наступного кроку $T\Delta t^k$ поки не буде підібрано такий крок при якому відбудеться об'єднання двох кривих з послідовності сегментів.

Крок. 3. Вилучення останнього блоку a_i^l виконується шляхом перебирання усіх крокі $T\Delta t^k$, $k = \overline{1,2^l}$, доки не буде виконуватися рівність $t_i - T\Delta t^k = t_0$.

В даному підрозділі запропоновано стеганографічний метод шаблонного приховування інформації в структуру векторного зображення, який, за рахунок визначення таблиці співвідношень значень елементів шаблону TV_l^k з різними кроками $T\Delta t^k$ зміни параметра побудови кривої t, дозволяє вбудовувати цілі блоки біт секретної інформації в криві шляхом поділу їх на сукупності сегментів.

2.5. Розробка структурної моделі процесу прихованої передачі інформації резервним каналом зв'язку

Теоретико-множинна модель процесу прихованої передачі інформації

резервним каналом зв'язку, що дозволяє вбудовувати інформацію у криві векторних зображень з урахуванням стійкості до атак на основі афінних перетворень (застосовуваних до стеганоконтейнерів у відкритих каналах передачі даних), описується наступним чином [12, 13, 35, 41, 44]:

$$\mathbf{MVZ} = \left\{\mathbf{M}, \mathbf{I}, \mathbf{K}, \mathbf{V}, \mathbf{S}, \mathbf{T}, \mathbf{S}^*, F_M, F_N, F_I, F_H, F_S, F_L, F_E\right\}, \qquad (2.2)$$

де $\mathbf{M} = \{M_a\}$, $a = \overline{1, A}$ – множина приховуваних повідомлень;

A – кількість повідомлень у множині $\mathbf{M}$;

$\mathbf{I} = \{I_b\}$, $b = \overline{1, B}$ – множина векторних зображень (контейнерів);

B – кількість контейнерів у множині $\mathbf{I}$;

$\mathbf{K} = \{K_c\}$, $c = \overline{1, C}$ – множина стеганоключів;

C – кількість стеганоключів у множині $\mathbf{K}$;

$\mathbf{V} = \{V_j\}$, $j = \overline{1, 5}$ – множина параметрів приховування інформації (див. підрозділ 2.2);

$\mathbf{S} = \{S_x\}$, $x = \overline{1, X}$ – множина стеганоконтейнерів;

X – кількість стеганоконтейнерів у множині $\mathbf{S}$;

$\mathbf{T} = \{T_y\}$, $y = \overline{1, 4}$ – множина атак на основі афінних перетворень: T_1 – перенесення, T_2 – поворот, T_3 – зсув, T_4 – масштабування;

$\mathbf{S}^* = \{S_x^*\}$, $x = \overline{1, X}$ – множина одержаних стеганоконтейнерів через відкритий канал зв'язку;

X – кількість стеганоконтейнерів у множині $\mathbf{S}^*$;

$F_M(\mathbf{M}) : \mathbf{M} \to \mathbf{P}$ – функція перетворення (оброблення) приховуваних повідомлень з множини $\mathbf{M}$;

$F_N(\mathbf{P}, \mathbf{V}) : \mathbf{P} \times \mathbf{V} \to \mathbf{D}$ – функція поділу перетворених повідомлень з множини $\mathbf{P}$ на частини за визначеним параметром V_3;

$F_I(\mathbf{I}, \mathbf{V}) : \mathbf{I} \times \mathbf{V} \to \mathbf{G}$ – функція відбору допустимих з множини $\mathbf{I}$ контейнерів для приховування інформації за встановленими параметрами V_1 та

V_2;

$F_H(\mathbf{D},\mathbf{G},\mathbf{V},\mathbf{K}):\mathbf{D}\times\mathbf{G}\times\mathbf{V}\times\mathbf{K}\to\mathbf{S}$ – функція приховування інформації у відібрані контейнери з множини $\mathbf{G}$ за розробленими методами;

$F_S(\mathbf{S}^*,\mathbf{V},\mathbf{K}):\mathbf{S}^*\times\mathbf{V}\times\mathbf{K}\to\mathbf{D}$ – функція вилучення прихованої інформації з множини одержаних стеганоконтейнерів $\mathbf{S}^*$ за розробленими методами;

$F_L(\mathbf{D}):\mathbf{D}\to\mathbf{P}$ – функція збирання частин перетворених повідомлень в одне ціле;

$F_E(\mathbf{P}):\mathbf{P}\to\mathbf{M}$ – функція відтворення прихованих повідомлень.

Запропонованій теоретико-множинній моделі процесу прихованої передачі інформації резервним каналом зв'язку відповідає наступна структурна модель представлена на рис. 2.7.

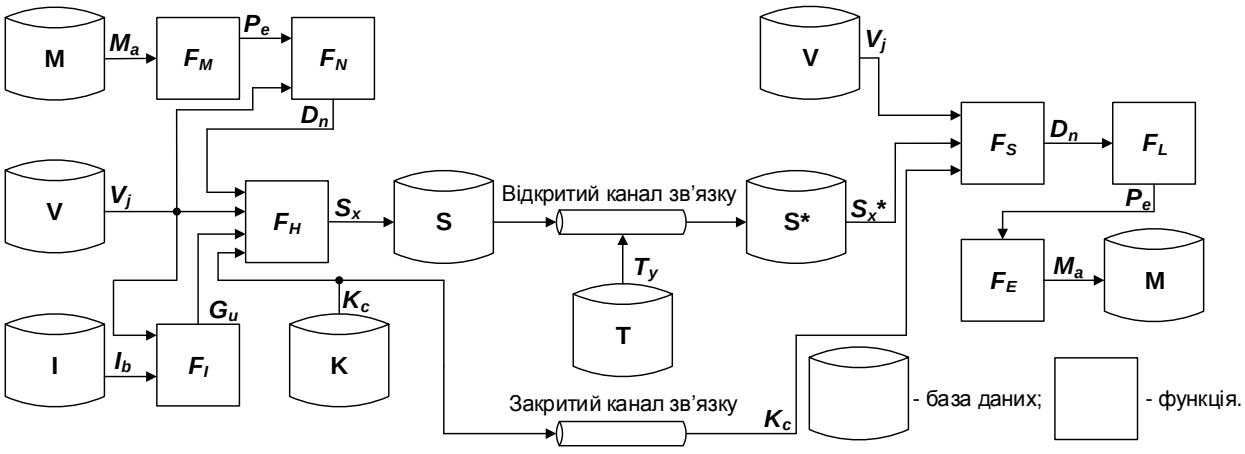


Рис. 2.7. Структурна модель прихованої передачі інформації

Тоді, сам процес прихованої передачі інформації резервним каналом зв'язку буде наступним:

1. За допомогою функції $F_M(\mathbf{M})$ приховувані повідомлення з множини $\mathbf{M}$ можуть піддаватися:

а) стисненню, для зменшення кількості приховуваної інформації [60];

б) криптографічним операціям (накладання цифрового електронного підпису, шифрування та створення хеш-образу) для підвищення захищеності прихованої інформації [4, 27, 72, 73];

в) внесенню завадостійкого кодування, для запобігання втрати частин приховуваної інформації при її передачі [33, 40].

Функція F_M змінює розмір прихованої інформації (наприклад, при використанні симетричного шифрування вноситься надлишковість за рахунок вирівнювання блоків даних), а її результатом являється множина оброблених повідомлень $\mathbf{P}$.

2. За допомогою функції $F_N(\mathbf{P},\mathbf{V})$ оброблені повідомлення з множини $\mathbf{P}$ діляться на частини за встановленим параметром приховування V_3. Результатом виконання функції F_N є одержання множини частин оброблених повідомлень $\mathbf{D}$.

3. За допомогою функції $F_I(\mathbf{I},\mathbf{V})$, при встановлених параметрах V_1 та V_2, визначаються допустимі контейнери з множини $\mathbf{I}$ в структуру яких буде приховуватися інформація. В результаті виконання функції F_I одержується множина допустимих контейнерів $\mathbf{G}$.

4. За допомогою функції $F_H(\mathbf{D},\mathbf{G},\mathbf{V},\mathbf{K})$ виконується приховування частин оброблених повідомлень $\mathbf{D}$ у відібрані контейнери $\mathbf{G}$ при використанні визначених стеганоключів $\mathbf{K}$ та параметрів приховування $\mathbf{V}$. Результатом виконання функції F_H є одержання множини стеганоконтейнерів $\mathbf{S}$.

Далі, одержанні стеганоконтейнери з множини $\mathbf{S}$ передаються відкритим каналом зв'язку, де на них можуть накладатися афінні перетворення з множини $\mathbf{T}$. В результаті чого, на приймаючій стороні одержуються змінені (модифіковані) стеганоконтейнери $\mathbf{S}^*$. Закритим каналом зв'язку передаються використовувані при вбудовуванні інформації стеганоключі $\mathbf{K}$ та параметри приховування $\mathbf{V}$. Параметри з множини $\mathbf{V}$ можуть бути заздалегідь зафіксованими та не передаватися між сторонами обміну інформації.

5. За допомогою функції $F_S(\mathbf{S}^*,\mathbf{V},\mathbf{K})$ виконується вилучення прихованих частин оброблених повідомлень з множини $\mathbf{S}^*$ при використанні

стеганоключів $\mathbf{K}$ та параметрів приховування $\mathbf{V}$. В результаті виконання функції F_S одержується множина частин оброблених повідомлень $\mathbf{D}$.

6. За допомогою функції $F_L(\mathbf{D})$ виконується об'єднання частин оброблених повідомлень в єдине ціле. Результатом даної функції F_L є одержання множини оброблених повідомлень $\mathbf{P}$.

7. За допомогою функції $F_E(\mathbf{P})$ виконується відтворення прихованих повідомлень, в результаті чого одержується множина $\mathbf{M}$.

У функції $F_M(\mathbf{M})$, як вже зазначалось, може попередньо виконуватися шифрування секретного повідомлення $\mathbf{M}$ криптографічними алгоритмами. Зокрема, можуть використовуватися симетричні блокові шифри Luna і Neptun [4, 27]. Псевдокоди процедури зашифровування яких наведено на рис. 2.8 та рис. 2.9. Дані шифри працюють із 128-бітними блоками даних з підтримкою секретного ключа довжиною 128, 256 та 512 бітів. Кількість раундів шифрування r залежить від довжини секретного ключа. При довжині секретного ключа 128, 256 та 512 біт у шифрі Luna $r = 7,\ 9,\ 17$, а у шифрі Neptun $r = 7,\ 9,\ 19$ відповідно.

Luna
Input: 128-бітний вхідний блок даних $state$,
128-бітні раунд. ключі $subkey[i]$, $i = \overline{0,r}$.
Output: 128-бітний вихідний блок даних.

1. $AddKeyMod2(state, subkey[0])$;
2. $For \quad j = 0, \quad j < r-1, \quad j{+}{+} \quad do$
2.1. $SubBytes_{Luna}(state)$;
2.2. $ShiftRows(state)$;
2.3. $MixColumns(state)$;
2.4. $AddKeyMod2(state, subkey[j+1])$;
3. $SubBytes_{Luna}(state)$;
4. $ShiftRows(state)$;
5. $AddKeyMod2(state, subkey[r])$;
6. $return\ state$.

Рис.2.8. Псевдокод процедури шифрування алгоритму шифрування Luna

Neptun
Input: 128-бітний вхідний блок даних $state$,
128-бітні раунд. ключі $subkey[i]$, $i=\overline{0,r+1}$.
Output: 128-бітний вихідний блок даних.

1. $AddKeyMod2(state, subkey[0])$;
2. $For \quad j=0, \quad j<r-1, \quad j++ \quad do$
2.1. $SubBytes_{Neptun}(state, subkey[r+1])$;
2.2. $ShiftRows(state)$;
2.3. $MixColumns(state)$;
2.4. $AddKeyMod2(state, subkey[j+1])$;
3. $SubBytes_{Neptun}(state, subkey[r+1])$;
4. $ShiftRows(state)$;
5. $AddKeyMod2(state, subkey[r])$;
6. *return state* .

Рис.2.9. Псевдокод процедури шифрування алгоритму шифрування Neptun

Операція $AddKeyMod2(x, y)$ виконує побітове додавання за модулем 2 відповідних бітів x та y. Операція $MixColumns(x)$ являє собою лінійне перетворення матриці x [4, 27]. У даній операції блок даних x розбивається на дві частини по 8 байт, кожна з яких розглядається як поліном над полем $GF(2^8)$ з 8 термами, який перемножується за модулем x^8+1 з фіксованим поліномом $c(x)$ степені 7. В операціях $SubBytes_{Luna}(x)$ і $SubBytes_{Neptun}(x, y)$ виконується таблична заміна відповідно кожних 16 та 8 біт блоку даних x [4, 27]. У шифрі Luna використовується одна таблиця замін на множині V_{16} ($V_n = \{0,1\}^n$), а у шифрі Neptun − 16 таблиць на множині V_8, при чому вибір конкретної таблиці у кожному раунді залежить від раундового ключа y.

Блокові шифри Luna та Neptun забезпечують високу швидкість шифрування інформації, що не суттєво вплине на швидкість вбудовування даних.

Отже, у даному підрозділі розроблена структурна моделі процесу прихованої передачі інформації резервним каналом зв’язку. Вбудовування

інформації відбувається у криві векторних зображень за використанням стеганоключів та параметрів приховування з множини $\mathbf{V}$.

2.6. Висновки до другого розділу

У другому розділі дисертаційної роботи були отримані наступні результати:

1. Запропоновано принцип вбудовування інформації у векторні зображення, який, за допомогою особливостей побудови та властивостей точково-заданих кривих, забезпечить стійкість до афінних перетворень. Основна ідея якого полягає на можливості розбивати та подавати криві різного ступеня через сукупності сегментів, в яких міститиметься прихована інформація. Для цього фіксується деякий крок Δt зміни параметра побудови кривої t, при певних значеннях якого може відбуватися розбиття кривої на два сегмента, що візуально виглядатимуть однаково, і таким чином приховуватися секретна інформація.

2. Визначені параметри приховування інформації у криві векторного зображення, які дозволяють впливати на вибір допустимого контейнера (види кривих різного ступеня, їх допустима довжина відносно опорних точок) та на процес вбудовування інформації при розбитті кривих на сукупності сегментів (точність координат опорних точок, допустима похибка при відтворенні інформації та кількості вбудовуваних даних в одну криву).

3. Розроблено стеганографічний метод побітового приховування інформації у векторні зображення, який за рахунок фіксування різних кроків Δt^j зміни параметра побудови кривої t, дозволяє вбудовувати інформацію в криві шляхом поступового їх поділу на сукупності сегментів.

4. Розроблено стеганографічний метод шаблонного приховування інформації у векторні зображення, який за рахунок визначення таблиці співвідношень значень елементів шаблону TV_l^k з різними кроками $T\Delta t^k$ зміни

параметра побудови кривої t, дозволяє вбудовувати цілі блоки біт секретної інформації в криві шляхом поступового поділу їх на сегменти.

5. Запропоновано структурну модель процесу прихованої передачі інформації резервним каналом зв’язку, що дозволяє формувати множину стеганоконтейнерів, стійких до афінних перетворень.

РОЗДІЛ 3

АЛГОРИТМИ ПРИХОВУВАННЯ ІНФОРМАЦІЇ У КРИВІ БЕЗ'Є ВЕКТОРНИХ ЗОБРАЖЕНЬ

3.1. Дослідження типів кривих векторної графіки

Для роботи з векторними зображеннями використовуються спеціальні графічні пакети: Adobe Illustrator, CorelDRAW, Adobe Flash Professional, AutoCAD, KOMPAS-3D, Autodesk 3ds Max, Inkscape та інші. Вони підтримують велику кількість форматів представлення даних [21]: AI, CDR, CDW, CDT, DWG, DXF, WMF, FLA, FH, SVG, SWF, 3DM та інші. В структурі кожного векторного формату використовуються різні типи кривих. Для приховування інформації за побітовим та шаблонним методами (див. підрозділ 2.3, 2.4) дані криві повинні володіти властивістю інваріантності до афінних перетворень та можливістю поділу на сегменти або побудови складених кривих. До них слід віднести криві Без'є, B-сплайни, інтерполяційні криві Ерміта [5, 39, 43, 57, 75, 80]. В табл. 3.1 представленні властивості на прикладі різних типів кубічних кривих [43].

Оскільки, в роботі розв'язується задача приховування інформації у векторні зображення, то слід використовувати найпоширеніші криві, що підтримуються більшістю графічними пакетами, а саме криві Без'є. Інші типи кривих можуть використовуватися в менш поширених форматах представлення зображень, наприклад: формат 3DM використовує NURBS-криві.

Крива Без'є – це параметрична крива, яка задається наступним рівнянням [5, 26, 39, 43, 57, 63]:

$$B(t)=\sum_{i=0}^{n} b_{i,n}(t)P_i,\ t=t+\Delta t,\ t\in[0,1] \qquad (3.1),$$

де P_i – опорні точки, $i\in\overline{0,n}$, i – індекс опорних точок, n – ступінь поліноміальної кривої, порядок визначальної функції базису Бернштейна і кількості її сегментів, $n+1$ – кількість опорних точок, t – параметр побудови

кривої, Δt – крок зміни параметра побудови кривої t, $b_{i,n}(t)$ – поліноми Бернштейна, що визначаються за наступним рівнянням:

$$b_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}, \quad \binom{n}{i} = \frac{n!}{i!(n-i)!} \qquad (3.2).$$

Таблиця 3.1

Властивості типів кубічних кривих векторної графіки

Назва	Властивості
Криві Без'є	1. Крива Без'є інваріантна відносно афінних перетворень. 2. Складена кубічна крива Без'є – неперервна крива γ, що є об'єднанням елементарних кубічних кривих: $\gamma = \gamma^0 \cup \gamma^1 \cup ... \cup \gamma^l$, де l – кількість елементарних кубічних кривих, кожна з яких містить по чотири опорних точки з множини $\mathbf{P}$, так, що остання точка попередньої кривої – перша точка наступної кривої.
Інтерполяційні криві Ерміта	1. Кубічна крива Ерміта – афінно-інваріантна, але не проективно інваріантна. 2. Складена кубічна крива Ерміта – крива γ, що визначається масивом опорних точок з множини $\mathbf{P} = \{P_i\}$ і довільним набором ненульових векторів з множини $\mathbf{Q} = \{Q_i\}$, $i = \overline{0,m}$, $m \geq 1$, m – кількість опорних точок та векторів, яку можна подати у вигляді об'єднання елементарних кубічних кривих Ерміта: $\gamma = \gamma^0 \cup \gamma^1 \cup ... \cup \gamma^m$.
B-сплайни та NURBS-криві	1. Кубічна B-сплайнова крива є афінно-інваріантною. 2. Складена кубічна B-сплайнова крива – крива γ, що визначається масивом опорних точок з множини $\mathbf{P} = \{P_i\}$, $i = \overline{0,m}$, m – кількість опорних точок, яку можна подати у вигляді об'єднання елементарних B-сплайнових кривих: $\gamma = \gamma^1 \cup \gamma^2 \cup ... \cup \gamma^{m-2}$, де γ^j, $j = \overline{1,m-2}$ будується як елементарний B-сплайн на точках P_{j-1}, P_j, P_{j+1}, P_{j+2}. 3. Нерівномірні раціональні B-сплайни називаються NURBS-кривими, які зберігають властивості B-сплайнів та володіють властивістю афінно-інваріантності.

У векторній графіці, криві Без'є використовуються для моделювання гладких кривих, застосовуваних при побудові текстів, рухливих анімацій, векторизації растрових зображень [10, 21, 102]. Метод побудови кривої Без'є в графічних пакетах базується на використанні дотичних управляючих ліній, що

проведені до сегмента кривої на його кінцях (вузлах) [43]. При цьому форму кривої Без'є моделюють шляхом зміни нахилу дотичних і довжини відрізка, які визначаються контрольними точками. Криві Без'є характеризуються ступенем порядку, тобто кількістю опорних точок з множини **P** необхідних для їх представлення [39, 43]. Кількість опорних точок множини **P** суворо визначає порядок побудови кривої Без'є. Довільна точка на кривій Без'є залежить від всіх опорних точок, тому при зміні однієї точки змінюється вся крива [43].

Криві Без'є будуються за наступним алгоритмом (рис. 3.1) [43]:

1. Задані опорні точки кривої Без'є послідовно з'єднуються відрізками, утворюючи дотичні лінії.

2. Кожна сторона дотичної лінії ділиться у відношенні параметра побудови t, де кожне значення параметра $t \in [0,1]$ є точка поділу.

3. Точки поділу з'єднуються відрізками прямих і утворюють нову дотичну лінію. При цьому кількість вузлів нового сегменту на одиницю менша за кількість вузлів попереднього сегменту.

4. Сторони нового сегменту знову діляться у тому ж відношенні. Цей процес продовжується доти, поки не буде отримана єдина точка, що й буде точкою кривої Без'є.

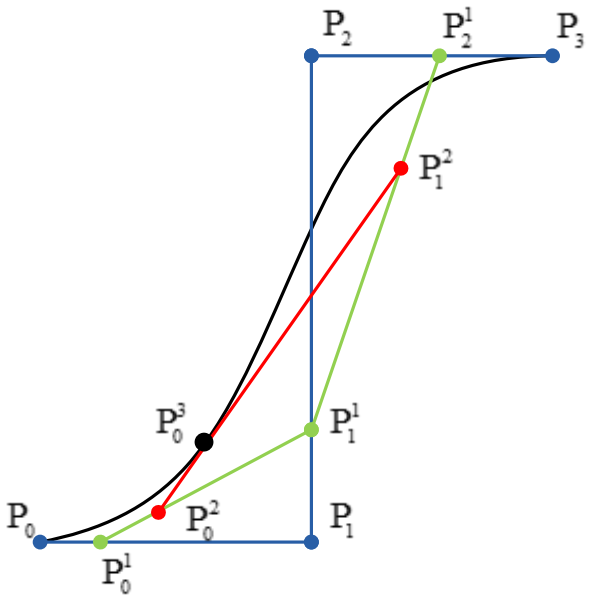


Рис. 3.1. Приклад побудови кривої Без'є третього ступеня

Властивості кривих Без'є, що наведені в табл. 3.1, можна використати для приховування секретного повідомлення в них. Так, другу властивість слід використати для вбудовування інформації у криві Без'є векторного зображення, при заміні початкових кривих сукупністю їх сегментів, що будуть формуватися певним чином на основі приховуваних даних та стеганоключів. За допомогою першої властивості буде забезпечуватися стійкість до активних атак на основі афінних перетворень.

Розбиття кривої Без'є на сегменти можна здійснити за алгоритмом де Кастельжо [5, 26, 39, 63, 80]. Розбиття кривої Без'є n-ступеня на два сегменти проходитиме в n кроків, де на кожному кроці обраховуватимуться допоміжні координати точок необхідні для відшукання заданої точки розбиття [21, 39]. На рис. 3.2 представлена схематична ілюстрація алгоритму де Кастельжо.

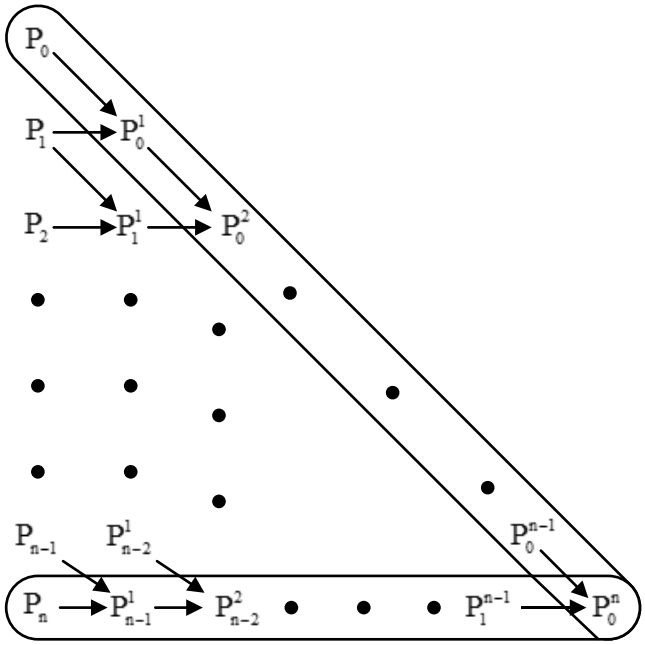


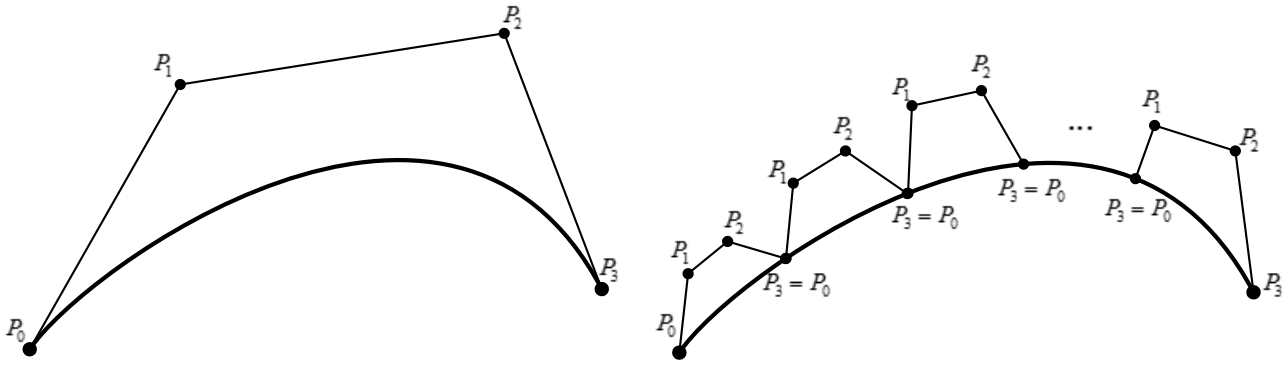
Рис. 3.2. Схематична ілюстрація алгоритму де Кастельжо

Описати аналітично алгоритм де Кастельжо можна так [26, 39]: координати точки $B(t)$ на кривій Без'є при певному значенні параметра t рівні координатам точки P_0^x ($B(t) = P_0^x$), $x = n+1$ – кількість опорних точок кривої, що обчислюється за наступною рекурсивною формулою:

$$P_i^r = (1-t)P_i^{r-1} + tP_{i+1}^{r-1},\ r \in \overline{1,x},\ i \in \overline{0,x-r}, \qquad (2.3)$$

де початкові значення P_i^0 – координати опорних точок P_i ($P_i^0 = P_i$).

На рис. 3.3 наведена схематична ілюстрація представлення кривої Без'є третього ступеня через сукупність сегментів.



а *б*

Рис. 3.3. Представлення кривої Без'є третього ступеня через сукупність сегментів: *а* – початкова крива; *б* – сукупність сегментів

Алгоритм де Кастельжо також можна застосувати для відтворення початкових кривих з сукупності сегментів [39] і таким чином здійснювати вилучення прихованої інформації.

Отже, у даному підрозділі виділені основні типи кривих, які володіють властивістю інваріантності до афінних перетворень та можливістю поділу на сегменти. Особлива увага була приділена кривим Без'є, що підтримуються більшістю графічними пакетами. Були розглянуті способи побудови кривих Без'є та розбиття їх на сукупності сегментів за допомогою алгоритму де Кастельжо, що дозволяє приховувати/відтворювати секретну інформацію при роботі з векторними зображеннями, де використовуються криві Без'є.

3.2. Процес вбудовування даних у криві Без'є за побітовим методом приховування інформації

На основі запропонованого методу **побітового приховування інформації** (див. підрозділ 2.3) реалізовано алгоритм вбудовування даних StegoBIT у криві Без'є третього ступеня [21, 22, 24, 28, 29, 31, 63].

Даний алгоритм дозволяє приховувати секретні повідомлення у криві Без'є векторних зображень, забезпечуючи при цьому стійкість до активних атак на основі афінних перетворень. У якості повідомлення можуть бути вбудовані будь-які двійкові дані, які попередньо можуть бути стиснуті [60], зашифровані [4, 11, 27, 72, 73], до них можуть бути застосовані методи завадостійкого кодування і перевірки цілісності [33, 40, 72, 73].

Приховування повідомлень відбувається при поступовому поділі початкових кривих на сегменти, при цьому саме повідомлення впливає на процес сегментації (розбиття кривих на сегменти відбувається лише при вбудовуванні нульового/одиничного біта приховуваної послідовності даних). При поділі кривих на сегменти зображення візуально не змінюється, а стійкість до афінних перетворень забезпечується властивістю афінно-інваріантності кривих Без'є. У якості контейнера може бути використане будь-яке векторне зображення, у складі якого є криві Без'є третього ступеня.

Слід зауважити, що в даній роботі не досліджується стійкість до пасивних атак розробленого алгоритму. Проте, якщо вбудовувати у одне векторне зображення повідомлення невеликого розміру (після приховування інформації одержаний стеганоконтейнер не перевищуватиме середньостатистичного розміру векторного зображення), то виявити факт присутності секретного повідомлення буде досить складно.

Вхідними даними для реалізації алгоритму за методом побітового приховування інформації є:

1. Сукупність векторних зображень, у структурі яких є криві Без'є.
2. Параметри приховування інформації у векторні зображення з множини $\mathbf{V}$ ($\mathbf{V} = \{V_j\}$, $j \in \overline{1,5}$) (див. підрозділ 2.2), що визначатимуть: ступінь кривих $V_1 = 3$ в які буде приховуватись інформація; допустиму відстань між опорними точками V_2 кожної кривої; максимальну кількість інформації V_3, що буде прихована в одну криву; точність координат опорних точок V_4; максимально допустиму похибку V_5 при відтворенні початкової кривої.

Приховування інформації у криві Без'є третього ступеня відбуватиметься за алгоритмом StegoBIT наступним чином [21, 22, 24, 28, 29, 31, 63]:

Крок 1. Секретне повідомлення $a=\{a_i\}$, $a_i\in\{0,1\}$, $i=\overline{1,h}$, a_i – біт секретного повідомлення, h – кількість біт повідомлення a, ділиться на частини $a=\{a_1^1,...,a_{V_3}^1,a_1^2,...,a_{V_3}^2,...,a_1^j,...,a_{V_3}^j\}$, $a^j=\{a_1^j,...,a_{V_3}^j\}$, $a_i^j\in\{0,1\}$, $i=\overline{1,V_3}$, $j=\overline{1,m}$, $m=\lceil h/V_3\rceil$, де m – кількість кривих Без'є із сукупності векторних зображень, в які приховуються частини повідомлення.

Крок 2. Для кожної послідовності a^j визначається стеганоключ Δt^j, $j=\overline{1,m}$ кроку зміни параметра побудови кривої t, де кожний $\Delta t^j<1/V_3$.

Крок 3. Приховування кожної послідовності a_i^j, $i=\overline{1,V_3}$ в криву D_j, $j=\overline{1,m}$ виконується шляхом розбиття її на послідовність сегментів $D_j^*=D_{V_1}^0\cup D_{V_1}^1\cup...\cup D_{V_1}^w$, де w – індекс послідовності сегментів кривої D_j^*, $w\in N$ (перед приховуванням $w=0$ і $D_{V_1}^0=D_j$), $V_1=3$ – ступінь кривої Без'є, що впливає на кількість операцій необхідних для розбиття кривої. Розбиття виконується при заданому параметрі t_i ($t_i=t_{i-1}+\Delta t^j$, де t_0 – довільне початкове значення, $0\le t_0<1-V_3\cdot\Delta t^j$):

3.1. При приховуванні біта $a_i^j=0$ ($a_i^j=1$ – в залежності від вибору значення біта, при якому відбувається поділ кривих) в заданій точці розбиття t_i крива Без'є не ділиться, а відбувається перехід до наступного біта a_{i+1}^j.

3.2. При приховуванні біта $a_i^j=1$ ($a_i^j=0$ – в залежності від вибору значення біта, при якому відбувається поділ кривих) в заданій точці розбиття t_i виконується поділ кривої Без'є $D_{V_1}^w$ з заданим індексом w на два сегмента $D_{V_1}^w$ і $D_{V_1}^{w+1}$ за алгоритмом де Кастельжо [5, 21, 39, 80]. Подальше внесення наступного біта a_{i+1}^j відбувається при наступному значенні t_{i+1} в отриманий сегмент $D_{V_1}^{w+1}$. Кожний поділ кривої Без'є призводить до збільшення кількості

послідовності сегментів на один ($w = w+1$), що відбувається при заданому значенні t_i наступним чином:

3.2.1. З початкової кривої $D_{V_1}^w$ заданої опорними точками P_0, P_1, P_2, P_3 обчислюються точки $P_0^1 = (1-t_i)P_0 + t_i P_1$, $P_1^1 = (1-t_i)P_1 + t_i P_2$ та $P_2^1 = (1-t_i)P_2 + t_i P_3$.

3.2.2. З отриманих координат точок P_0^1, P_1^1, P_2^1 розраховуються точки $P_0^2 = (1-t_i)P_0^1 + t_i P_1^1$ та $P_1^2 = (1-t_i)P_1^1 + t_i P_2^1$.

3.2.3. З отриманих координат точок P_0^2, P_1^2 отримуються координати останньої (шуканої) точки $P_0^3 = (1-t_i)P_0^2 + t_i P_1^2$.

3.2.4. За даними точками початкова крива Без'є ділиться на два сегмента, з яких перший сегмент $D_{V_1}^w$ будується по точкам P_0, P_0^1, P_0^2, P_0^3, а другий $D_{V_1}^{w+1}$ – по точкам P_0^3, P_1^2, P_2^1, P_3.

3.3. Після приховування послідовності a^j у D_j криву, отримана послідовність сегментів D_j^* записується до стеганоконтейнера замість D_j кривої.

Псевдокоди процедури вбудовування інформації за даним алгоритмом представлено на рис. 3.4.

Під операцією $SplitMessage(x, y)$ мається на увазі побітове розбиття вхідного повідомлення x на частини розмірністю y біт. Операція $SelectImage(x, y, z)$ виконує відбір допустимих контейнерів з множини x необхідних для вбудовування y частин прихованого повідомлення, в структурі яких містяться криві Без'є ступеня z. Операція $SelectCurves(x, y, z, h)$ визначає з множини зображень x кількість кривих Без'є ступеня z необхідних для вбудовування y частин прихованого повідомлення, що відповідають мінімальній допустимій відстані h між опорними точками. Операція $SelectKey(x, y)$ використовується для визначення з множини x допустимих

стеганоключів Δt^j, необхідних для вбудовування частин приховуваної інформації розмірності y в криві Без'є.

Input: Секретне повідомлення $a=\{a_i\}$, $a_i \in \{0,1\}$, $i=\overline{1,h}$, $h \in N$;
множина векторних зображень (контейнерів) $I=\{I_b\}$, $b \in N$;
множина стеганоключів $T=\{\Delta t^u\}$, $\Delta t^u \in (0,1)$, $u \in N$;
множина параметрів $V=\{V_i\}$, $i=\overline{1,4}$, $V_i \in N$, $V_1=3$;
значення біту при якому буде відбуватись поділ кривих Без'є Q, $Q \in \{0,1\}$.
Output: множина стеганоконтейнерів $S=\{S_x\}$, $x \in N$.
1. $\{a^j\}=SplitMessage(a,V_3)$, $a^j=\{a_1^j,...,a_{V_3}^j\}$, $a_i^j \in \{0,1\}$, $i=\overline{1,V_3}$, $j=\overline{1,m}$, $m=\lceil h/V_3 \rceil$;
2. $S=SelectImage(I,m,V_1)$, $S=\{S_x\}$, $x \in N$, $x \le m$, $m=\lceil h/V_3 \rceil$;
3. $D=SelectCurves(S,m,V_1,V_2)$, $D=\{D_j\}$, $j=\overline{1,m}$, $m=\lceil h/V_3 \rceil$;
2. $For\ (j=1;\ j \le m;\ j++)$
2.1. $\Delta t^j=SelectKey(T,V_3)$, $\Delta t^j < 1/V_3$;
2.2. $w=0$; $t=0$; $D_{V_1}^w=D_j$;
2.3. $For\ (i=1; i \le V_3; i++)$
2.3.1. $t=t+\Delta t^j$;
2.3.2. $if\ (a_i^j == Q)$
2.3.2.1. $\{P_e\}=GetPoints(D_{V_1}^w)$, $e=\overline{0,3}$;
2.3.2.2. $P_0^1=(1-t)\cdot P_0+t\cdot P_1$;
2.3.2.3. $P_1^1=(1-t)\cdot P_1+t\cdot P_2$;
2.3.2.4. $P_2^1=(1-t)\cdot P_2+t\cdot P_3$;
2.3.2.5. $P_0^2=(1-t)\cdot P_0^1+t\cdot P_1^1$;
2.3.2.6. $P_1^2=(1-t)\cdot P_1^1+t\cdot P_2^1$;
2.3.2.7. $P_0^3=(1-t)\cdot P_0^2+t\cdot P_1^2$;
2.3.2.8. $D_{V_1}^w=CreateCurve(P_0, P_0^1, P_0^2, P_0^3)$;
2.3.2.9. $D_{V_1}^{w+1}=CreateCurve(P_0^3, P_1^2, P_2^1, P_3)$;
2.3.2.10. $w=w+1$;
2.4. $D_j=ReplaceCurve(S, D_j, \{D_{V_1}^i\})$, $i=\overline{0,w}$;
3. $Return\ S$;

Рис. 3.4. Псевдокод процедури приховування інформації алгоритму StegoBIT

Операція $GetPoints(x)$ використовується для одержання координат опорних точок кривої x. Операція $CreateCurve(x, y, z, h)$ використовується для побудови кривої Без'є за координатами опорних точок x, y, z та h. Операція

$ReplaceCurve(x, y, z)$ виконує заміну в зображеннях x початкових кривих y на сукупності сегментів z, що містять приховану інформацію.

Вилучення інформації a^j ($a^j = \{a_1^j, ..., a_{V_3}^j\}$) виконується при відтворені початкових кривих Без'є D_j з сукупностей сегментів D_j^*, $j = \overline{1,m}$, що здійснюється в процесі поступового об'єднання двох останніх кривих $D_{V_1}^{w-1}$ та $D_{V_1}^w$ послідовності D_j^*. Вилучення кожного біта a_i^j, $i = \overline{V_3,1}$, $j = \overline{1,m}$ з послідовності сегментів D_j^* виконується за таким алгоритмом:

Крок 1. Визначається значення $t_i = t_{i+1} - \Delta t^j$, $i = \overline{V_3,1}$, де t_{V_3+1} обраховується за формулою $t_{V_3+1} = \Delta t^j \cdot (V_3 + 1)$ або передається по захищеному каналі.

Крок 2. За отриманим значенням параметра t_i та опорними точками двох останніх кривих Без'є $D_{V_1}^{w-1}$ (P_0, P_0^1, P_0^2, P_0^3) та $D_{V_1}^w$ (P_0^3, P_1^2, P_2^1, P_3) з послідовності сегментів D_j^*, де $V_1 = 3$, обчислюються координати спільної точки P_0^2 за двома способами [21]:

Спосіб 1. За координатами P_0^3 та P_1^2 обраховується $P_{0\,(2)}^2 = \left(P_0^3 - t_i P_1^2\right) / \left(1 - t_i\right)$.

Спосіб 2. Спочатку обраховується додаткова координата $P_1^1 = \left(P_1^2 - t_i P_2^1\right) / \left(1 - t_i\right)$, за допомогою якої визначається $P_{0\,(3)}^2 = \left(1 - t_i\right) P_0^1 + t_i P_1^1$.

Крок 3. В залежності від виконання нерівностей $\left|P_{0\,(2)}^2 - P_{0\,(3)}^2\right| \le V_5$, $\left|P_0^2 - P_{0\,(2)}^2\right| \le V_5$, $\left|P_0^2 - P_{0\,(3)}^2\right| \le V_5$ відбувається наступне:

3.1. При виконанні однієї з заданих вище нерівностей виконується об'єднання двох останніх кривих послідовності сегментів в одну криву Без'є, при цьому зменшуючи кількість сегментів на одиницю ($w = w - 1$). У такому випадку $a_i^j = 1$ ($a_i^j = 0$ – в залежності від вибору значення біта, при якому

відбувається поділ кривих). Об’єднання двох кривих при заданому значенні t_i виконується наступним чином:

3.1.1. З координат точок P_2^1, P_3 розраховується точка $P_2 = \left(P_2^1 - t_i P_3\right) / \left(1 - t_i\right)$.

3.1.2. З координат точок P_1^2, P_2^1 обраховується додаткова точка $P_1^1 = \left(P_1^2 - t_i P_2^1\right) / \left(1 - t_i\right)$. Далі, за точками P_2, P_1^1 знаходиться точка $P_1 = \left(P_1^1 - t_i P_2\right) / \left(1 - t_i\right)$.

3.1.3. За координатами точок P_0, P_1, P_2, P_3 задається (утворюється) об’єднана крива Без’є.

3.2. Якщо не виконується жодна з наведених вище нерівностей, то у такому випадку $a_i^j = 0$ ($a_i^j = 1$ – в залежності від вибору значення біта, при якому відбувається поділ кривих), а кількість кривих Без’є з послідовності сегментів не зменшується.

Приклад приховування та вилучення інформації.

Нехай є SVG-зображення, в структурі якого є одна крива Без’є третього ступеня. На рис. 3.5 наведена дана крива і її представлення у вигляді команд побудови з опорними точками заданих в об’єкті *path* параметра *d* SVG-зображення [10, 76, 102].

Нехай потрібно приховати у дану криву секретне повідомлення $a^j = \left\{a_1^j, ..., a_8^j\right\} = 01010101$, де $j = 1$ – кількість кривих Без’є в зображенні. Приховування інформації відбуватиметься при наступних параметрах: $V_1 = 3$, $V_2 = 2$, $V_3 = 8$, $V_4 = 6$, $t_0 = 0$ та $\Delta t^j = 0{,}01$. Тоді дана крива буде поділена лише 4 рази – залежить від кількості одиничних біт (нехай при одиничному біті секретного повідомлення відбуватиметься поділ кривої на сегменти) в прихованому повідомленні.

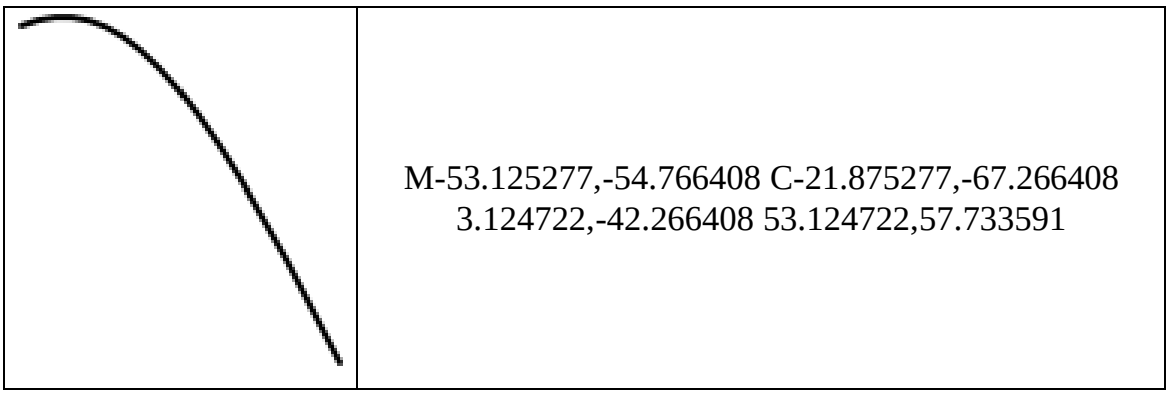


Рис. 3.5.Початкове SVG-зображення

Спочатку індекс послідовності сегментів кривої $D_j^* = D_{V_1}^0 \cup D_{V_1}^1 \cup ... \cup D_{V_1}^w$ приймає значення $w = 0$, а перший сегмент $D_{V_1}^0$ відповідатиме початковій кривій D_j ($D_{V_1}^0 = D_j$). Для приховування повідомлення $a^j = \left\{a_1^j, ..., a_8^j\right\} = 01010101$ виконуються наступні дії:

1. Для приховання a_1^j визначається t_1: $t_1 = t_0 + \Delta t^j = 0,01$. Перевіряється чи a_1^j дорівнює одиниці. В даному випадку $a_1^j = 0$, то ніякого поділу на сегменти кривої $D_{V_1}^0$ в точці t_1 не відбувається.

2. Для приховання a_2^j визначається t_2: $t_2 = t_1 + \Delta t^j = 0,02$. Перевіряється чи a_2^j дорівнює одиниці. В даному випадку $a_2^j = 1$, то крива $D_{V_1}^0$ ділиться в точці t_2 на 2 сегменти $D_{V_1}^0$ і $D_{V_1}^1$, $w = w + 1 = 1$. Наступне приховування відбуватиметься вже в одержаний $D_{V_1}^1$ сегмент.

3. Для приховання a_3^j визначається t_3: $t_3 = t_2 + \Delta t^j = 0,03$. Перевіряється чи a_3^j дорівнює одиниці. В даному випадку $a_3^j = 0$, то ніякого поділу на сегменти кривої $D_{V_1}^1$ в точці t_3 не відбувається.

4. Для приховання a_4^j визначається t_4: $t_4 = t_3 + \Delta t^j = 0,04$. Перевіряється чи a_4^j дорівнює одиниці. В даному випадку $a_4^j = 1$, то крива $D_{V_1}^1$ ділиться в точці t_4 на 2 сегменти $D_{V_1}^1$ і $D_{V_1}^2$, $w = w + 1 = 2$. Наступне приховування відбуватиметься вже в $D_{V_1}^2$ криву.

5. Для приховання a_5^j визначається t_5: $t_5 = t_4 + \Delta t^j = 0,05$. Перевіряється чи a_5^j дорівнює одиниці. В даному випадку $a_5^j = 0$, то ніякого поділу на сегменти кривої $D_{V_1}^2$ в точці t_5 не відбувається.

6. Для приховання a_6^j визначається t_6: $t_6 = t_5 + \Delta t^j = 0,06$. Перевіряється чи a_6^j дорівнює одиниці. В даному випадку $a_6^j = 1$, то крива $D_{V_1}^2$ ділиться в точці t_6 на 2 сегменти $D_{V_1}^2$ і $D_{V_1}^3$, $w = w + 1 = 3$. Наступне приховування відбуватиметься вже в $D_{V_1}^3$ криву.

7. Для приховання a_7^j визначається t_7: $t_7 = t_6 + \Delta t^j = 0,07$. Перевіряється чи a_7^j дорівнює одиниці. В даному випадку $a_7^j = 0$, то ніякого поділу на сегменти кривої $D_{V_1}^3$ в точці t_7 не відбувається.

8. Для приховання a_8^j визначається t_8: $t_8 = t_7 + \Delta t^j = 0,08$. Перевіряється чи a_8^j дорівнює одиниці. В даному випадку $a_8^j = 1$, то крива $D_{V_1}^3$ ділиться в точці t_8 на 2 сегменти $D_{V_1}^3$ і $D_{V_1}^4$, $w = w + 1 = 4$.

9. Початкова крива D_j зображення заміняєм на сукупність утворених сегментів $D_j^* = D_{V_1}^0 \cup D_{V_1}^1 \cup D_{V_1}^2 \cup D_{V_1}^3 \cup D_{V_1}^4$.

На рис. 3.6 наведено змінене векторне зображення та його представлення у вигляді команд побудови і опорних точок кривих в об’єкті *path* параметра *d* SVG-зображення. Візуально воно буде ідентичним початковому, однак описується збільшеною кількістю опорних точок.

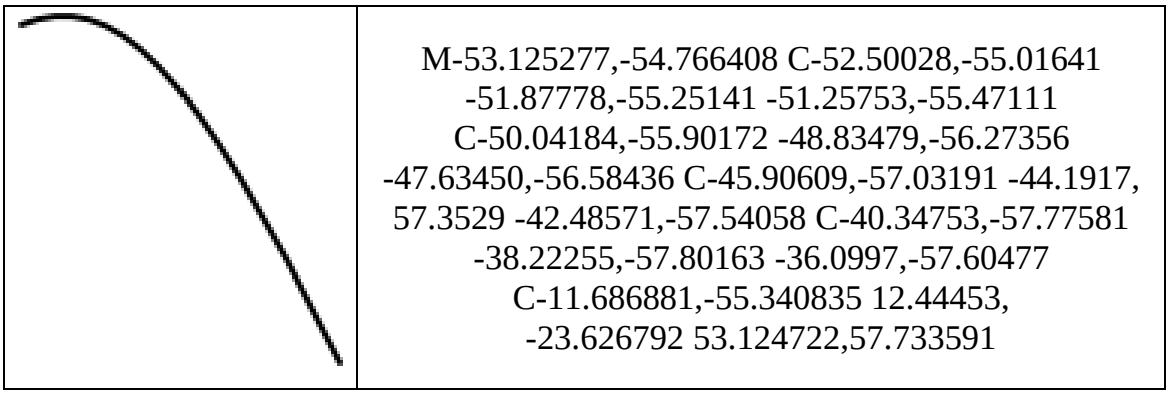


Рис. 3.6. Змінене SVG-зображення в результаті вбудовування інформації

Вилучення прихованої інформації $a^j = \{a_i\}$, $i = \overline{V_3,1}$, $j = 1$, $V_3 = 8$ з послідовності сегментів $D_j^* = D_{V_1}^0 \cup D_{V_1}^1 \cup D_{V_1}^2 \cup D_{V_1}^3 \cup D_{V_1}^4$ з індексом $w = 4$ виконується з наперед визначеним кроком $\Delta t^j = 0,01$, значенням параметра $t_{V_3+1} = 0,01 \cdot (V_3 + 1) = 0,09$ та допустимою похибкою $V_5 = 0,00004$ при виконанні наступних дій:

1. Для вилучення a_8^j визначається t_8: $t_8 = t_{V_3+1} - \Delta t^j = 0,08$. Далі, беруться координати опорних точок двох останніх сегментів $D_{V_1}^4$ (P_0^3, P_1^2, P_2^1, P_3), $D_{V_1}^3$ (P_0, P_0^1, P_0^2, P_0^3) і обчислюються при заданому t_8 точки $P_{0\,(2)}^2$ та $P_{0\,(3)}^2$. За обчисленими точками перевіряються нерівності $\left|P_{0\,(2)}^2 - P_{0\,(3)}^2\right| \le V_5$, $\left|P_0^2 - P_{0\,(2)}^2\right| \le V_5$, $\left|P_0^2 - P_{0\,(3)}^2\right| \le V_5$. В даному випадку хоч одна з нерівностей виконується і відбувається об'єднання двох сегментів $D_{V_1}^4$, $D_{V_1}^3$ в одну криву $D_{V_1}^3$, $w = w - 1 = 3$ та вилучення $a_8^j = 1$.

2. Для вилучення a_7^j визначається t_7: $t_7 = t_8 - \Delta t^j = 0,07$. Далі, беруться координати опорних точок двох останніх сегментів $D_{V_1}^3$ (P_0^3, P_1^2, P_2^1, P_3), $D_{V_1}^2$ (P_0, P_0^1, P_0^2, P_0^3) і обчислюються при заданому t_7 точки $P_{0\,(2)}^2$ та $P_{0\,(3)}^2$. За обчисленими точками перевіряються нерівності $\left|P_{0\,(2)}^2 - P_{0\,(3)}^2\right| \le V_5$, $\left|P_0^2 - P_{0\,(2)}^2\right| \le V_5$, $\left|P_0^2 - P_{0\,(3)}^2\right| \le V_5$. В даному випадку жодна з нерівностей не виконується, тому вилучається $a_7^j = 0$.

3. Для вилучення a_6^j визначається t_6: $t_6 = t_7 - \Delta t^j = 0,06$. Далі, беруться координати опорних точок двох останніх сегментів $D_{V_1}^3$ (P_0^3, P_1^2, P_2^1, P_3), $D_{V_1}^2$ (P_0, P_0^1, P_0^2, P_0^3) і обчислюються при заданому t_6 точки $P_{0\,(2)}^2$ та $P_{0\,(3)}^2$. За обчисленими точками перевіряються нерівності $\left|P_{0\,(2)}^2 - P_{0\,(3)}^2\right| \le V_5$, $\left|P_0^2 - P_{0\,(2)}^2\right| \le V_5$, $\left|P_0^2 - P_{0\,(3)}^2\right| \le V_5$. В даному випадку хоч одна з нерівностей

виконується і відбувається об'єднання двох сегментів $D^3_{V_1}$, $D^2_{V_1}$ в одну криву $D^2_{V_1}$, $w = w - 1 = 2$ та вилучення $a^j_6 = 1$.

4. Для вилучення a^j_5 визначається t_5: $t_5 = t_6 - \Delta t^j = 0{,}05$. Далі, беруться координати опорних точок двох останніх сегментів $D^2_{V_1}$ (P^3_0, P^2_1, P^1_2, P_3), $D^1_{V_1}$ (P_0, P^1_0, P^2_0, P^3_0) і обчислюються при заданому t_5 точки $P^2_{0\,(2)}$ та $P^2_{0\,(3)}$. За обчисленими точками перевіряються нерівності $\left|P^2_{0\,(2)} - P^2_{0\,(3)}\right| \le V_5$, $\left|P^2_0 - P^2_{0\,(2)}\right| \le V_5$, $\left|P^2_0 - P^2_{0\,(3)}\right| \le V_5$. В даному випадку жодна з нерівностей не виконується, тому вилучається $a^j_5 = 0$.

5. Для вилучення a^j_4 визначається t_4: $t_4 = t_5 - \Delta t^j = 0{,}04$. Далі, беруться координати опорних точок двох останніх сегментів $D^2_{V_1}$ (P^3_0, P^2_1, P^1_2, P_3), $D^1_{V_1}$ (P_0, P^1_0, P^2_0, P^3_0) і обчислюються при заданому t_4 точки $P^2_{0\,(2)}$ та $P^2_{0\,(3)}$. За обчисленими точками перевіряються нерівності $\left|P^2_{0\,(2)} - P^2_{0\,(3)}\right| \le V_5$, $\left|P^2_0 - P^2_{0\,(2)}\right| \le V_5$, $\left|P^2_0 - P^2_{0\,(3)}\right| \le V_5$. В даному випадку хоч одна з нерівностей виконується і відбувається об'єднання двох сегментів $D^2_{V_1}$, $D^1_{V_1}$ в одну криву $D^1_{V_1}$, $w = w - 1 = 1$ та вилучення $a^j_4 = 1$.

6. Для вилучення a^j_3 визначається t_3: $t_3 = t_4 - \Delta t^j = 0{,}03$. Далі, беруться координати опорних точок двох останніх сегментів $D^1_{V_1}$ (P^3_0, P^2_1, P^1_2, P_3), $D^0_{V_1}$ (P_0, P^1_0, P^2_0, P^3_0) і обчислюються при заданому t_3 точки $P^2_{0\,(2)}$ та $P^2_{0\,(3)}$. За обчисленими точками перевіряються нерівності $\left|P^2_{0\,(2)} - P^2_{0\,(3)}\right| \le V_5$, $\left|P^2_0 - P^2_{0\,(2)}\right| \le V_5$, $\left|P^2_0 - P^2_{0\,(3)}\right| \le V_5$. В даному випадку жодна з нерівностей не виконується, тому вилучається $a^j_3 = 0$.

7. Для вилучення a^j_2 визначається t_2: $t_2 = t_3 - \Delta t^j = 0{,}02$. За обчисленими точками беруться координати опорних точок двох останніх

сегментів $D^1_{V_1}$ (P^3_0, P^2_1, P^1_2, P_3), $D^0_{V_1}$ (P_0, P^1_0, P^2_0, P^3_0) і обчислюються при заданому t_2 точки $P^2_{0\,(2)}$ та $P^2_{0\,(3)}$. За обчисленими точками перевіряються нерівності $\left|P^2_{0\,(2)} - P^2_{0\,(3)}\right| \le V_5$, $\left|P^2_0 - P^2_{0\,(2)}\right| \le V_5$, $\left|P^2_0 - P^2_{0\,(3)}\right| \le V_5$. В даному випадку хоч одна з нерівностей виконується і відбувається об'єднання двох сегментів $D^1_{V_1}$, $D^0_{V_1}$ в одну криву $D^0_{V_1}$, $w = w - 1 = 0$ та вилучення $a^j_2 = 1$.

8. Для вилучення a^j_1 визначається t_1: $t_1 = t_2 - \Delta t^j = 0{,}01$. Оскільки, була відтворена початкова крива $D_j = D^0_{V_1}$, то $a^j_1 = 0$.

В даному підрозділі за методом побітового приховування інформації розроблено алгоритм вбудовування даних StegoBIT у криві Без'є векторних зображень. Відповідно якому приховування інформації здійснюється у криві третього ступеня шляхом розбиття їх на сукупності сегментів за алгоритмом де Кастельжо. Також, розроблено алгоритм вилучення прихованої інформації з сукупностей сегментів кривих Без'є, що містять приховане повідомлення.

3.3. Процес вбудовування даних у криві Без'є за шаблонним методом приховування інформації

На основі запропонованого методу **шаблонного приховування інформації** (див. підрозділ 2.3) реалізовано алгоритм вбудовування даних StegoTEMPL у криві Без'є третього ступеня [20, 23, 25, 26, 32, 87].

Даний алгоритм дозволяє приховувати секретне повідомлення у криві Без'є третього ступеня векторних зображень. У якості повідомлення можуть бути вбудовані будь-які двійкові дані, які попередньо можуть бути стиснуті [60], зашифровані [4, 11, 27, 72, 73], до них можуть бути застосовані методи завадостійкого кодування і перевірки цілісності [33, 40, 72, 73]. Приховування повідомлення відбувається при поступовому поділі початкових кривих на сукупності сегменти, при цьому саме повідомлення впливає на процес сегментації кривої. При поділі кривих на сегменти зображення

візуально не змінюється. За допомогою властивості афінно-інваріантності кривих Без'є забезпечується стійкість до атак на основі афінних перетворень.

Особливістю розробленого алгоритму є можливість визначати наперед різні кроки зміни параметра побудови кривої відповідно до кожного значення елемента шаблону, що дозволить приховувати цілий блок бітів при одному розбитті кривих Без'є третього ступеня на два сегменти. Для приховування інформації за алгоритмом StegoTEMPL рекомендується використовувати дані кратні 8 бітам для можливості формування таблиць шаблонів з різними кількостями біт значень їх елементів (наприклад, 2 біта, 4 біта, 8 біт, і т. д.).

Слід зауважити, що в даній роботі не досліджується стійкість до пасивних атак розробленого алгоритму. Проте, якщо вбудовувати у одне векторне зображення повідомлення невеликого розміру (після приховування інформації одержаний стеганоконтейнер не повинен перевищувати середньостатистичного розміру векторного зображення), то виявити факт присутності секретного повідомлення буде досить складно.

Вхідними даними для реалізації алгоритму за методом шаблонного приховування інформації є:

1. Сукупність векторних зображень, у структурі яких є криві Без'є.
2. Параметри приховування інформації у векторні зображення з множини $\mathbf{V}$ ($\mathbf{V} = \{V_j\}$, $j \in \overline{1,5}$) (див. підрозділ 2.2), що визначатимуть: ступінь кривих Без'є $V_1 = 3$, що будуть використовуватись для приховування інформації; допустиму відстань між опорними точками V_2 кожної кривої; максимальна кількість вбудовуваної інформації V_3 в одну криву; точність координат опорних точок V_4; допустима похибка V_5 враховувана при відтворенні початкової кривої.

Приховування інформації у криві Без'є третього ступеня відбуватиметься за алгоритмом StegoTEMPL наступим чином [20, 23, 25, 26, 32, 87]:

Крок 1. Секретне повідомлення $a = \{a_i\}$, $a_i \in \{0,1\}$, $i = \overline{1,h}$, a_i – біт секретного повідомлення, h – кількість біт повідомлення a, ділиться на

частини $a=\left\{a_1^1,...,a_{V_3}^1,a_1^2,...,a_{V_3}^2,...,a_1^j,...,a_{V_3}^j\right\}$, $a^j=\left\{a_1^j,...,a_{V_3}^j\right\}$ – послідовність біт, $a_i^j\in\{0,1\}$, $i=\overline{1,V_3}$, $j=\overline{1,m}$, $m=\lceil h/V_3\rceil$, де m – кількість допустимих кривих Без'є у векторному зображенні, в які приховуються частини повідомлення. Після чого, кожна a^j послідовність ділиться на блоки $a^j=\left\{a_1^l,a_2^l,...,a_i^l...,a_z^l\right\}$, $z=V_3/l$, де a_i^l – i-та частина блоку a^j довжиною l біт, $i=\overline{1,z}$. Кожному a_i^l відповідає свій елемент з таблиці шаблонів TV_l^k, $k=\overline{1,2^l}$.

Крок 2. Визначається максимально допустиме значення $\max\Delta t=l/V_3$ та встановлюється кожному елементу TV_l^k ($k=\overline{1,2^l}$) шаблону власний крок $T\Delta t^k$ зміни параметра побудови кривої t, де кроки $T\Delta t^k$ не повторюються та $T\Delta t^k\le\max\Delta t$.

Крок 3. Виконується приховування кожного a^j секретного повідомлення в криву Без'є D_j, $j=\overline{1,m}$, шляхом розбиття її на послідовність сегментів $D_j^*=D_{V_1}^0\cup D_{V_1}^1\cup...\cup D_{V_1}^w$, де w – індекс послідовності створених сегментів кривої D_j^*, $w\in N$ (перед приховуванням $w=0$ і $D_{V_1}^0=D_j$), $V_1=3$ – ступінь кривої Без'є, що впливає на кількість операцій, необхідних для розбиття кривої, при різних значеннях параметра t:

3.1. Приховування кожного елемента a_i^l ($a^j=\left\{a_1^l,a_2^l,...,a_i^l...,a_z^l\right\}$, $i=\overline{1,z}$) послідовності a^j при певному значенні t_i ($t_i=t_{i-1}+T\Delta t^k$, де t_0 – довільне початкове значення, $0\le t_0<1-z\cdot\max\Delta t$, $T\Delta t^k$ відповідає кроку зміни параметра t для приховування a_i^l) відбувається шляхом розбиття кривої Без'є $D_{V_1}^w$ з заданим індексом w на два сегменти $D_{V_1}^w$ і $D_{V_1}^{w+1}$ в точці t_i за алгоритмом де Кастельжо [5, 21, 39, 80]. Подальше внесення наступних елементів a_{i+1}^l шаблону з послідовності a^j буде відбуватися при наступному значенні точки розбиття t_{i+1} в отриманий другий сегмент $D_{V_1}^{w+1}$. Кожне розбиття кривої

збільшує кількість кривих в послідовності сегментів на один ($w = w+1$), що виконується при заданому значенні t_i наступним чином:

3.2.1. З початкової кривої $D_{V_1}^w$ заданої опорними точками P_0, P_1, P_2, P_3 обчислюються точки $P_0^1 = (1-t_i)P_0 + t_i P_1$, $P_1^1 = (1-t_i)P_1 + t_i P_2$ та $P_2^1 = (1-t_i)P_2 + t_i P_3$.

3.2.2. З отриманих координат точок P_0^1, P_1^1, P_2^1 розраховуються точки $P_0^2 = (1-t_i)P_0^1 + t_i P_1^1$ та $P_1^2 = (1-t_i)P_1^1 + t_i P_2^1$.

3.2.3. З отриманих координат точок P_0^2, P_1^2 отримуються координати останньої (шуканої) точки $P_0^3 = (1-t_i)P_0^2 + t_i P_1^2$.

3.2.4. За даними точками початкова крива Без'є ділиться на два сегмента, з яких перший сегмент $D_{V_1}^w$ будується по точкам P_0, P_0^1, P_0^2, P_0^3, а другий $D_{V_1}^{w+1}$ – по точкам P_0^3, P_1^2, P_2^1, P_3.

3.2. Після приховування послідовності a^j у D_j криву, отримана послідовність сегментів $D_j^* = D_{V_1}^0 \cup D_{V_1}^1 \cup ... \cup D_{V_1}^w$ записується до стеганоконтейнера замість D_j кривої.

Псевдокоди процедури вбудовування інформації за даним алгоритмом представлено на рис. 3.7.

Під операцією $SplitMessage(x, y)$ мається на увазі побітове розбиття вхідного повідомлення x на частини розмірністю y біт. Операція $SelectImage(x, y, z)$ виконує відбір допустимих контейнерів з множини x необхідних для вбудовування y частин прихованого повідомлення, в структурі яких містяться криві Без'є ступеня z. Операція $SelectCurves(x, y, z, h)$ визначає з множини зображень x кількість кривих Без'є ступеня z необхідних для вбудовування y частин прихованого повідомлення, що відповідають мінімальній допустимій відстані h між опорними точками. Операція

$GetPoints(x)$ використовується для одержання координат опорних точок кривої x.

Input: Секретне повідомлення $a=\{a_i\}$, $a_i \in \{0,1\}$, $i=\overline{1,h}$, $h \in N$;

множина векторних зображень (контейнерів) $I=\{I_b\}$, $b \in N$;

множина параметрів $V=\{V_i\}$, $i=\overline{1,4}$, $V_i \in N$, $V_1=3$;

таблиця шаблону $TV_l[i]$, що виступатиме стеганоключем, $i=\overline{0,2^l-1}$, $TV_l[i] \in (0,\ l/V_3)$, $l \in N$, $c=V_3/l$.

Output: множина стеганоконтейнерів $S=\{S_x\}$, $x \in N$.

1. $\{a^j\}=SplitMessage(a,V_3)$, $a^j=\{a_1^j,...,a_{V_3}^j\}$, $a_i^j \in \{0,1\}$, $i=\overline{1,V_3}$, $j=\overline{1,m}$, $m=\lceil h/V_3 \rceil$;
2. $S=SelectImage(I,m,V_1)$, $S=\{S_x\}$, $x \in N$, $x \le m$, $m=\lceil h/V_3 \rceil$;
3. $D=SelectCurves(S,m,V_1,V_2)$, $D=\{D_j\}$, $j=\overline{1,m}$, $m=\lceil h/V_3 \rceil$;
2. $For\ (j=1;\ j \le m;\ j++)$
2.1. $\{A_i^j\}=SplitMessage(a^j,l)$, $A_i^j \in \{0,1\}^l$, $i=\overline{1,c}$;
2.2. $w=0$; $t=0$; $D_{V_1}^w=D_j$;
2.3. $For\ (i=1; i \le c; i++)$
2.3.1. $t=t+TV_l[A_i^j]$;
2.3.2. $\{P_e\}=GetPoints(D_{V_1}^w)$, $e=\overline{0,3}$;
2.3.3. $P_0^1=(1-t)\cdot P_0+t\cdot P_1$;
2.3.4. $P_1^1=(1-t)\cdot P_1+t\cdot P_2$;
2.3.5. $P_2^1=(1-t)\cdot P_2+t\cdot P_3$;
2.3.6. $P_0^2=(1-t)\cdot P_0^1+t\cdot P_1^1$;
2.3.7. $P_1^2=(1-t)\cdot P_1^1+t\cdot P_2^1$;
2.3.8.. $P_0^3=(1-t)\cdot P_0^2+t\cdot P_1^2$;
2.3.9. $D_{V_1}^w=CreateCurve(P_0, P_0^1, P_0^2, P_0^3)$;
2.3.10. $D_{V_1}^{w+1}=CreateCurve(P_0^3, P_1^2, P_2^1, P_3)$;
2.3.11. $w=w+1$;
2.4. $D_j=ReplaceCurve(S, D_j, \{D_{V_1}^i\})$, $i=\overline{0,w}$;
3. $Return\ S$;

Рис. 3.7. Псевдокод процедури приховування інформації алгоритму StegoTEMPL

Операція $CreateCurve(x,y,z,h)$ використовується для побудови кривої Без'є за координатами опорних точок x, y, z та h. Операція

$ReplaceCurve(x, y, z)$ виконує заміну в зображеннях x початкових кривих y на сукупності сегментів z, що містять приховану інформацію.

Вилучення інформації a^j ($a^j = \{a_1^l, a_2^l, ..., a_i^l ..., a_z^l\}$) виконується при відтворені початкових кривих Без'є D_j з сукупностей сегментів D_j^*, $j = \overline{1,m}$, що здійснюється в процесі поступового об'єднання двох останніх кривих $D_{V_1}^{w-1}$, $D_{V_1}^w$ послідовності D_j^*. Вилучення кожного блоку a_i^l, $i = \overline{V_3 / l, 1}$, з послідовності сегментів D_j^*, $j = \overline{1,m}$, виконується за наступним алгоритмом:

Крок 1. За отриманим значенням параметра t_i (який передається закритим каналом) та опорними точками двох останніх сегментів $D_{V_1}^{w-1}$ (P_0, P_0^1, P_0^2, P_0^3) та $D_{V_1}^w$ (P_0^3, P_1^2, P_2^1, P_3), $V_1 = 3$, виконується їх об'єднання в одну криву наступним чином (зменшуючи кількість кривих в сукупності сегментів на один ($w = w - 1$)):

1.1. З координат точок P_2^1, P_3 розраховується точка $P_2 = \left(P_2^1 - t_i P_3\right) / \left(1 - t_i\right)$.

1.2. З координат точок P_1^2, P_2^1 обраховується додаткова точка $P_1^1 = \left(P_1^2 - t_i P_2^1\right) / \left(1 - t_i\right)$. Далі, за точками P_2, P_1^1 знаходиться точка $P_1 = \left(P_1^1 - t_i P_2\right) / \left(1 - t_i\right)$.

1.3. За координатами точок P_0, P_1, P_2, P_3 задається (утворюється) об'єднана крива Без'є.

Крок 2. Перебираються усі кроки $T\Delta t^k$, $k = \overline{1, 2^l}$, зміни параметра побудови кривої t, для яких виконується наступна послідовність дій:

2.1. Визначається наступне значення параметра t_i: $t_i = t_{i+1} - T\Delta t^k$, $i = \overline{V_3 / l - 1, 1}$.

2.2. За отриманим значенням параметра t_i та опорними точками двох останніх кривих $D_{V_1}^{w-1}$ (P_0, P_0^1, P_0^2, P_0^3) та $D_{V_1}^{w}$ (P_0^3, P_1^2, P_2^1, P_3), де $V_1 = 3$, обчислюються координати точки P_0^2 за двома способами [21]:

Спосіб 1. За координатами P_0^3 та P_1^2 обраховується $P_{0\,(2)}^2 = \left(P_0^3 - t_i P_1^2\right) / \left(1 - t_i\right)$.

Спосіб 2. Спочатку обраховується додаткова координата $P_1^1 = \left(P_1^2 - t_i P_2^1\right) / \left(1 - t_i\right)$, за допомогою якої визначається $P_{0\,(3)}^2 = \left(1 - t_i\right) P_0^1 + t_i P_1^1$.

2.3. В залежності від виконання нерівностей $\left|P_{0\,(2)}^2 - P_{0\,(3)}^2\right| \le V_5$, $\left|P_0^2 - P_{0\,(2)}^2\right| \le V_5$, $\left|P_0^2 - P_{0\,(3)}^2\right| \le V_5$, відбувається наступне:

2.3.1. При виконанні однієї з заданих вище нерівностей відбувається вилучення прихованого блоку $a_i^l = TV_l^k$, який відповідає кроку зміни $T\Delta t^k$, що використовувався при обчисленні значенням параметра t_i, та виконується об’єднання двох останніх кривих Без’є послідовності сегментів в одну криву (відповідно пунктам 1.1-1.3) зменшуючи кількості кривих в сукупності сегментів на один ($w = w - 1$).

2.3.2. Якщо не виконується задана нерівність, то виконується перехід до наступного кроку $T\Delta t^k$ ($k = k + 1$) поки не буде підібрано такий крок, при якому відбудеться об’єднання двох кривих Без’є з послідовності сегментів.

Крок. 3. Вилучення останнього блоку a_i^l виконується шляхом перебирання усіх крокі $T\Delta t^k$, $k = \overline{1, 2^l}$, доки не буде виконуватися рівність $t_i - T\Delta t^k = t_0$. При виконання якої $a_i^l = TV_l^k$ при відповідному значенні k.

Приклад приховування та вилучення інформації.

Нехай є SVG-зображення, в якому задана одна крива Без’є третього ступеня. На рис. 3.5 наведена дана крива та її представлення у вигляді команд побудови з опорними точками заданих в об’єкті *path* параметра *d* SVG-зображення [10, 76, 102].

Нехай потрібно приховати у дану криву секретне повідомлення $a^j = \left\{a_1^j, ..., a_8^j\right\} = 01010111$, де $j = 1$ – кількість кривих Без'є в зображенні. Приховування інформації відбуватиметься при використанні наступних параметрів: $V_1 = 3$, $V_2 = 2$, $V_3 = 8$, $V_4 = 6$, $t_0 = 0{,}0005$, $l = 4$, $k = 2^l = 16$.

Нехай в табл. 3.2. наведена таблиця значень шаблону для $l = 4$, що буде використовуватися для приховування повідомлення a^j.

Оскільки $l = 4$, то повідомлення a^j представляється у вигляді двох 4-бітних блоків $a^j = \left\{a_1^l, a_2^l\right\} = \{0101, 0111\}$. Зважаючи на це, крива Без'є D_j буде поділена лише 2 рази.

Таблиця 3.2

Таблиця співвідношень значень елементів шаблону різним крокам зміни параметра побудови кривих Без'є

k	TV_l^k	$T\Delta t^k$
1	0000	0,0075
2	0001	0,0095
3	0010	0,009
4	0011	0,0085
5	0100	0,008
6	0101	0,002
7	0110	0,007
8	0111	0,0065
9	1000	0,006
10	1001	0,0055
11	1010	0,005
12	1011	0,0045
13	1100	0,004
14	1101	0,0035
15	1110	0,003
16	1111	0,0025

Спочатку індекс послідовності сегментів кривої $D_j^* = D_{V_1}^0 \cup D_{V_1}^1 \cup ... \cup D_{V_1}^w$ приймає значення $w = 0$, а сегмент $D_{V_1}^0$ відповідатиме початковій кривій D_j ($D_{V_1}^0 = D_j$). Для приховування повідомлення $a^j = \left\{a_1^l, a_2^l\right\} = \{0101, 0111\}$ виконуються наступні дії:

1. Приховування блоку $a_1^l = 0101$ при $t_1 = t_0 + T\Delta t^k = 0{,}0025$, де крок $T\Delta t^k = 0{,}002$, $k = 6$ береться з табл. 3.2 при відповідності прихованого блоку до

певного елемента шаблону ($a_1^l = TV_l^k$), виконується розбиттям кривої $D_{V_1}^0$ на 2 сегменти $D_{V_1}^0$ і $D_{V_1}^1$, $w = w+1 = 1$.

2. Приховування блоку $a_2^l = 0111$ при кроці $t_2 = t_1 + T\Delta t^k = 0,009$, де крок $T\Delta t^k = 0,0065$, $k = 8$ береться з табл. 3.2 при відповідності прихованого блоку до певного елемента шаблону ($a_1^l = TV_l^k$), виконується розбиттям кривої $D_{V_1}^1$ на 2 сегменти $D_{V_1}^1$ і $D_{V_1}^2$, $w = w+1 = 2$.

3. Початкова крива D_j зображення заміняєм на сукупність утворених сегментів $D_j^* = D_{V_1}^0 \cup D_{V_1}^1 \cup D_{V_1}^2$.

На рис. 3.8 наведено змінене векторне зображення та його представлення у вигляді команд побудови і координат опорних точок кривих Без'є в об'єкті *path* параметра *d* SVG-зображення. Візуально одержане зображення ідентичне початковому, однак описується збільшеною кількістю опорних точок.

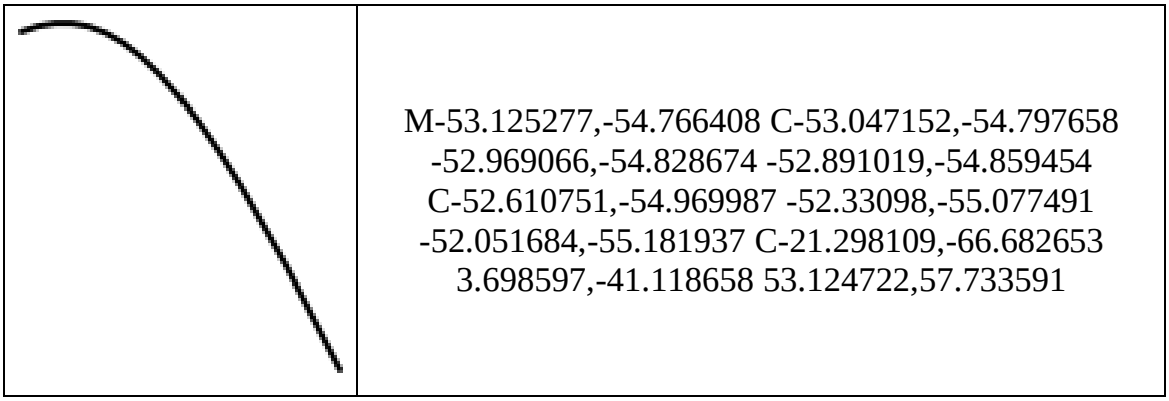


Рис. 3.8. Змінене SVG-зображення в результаті вбудовування інформації

Вилучення прихованої інформації a^j, $j = 1$, з послідовності сегментів $D_j^* = D_{V_1}^0 \cup D_{V_1}^1 \cup D_{V_1}^2$ з індексом $w = 2$ виконується з наперед визначеною таблицею співвідношень табл. 3.2, кількістю вбудованої інформації в одну криву $V_3 = 8$, кінцевим значенням параметра $t_{V_3/l} = t_2 = 0,009$, початковим значенням $t_0 = 0,0005$ та допустимою похибкою $V_5 = 0,00004$ при виконанні наступних дій:

1. При заданому значенні t_2 виконується об'єднання двох сегментів $D^2_{V_1}$, $D^1_{V_1}$ в одну криву $D^1_{V_1}$, $w = w - 1 = 1$.

2. Для вилучення блоку a^l_2 визначається t_1: $t_1 = t_2 - T\Delta t^k$, $k = \overline{1,16}$.

2.1. Для кожного значення t_1 беруться координати опорних точок двох останніх сегментів $D^1_{V_1}$ (P^3_0, P^2_1, P^1_2, P_3), $D^0_{V_1}$ (P_0, P^1_0, P^2_0, P^3_0) і обчислюються точки $P^2_{0\,(2)}$ та $P^2_{0\,(3)}$. За обчисленими точками перевіряються нерівності $\left|P^2_{0\,(2)} - P^2_{0\,(3)}\right| \le V_5$, $\left|P^2_0 - P^2_{0\,(2)}\right| \le V_5$, $\left|P^2_0 - P^2_{0\,(3)}\right| \le V_5$.

2.2. При значенні $k = 8$, $t_1 = t_2 - 0{,}0065 = 0{,}0025$ хоч одна з нерівностей виконається і відбувається об'єднання двох сегментів $D^1_{V_1}$, $D^0_{V_1}$ в одну криву $D^0_{V_1}$, $w = w - 1 = 1$ та вилучиться блок $a^l_2 = 0111$.

3. Для вилучення блоку a^l_1 виконується перевірка рівності $t_1 - T\Delta t^k = t_0$, $k = \overline{1,16}$.

3.1. При значенні $k = 6$ рівність виконується $t_1 - 0{,}002 = 0{,}0005$, а блок $a^l_1 = 0101$.

В даному підрозділі за методом шаблонного приховування інформації розроблено алгоритм вбудовування даних StegoTEMPL у криві Без'є векторних зображень. Відповідно алгоритму приховування інформації здійснюється у криві третього ступеня шляхом розбиття їх на сукупності. Для розбиття кривих використовується наперед визначена таблиця співвідношень значень елементів шаблону з різними кроками побудови кривих Без'є. Також, розроблено алгоритм вилучення прихованої інформації з сукупностей сегментів з прихованою інформацією, що складаються з кривих Без'є третього ступеня.

3.4. Висновки до третього розділу

Таким чином, у третьому розділі дисертаційної роботи були отримані наступні результати:

1. Проведено дослідження різних типів кубічних кривих векторної графіки, що дозволяють будувати складені криві шляхом їх поділу на сукупності сегментів і таки чином вбудовувати секретну інформацію в них. Розглянуті криві володіють властивістю афінно-інваріантності за якою забезпечуватиметься стійкість до афінних перетворень. Особлива увага була приділена кривим Без'є, а саме їх принципам побудови та розбиття на сегменти за алгоритмом де Кастельжо.

2. Розроблено алгоритм побітового приховування інформації у криві Без'є третього ступеня векторних зображень – StegoBIT, який, за рахунок впливу послідовності даних на процес сегментації кривих з фіксованим кроком зміни параметра побудови кривих (розбиття кривих на сегменти відбувається лише при вбудовуванні нульового/одиничного біта приховуваної послідовності даних), дозволяє вбудовувати дані у криві Без'є з забезпеченням стійкості до афінних перетворень.

3. Розроблено алгоритм шаблонного приховування інформації у криві Без'є третього ступеня векторних зображень – StegoTEMPL, який, за рахунок впливу послідовності даних на процес сегментації кривих згідно визначеної таблиці співвідношень значень елементів шаблону різним крокам зміни параметра побудови кривих (при розбитті кривої на два сегменти вбудовується блок даних), дозволяє вбудовувати дані у криві Без'є з забезпеченням стійкості до афінних перетворень.

РОЗДІЛ 4

ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ РОЗРОБЛЕНИХ РІШЕНЬ

4.1. Методика проведення експериментального дослідження

Методика експерименту – визначена послідовність процесів, у результаті якої досягається мета дослідження. Метою експериментального дослідження є вивчення якостей оцінюваних об’єктів, перевірка правильності формування та достовірності гіпотез, глибоке вивчення досліджуваної наукової тематики. В залежності від галузі науки, структури об’єктів досліджень, мети дослідження, організаційних явищ, існує багато різних класифікації експериментів. Правильний вибір методики експерименту займає особливе значення при його проведенні. Цінність експерименту визначає правильність розробки методики його дослідження.

Перший крок у проведенні експериментального дослідження займає складання плану – програми дослідження. Другий крок, здійснюється після затвердження методики, – це визначення об’єму експериментальних досліджень та необхідних програмних засобів. Третім кроком є безпосереднє проведення експерименту, а заключним кроком – обробка експериментальних даних, систематизація усіх числових даних, перевірка зведення до єдиної системи одиниць, побудова графіків залежностей, таблиць, діаграм.

Програма дослідження

1. *Мета і задачі експерименту*.

Метою експерименту є перевірка ефективності застосування на практиці запропонованих алгоритмів для приховування даних у векторні зображення та розробка практичних рекомендацій щодо їх використання.

Задачі експерименту:

а) дослідження швидкостей вбудовування та вилучення інформації при застосуванні запропонованих алгоритмів (при різних значеннях стеганоключів та параметрів приховування з множини $\mathbf{V} = \{V_i\}$, $i = \overline{1,5}$);

б) дослідження коефіцієнту візуального спотворення зображень після вбудовування інформації алгоритмів (при різних значеннях стеганоключів та параметрів приховування з множини $\mathbf{V} = \{V_i\},\ i = \overline{1,5}$);

в) дослідження зміни розмірів стеганоконтейнерів (при різних значеннях алгоритмів (при різних значеннях стеганоключів та параметрів приховування з множини $\mathbf{V} = \{V_i\},\ i = \overline{1,5}$);

г) дослідження стійкості запропонованих алгоритмів до афінних та майже афінних перетворень (при різних значеннях стеганоключів та параметрів приховування з множини $\mathbf{V} = \{V_i\},\ i = \overline{1,5}$);

д) порівняння отриманих результатів стійкості до даних атак з аналогічними результатами метода Карпінцева-Яремчука;

е) розробка практичних рекомендацій.

2. *Вибір вхідних та вихідних параметрів*:

а) *вхідні параметри*:

– множина із 30-и векторних зображень (контейнерів) в які буде приховуватись секретне повідомлення;

– множини стеганоключів для алгоритмів StegoBIT та StegoTEMPL при використанні яких буде проводитись експериментальне дослідження;

– множини параметрів приховування $\mathbf{V} = \{V_i\},\ i = \overline{1,5}$ при яких буде проводитись дослідження.

б) *вихідні параметри*:

– середні швидкісні характеристики розроблених алгоритмів;

– середній коефіцієнт візуального спотворення зображень після вбудовування інформації;

– середні розміри стеганоконтейнерів при вбудовуванні повідомлень запропонованими алгоритмами;

– експериментальні дані стійкості запропонованих алгоритмів до афінних перетворень;

– практичні рекомендації щодо використання розроблених алгоритмів приховування інформації при різних значеннях параметрів V_i.

3. Послідовність дій:

1. Розробити програмні засоби, що реалізують вбудовування інформації у множину векторних зображень запропонованими алгоритмами.

2. Для кожної комбінації множини параметрів приховування $\mathbf{V} = \{V_i\}$, $i = \overline{1,5}$ провести вбудовування даних у множину із 30 векторних зображень, дослідити розміри утворених стеганоконтейнерів.

3. Для кожної комбінації параметрів V_i дослідити швидкість вбудовування ти вилучення інформації при різних процесорах. Дослідження буде проводитись на: AMD A10-4600M, 2.3GHz, AMD Phenom(tm) II X4 945 Processor, 3.00 GHz, Intel® Pentium® quad core processon N3540, up to 2.66 GHz.

4. Для кожної комбінації параметрів V_i, при використані програми AntiDupl.NET, дослідити коефіцієнт візуального спотворення зображень після вбудовування інформації.

5. Для кожної комбінації множини параметрів V_i на отриманими стеганоконтейнерами, при використанні програми Inkscape, виконати: афінні перетворення типу перенесення, поворот, зсув за віссю абсцис і ординат та пропорційне і непропорційне масштабування.

6. Порівняти отримані результати пункти 2-4 із результатами метода Карпінцева-Яремчука.

7. На основі результатів пункти 2-5 розробити практичні рекомендації.

4. *Вибір кроку зміни параметрів приховування*.

Визначення швидкісних характеристик, розмірів та коефіцієнтів візуального спотворення стеганоконтейнерів, експериментальних даних стійкості до афінних та майже афінних перетворень розроблених алгоритмів StegoBIT та StegoTEMPL будуть виконуватись при різних стеганоключах і значеннях параметрів V_i.

Значення параметра V_1, що визначає ступінь кривої Без'є, фіксується і не змінюється – $V_1 = 3$ (див. підрозділи 3.2, 3.3). Це пояснюється тим, що на практиці (наприклад, SVG-зображеннях) частіше всього використовуються криві 3 ступеня, при використанні кривих вищих порядків збільшується: кількість операцій необхідних для їх обробки та кількість точок необхідних для задання таких кривих Без'є [43]. Параметр V_2, що визначає допустима відстань між опорними точками кривої, буде довільним і прийматиме значення $V_2 \geq 1$. Параметр V_3, що визначає кількість вбудовуваної інформації в одну криву, буде змінюватись від 40 до 80 байт (з кроком зміни 20 байт). Параметр V_4, що визначає точність координат опорних точок, буде змінюватись від 10^{-5} до 10^{-6} байт (з кроком зміни 10^{-m}, $m \in \{5,6\}$). Параметр V_5, що визначає максимально допустиму похибку при відтворенні початкової кривої, буде змінюватись з $2 \cdot 10^{-5}$ до $4 \cdot 10^{-5}$ (з кроком зміни $2 \cdot 10^{-5}$).

5. *Використовувані засоби:*

– Visual Studio 2012 (для створення програмних засобів, що дозволяють провести дослідження);

– AntiDupl.NET-2.3.6 (для дослідження коефіцієнту візуального спотворення зображень);

– Inkscape-0.48.5 (для виконання афінних перетворень).

6. *Аналіз результатів*.

Відповідно до обраної методики проведено експериментальні дослідження, опис та обробка яких містяться в підрозділі 4.3.

4.2. Розробка програмного забезпечення для проведення експериментів

Усі програми описані в цьому підрозділі були розроблені на мові програмування C++ в середовищі Microsoft Visual Studio 2012. Кожний засіб спроектований у вигляді спеціального класу, що може бути підключений для

використання в різноманітних програмних засобах (вихідні коди програм наведені в додатку В).

StegoInSVG-Bitwise

На основі запропонованого алгоритму StegoBIT (див. підрозділ 3.2) було розроблено ПЗ *StegoInSVG-Bitwise* (рис. 4.1). Даний ПЗ приховує інформацію у векторні зображення формату SVG. Вбудовування інформації здійснюється в об’єкти *path* SVG зображень, де містяться команди побудови кривих Без’є.

Для приховування текстових файлів з секретною інформацією потрібно виконати наступну послідовність дій:

1. В полі «Режим роботи програми» вибрати режим «Приховування».
2. В полі «Приховування інформації / Дані вилучення» вибрати приховуваний файл.
3. В полі «SVG зображення» вибирати зображення в яке буде здійснюватися вбудовування даних.
4. В полі «Місце збереження результату» вказати каталог куди буде збережений стеганоконтейнер та текстовий файл з додатковою інформацією про вилучення.
5. В полі «Ступінь кривої Без’є» обрати ступінь кривих, в які буде відбуватися вбудовування даних.
6. В полі «Мінімальна довжина кривої» слід визначити допустиму відстань між опорними точками кривих Без’є, в які буде здійснюватися вбудовування інформації.
7. В полі «Кількість вбудовуваних даних в одну криву Без’є» зазначити яка кількість інформації буде вбудовуватися одну криву.

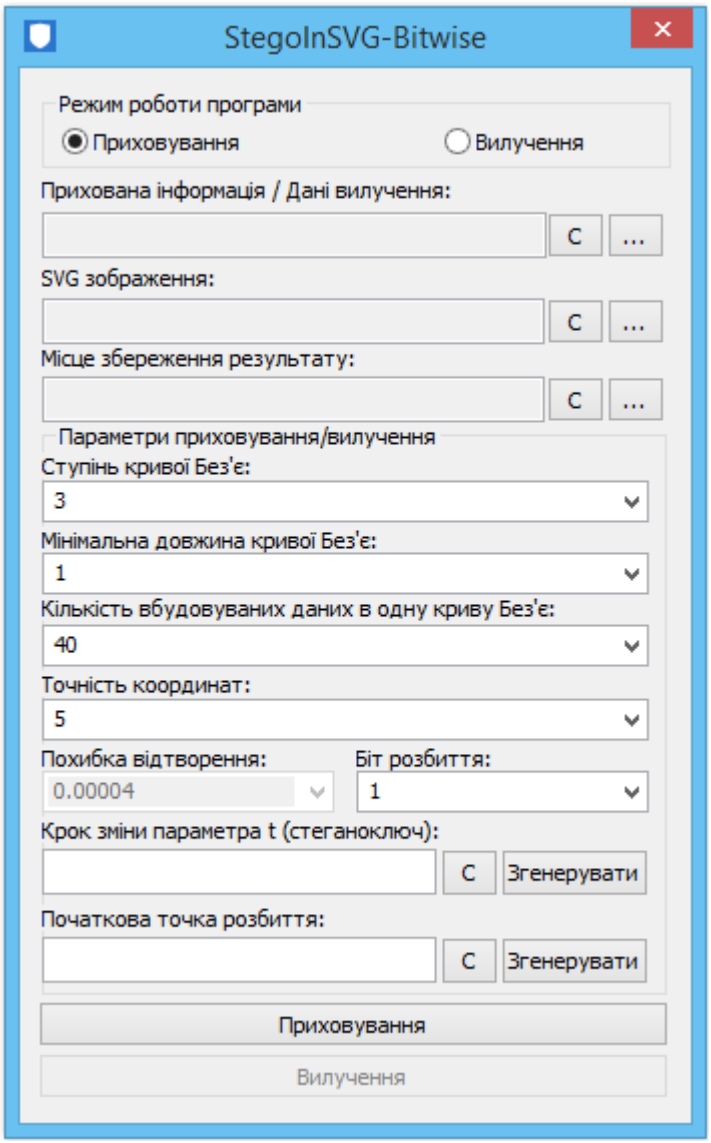


Рис. 4.1. Вікно програми *StegoInSVG-Bitwise*

8. В полі «Точність координат» встановити точність дробової частини координат опорних точок, необхідного для проведення аналітичних розрахунків над кривими Без’є.

9. В полі «Біт розбиття» встановити біт при якому буде відбуватися поділ кривих на сегменти.

10. В полі «Крок зміни параметра t (стеганоключ)» визначити або згенерувати (натиснувши кнопку «Згенерувати») крок зміни параметра побудови (розбиття) t кривих Без’є, за яким буде здійснюється приховування інформації.

11. В полі «Початкова точка розбиття» встановити або згенерувати (натиснувши кнопку «Згенерувати») початкове значення параметра t, за яким

відбуватиметься перший поділ кривих Без'є векторного зображення на сегменти.

12. Натиснувши кнопку «Приховування» відбудеться вбудовування даних в SVG зображення. Після виконання операції одержаний стеганоконтейнер і текстовий файл з додатковою інформацією збережеться в обраний каталог.

Для відтворення прихованої інформації з стеганоконтейнера потрібно виконати наступні дії:

1. В груповому полі «Режим роботи програми» вибрати режим «Вилучення».

2. В полі «Приховування інформації / Дані вилучення» вибрати текстовий файл з додатковою інформацією для вилучення прихованих даних.

3. В полі «SVG зображення» вибирати відповідний стеганоконтейнер з прихованими даними.

4. В полі «Місце збереження результату» вказати каталог куди буде збережений текстовий файл з прихованою інформацією.

5. В полі «Похибка відтворення» вказати допустиму похибку відтворення, що враховується при відтворенні прихованої інформації.

6. В полі «Біт розбиття» встановити біт при якому буде відбуватися об'єднання кривих з сукупності сегментів і вилучатися прихована інформація.

7. Натиснувши кнопку «Вилучення» відбудеться вилучення прихованої інформації з стеганоконтейнера. Після виконання операції текстовий файл з прихованою інформацією збережеться в обраний каталог.

StegoInSVG-Template

На основі запропонованого алгоритму StegoTEMPL (див. підрозділ 3.3) було розроблено ПЗ *StegoInSVG-Template* (рис. 4.2), що приховує дані у зображення формату SVG. Вбудовування даних здійснюється в об'єкти *path* SVG зображень, де містяться команди побудови кривих Без'є.

Для приховування текстових файлів з секретною інформацією потрібно виконати наступну послідовність дій:

1. В полі «Режим роботи програми» вибрати режим «Приховування».

2. В полі «Приховування інформації / Дані вилучення» вибрати приховуваний файл.

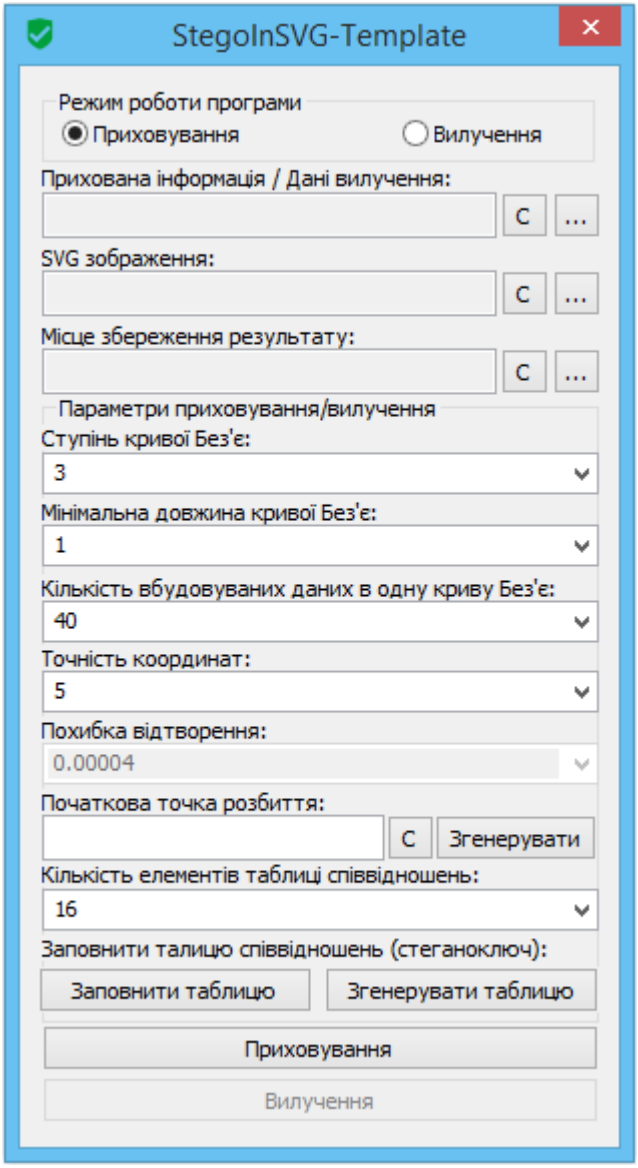


Рис. 4.2. Вікно програми *StegoInSVG-Template*

3. В полі «SVG зображення» вибирати зображення в яке буде здійснюватися вбудовування інформації.

4. В полі «Місце збереження результату» вказати каталог куди буде збережений стеганоконтейнер та текстовий файл з додатковою інформацією про вилучення.

5. В полі «Ступінь кривої Без’є» обрати ступінь кривих, в які буде вбудовуватися прихована інформація.

6. В полі «Мінімальна довжина кривої» слід визначити допустима відстань між опорними точками кривих Без’є, в які буде здійснюватися вбудовування інформації.

7. В полі «Кількість вбудовуваних даних в одну криву Без’є» зазначити яка кількість інформації буде вбудовуватися одну криву.

8. В полі «Точність координат» встановити точність дробової частини координат опорних точок, необхідного для проведення аналітичних розрахунків над кривими Без’є.

9. В полі «Початкова точка розбиття» встановити або згенерувати (натиснувши кнопку «Згенерувати») початкове значення параметра t, за яким відбуватиметься перший поділ кривих Без’є векторного зображення на сегменти.

10. В полі «Кількість елементів таблиці співвідношень» встановити відповідний розмір таблиці співвідношень.

11. Натиснувши кнопку «Заповнити таблицю» або «Згенерувати таблицю» відбувається заповнення таблиці співвідношень різних значень елементів шаблону із різними кроками побудови кривих Без’є (рис. 4.3), за якою буде здійснюється приховування інформації.

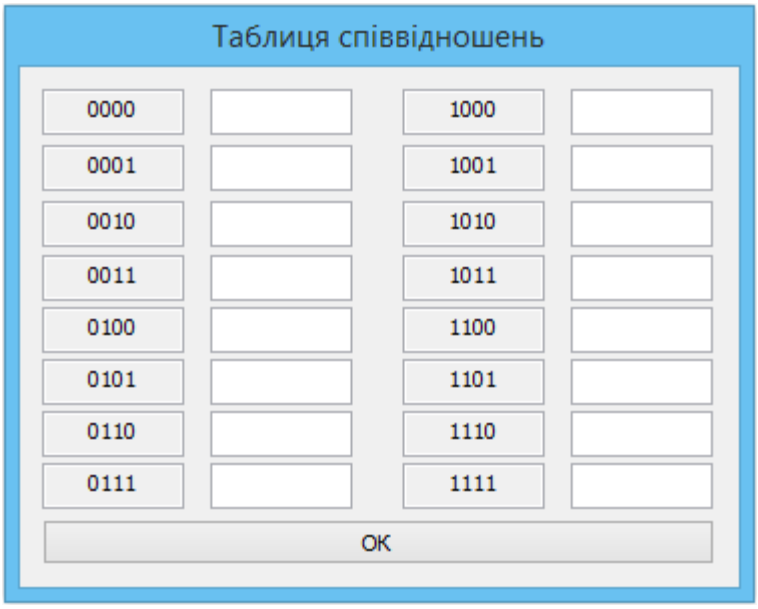


Рис. 4.3. Вікно таблиці співвідношень ПЗ *StegoInSVG-Template*

12. Натиснувши кнопку «Приховування» відбудеться вбудовування даних в SVG зображення. Після виконання операції одержаний стеганоконтейнер і текстовий файл з додатковою інформацією збережеться в обраний каталог.

Для відтворення прихованої інформації з стеганоконтейнера потрібно виконати наступні дії:

1. В полі «Режим роботи програми» вибрати режим роботи «Вилучення».

2. В полі «Приховування інформації / Дані вилучення» вибрати текстовий файл з додатковою інформацією для вилучення.

3. В полі «SVG зображення» вибирати стеганоконтейнер з прихованою інформацією.

4. В полі «Місце збереження результату» вказати каталог куди буде збережений текстовий файл з прихованою інформацією.

8. В полі «Похибка відтворення» вказати допустиму похибку відтворення, що враховується при відтворенні прихованої інформації.

5. Натиснувши кнопку «Вилучення» відбудеться вилучення прихованої інформації з стеганоконтейнера. Після виконання операції текстовий файл з прихованою інформацією збережеться в обраний каталог.

4.3. Дослідження швидкісних характеристик, стійкості до афінних перетворень, коефіцієнту візуального спотворення та розмірів стеганоконтейнерів запропонованих алгоритмів

У даному підрозділі проводиться експериментальне дослідження щодо визначення швидкісних характеристик, стійкості до афінних та майже афінних перетворень, коефіцієнту візуального спотворення та розмірів стеганоконтейнерів при застосування запропонованих алгоритмів з різними значеннями параметрів V_i, $i = \overline{1,5}$ [20-24, 26, 87].

Дослідження проводилось на 30 довільних SVG зображення в структурі яких містилися команди побудови кривих Без'є третього ступеня [10, 21, 76, 102]. В кожне з даних SVG зображень за алгоритмами StegoBIT та StegoTEMPL було вбудовано однакову інформацію розміром 2^{10} біт. При вбудовуванні інформації використовувалися наступні значення стеганоключів:

– за алгоритмом StegoBIT $\Delta t = \left\{2 \cdot 10^{-3}, 1 \cdot 10^{-3}, 5 \cdot 10^{-4}\right\}$;

– за алгоритмом StegoTEMPL таблиці співвідношень представленні в табл. 4.1 при $l = 4$, $k = 16$.

Таблиця 4.1

Таблиці співвідношень значень елементів шаблону з різними кроками побудови кривої Без'є

Індекс k	Шаблон TV_l^k, біт	Крок зміни $T\Delta t^k$
1	0000	0,009
2	0001	0,0085
3	0010	0,008
4	0011	0,0076
5	0100	0,007
6	0101	0,0065
7	0110	0,006
8	0111	0,0055
9	1000	0,005
10	1001	0,0045
11	1010	0,004
12	1011	0,0035
13	1100	0,003
14	1101	0,0025
15	1110	0,002
16	1111	0,0015

Індекс k	Шаблон TV_l^k, біт	Крок зміни $T\Delta t^k$
1	0000	0,007
2	0001	0,0066
3	0010	0,0062
4	0011	0,0058
5	0100	0,0054
6	0101	0,005
7	0110	0,0046
8	0111	0,0042
9	1000	0,0038
10	1001	0,0034
11	1010	0,003
12	1011	0,0026
13	1100	0,0022
14	1101	0,0018
15	1110	0,0014
16	1111	0,001

Індекс k	Шаблон TV_l^k, біт	Крок зміни $T\Delta t^k$
1	0000	0,001
2	0001	0,0017
3	0010	0,0024
4	0011	0,0031
5	0100	0,0038
6	0101	0,0045
7	0110	0,0052
8	0111	0,0059
9	1000	0,0066
10	1001	0,0073
11	1010	0,008
12	1011	0,0087
13	1100	0,0094
14	1101	0,0101
15	1110	0,0108
16	1111	0,0115

Для кожної комбінації параметрів V_i при визначенні ***швидкісних характеристик*** програмно замірявся час вбудовування та вилучення інформації за алгоритмами StegoBIT та StegoTEMPL для кожного із 30 SVG зображень. Кількість виконаних ітерацій 100. Вираховувався середній час вбудовування і вилучення та розраховувались середня швидкість в бітах за секунду.

Для кожної комбінації параметрів V_i при визначенні ***коефіцієнту візуального спотворення стеганоконтейнерів*** використовувалась програма AntiDupl.NET (за допомогою програми Inkscape створювались растрові копії стеганоконтейнерів). При порівняні контейнерів та утворених стеганоконтейнерів дана програма виконує наступне:

1. Приводить всі зображень до одного розміру (32x32).
2. Відкидає колірну інформації (перетворює в сіре зображення).
3. Знаходить середньоквадратичну різницю для кожної пари зменшених сірих зображень.
4. Порівнюються отримані середньоквадратичні різниці з деяким порогом. На виході видає коефіцієнту візуального спотворення.

Для кожного з утворених стеганоконтейнерів визначався даний коефіцієнт, після чого розраховувалось його середня значення.

При різних комбінаціях параметрів V_i визнались ***розміри утворених стеганоконтейнерів*** кожного із обраних SVG зображень. Після чого виконувалось порівняння із оригіналом для визначення у скільки разів збільшується розмір стеганоконтейнерів, у таблиці заносився середній результат. Додатково проводилось дослідження по стисненню стеганоконтейнерів за допомогою Zip архіватора.

В теорії стійкість до ***афінних перетворень*** гарантується інваріантністю (незмінністю) кривих Без'є відносно них. На практиці точність координат опорних точок, що задають криві обмежена, тому координати опорних точок після виконання афінних перетворень можуть бути не точними (виникає похибка округлення), що може призвести до спотворення секретного повідомлення при його вилученні. У роботі для кожної комбінації параметрів V_i виконувались афінні та майже афінні перетворення над отриманими стеганоконтейнерами, а при вилученні підраховувалось кількість спотворених біт секретного повідомлення. В результаті отримали експериментальні дані з стійкості запропонованих алгоритмів до атак на основі афінних перетворень.

У підрозділі 1.3 наводиться загальна формула та часткові випадки виконання афінних перетворень, до кожної координати (точки) зображення. У формулі (4.1) наведено загальний принцип проведення експериментального дослідження, щодо перевірки стійкості запропонованих алгоритмів до афінних перетворень:

$$S_k = F\left(S_{k-1}, \alpha_{1,1}^k, \alpha_{1,2}^k, \alpha_{2,1}^k, \alpha_{2,2}^k, \beta_1^k, \beta_2^k, \varepsilon_1^k, \varepsilon_2^k, \varepsilon_3^k, \varepsilon_4^k, \varepsilon_5^k, \varepsilon_6^k\right),\ k = k+1, \quad (4.1)$$

де F – афінне перетворення, S_0 – початковий стеганоконтейнер, $\alpha_{1,1}^k, \alpha_{1,2}^k, \alpha_{2,1}^k, \alpha_{2,2}^k, \beta_1^k, \beta_2^k, \varepsilon_1^k, \varepsilon_2^k, \varepsilon_3^k, \varepsilon_4^k, \varepsilon_5^k, \varepsilon_6^k \in R, \quad k \in N$. В залежності від підібраних коефіцієнтів $\alpha_{1,1}^k$, $\alpha_{1,2}^k$, $\alpha_{2,1}^k$, $\alpha_{2,2}^k$, β_1^k, β_2^k, ε_1^k, ε_2^k, ε_3^k, ε_4^k, ε_5^k, ε_6^k виконувалось те чи інше афінне перетворення (перенесення, повороту, майже повороту, зсуву за віссю абсцис і ординат, пропорційного та непропорційного масштабування). Вилучення секретного повідомлення та підрахунок кількості помилкових біт виконувалось при кожному $k \geq 1$. На основі даних вилучення будувались графіки для кожного виду афінних перетворень.

Афінні перетворення виконувались при наступних коефіцієнтах $\alpha_{1,1}^k$, $\alpha_{1,2}^k$, $\alpha_{2,1}^k$, $\alpha_{2,2}^k$, β_1^k, β_2^k, ε_1^k, ε_2^k, ε_3^k, ε_4^k, ε_5^k, ε_6^k згідно (4.1):

а) *перетворення перенесення*: $\alpha_{1,1}^k = \alpha_{1,2}^k = \alpha_{2,1}^k = \alpha_{2,2}^k = 0$, $\beta_1^k \in [-500, 500]$, $\beta_2^k \in [-500, 500]$, $\varepsilon_1^k = \varepsilon_2^k = \varepsilon_3^k = \varepsilon_4^k = \varepsilon_5^k = \varepsilon_6^k = 0$, $k = \overline{1,100}$;

б) *перетворення повороту* навколо точки $(0,0,1)$: $\alpha_{1,1}^k = \cos\theta$, $\alpha_{1,2}^k = -\sin\theta$, $\alpha_{2,1}^k = \sin\theta$, $\alpha_{2,2}^k = -\cos\theta$, $\theta = 1$, $\beta_1^k = \beta_2^k = 0$, $\varepsilon_1^k = \varepsilon_2^k = \varepsilon_3^k = \varepsilon_4^k = \varepsilon_5^k = \varepsilon_6^k = 0$, $k = \overline{1,360}$;

в) *перетворення зсуву за віссю абсцис*: $\alpha_{1,1}^k = \alpha_{2,2}^k = 1$, $\alpha_{1,2}^k = 0{,}01$, $\alpha_{2,1}^k = \beta_1^k = \beta_2^k = 0$, $\varepsilon_1^k = \varepsilon_2^k = \varepsilon_3^k = \varepsilon_4^k = \varepsilon_5^k = \varepsilon_6^k = 0$, $k = \overline{1,100}$; *перетворення зсуву за віссю ординат*: $\alpha_{1,1}^k = \alpha_{2,2}^k = 1$, $\alpha_{1,2}^k = \beta_1^k = \beta_2^k = 0$, $\alpha_{2,1}^k = 0{,}01$, $\varepsilon_1^k = \varepsilon_2^k = \varepsilon_3^k = \varepsilon_4^k = \varepsilon_5^k = \varepsilon_6^k = 0$, $k = \overline{1,100}$;

г) *перетворення пропорційного масштабування для стиснення зображення*: $\alpha_{1,1}^k = \alpha_{2,2}^k = 0{,}99$, $\alpha_{1,2}^k = \alpha_{2,1}^k = \beta_1^k = \beta_2^k = 0$, $\varepsilon_1^k = \varepsilon_2^k = \varepsilon_3^k = \varepsilon_4^k = \varepsilon_5^k = \varepsilon_6^k = 0$, $k = \overline{1,99}$; *перетворення пропорційного масштабування для розширення зображення*: $\alpha_{1,1}^k = \alpha_{2,2}^k = 1{,}01$, $\alpha_{1,2}^k = \alpha_{2,1}^k = \beta_1^k = \beta_2^k = 0$, $\varepsilon_1^k = \varepsilon_2^k = \varepsilon_3^k = \varepsilon_4^k = \varepsilon_5^k = \varepsilon_6^k = 0$, $k = \overline{1,100}$;

д) *перетворення непропорційного масштабування за віссю абсцис для стиснення зображення*: $\alpha_{1,1}^k = 0{,}99$, $\alpha_{1,2}^k = \alpha_{2,1}^k = \beta_1^k = \beta_2^k = 0$, $\alpha_{2,2}^k = 1$, $\varepsilon_1^k = \varepsilon_2^k = \varepsilon_3^k = \varepsilon_4^k = \varepsilon_5^k = \varepsilon_6^k = 0$, $k = \overline{1,99}$; *перетворення непропорційного масштабування за віссю абсцис для розширення зображення*: $\alpha_{1,1}^k = 1{,}01$, $\alpha_{1,2}^k = \alpha_{2,1}^k = \beta_1^k = \beta_2^k = 0$, $\alpha_{2,2}^k = 1$, $\varepsilon_1^k = \varepsilon_2^k = \varepsilon_3^k = \varepsilon_4^k = \varepsilon_5^k = \varepsilon_6^k = 0$, $k = \overline{1,100}$;

е) *перетворення непропорційного масштабування за віссю ординат для стиснення зображення*: $\alpha_{1,1}^k = 1$, $\alpha_{1,2}^k = \alpha_{2,1}^k = \beta_1^k = \beta_2^k = 0$, $\alpha_{2,2}^k = 0{,}99$, $\varepsilon_1^k = \varepsilon_2^k = \varepsilon_3^k = \varepsilon_4^k = \varepsilon_5^k = \varepsilon_6^k = 0$, $k = \overline{1,99}$; *перетворення непропорційного масштабування за віссю ординат для розширення зображення*: $\alpha_{1,1}^k = 1$, $\alpha_{1,2}^k = \alpha_{2,1}^k = \beta_1^k = \beta_2^k = 0$, $\alpha_{2,2}^k = 1{,}01$, $\varepsilon_1^k = \varepsilon_2^k = \varepsilon_3^k = \varepsilon_4^k = \varepsilon_5^k = \varepsilon_6^k = 0$, $k = \overline{1,100}$;

є) *майже афінне перетворення повороту* навколо точки $(0,0,1)$: $\alpha_{1,1}^k = \cos\theta$, $\alpha_{1,2}^k = -\sin\theta$, $\alpha_{2,1}^k = \sin\theta$, $\alpha_{2,2}^k = -\cos\theta$, $\theta = 1$, $\beta_1^k = \beta_2^k = 0$, $\varepsilon_1^k = \varepsilon_4^k = \varepsilon_6^k = -0{,}0001$, $\varepsilon_2^k = \varepsilon_3^k = \varepsilon_5^k = 0{,}0001$, $k = \overline{1,360}$.

Загалом проведено 36 експериментів з визначення коефіцієнту візуального спотворення, зміну розмірів стеганоконтейнерів, швидкостей вбудовування/вилучення інформації та стійкості алгоритмів StegoBIT і StegoTEMPL до афінних та майже афінних перетворень при різних комбінаціях параметрів V_i, $i = \overline{1,5}$ та описаних вище стеганоключів. При проведенні кожного експерименту відбувалася зміна одного з параметрів приховування V_i (відповідно крокам їх зміни визначених в підрозділі 4.1) або використовуваного стеганоключа.

Опишемо одержанні середні результати експериментів (кожних 6-ти проведених) згрупованих по різним значенням параметрів V_3 та V_5. Оригінальні середні результати стійкості до атак перенесення, повороту, майже повороту, зсуву, пропорційного та непропорційного масштабування розроблених алгоритмів кожного експерименту приведені в додатку Б.

Експерименти 1-6. Приховування інформації у SVG зображення здійснювалося за обраними стеганоключами та наступними параметрами: $V_1 = 3$, $V_2 \geq 1$, $V_3 = 40$, $V_4 = \{5,6\}$. Вилучення вбудованої інформації здійснювалося за параметром $V_5 = 2 \cdot 10^{-5}$.

При кожному вбудовуванні і вилученні прихованої інформації з кожного стеганоконтейнера програмно визначався час виконання даних операцій. У табл. 4.2 наведені загальні середні результати затрати часу за кожним з алгоритмів та їх *швидкісні характеристики* (після виконання 100 ітерацій для кожного SVG зображення).

Таблиця 4.2

Середні швидкісні характеристики розроблених алгоритмів

Алгоритм приховування	Процесор	Розмір прихованої інформації, біт	Час приховування, с	Швидкість вбудовування, біт/с	Час вилучення, с	Швидкість вилучення, біт/с
StegoBIT	1	1024	0,1451	7057,20	2,6979	379,55
	2		0,1241	8251,41	1,14565	893,82
	3		0,2316	4421,42	1,1538	887,50
StegoTEMPL	1		0,1271	8056,65	0,74178	1380,46
	2		0,2327	4400,52	2,3516	435,45
	3		0,1305	7846,74	1,12	914,20
1 – AMD A10-4600M, 2.3 GHz; 2 – AMD Phenom(tm) II X4 945 Processor, 3.00 GHz; 3 – Intel® Pentium® quad core processon N3540, up to 2.66 GHz.						

Результати порівнянь середнього значення *розміру всіх стеганоконтейнерів та їх стиснутих копій* (за допомогою Zip архіватора) *відносно контейнерів-оригіналів* наведені в табл. 4.3.

Середнє значення порівняння візуального спотворення всіх стеганоконтейнерів відносно контейнера-оригінала наведені в табл. 4.4. Результати отримані при використанні ПЗ AntiDupl.NET.

Таблиця 4.3

Порівняння середніх значень розмірів одержаних стеганоконтейнерів

Алгоритм приховування	Розмір прихованої інформації, біт	Розмір контейнера «до», КБайт	Розмір контейнера «після», КБайт	Розмір стиснутого контейнера «після», КБайт	Збільшення розміру контейнера «після» відносно контейнера «до», раз	Збільшення розміру стиснутого контейнера «після» відносно контейнера «до», раз
StegoBIT	1024	26,81	55,91	24,33	2,09	0,91
StegoTEMPL			43,25	18,98	1,61	0,71

Таблиця 4.4

Порівняння середніх значень коефіцієнту візуального спотворення

Алгоритм приховування	Коефіцієнт візуального спотворення контейнера «після» відносно контейнера «до», %
StegoBIT	0,12
StegoTEMPL	0,12

На рис. 4.4 наведені результати порівняння кількості втрачених біт секретного повідомлення при накладанні на стеганоконтейнери, за формулою (4.1), перетворень перенесення.

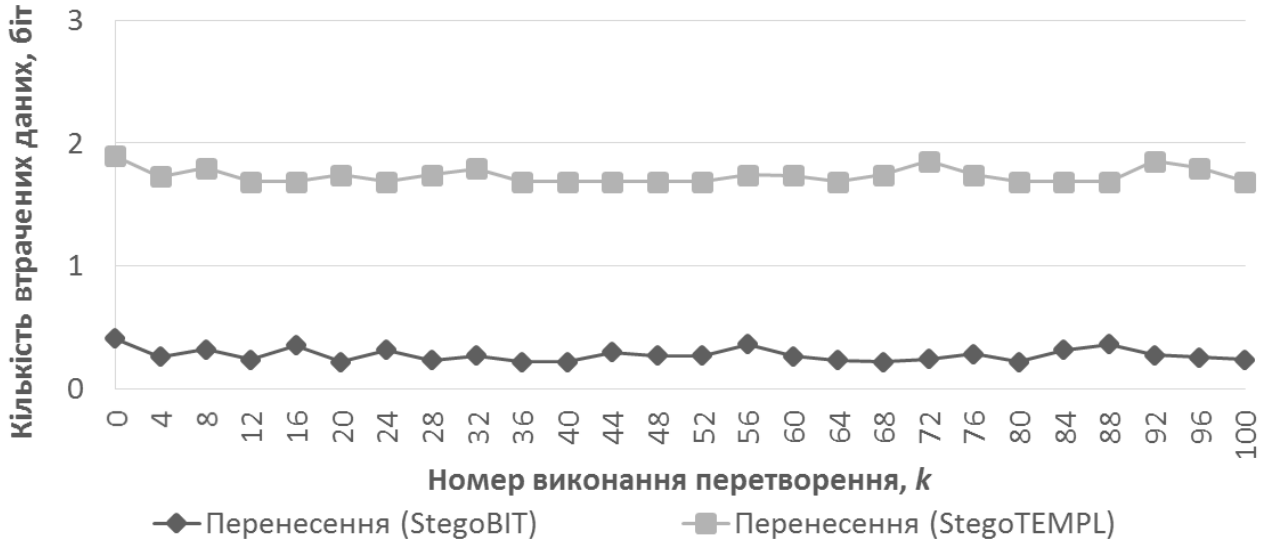


Рис. 4.4. Результати вилучення інформації з стеганоконтейнерів після виконання перетворень перенесення

На рис. 4.5 наведені результати порівняння кількості втрачених біт секретного повідомлення при накладанні на стеганоконтейнери, за формулою (4.1), перетворень повороту та майже повороту.

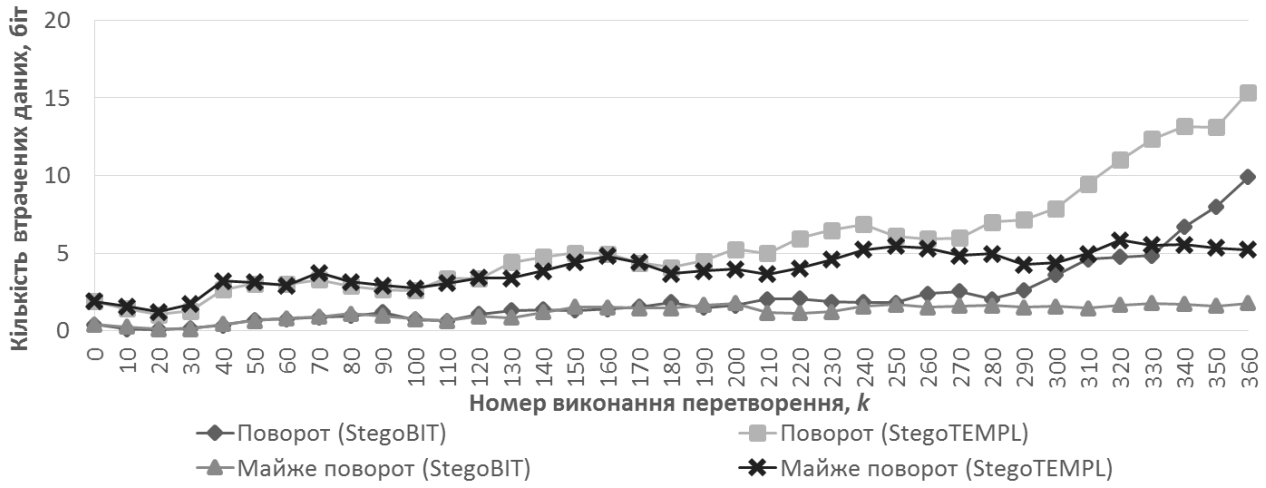


Рис. 4.5. Результати вилучення інформації з стеганоконтейнерів після виконання перетворень повороту та майже повороту

На рис. 4.6 та рис. 4.7 наведені результати порівняння кількості втрачених біт секретного повідомлення при поступовому стисненні/розширенні стеганоконтейнерів за перетвореннями пропорційного і непропорційного масштабування, відповідно формулі (4.1).

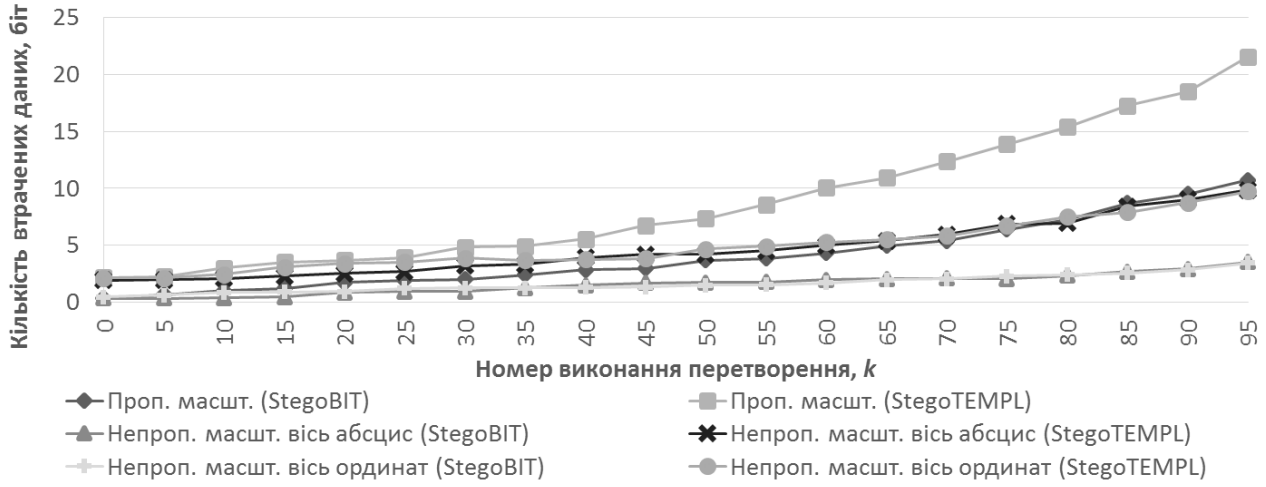


Рис. 4.6. Результати вилучення інформації з стиснених стеганоконтейнерів після виконання перетворень пропорційного і непропорційного масштабування

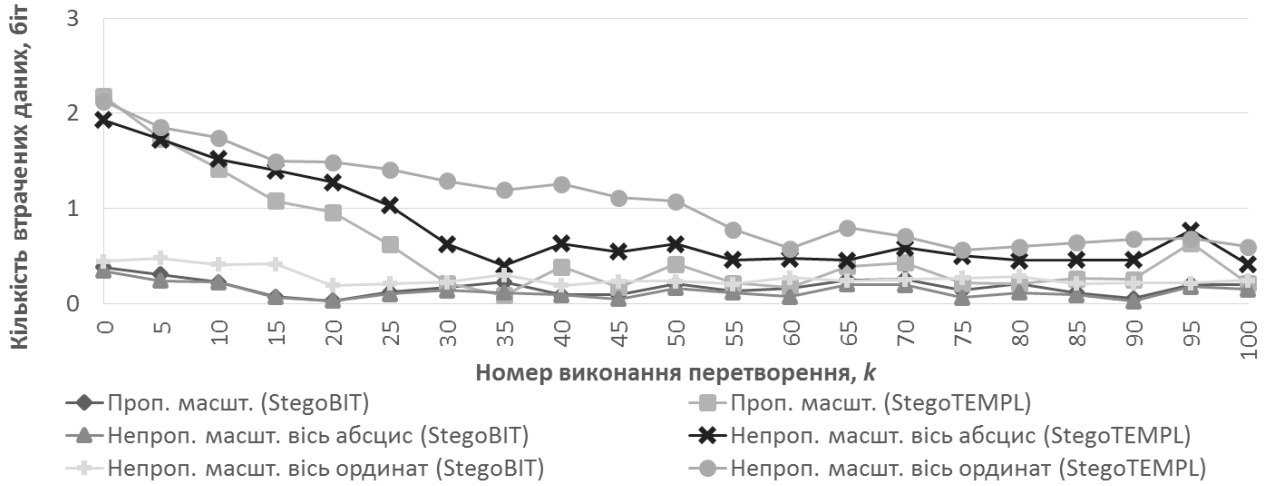


Рис. 4.7. Результати вилучення інформації з розширених стеганоконтейнерів після виконання перетворень пропорційного і непропорційного масштабування

На рис. 4.8 наведені результати порівняння кількості втрачених біт секретного повідомлення при накладанні на стеганоконтейнери, за формулою (4.1), перетворень зсуву.

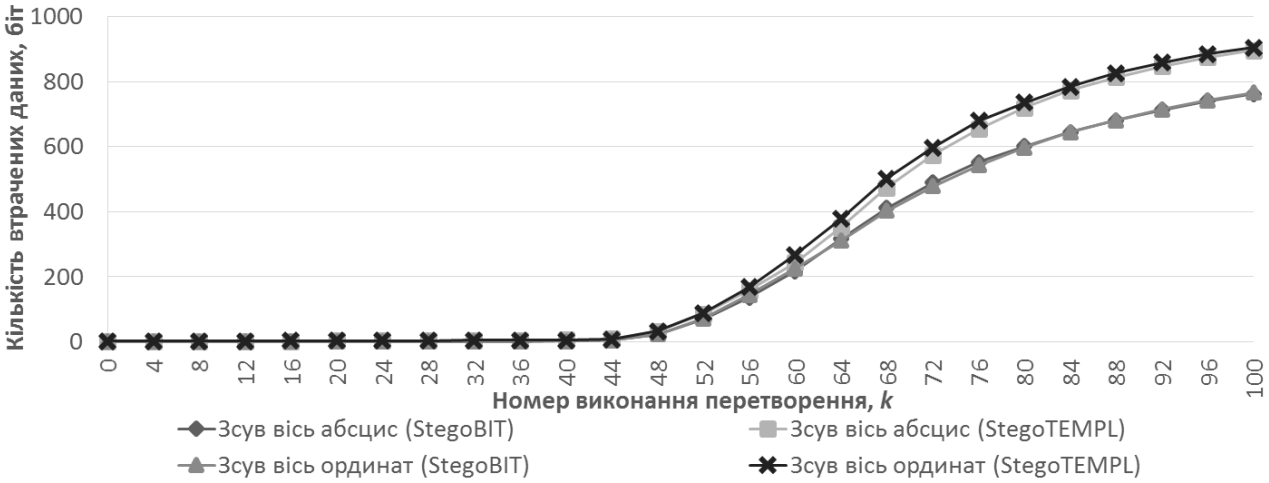


Рис. 4.8. Результати вилучення інформації з стеганоконтейнерів після виконання перетворень зсуву

***Згідно результатів експериментів* 1-6** визначено, що середня швидкість вбудовування/вилучення інформації алгоритмом StegoTEMPL на 2,91% і 26,34% відповідно кращі ніж в StegoBIT. За алгоритмом StegoTEMPL розміри стеганоконтейнерів в середньому менші на 23%. Середній коефіцієнт спотворення зображень після вбудовування інформації для обох алгоритмів менший за 0,5%. У алгоритмів StegoBIT стійкість до перенесення, повороту і

майже повороту, зсуву, пропорційного і непропорційного масштабування краща ніж в StegoTEMPL.

Експерименти 7-12. Приховування інформації у SVG зображення здійснювалося за обраними стеганоключами та наступними параметрами: $V_1 = 3$, $V_2 \geq 1$, $V_3 = 40$, $V_4 = \{5,6\}$. Вилучення вбудованої інформації здійснювалося за параметром $V_5 = 4 \cdot 10^{-5}$.

При кожному вбудовуванні і вилученні прихованої інформації з кожного стеганоконтейнера програмно визначався час виконання даних операцій. У табл. 4.5 наведені загальні середні результати затрати часу за кожним з алгоритмів та їх *швидкісні характеристики* (після виконання 100 ітерацій для кожного SVG зображення).

Таблиця 4.5

Середні швидкісні характеристики розроблених алгоритмів

Алгоритм приховування	Процесор	Розмір прихованої інформації, біт	Час приховування, с	Швидкість вбудовування, біт/с	Час вилучення, с	Швидкість вилучення, біт/с
StegoBIT	1	1024	0,1585	6460,57	2,7214	376,28
	2		0,1239	8264,73	1,2421	824,41
	3		0,2688	3809,52	1,4541	704,22
StegoTEMPL	1		0,1312	7804,88	0,7289	1404,86
	2		0,2461	4160,91	1,8891	542,06
	3		0,1358	7540,50	1,3827	740,58
1 – AMD A10-4600M, 2.3 GHz; 2 – AMD Phenom(tm) II X4 945 Processor, 3.00 GHz; 3 – Intel® Pentium® quad core processon N3540, up to 2.66 GHz.						

Результати порівнянь середнього значення *розміру всіх стеганоконтейнерів та їх стиснутих копій* (за допомогою Zip архіватора) *відносно контейнерів-оригіналів* наведені в табл. 4.6.

Таблиця 4.6

Порівняння середніх значень розмірів одержаних стеганоконтейнерів

Алгоритм приховування	Розмір прихованої інформації, біт	Розмір контейнера «до», КБайт	Розмір контейнера «після», КБайт	Розмір стиснутого контейнера «після», КБайт	Збільшення розміру контейнера «після» відносно контейнера «до», раз	Збільшення розміру стиснутого контейнера «після» відносно контейнера «до», раз
StegoBIT	1024	26,81	55,91	24,34	2,09	0,91
StegoTEMPL			43,25	18,98	1,61	0,71

Середнє значення порівняння візуального спотворення всіх стеганоконтейнерів відносно контейнера-оригінала наведені в табл. 4.7. Результати отримані при використанні ПЗ AntiDupl.NET.

Таблиця 4.7

Порівняння середніх значень коефіцієнту візуального спотворення

Алгоритм приховування	Коефіцієнт візуального спотворення контейнера «після» відносно контейнера «до», %
StegoBIT	0,12
StegoTEMPL	0,12

На рис. 4.9 наведені результати порівняння кількості втрачених біт секретного повідомлення при накладанні на стеганоконтейнери, за формулою (4.1), перетворень перенесення.

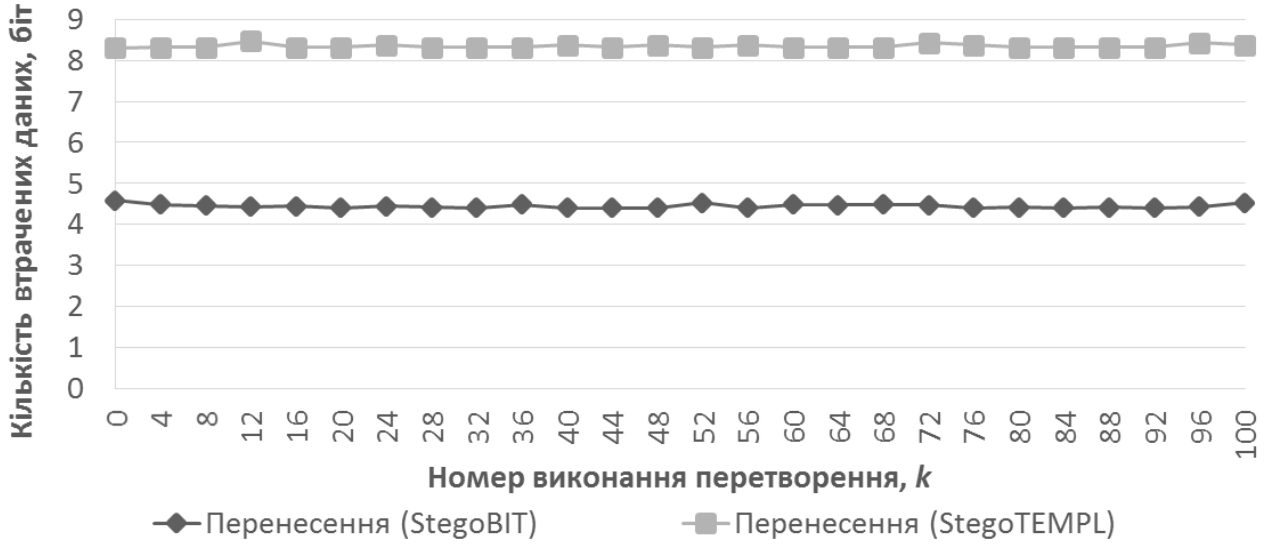


Рис. 4.9. Результати вилучення інформації з стеганоконтейнерів після виконання перетворень перенесення

На рис. 4.10 наведені результати порівняння кількості втрачених біт секретного повідомлення при накладанні на стеганоконтейнери, за формулою (4.1), перетворень повороту та майже повороту.

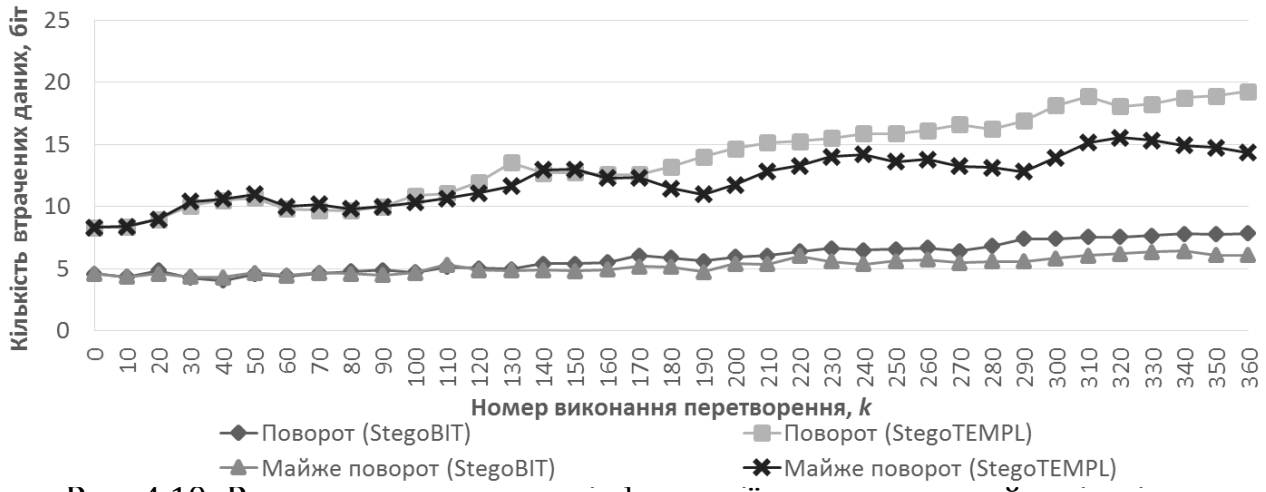


Рис. 4.10. Результати вилучення інформації з стеганоконтейнерів після виконання перетворень повороту та майже повороту

На рис. 4.11 та рис. 4.12 наведені результати порівняння кількості втрачених біт секретного повідомлення при поступовому стисненні/розширенні стеганоконтейнерів за перетвореннями пропорційного і непропорційного масштабування, відповідно формулі (4.1).

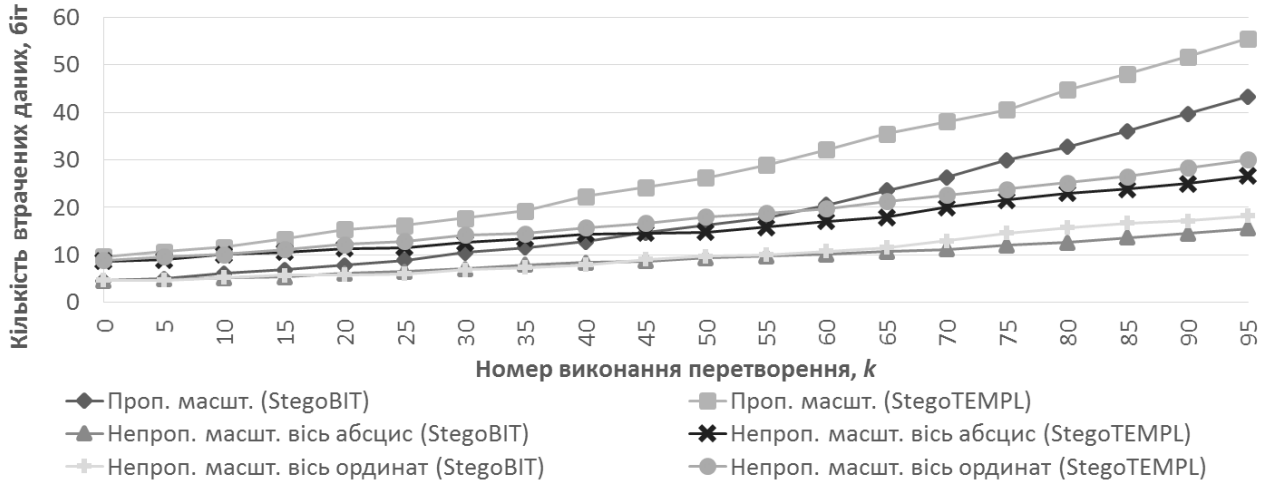


Рис. 4.11. Результати вилучення інформації з стиснутих стеганоконтейнерів після виконання перетворень пропорційного і непропорційного масштабування

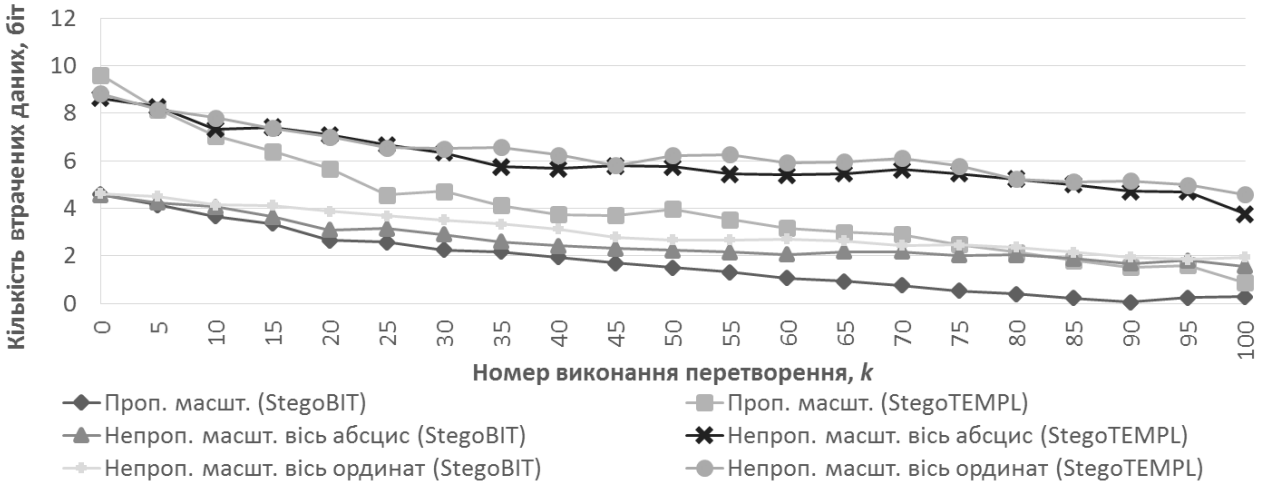


Рис. 4.12. Результати вилучення інформації з розтягнутих стеганоконтейнерів після виконання перетворень пропорційного і непропорційного масштабування

На рис. 4.13 наведені результати порівняння кількості втрачених біт секретного повідомлення при накладанні на стеганоконтейнери, за формулою (4.1), перетворень зсуву.

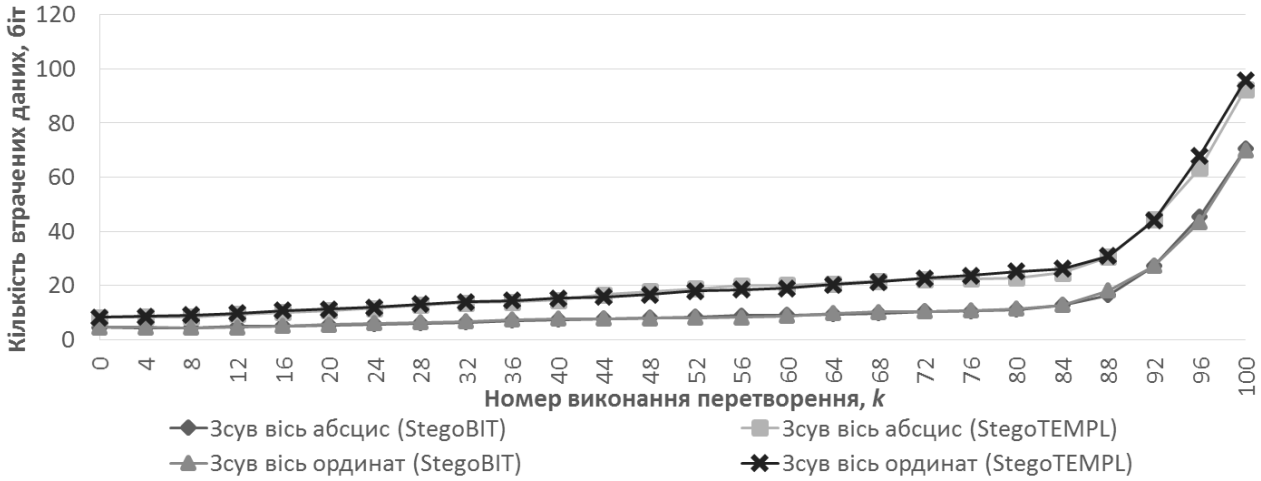


Рис. 4.13. Результати вилучення інформації з стеганоконтейнерів після виконання перетворень зсуву

***Згідно результатів експериментів* 7-12** визначено, що середня швидкість вбудовування/вилучення інформації алгоритмом StegoTEMPL на 5,27% і 41,08% відповідно кращі ніж в StegoBIT. За алгоритмом StegoTEMPL розміри стеганоконтейнерів в середньому менші на 23%. Середній коефіцієнт спотворення зображень після вбудовування інформації для обох алгоритмів менший за 0,5%. У алгоритмів StegoBIT стійкість до перенесення, повороту і

майже повороту, зсуву, пропорційного і непропорційного масштабування краща ніж в StegoTEMPL.

Експерименти 13-18. Приховування інформації у SVG зображення здійснювалося за обраними стеганоключами та наступними параметрами: $V_1 = 3$, $V_2 \geq 1$, $V_3 = 60$, $V_4 = \{5,6\}$. Вилучення вбудованої інформації здійснювалося за параметром $V_5 = 2 \cdot 10^{-5}$.

При кожному вбудовуванні і вилученні прихованих даних з кожного стеганоконтейнера програмно визначався час виконання даних операцій. У табл. 4.8 наведені загальні середні результати затрати часу за кожним з алгоритмів та їх *швидкісні характеристики* (після виконання 100 ітерацій для кожного SVG зображення).

Таблиця 4.8

Середні швидкісні характеристики розроблених алгоритмів

Алгоритм приховування	Процесор	Розмір прихованої інформації, біт	Час приховування, с	Швидкість вбудовування, біт/с	Час вилучення, с	Швидкість вилучення, біт/с
StegoBIT	1	1024	0,1697	6034,18	0,75	1365,33
	2		0,1413	7246,99	1,6434	623,10
	3		0,2218	4616,77	0,8781	1166,15
StegoTEMPL	1		0,0948	10801,69	1,0442	980,66
	2		0,3271	3130,54	0,8768	1167,88
	3		0,1471	6961,25	1,2303	832,32
1 – AMD A10-4600M, 2.3 GHz; 2 – AMD Phenom(tm) II X4 945 Processor, 3.00 GHz; 3 – Intel® Pentium® quad core processon N3540, up to 2.66 GHz.						

Результати порівнянь середнього значення *розміру всіх стеганоконтейнерів та їх стиснутих копій* (за допомогою Zip архіватора) *відносно контейнерів-оригіналів* наведені в табл. 4.9.

Таблиця 4.9

Порівняння середніх значень розмірів одержаних стеганоконтейнерів

Алгоритм приховування	Розмір прихованої інформації, біт	Розмір контейнера «до», КБайт	Розмір контейнера «після», КБайт	Розмір стиснутого контейнера «після», КБайт	Збільшення розміру контейнера «після» відносно контейнера «до», раз	Збільшення розміру стиснутого контейнера «після» відносно контейнера «до», раз
StegoBIT	1024	26,81	55,76	24,44	2,08	0,91
StegoTEMPL			43,09	19,03	1,61	0,71

Середнє значення порівняння візуального спотворення всіх стеганоконтейнерів відносно контейнера-оригінала наведені в табл. 4.10. Результати отримані при використанні ПЗ AntiDupl.NET.

Таблиця 4.10

Порівняння середніх значень коефіцієнту візуального спотворення

Алгоритм приховування	Коефіцієнт візуального спотворення контейнера «після» відносно контейнера «до», %
StegoBIT	0,1
StegoTEMPL	0,09

На рис. 4.14 наведені результати порівняння кількості втрачених біт секретного повідомлення при накладанні на стеганоконтейнери, за формулою (4.1), перетворень перенесення.

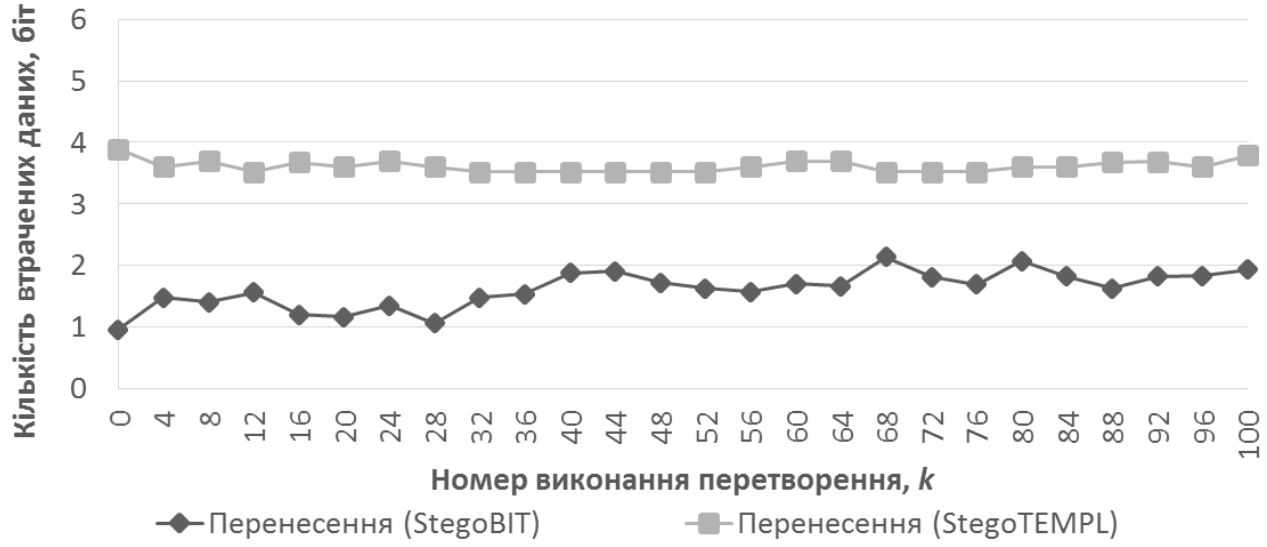


Рис. 4.14. Результати вилучення інформації з стеганоконтейнерів після виконання перетворень перенесення

На рис. 4.15 наведені результати порівняння кількості втрачених біт секретного повідомлення при накладанні на стеганоконтейнери, за формулою (4.1), перетворень повороту та майже повороту.

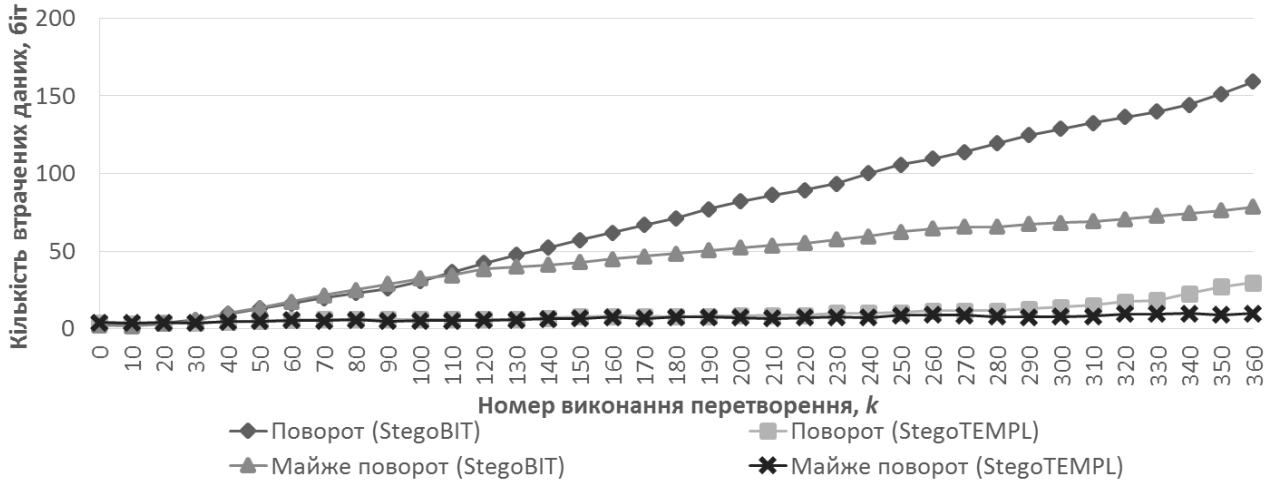


Рис. 4.15 Результати вилучення інформації з стеганоконтейнерів після виконання перетворень повороту та майже повороту

На рис. 4.16 та рис. 4.17 наведені результати порівняння кількості втрачених біт секретного повідомлення при поступовому стисненні/розширенні стеганоконтейнерів за перетвореннями пропорційного і непропорційного масштабування, відповідно формулі (4.1).

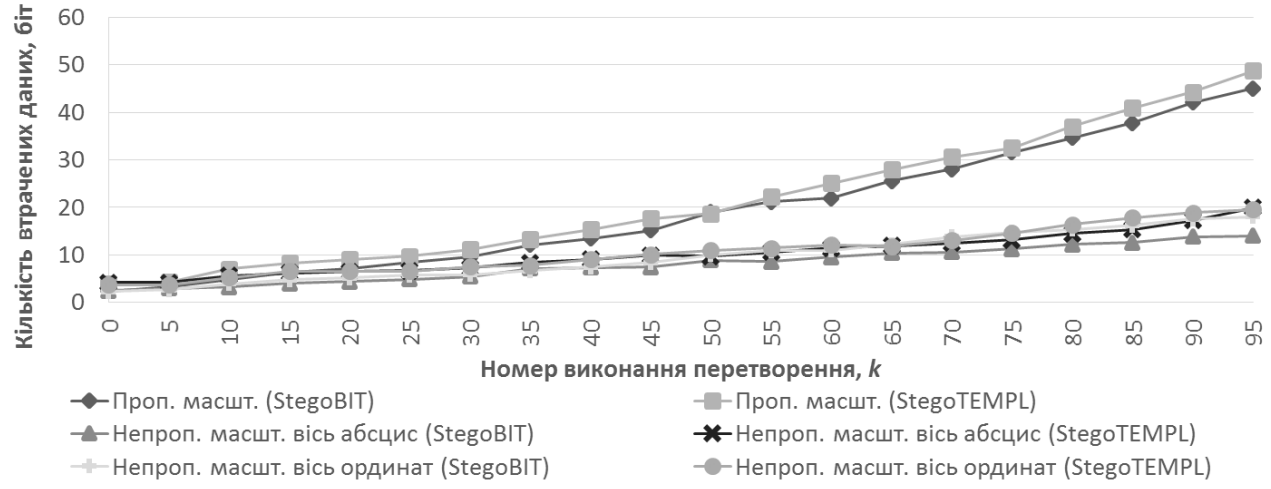


Рис. 4.16. Результати вилучення інформації з стиснутих стеганоконтейнерів після виконання перетворень пропорційного і непропорційного масштабування

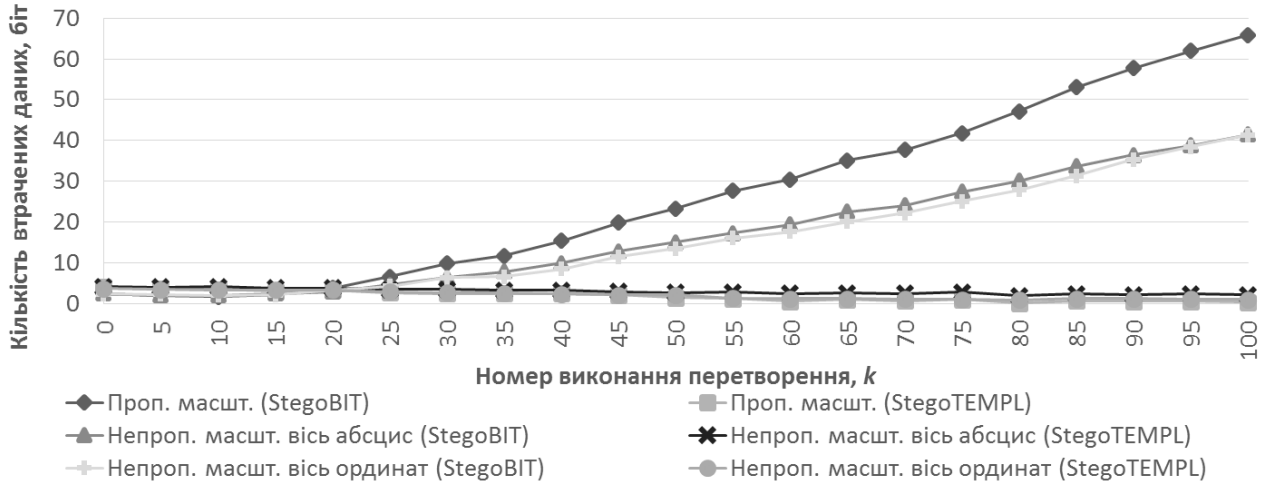


Рис. 4.17. Результати вилучення інформації з розтягнутих стеганоконтейнерів після виконання перетворень пропорційного і непропорційного масштабування

На рис. 4.18 наведені результати порівняння кількості втрачених біт секретного повідомлення при накладанні на стеганоконтейнери, за формулою (4.1), перетворень зсуву.

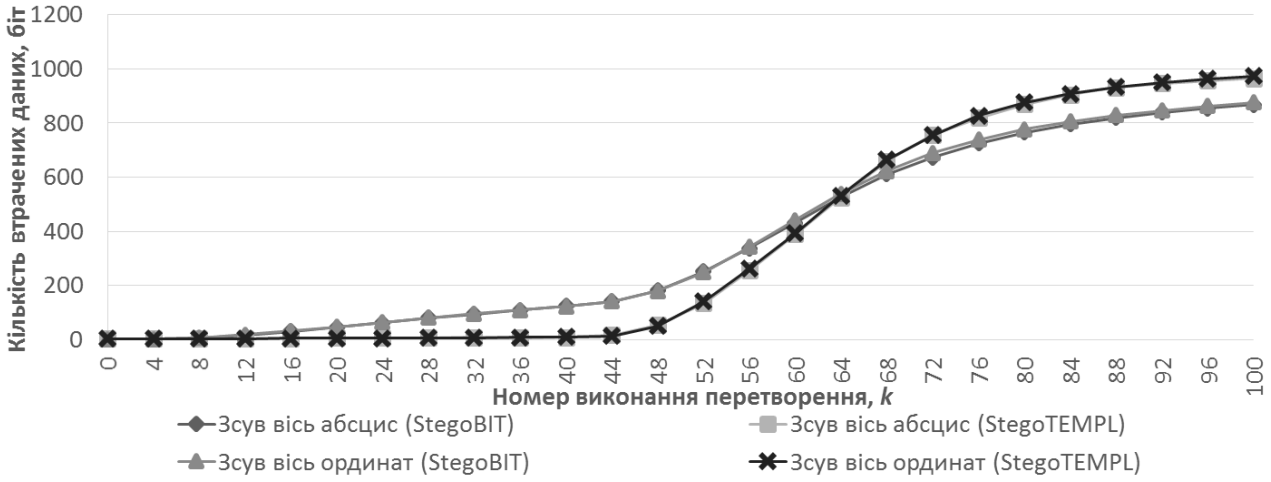


Рис. 4.18. Результати вилучення інформації з стеганоконтейнерів після виконання перетворень зсуву

***Згідно результатів експериментів* 13-18** визначено, що середня швидкість вбудовування інформації алгоритмом StegoTEMPL краща на 16,76% ніж в StegoBIT, проте швидкість вилучення даних гірша на 5,51%. За алгоритмом StegoTEMPL розміри стеганоконтейнерів в середньому менші на 23%. Середній коефіцієнт спотворення зображень після вбудовування інформації для обох алгоритмів менший за 0,5%. У алгоритмів StegoTEMPL

стійкість до повороту, майже повороту, пропорційного і непропорційного масштабування краща ніж в StegoBIT, проте дещо гірша до атаки перенесення.

Експерименти 19-24. Приховування інформації у SVG зображення здійснювалося за обраними стеганоключами та наступними параметрами: $V_1 = 3$, $V_2 \geq 1$, $V_3 = 60$, $V_4 = \{5,6\}$. Вилучення вбудованої інформації здійснювалося за параметром $V_5 = 4 \cdot 10^{-5}$.

При кожному вбудовуванні і вилученні прихованих даних з кожного стеганоконтейнера програмно визначався час виконання даних операцій. У табл. 4.11 наведені загальні середні результати затрати часу за кожним з алгоритмів та їх *швидкісні характеристики* (після виконання 100 ітерацій для кожного SVG зображення).

Таблиця 4.11

Середні швидкісні характеристики розроблених алгоритмів

Алгоритм приховування	Процесор	Розмір прихованої інформації, біт	Час приховування, с	Швидкість вбудовування, біт/с	Час вилучення, с	Швидкість вилучення, біт/с
StegoBIT	1	1024	0,1997	5127,69	0,8783	1165,89
	2		0,1467	6980,23	1,7449	586,85
	3		0,2216	4620,94	0,9002	1137,52
StegoTEMPL	1		0,1085	9437,79	1,1736	872,53
	2		0,3912	2617,59	0,9851	1039,49
	3		0,1555	6585,21	1,3596	753,16
1 – AMD A10-4600M, 2.3 GHz; 2 – AMD Phenom(tm) II X4 945 Processor, 3.00 GHz; 3 – Intel® Pentium® quad core processon N3540, up to 2.66 GHz.						

Результати порівнянь середнього значення *розміру всіх стеганоконтейнерів та їх стиснутих копій* (за допомогою Zip архіватора) *відносно контейнерів-оригіналів* наведені в табл. 4.12.

Таблиця 4.12

Порівняння середніх значень розмірів одержаних стеганоконтейнерів

Алгоритм приховування	Розмір прихованої інформації, біт	Розмір контейнера «до», КБайт	Розмір контейнера «після», КБайт	Розмір стиснутого контейнера «після», КБайт	Збільшення розміру контейнера «після» відносно контейнера «до», раз	Збільшення розміру стиснутого контейнера «після» відносно контейнера «до», раз
StegoBIT	1024	26,81	55,76	24,44	2,08	0,91
StegoTEMPL			43,09	19,03	1,61	0,71

Середнє значення порівняння візуального спотворення всіх стеганоконтейнерів відносно контейнера-оригінала наведені в табл. 4.13. Результати отримані при використанні ПЗ AntiDupl.NET.

Таблиця 4.13

Порівняння середніх значень коефіцієнту візуального спотворення

Алгоритм приховування	Коефіцієнт візуального спотворення контейнера «після» відносно контейнера «до», %
StegoBIT	0,1
StegoTEMPL	0,09

На рис. 4.19 наведені результати порівняння кількості втрачених біт секретного повідомлення при накладанні на стеганоконтейнери, за формулою (4.1), перетворень перенесення.

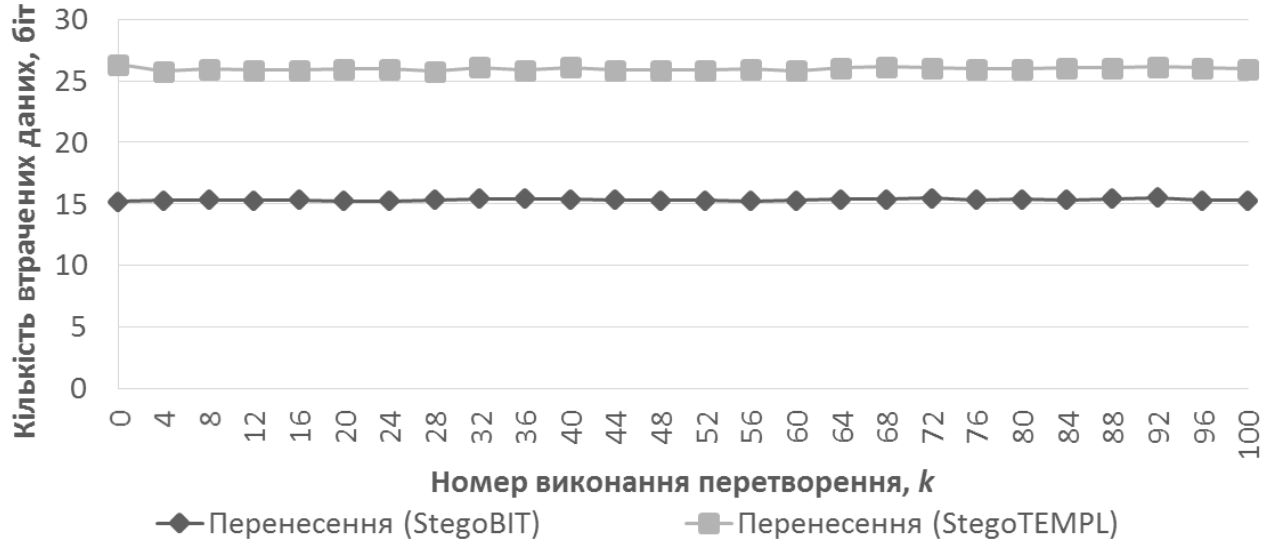


Рис. 4.19. Результати вилучення інформації з стеганоконтейнерів після виконання перетворень перенесення

На рис. 4.20. наведені результати порівняння кількості втрачених біт секретного повідомлення при накладанні на стеганоконтейнери, за формулою (4.1), перетворень повороту та майже повороту.

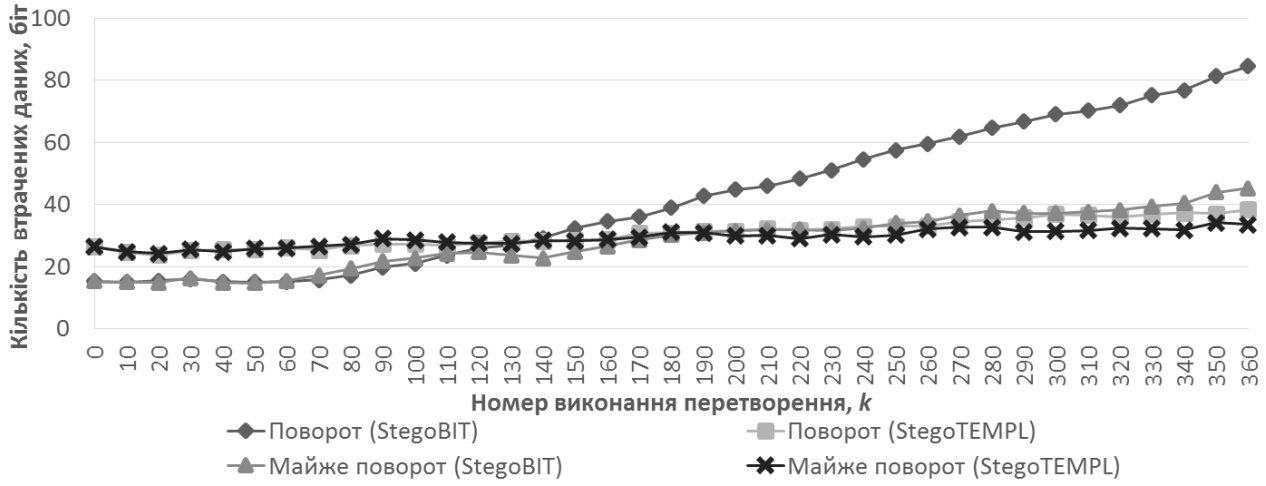


Рис. 4.20. Результати вилучення інформації з стеганоконтейнерів після виконання перетворень повороту та майже повороту

На рис. 4.21 та рис. 4.22 наведені результати порівняння кількості втрачених біт секретного повідомлення при поступовому стисненні/розширенні стеганоконтейнерів за перетвореннями пропорційного і непропорційного масштабування, відповідно формулі (4.1).

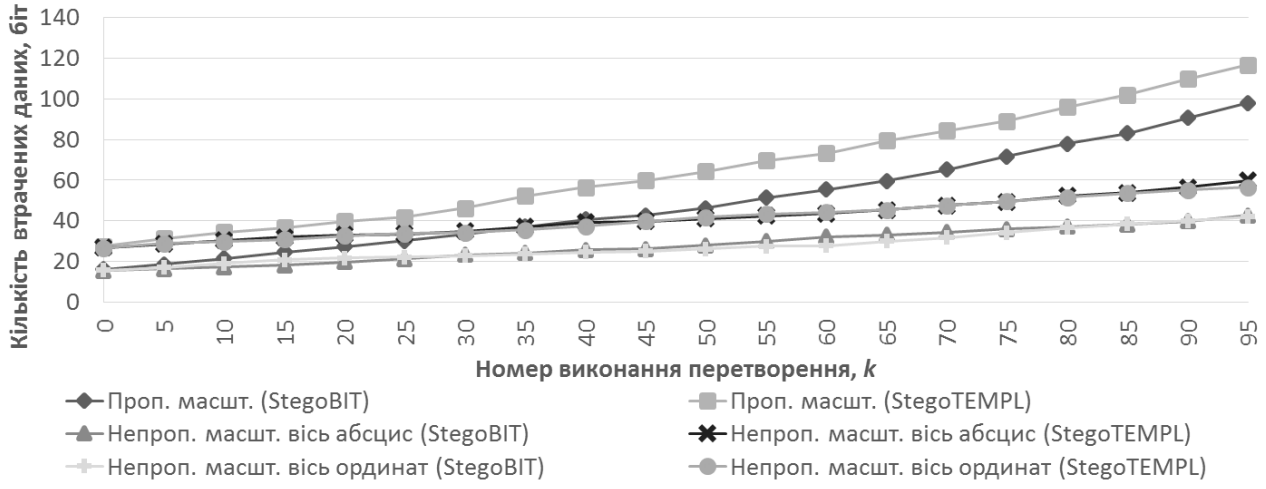


Рис. 4.21. Результати вилучення інформації з стиснутих стеганоконтейнерів після виконання перетворень пропорційного і непропорційного масштабування

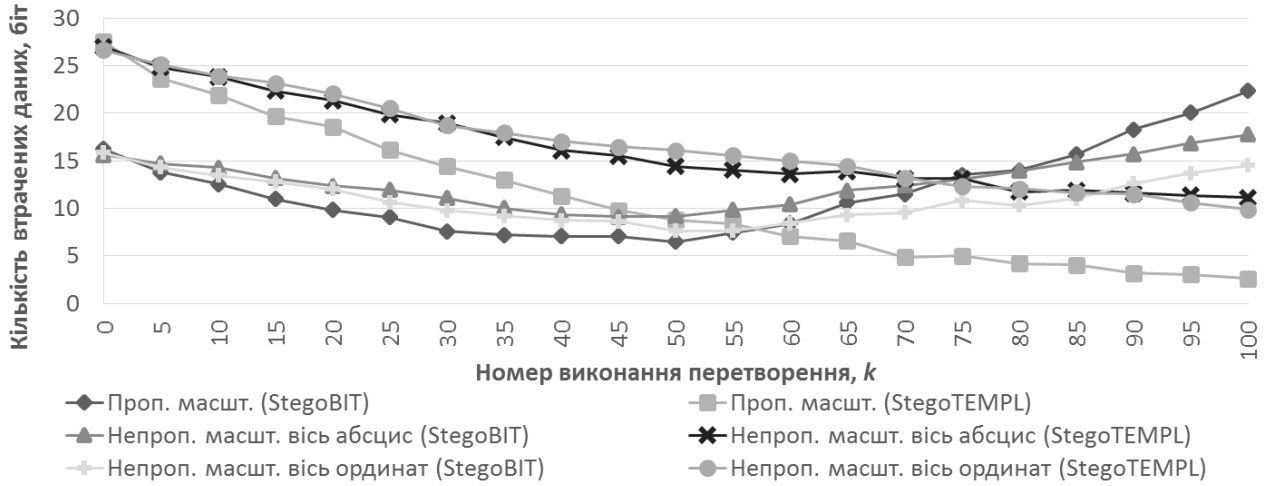


Рис. 4.22. Результати вилучення інформації з розтягнутих стеганоконтейнерів після виконання перетворень пропорційного і непропорційного масштабування

На рис. 4.23 наведені результати порівняння кількості втрачених біт секретного повідомлення при накладанні на стеганоконтейнери, за формулою (4.1), перетворень зсуву.

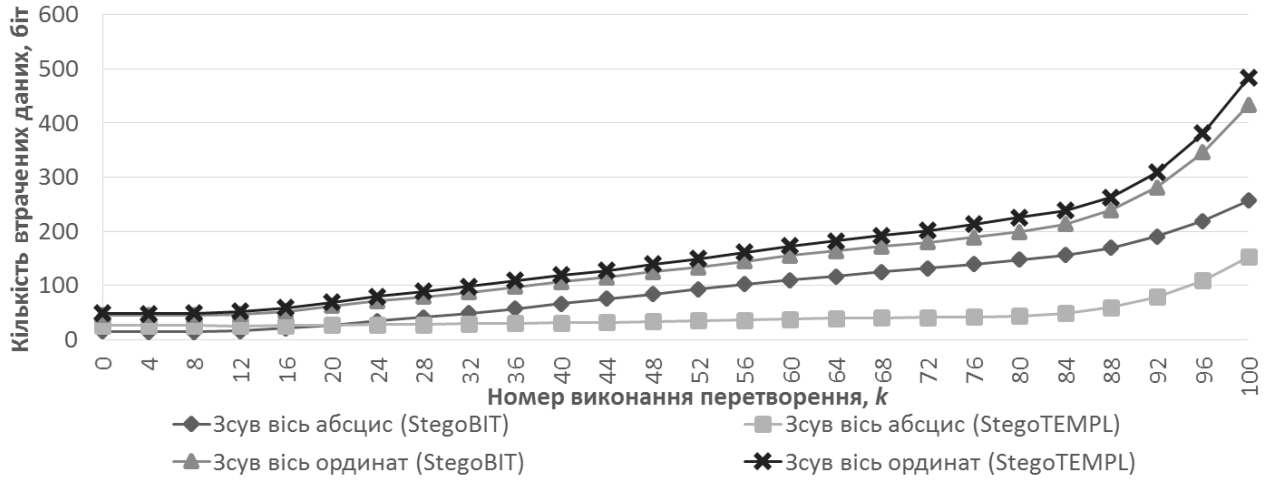


Рис. 4.23. Результати вилучення інформації з стеганоконтейнерів після виконання перетворень зсуву

***Згідно результатів експериментів* 19-24** визначено, що середня швидкість вбудовування інформації алгоритмом StegoTEMPL краща на 11,4% ніж в StegoBIT, проте швидкість вилучення даних гірша на 7,78%. За алгоритмом StegoTEMPL розміри стеганоконтейнерів в середньому менші на 23%. Середній коефіцієнт спотворення зображень після вбудовування інформації для обох алгоритмів менший за 0,5%. У алгоритмів StegoBIT

стійкість до перенесення, пропорційного і непропорційного масштабування краща ніж в StegoTEMPL, проте гірша до атаки повороту та майже повороту.

Експерименти 25-30. Приховування інформації у SVG зображення здійснювалося за обраними стеганоключами та наступними параметрами: $V_1 = 3$, $V_2 \geq 1$, $V_3 = 80$, $V_4 = \{5,6\}$. Вилучення вбудованої інформації здійснювалося за параметром $V_5 = 2 \cdot 10^{-5}$.

При кожному вбудовуванні і вилученні прихованих даних з кожного стеганоконтейнера програмно визначався час виконання даних операцій. У табл. 4.14 наведені загальні середні результати затрати часу за кожним з алгоритмів та їх *швидкісні характеристики* (після виконання 100 ітерацій для кожного SVG зображення).

Таблиця 4.14

Середні швидкісні характеристики розроблених алгоритмів

Алгоритм приховування	Процесор	Розмір прихованої інформації, біт	Час приховування, с	Швидкість вбудовування, біт/с	Час вилучення, с	Швидкість вилучення, біт/с
StegoBIT	1	1024	0,1877	5455,51	1,2764	802,26
	2		0,1265	8094,86	1,0031	1020,84
	3		0,4539	2256,00	1,7998	568,95
StegoTEMPL	1		0,1448	7071,82	0,5969	1715,53
	2		0,2739	3738,59	0,5804	1764,30
	3		0,2202	4650,32	1,5839	646,51
1 – AMD A10-4600M, 2.3 GHz; 2 – AMD Phenom(tm) II X4 945 Processor, 3.00 GHz; 3 – Intel® Pentium® quad core processon N3540, up to 2.66 GHz.						

Результати порівнянь середнього значення *розміру всіх стеганоконтейнерів та їх стиснутих копій* (за допомогою Zip архіватора) *відносно контейнерів-оригіналів* наведені в табл. 4.15.

Таблиця 4.15

Порівняння середніх значень розмірів одержаних стеганоконтейнерів

Алгоритм приховування	Розмір прихованої інформації, біт	Розмір контейнера «до», КБайт	Розмір контейнера «після», КБайт	Розмір стиснутого контейнера «після», КБайт	Збільшення розміру контейнера «після» відносно контейнера «до», раз	Збільшення розміру стиснутого контейнера «після» відносно контейнера «до», раз
StegoBIT	1024	26,81	55,74	24,40	2,08	0,91
StegoTEMPL			43,06	19,05	1,61	0,71

Середнє значення порівняння візуального спотворення всіх стеганоконтейнерів відносно контейнера-оригінала наведені в табл. 4.16. Результати отримані при використанні ПЗ AntiDupl.NET.

Таблиця 4.16

Порівняння середніх значень коефіцієнту візуального спотворення

Алгоритм приховування	Коефіцієнт візуального спотворення контейнера «після» відносно контейнера «до», %
StegoBIT	0,09
StegoTEMPL	0,09

На рис. 4.24 наведені результати порівняння кількості втрачених біт секретного повідомлення при накладанні на стеганоконтейнери, за формулою (4.1), перетворень перенесення.

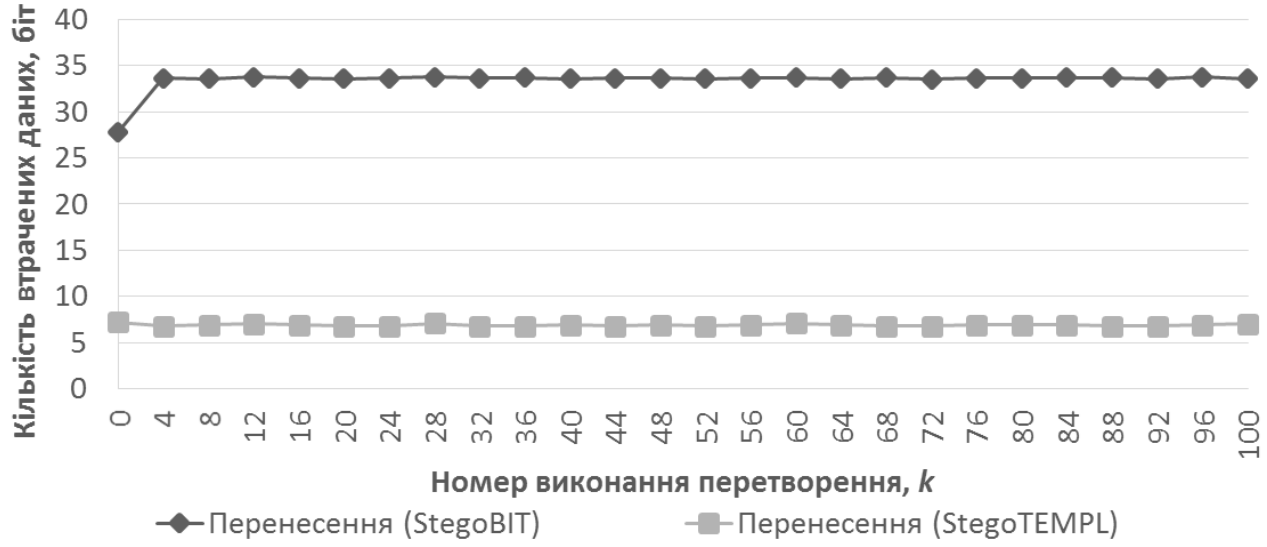


Рис. 4.24. Результати вилучення інформації з стеганоконтейнерів після виконання перетворень перенесення

На рис. 4.25 наведені результати порівняння кількості втрачених біт секретного повідомлення при накладанні на стеганоконтейнери, за формулою (4.1), перетворень повороту та майже повороту.

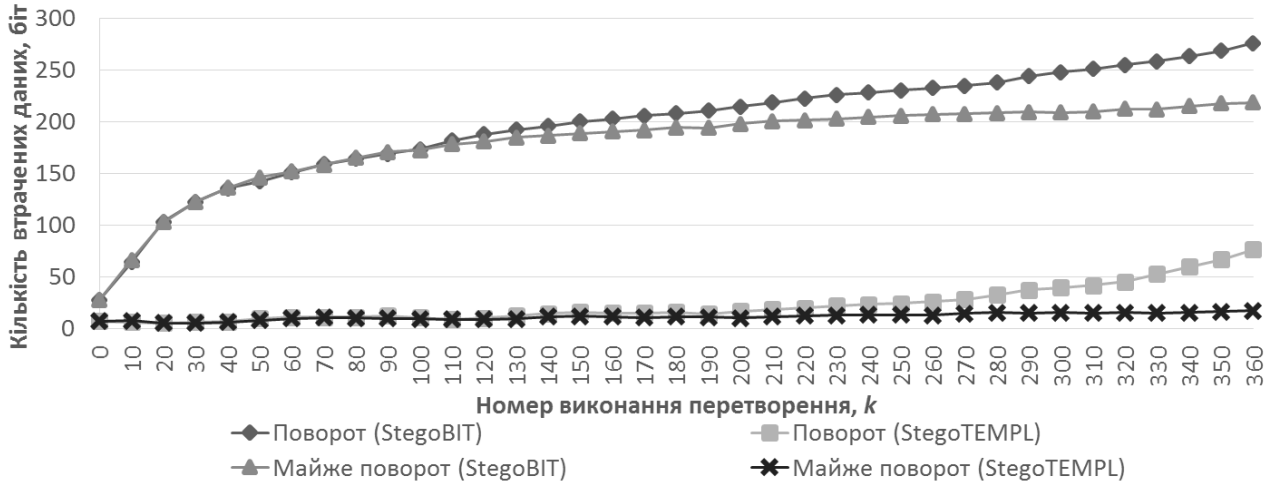


Рис. 4.25. Результати вилучення інформації з стеганоконтейнерів після виконання перетворень повороту та майже повороту

На рис. 4.26 та рис. 4.27 наведені результати порівняння кількості втрачених біт секретного повідомлення при поступовому стисненні/розширенні стеганоконтейнерів за перетвореннями пропорційного і непропорційного масштабування, відповідно формулі (4.1).

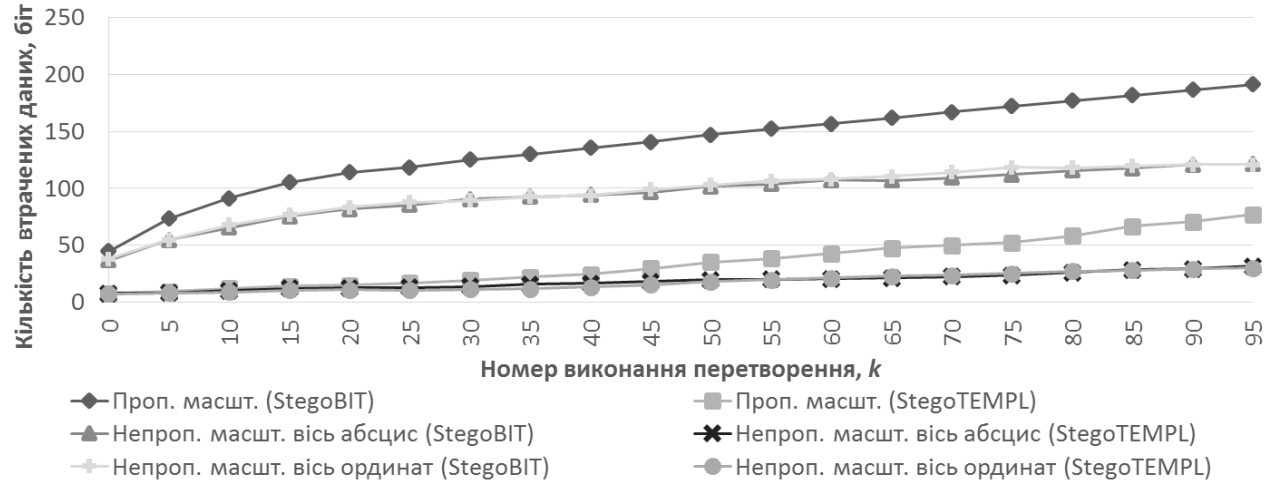


Рис. 4.26. Результати вилучення інформації з стиснутих стеганоконтейнерів після виконання перетворень пропорційного і непропорційного масштабування

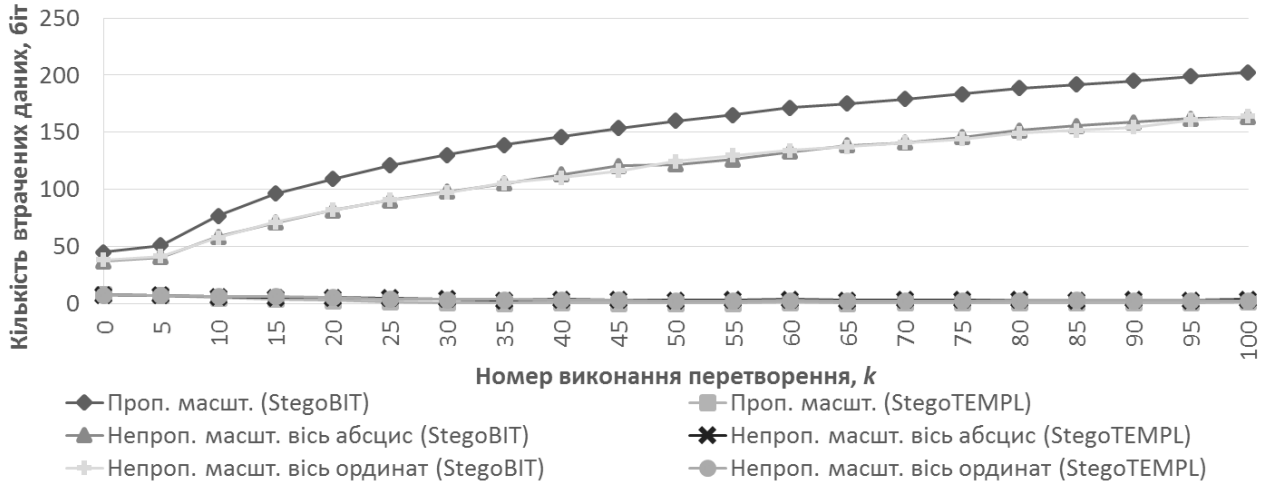


Рис. 4.27. Результати вилучення інформації з розтягнутих стеганоконтейнерів після виконання перетворень пропорційного і непропорційного масштабування

На рис. 4.28 наведені результати порівняння кількості втрачених біт секретного повідомлення при накладанні на стеганоконтейнери, за формулою (4.1), перетворень зсуву.

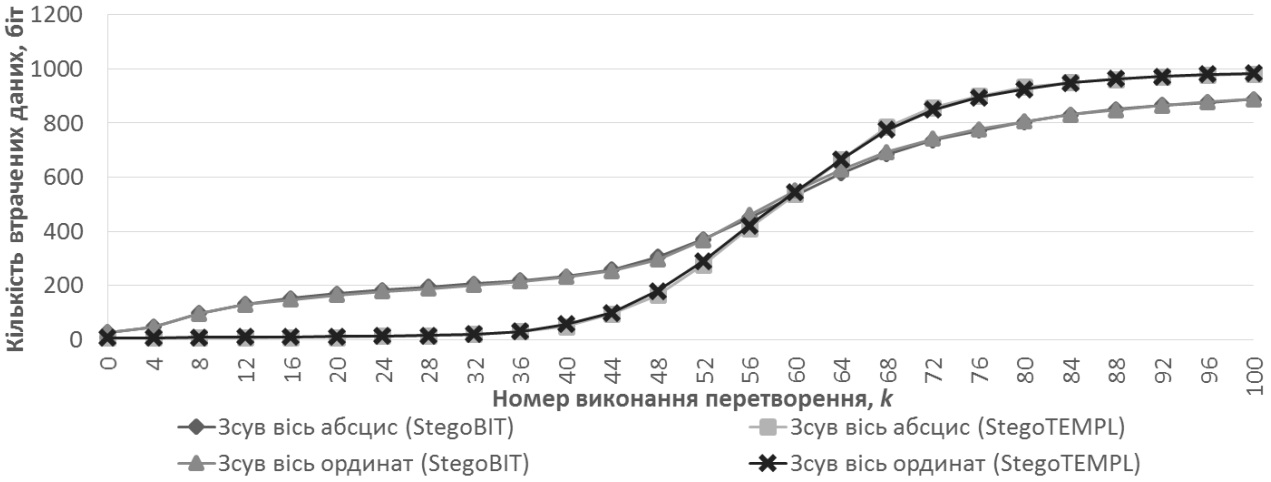


Рис. 4.28. Результати вилучення інформації з стеганоконтейнерів після виконання перетворень зсуву

***Згідно результатів експериментів* 25-30** визначено, що середня швидкість вилучення інформації алгоритмом StegoTEMPL краща на 72,5% ніж в StegoBIT, проте швидкість вбудовування даних гірша на 2,19%. За алгоритмом StegoTEMPL розміри стеганоконтейнерів в середньому менші на 23%. Середній коефіцієнт спотворення зображень після вбудовування інформації для обох алгоритмів менший за 0,5%. У алгоритмів StegoTEMPL

стійкість до перенесення, повороту і майже повороту, пропорційного і непропорційного масштабування краща ніж в StegoBIT.

Експерименти 31-36. Приховування інформації у SVG зображення здійснювалося за обраними стеганоключами та наступними параметрами: $V_1 = 3$, $V_2 \geq 1$, $V_3 = 80$, $V_4 = \{5,6\}$. Вилучення вбудованої інформації здійснювалося за параметром $V_5 = 4 \cdot 10^{-5}$.

При кожному вбудовуванні і вилученні прихованих даних з кожного стеганоконтейнера програмно визначався час виконання даних операцій. У табл. 4.17 наведені загальні середні результати затрати часу за кожним з алгоритмів та їх *швидкісні характеристики* (після виконання 100 ітерацій для кожного SVG зображення).

Таблиця 4.17

Середні швидкісні характеристики розроблених алгоритмів

Алгоритм приховування	Процесор	Розмір прихованої інформації, біт	Час приховування, с	Швидкість вбудовування біт/с	Час вилучення, с	Швидкість вилучення, біт/с
StegoBIT	1	1024	0,1942	5272,91	0,666	1537,54
	2		0,1284	7975,08	1,0969	933,54
	3		0,5158	1985,27	2,1198	483,06
StegoTEMPL	1		0,1489	6877,10	0,6069	1687,26
	2		0,2682	3818,05	0,6064	1688,65
	3		0,2796	3662,37	1,7443	587,05
1 – AMD A10-4600M, 2.3 GHz; 2 – AMD Phenom(tm) II X4 945 Processor, 3.00 GHz; 3 – Intel® Pentium® quad core processon N3540, up to 2.66 GHz.						

Результати порівнянь середнього значення *розміру всіх стеганоконтейнерів та їх стиснутих копій* (за допомогою Zip архіватора) *відносно контейнерів-оригіналів* наведені в табл. 4.18.

Таблиця 4.18

Порівняння середніх значень розмірів одержаних стеганоконтейнерів

Алгоритм приховування	Розмір прихованої інформації, біт	Розмір контейнера «до», КБайт	Розмір контейнера «після», КБайт	Розмір стиснутого контейнера «після», КБайт	Збільшення розміру контейнера «після» відносно контейнера «до», раз	Збільшення розміру стиснутого контейнера «після» відносно контейнера «до», раз
StegoBIT	1024	26,81	55,74	24,40	2,08	0,91
StegoTEMPL			43,06	19,05	1,61	0,71

Середнє значення порівняння візуального спотворення всіх стеганоконтейнерів відносно контейнера-оригінала наведені в табл. 4.19. Результати отримані при використанні ПЗ AntiDupl.NET.

Таблиця 4.19

Порівняння середніх значень коефіцієнту візуального спотворення

Алгоритм приховування	Коефіцієнт візуального спотворення контейнера «після» відносно контейнера «до», %
StegoBIT	0,09
StegoTEMPL	0,09

На рис. 4.29 наведені результати порівняння кількості втрачених біт секретного повідомлення при накладанні на стеганоконтейнери, за формулою (4.1), перетворень перенесення.

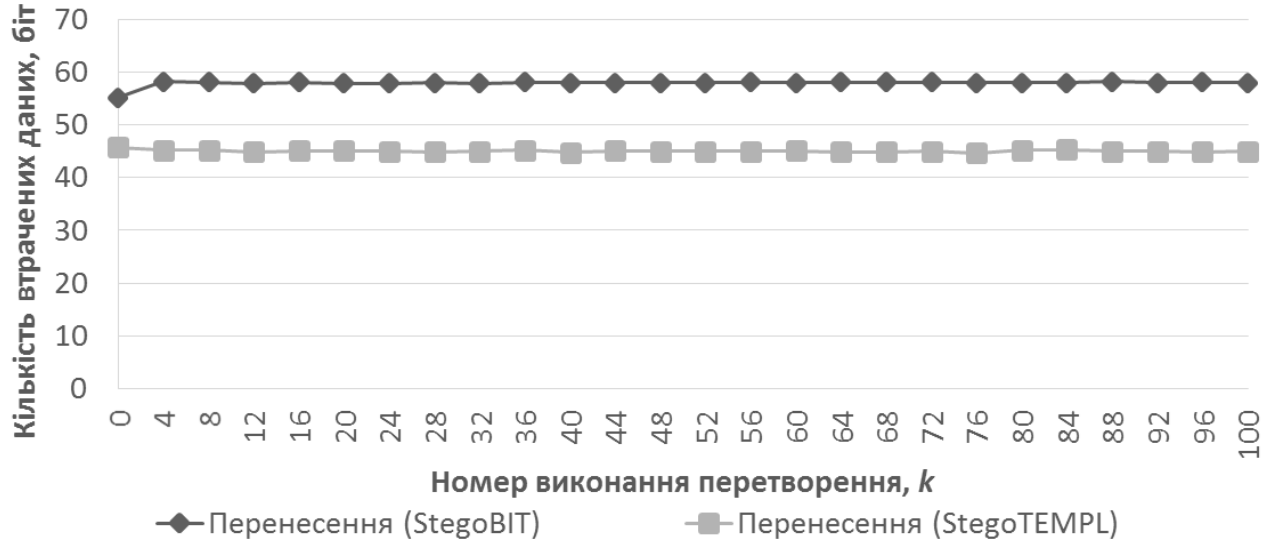


Рис. 4.29. Результати вилучення інформації з стеганоконтейнерів після виконання перетворень перенесення

На рис. 4.30 наведені результати порівняння кількості втрачених біт секретного повідомлення при накладанні на стеганоконтейнери, за формулою (4.1), перетворень повороту та майже повороту.

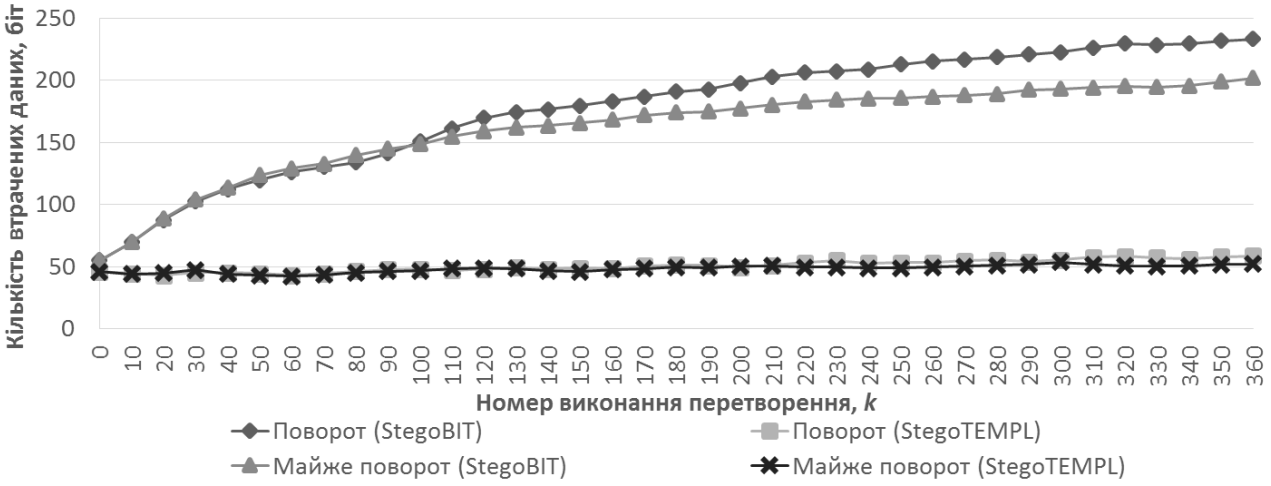


Рис. 4.30. Результати вилучення інформації з стеганоконтейнерів після виконання перетворень повороту та майже повороту

На рис. 4.31 та рис. 4.32 наведені результати порівняння кількості втрачених біт секретного повідомлення при поступовому стисненні/розширенні стеганоконтейнерів за перетвореннями пропорційного і непропорційного масштабування, відповідно формулі (4.1).

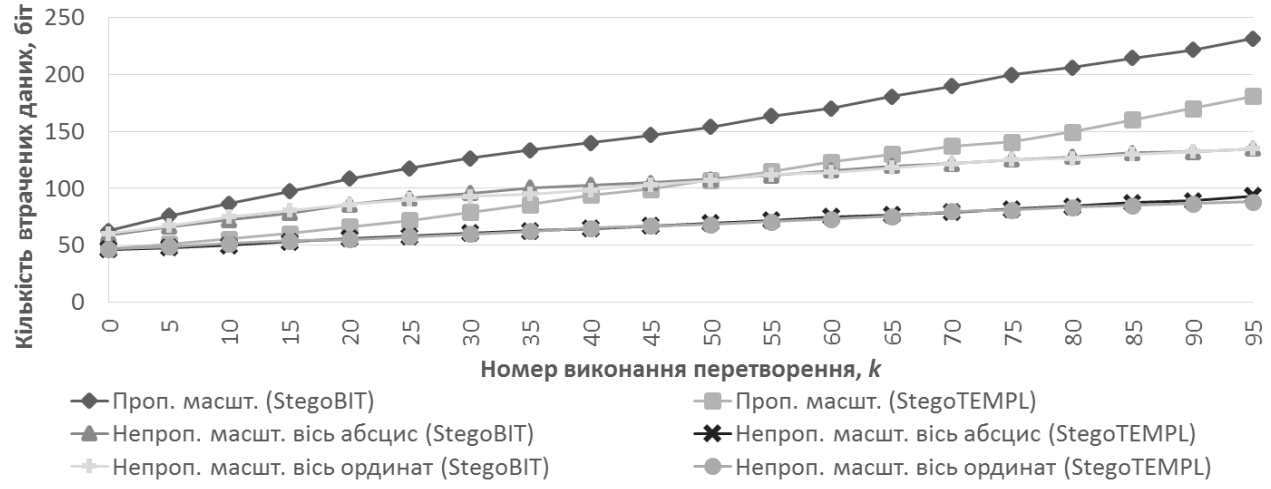


Рис. 4.31. Результати вилучення інформації з стиснутих стеганоконтейнерів після виконання перетворень пропорційного і непропорційного масштабування

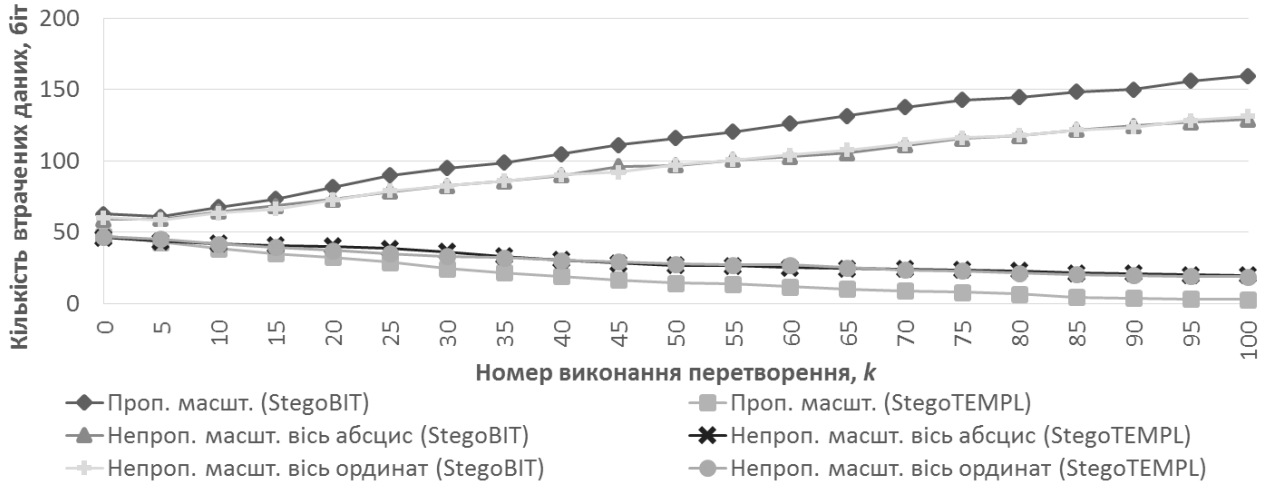


Рис. 4.32. Результати вилучення інформації з розтягнутих стеганоконтейнерів після виконання перетворень пропорційного і непропорційного масштабування

На рис. 4.33 наведені результати порівняння кількості втрачених біт секретного повідомлення при накладанні на стеганоконтейнери, за формулою (4.1), перетворень зсуву.

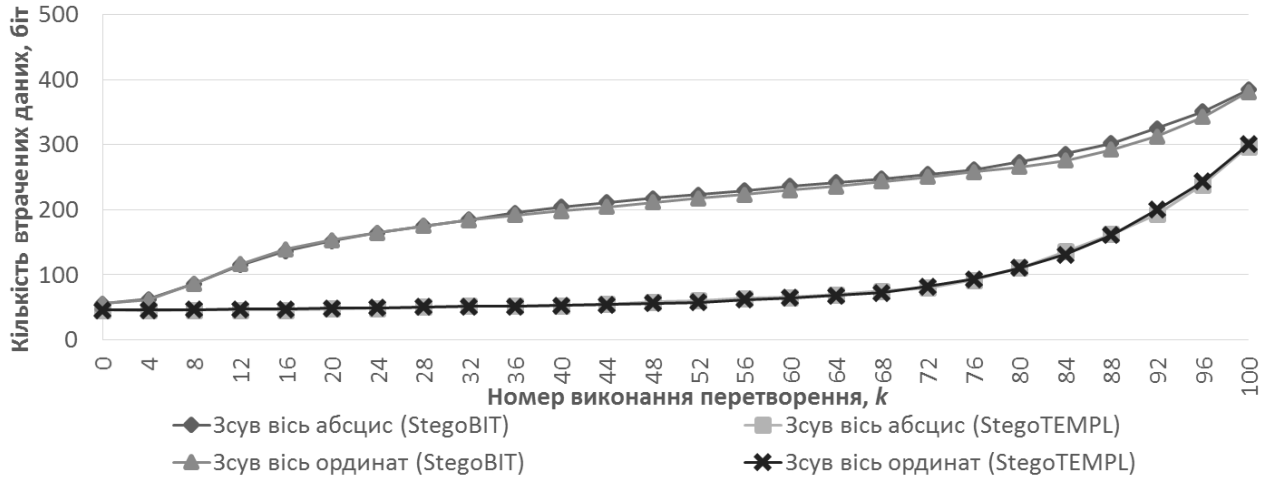


Рис. 4.33. Результати вилучення інформації з стеганоконтейнерів після виконання перетворень зсуву

Згідно результатів експериментів 31-36 визначено, що середня швидкість вилучення інформації алгоритмом StegoTEMPL краща на 34,15% ніж в StegoBIT, проте швидкість вбудовування даних гірша на 5,78%. За алгоритмом StegoTEMPL розміри стеганоконтейнерів в середньому менші на 23%. Середній коефіцієнт спотворення зображень після вбудовування інформації для обох алгоритмів менший 0,5%. У алгоритмів StegoTEMPL

стійкість до перенесення, повороту, майже повороту, зсуву, пропорційного і непропорційного масштабування краща ніж в StegoBIT.

Загальні середні результати експериментального дослідження

Середні результати швидкісних характеристик, коефіцієнт візуального спотворення, зміну розмірів стеганоконтейнерів розроблених алгоритмів по усім експериментам представленні в табл. 20-22.

Таблиця 4.20

Середні швидкісні характеристики розроблених алгоритмів

Алгоритм приховування	Процесор	Розмір прихованої інформації, біт	Час приховування, с	Швидкість вбудовування біт/с	Час вилучення, с	Швидкість вилучення, біт/с
StegoBIT	1	1024	0,1758	5824,80	1,4983	683,44
	2		0,1318	7769,35	1,3127	780,07
	3		0,3189	3211,04	1,3843	739,72
StegoTEMPL	1		0,1259	8133,44	0,8154	1255,83
	2		0,2899	3532,25	1,2149	842,87
	3		0,1781	5749,58	1,4035	729,60
1 – AMD A10-4600M, 2.3 GHz; 2 – AMD Phenom(tm) II X4 945 Processor, 3.00 GHz; 3 – Intel® Pentium® quad core processon N3540, up to 2.66 GHz.						

Таблиця 4.21

Порівняння середніх значень розмірів одержаних стеганоконтейнерів

Алгоритм приховування	Розмір прихованої інформації, біт	Розмір контейнера «до», КБайт	Розмір контейнера «після», КБайт	Розмір стиснутого контейнера «після», КБайт	Збільшення розміру контейнера «після» відносно контейнера «до», раз	Збільшення розміру стиснутого контейнера «після» відносно контейнера «до», раз
StegoBIT	1024	26,81	55,81	24,39	2,08	0,91
StegoTEMPL			43,13	19,02	1,61	0,71

Таблиця 4.22

Порівняння середніх значень коефіцієнту візуального спотворення

Алгоритм приховування	Коефіцієнт візуального спотворення контейнера «після» відносно контейнера «до», %
StegoBIT	0,11
StegoTEMPL	0,1

Середні результати стійкості алгоритмів StegoBIT та StegoTEMPL до перетворень перенесення, повороту, майже повороту по усім експериментам представлені на рис. 4.34-4.35.

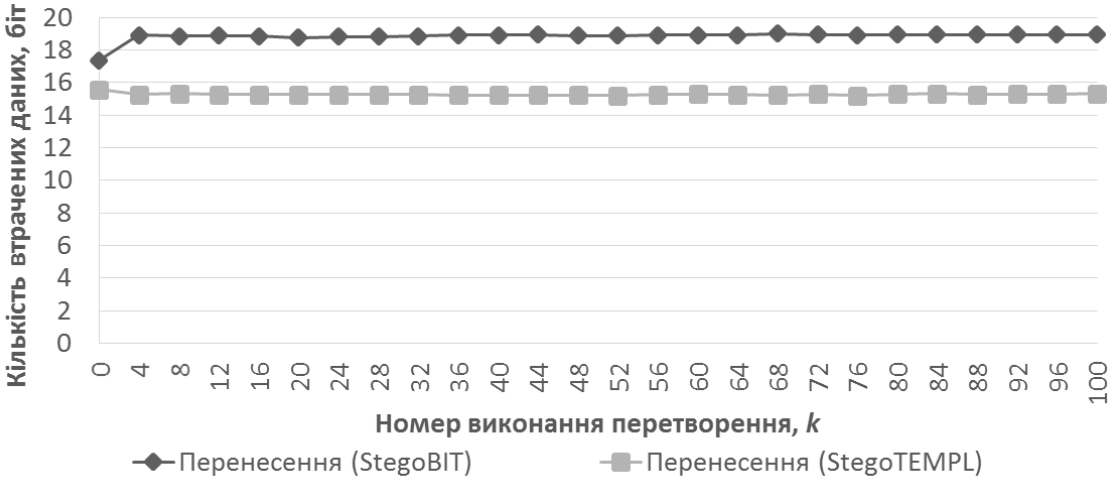


Рис. 4.34. Результати після виконання перетворень перенесення

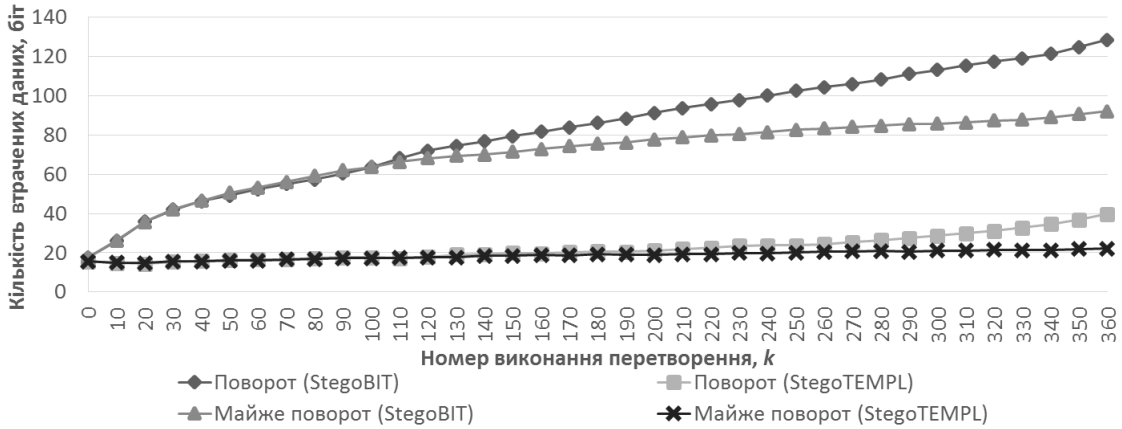


Рис. 4.35. Результати після виконання перетворень повороту та майже повороту

Середні результати стійкості алгоритмів StegoBIT та StegoTEMPL до перетворень пропорційного і непропорційного масштабування по усім експериментам представлені на рис. 4.36-4.37.

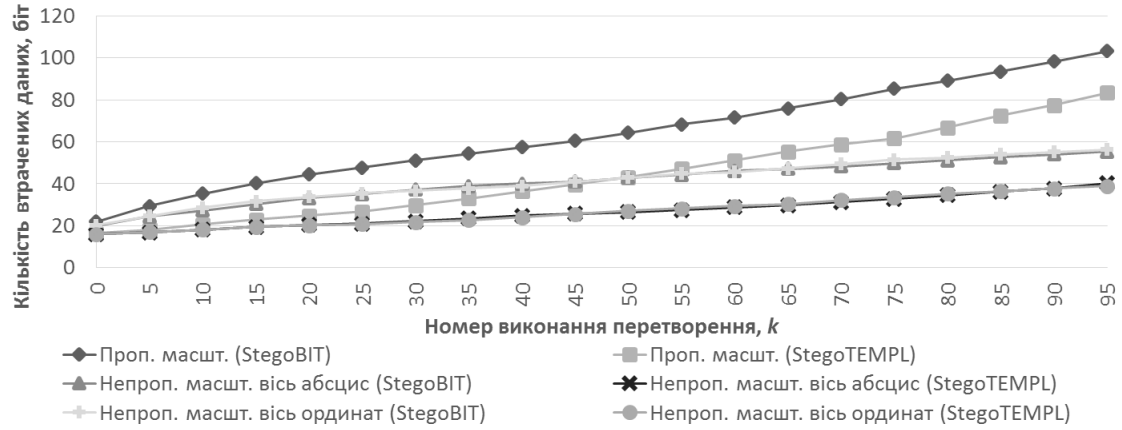


Рис. 4.36. Результати з стиснутих стеганоконтейнерів після виконання перетворень пропорційного і непропорційного масштабування

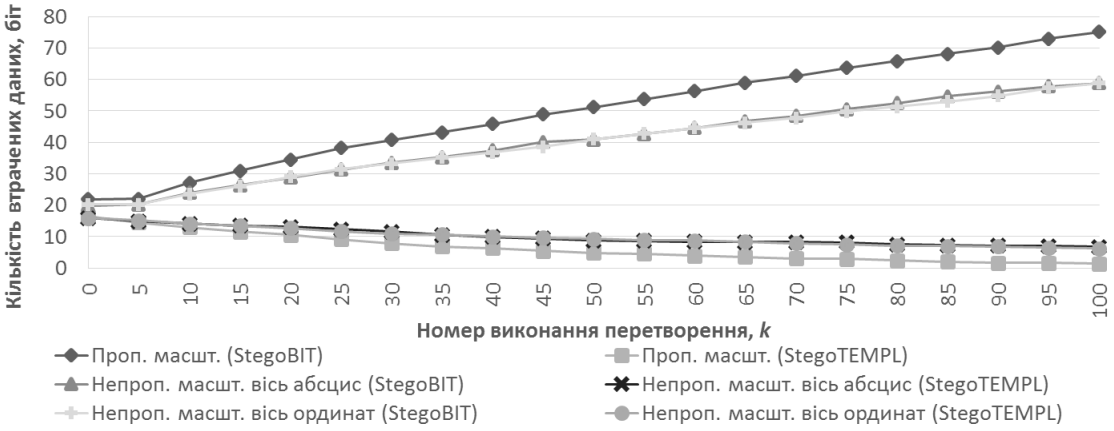


Рис. 4.37. Результати з розтягнутих стеганоконтейнерів після виконання перетворень пропорційного і непропорційного масштабування

Середні результати стійкості алгоритмів StegoBIT та StegoTEMPL до перетворення зсуву по усім експериментам представлені на рис. 4.38.

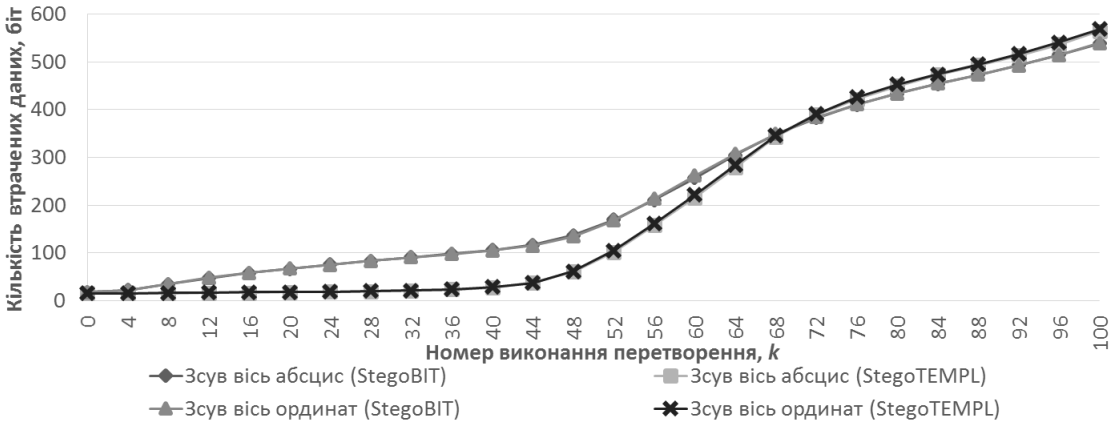


Рис. 4.38. Результати після виконання перетворень зсуву

Оцінки ефективності алгоритмів StegoBIT та StegoTEMPL

Ефективність алгоритмів будемо оцінювати за наступними показниками:

1. *Стійкістю до активних атак на основі афінних та майже афінних перетворень.* Згідно результатам вилучення прихованої інформації з стеганоконтейнерів після накладання афінних перетворень максимальний коефіцієнт втрат склав:

– за алгоритмом StegoBIT до атаки: перенесення – 1,85%; повороту – 12,54%; майже повороту – 8,98%; масштабування – 7,19%; зсуву – 52,61%;

– за алгоритмом StegoTEMPL до атаки: перенесення – 1,52%, повороту – 3,87%; майже повороту – 2,16%; масштабування – 5,29%; зсуву – 55,33%.

Причому, дані коефіцієнти отримані після багаторазового накладання ($k \geq 100$) даних афінних перетворень.

Найкращі результати стійкості алгоритмів StegoBIT та StegoTEMPL до усіх перетворень досягалися при використанні наступних параметрів: $V_1 = 3$, $V_2 \geq 1$, $V_3 = 40$, $V_4 = 6$, $V_5 = 4 \cdot 10^{-5}$.

2. *Розмірами стеганоконтейнера*. Згідно результатам вбудовування прихованого повідомлення розмірністю 2^{10} біт в контейнери призводило до збільшення їх розмірів у середньому:

– за алгоритмом StegoBIT у 2,08 рази, а стиснутого стеганоконтейнера у 0,91 рази;

– за алгоритмом StegoTEMPL у 1,61 рази, а стиснутого стеганоконтейнера у 0,71 рази.

3. *Коефіцієнтом загального спотворення*. Згідно результатам проведених експериментів середній коефіцієнт спотворення векторних зображень склав:

– за алгоритмом StegoBIT – 0,11%;

– за алгоритмом StegoTEMPL – 0,1%.

Порівняння стійкості алгоритмів StegoBIT та StegoTEMPL з методом Карпінцева-Яремчука

У роботах [15, 18, 19] наводяться результати дослідження метода Карпінцева-Яремчука щодо стійкості активним атакам на основі афінних перетворень. Оскільки, у відкритих джерелах це є єдина робота в якій автори проводять ряд експериментів з більш повною перевіркою стійкості до афінних перетворень їхнього векторного методу, то було вирішено порівняти його з алгоритмами StegoBIT та StegoTEMPL.

Порівняння стійкості до афінних перетворень відбувалося шляхом вбудовування інформації розмірністю 800 біт у довільно обране одне SVG зображення, при використанні наступних параметрів:

– для алгоритму StegoBIT: $V_1 = 3$, $V_2 \geq 1$, $V_3 = 40$, $V_4 = 5$, $V_5 = 4 \cdot 10^{-5}$, $\Delta t = \left\{2 \cdot 10^{-3}, 1 \cdot 10^{-3}, 5 \cdot 10^{-4}\right\}$;

– для алгоритму StegoTEMPL: $V_1 = 3$, $V_2 \geq 1$, $V_3 = 40$, $V_4 = 5$, $V_5 = 4 \cdot 10^{-5}$ та стеганоключів визначених в табл. 4.1;

– для методу Карпінцева-Яремчука (див. підрозділ 1.2): $P = \left\{3 \cdot 10^{-3}, 6 \cdot 10^{-3}, 7 \cdot 10^{-4}\right\}$, $P_h = \left\{8 \cdot 10^{-2}, 5 \cdot 10^{-2}, 3 \cdot 10^{-1}\right\}$, з встановленою точністю координат $V_4 = 5$.

Після чого, до одержаних стеганоконтейнерів поступово накладалися за формулою (4.1) та встановленими у даному розділі для перетворень перенесення, повороту, майже повороту, зсуву за віссю абсцис і ординат, пропорційного і непропорційного масштабування коефіцієнтами. Одержанні середні результати стійкості до перетворення перенесення, повороту і майже повороту при різних параметрах приховування наведені на рис. 4.39-4.41.

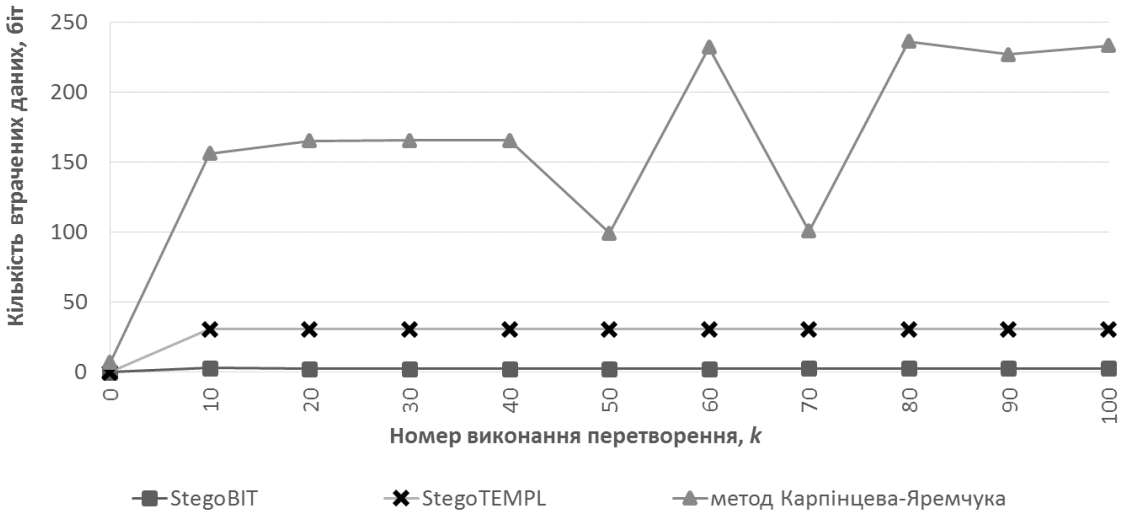


Рис. 4.39. Результати вилучення після виконання перетворень перенесення

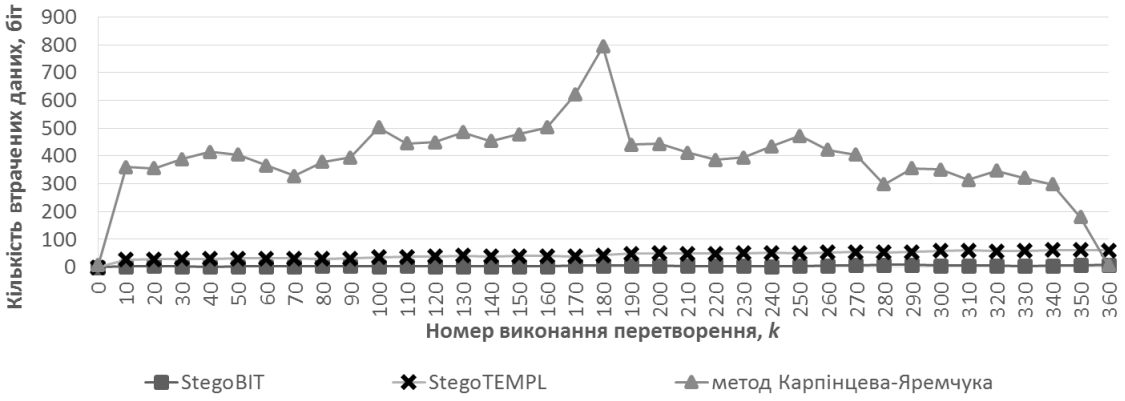


Рис. 4.40. Результати після виконання перетворень повороту

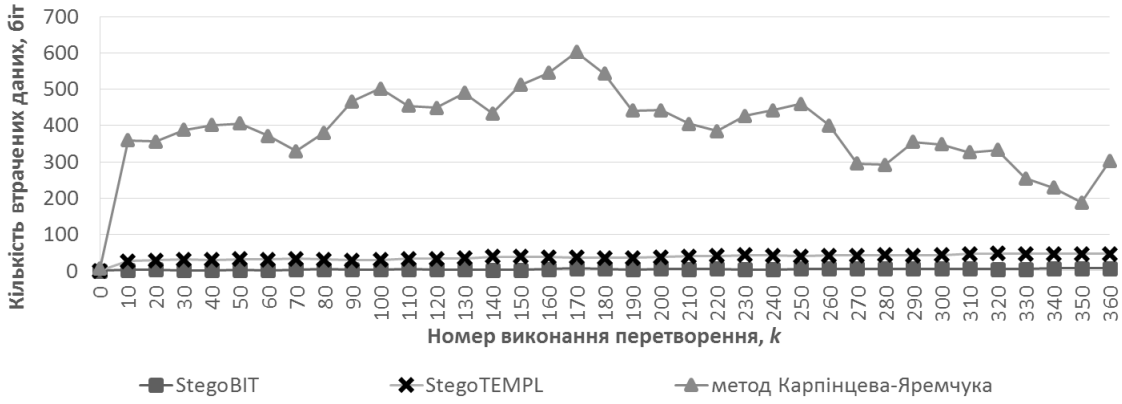


Рис. 4.41. Результати після виконання перетворень майже повороту

Одержанні середні результати стійкості до перетворення пропорційного масштабування при різних параметрах приховування наведені на рис. 4.42 та рис. 4.43.

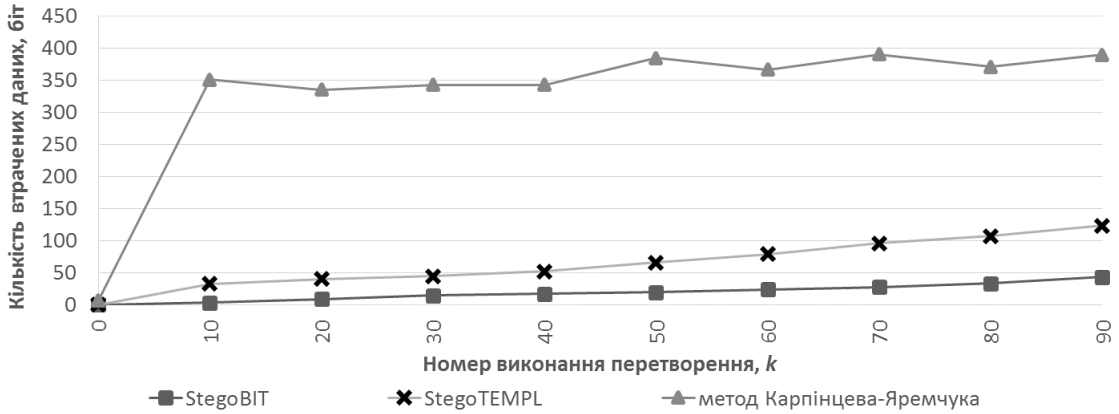


Рис. 4.42. Результати з стиснутих стеганоконтейнерів після виконання перетворень пропорційного масштабування

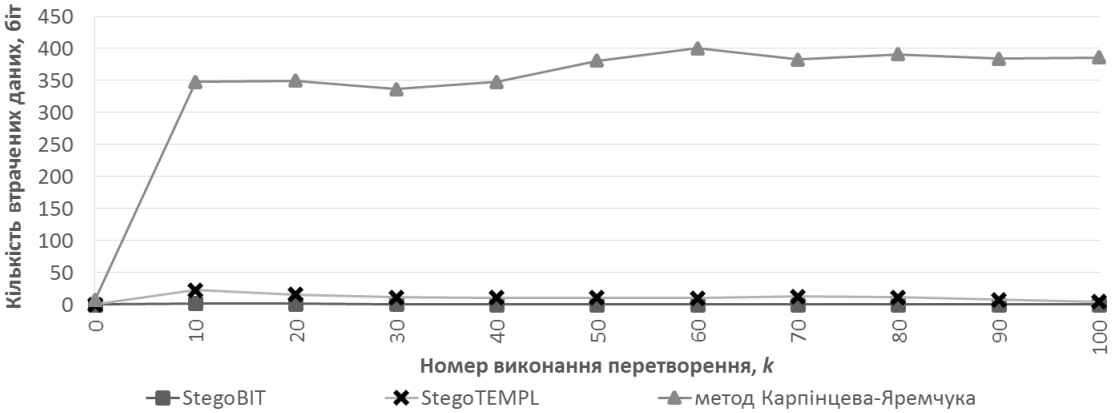


Рис. 4.43. Результати з розширених стеганоконтейнерів після виконання перетворень пропорційного масштабування

Одержанні середні результати стійкості до перетворення непропорційного масштабування за віссю абсцис при різних параметрах приховування наведені на рис. 4.44 та рис. 4.45.

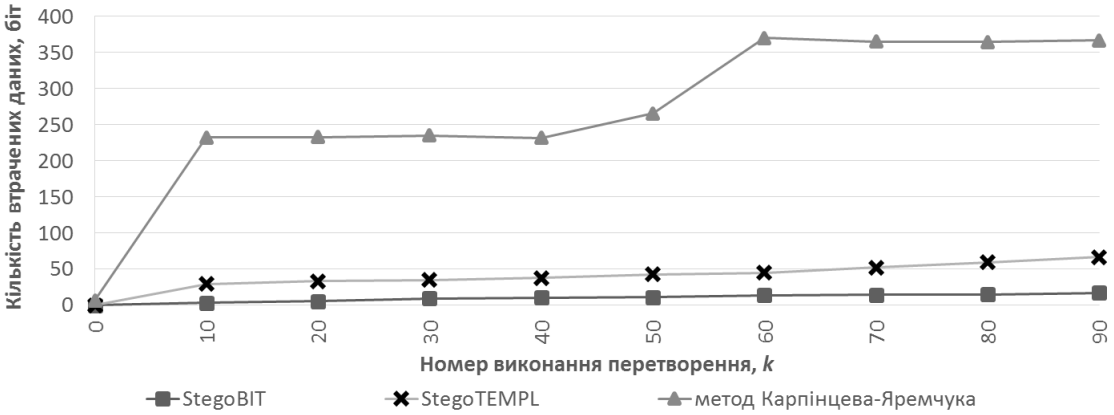


Рис. 4.44. Результати з стиснутих стеганоконтейнерів після виконання перетворень непропорційного масштабування за віссю абсцис

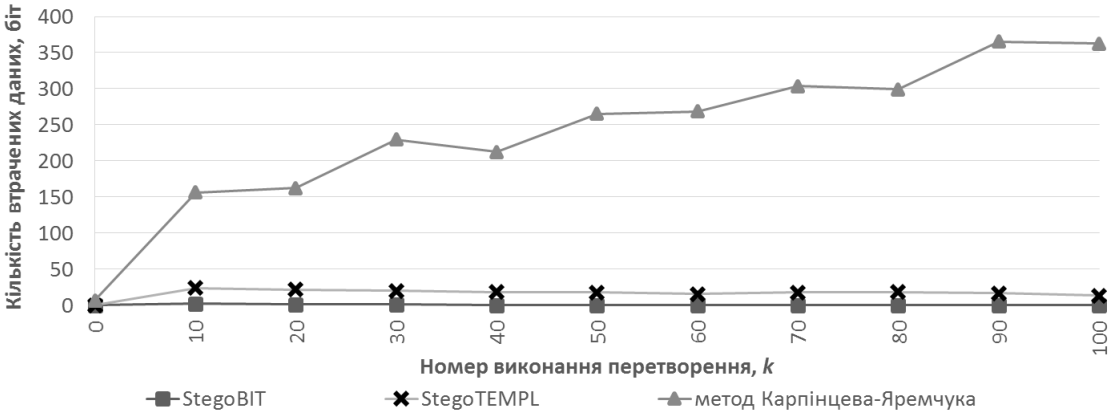


Рис. 4.45. Результати з розширених стеганоконтейнерів після виконання перетворень непропорційного масштабування за віссю абсцис

Одержанні середні результати стійкості до перетворення непропорційного масштабування за віссю ординат при різних параметрах приховування наведені на рис. 4.46 та рис. 4.47.

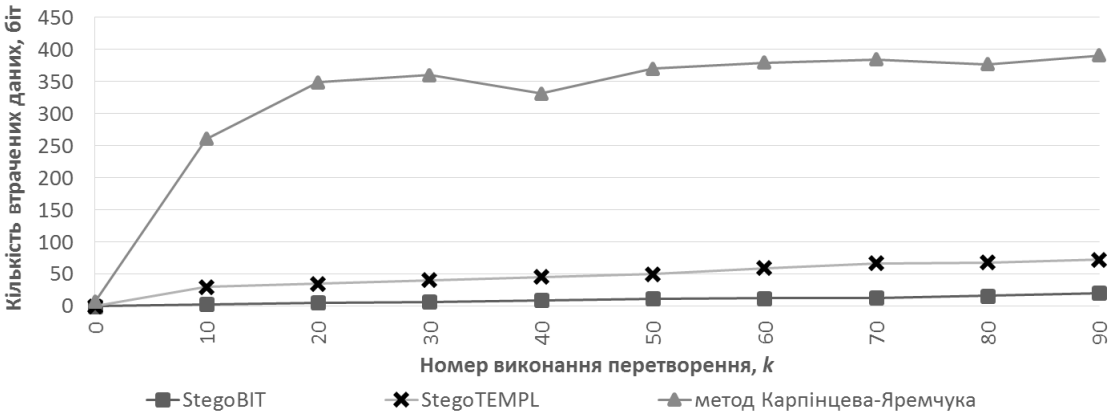


Рис. 4.46. Результати з стиснутих стеганоконтейнерів після виконання перетворень непропорційного масштабування за віссю ординат

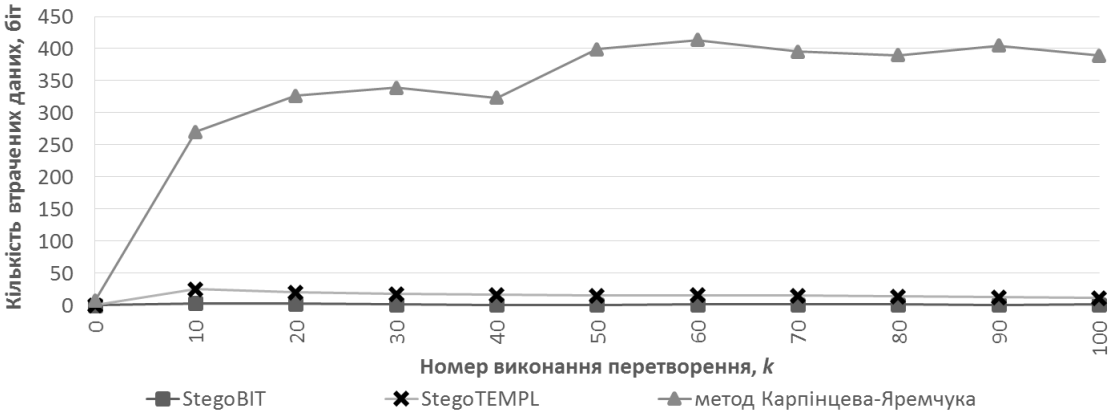


Рис. 4.47. Результати з розширених стеганоконтейнерів після виконання перетворень непропорційного масштабування за віссю ординат

Одержанні середні результати стійкості до перетворення зсуву за віссю абсцис та ординат при різних параметрах приховування наведені на рис. 4.48 та рис. 4.49.

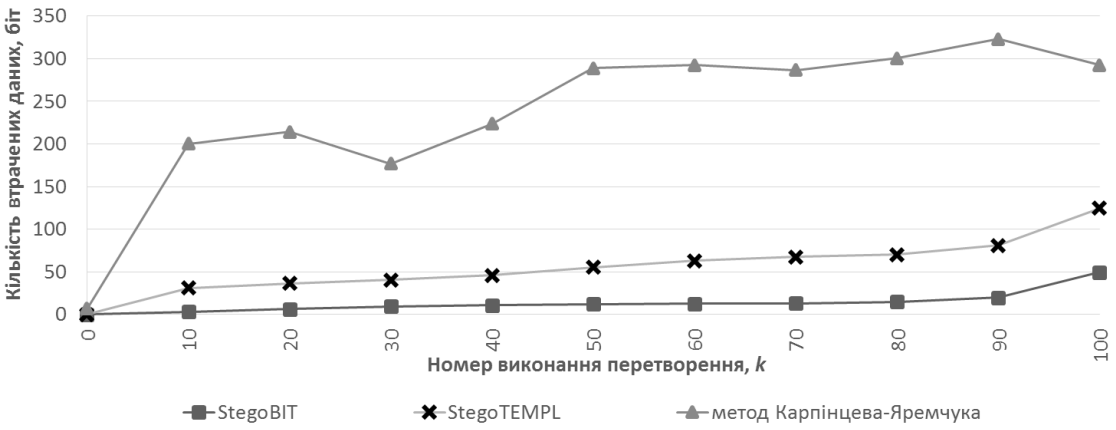


Рис. 4.48. Результати з стеганоконтейнерів після виконання перетворень зсуву за віссю абсцис

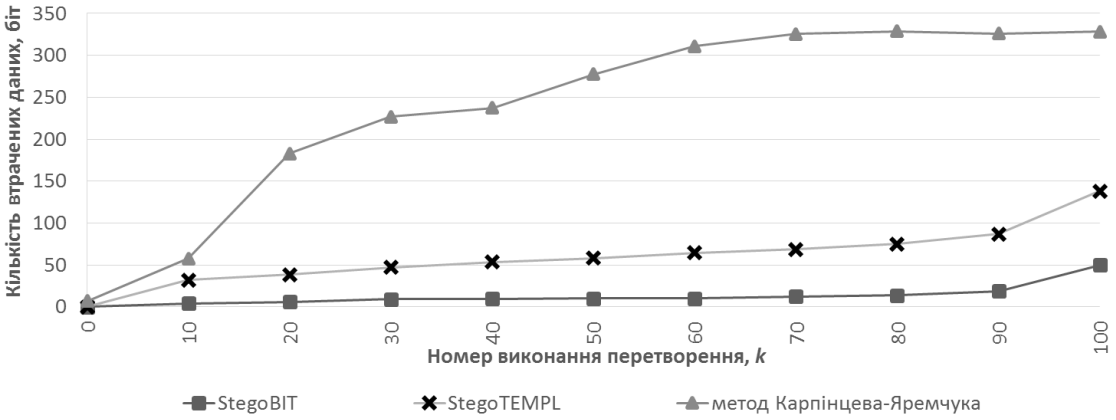


Рис. 4.49. Результати з стеганоконтейнерів після виконання перетворень зсуву за віссю ординат

Згідно результатам вилучення прихованої інформації з стеганоконтейнерів після накладання афінних перетворень середній коефіцієнт втрат склав:

– за алгоритмом StegoBIT до атаки: перенесення 0,3%, повороту 0,53%, майже повороту 0,55%, масштабування – 0,87%; зсуву – 1,69%;

– за алгоритмом StegoTEMPL до атаки: перенесення 3,47%, повороту 5,41%, майже повороту 4,64%, масштабування – 4,04%; зсуву – 7,27%;

– за методом Карпінчева-Яремчука до атаки: перенесення 20,3%, повороту 48,68%, майже повороту 48,38%, масштабування – 38,04%; зсуву – 29,62%.

Згідно результатам порівняння запропоновані алгоритми StegoBIT та StegoTEMP виявились стійкіші у порівняні з методом Карпінцева-Яремчука до атак на основі афінних перетворень. Середній коефіцієнт втрат інформації при застосуванні алгоритмів StegoBIT та StegoTEMPL зменшився: до атаки перенесення на 16,83%, до атаки повороту на 43,27%, до майже повороту на 43,74%, до атаки зсуву на 22,35% та до атаки масштабування на 34%.

Згідно отриманих результатів запропоновані наступні ***практичні рекомендації*** застосування алгоритмів StegoBIT та StegoTEMPL:

1. Розміри повідомлень, що приховуються запропонованими алгоритмами в один контейнер, не повинна перевищувати 2^{10} біт.

2. Розміри одержаних стеганоконтейнерів не повинні бути більшими за розміри середньостатистичного векторного зображення.

3. Параметр V_1 впливає на вибір кривих Без'є певного ступеня. Чим більша ступінь, тим меншою буде швидкість вбудовування інформації через збільшення кількості операцій необхідних для розбиття кривих на сегменти. Згідно результатам експерименту рекомендується приховувати інформацію в криві ступеня $V_1 = 3.$

4. Параметр V_2 впливає на вибір кривих Без'є у контейнері, чим даний параметр буде більший тим більше інформації можна приховати в одну криву. Згідно результатам проведених експериментів рекомендується використовувати довільні криві з відстанями між опорними точками $V_2 \geq 1.$

5. Чим більший параметр V_3 тим більші інформації можна приховати в одну криву, однак зменшуватиметься стійкість алгоритмів до афінних перетворень. Рекомендується вибирати параметр $V_3 \leq 40$ біт.

6. Від параметру V_4 залежить точність вилучення даних, чим більше V_4 тим точніше буде вилучення, але розміри стеганоконтейнерів будуть більшими. Як показали дослідження при $V_4 = \{5, 6\}$ досягається оптимальне відношення (розміри стеганоконтейнерів)/(стійкість до афінних перетворень).

7. Від параметру V_5 залежить правильність вилучення даних з сукупностей сегментів. Найбільш ефективніше вилучення інформації при застосуванні до стеганоконтейнерів різноманітних афінних перетворень досягалося при використанні значення $V_5 = 4 \cdot 10^{-5}$.

8. Для алгоритму StegoTEMPL рекомендовано використовувати таблицю шаблонів з кількістю значень елементів шаблону $k = 16$ та кількістю біт одного значення елементу шаблону $l = 4$.

4.4. Висновки до четвертого розділу

1. Запропоновано методику проведення експерименту, яка дозволяє дослідити швидкісні характеристики та стійкість до афінних перетворень запропонованих алгоритмів, коефіцієнт візуального спотворення та розміри стеганоконтейнерів.

2. Досліджено швидкісні характеристики запропонованих алгоритмів у криві Без'є. Середня швидкість вбудовування інформації за алгоритмом StegoBIT склала – 5601,73 біт/с, а за StegoTEMPL – 5805,09 біт/с. Середня швидкість вилучення інформації за алгоритмом StegoBIT склав – 734,41 біт/с, а за StegoTEMPL – 942,77 біт/с.

3. Досліджено коефіцієнт візуального спотворення, для обох запропонованих алгоритмів виявився не більшим ніж 0,5%.

4. При вбудовуванні повідомлень середній розмір стеганоконтейнерів збільшувався за алгоритмом StegoBIT у 2,08 рази, а за StegoTEMPL – у 1,61 рази.

5. Усі запропоновані алгоритми вбудовування інформації у криві Без'є показали гарну стійкість до усіх видів афінних перетворень порівняно з методом Карпінцева-Яремчук. Середній коефіцієнт втрат інформації при застосуванні алгоритмів StegoBIT та StegoTEMPL зменшився: до атаки перенесення на 16,83%, до атаки повороту на 43,27%, до майже повороту на 43,74%, до атаки зсуву на 22,35% та до атаки масштабування на 34%.

Максимальний коефіцієнт втрат при різних обсягах приховуваної інформації та багатократному застосуванні афінних перетворень до стеганоконтейнерів склав: за алгоритмом StegoBIT до атаки перенесення 1,85%, повороту 12,54%, майже повороту 8,98%, масштабування 7,19% та зсуву 52,61%; за алгоритмом StegoTEMPL до атаки перенесення 1,52%, повороту 3,87%, майже повороту 2,16%, масштабування 5,29%, а зсуву 55,33%.

6. Розроблено практичні рекомендації, щодо застосування алгоритмів StegoBIT та StegoTEMPL.

ВИСНОВКИ

У дисертаційній роботі розв’язано актуальне наукове завдання щодо розробки нових методів приховування даних у векторні зображення для підвищення стійкості стеганографічного захисту інформації до активних атак на основі афінних перетворень.

В ході розв’язання поставлених задач були отримані наступні наукові та практичні результати:

1. Проведено аналіз сучасних методів приховування інформації, що використовують у якості контейнера векторні зображення. Проведено порівняння стійкості даних методів до афінних перетворень, яке показало недостатню стійкість розглядуваних стеганографічних методів відносно даних атак. З огляду на це, постає необхідність розробки нових методів приховування інформації у векторні зображення, стійких до активних атак на основі афінних перетворень.

2. Формалізовано вимоги до вибору контейнера, визначено множину параметрів приховування інформації у векторні зображення, які, за рахунок врахування особливостей побудови векторних зображень та стеганографічних перетворень, дозволяють впливати на вибір допустимих контейнерів та на процес приховування даних у точково-задані криві.

3. Запропоновано метод побітового приховування інформації у точково-задані криві векторних зображень, який дозволяє приховувати дані шляхом поділу кривих на сукупності сегментів, не призводячи при цьому до погіршення якості самого зображення з забезпеченням стійкості до активних атак на основі афінних перетворень.

4. Запропоновано метод шаблонного приховування інформації у точково-задані криві векторних зображень з наперед визначеною таблицею співвідношень різних значень елементів шаблону різним крокам побудови точково-заданих кривих, який дозволяє за один поділ кривої на сегменти

здійснювати приховування цілого блоку біт з забезпеченням стійкості до активних атак на основі афінних перетворень.

5. Запропоновано структурну модель процесу прихованої передачі інформації резервним каналом зв’язку, що дозволяє формувати множину стеганоконтейнерів, стійких до афінних перетворень.

6. Розроблено алгоритми StegoBIT та StegoTEMPL, які підвищуються стійкість до атак на основі афінних перетворень. У порівняні з методом Карпінцева-Яремчука середній коефіцієнт втрат інформації зменшився: до атаки перенесення на 16,83%, до атаки повороту на 43,27%, до майже повороту на 43,74%, до атаки зсуву на 22,35% та до атаки масштабування на 34%. Максимальний коефіцієнт втрат при багаторазовому застосуванні афінних перетворень до стеганоконтейнерів склав: за алгоритмом StegoBIT до атаки перенесення 1,85%, повороту 12,54%, майже повороту 8,98%, масштабування 7,19% та зсуву 52,61%; за алгоритмом StegoTEMPL до атаки перенесення 1,52%, повороту 3,87%, майже повороту 2,16%, масштабування 5,29% та зсуву 55,33%.

7. Розроблено методику проведення експериментального дослідження, яка дозволяє дослідити розроблені. На основі запропонованої методики та алгоритмів розроблено програмні засоби, що дозволило верифікувати отримані результати. Упровадження зазначених розробок та їх експериментальне дослідження підтвердило достовірність теоретичних гіпотез і висновків дисертаційної роботи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Аграновский А.В. Основы компьютерной стеганографии / А.В. Аграновский, П.Н. Девянин, Р.А. Хади. — М. : Радио и связь, 2003. — 151 с.

2. Ажбаев Т.Г. Анализ стойкости современных стеганографических алгоритмов / Т.Г. Ажбаев, И.М. Ажмухамедов // Вестник Астраханского государственного технического университета. — 2008. — С. 56-61.

3. Бахрушина Г.И. Моделирование геометрических атак на основе аффинных преобразований / Г.И. Бахрушина // электронное научное издание «Ученые заметки ТОГУ». — 2013. — Т.4, №4. — С. 1291-1297.

4. Блоковий симетричний криптоалгоритм «LUNA» / В.М. Кінзерявий, В.П. Квасніков, С.О. Гнатюк, О.М. Кінзерявий // науково-практичний журнал «Захист інформації». — 2011. — Т.13, №3. — С. 77-86.

5. Васильков Д.М. Геометрическое моделирование и компьютерная графика. Вычислительные и алгоритмические основы : [курс лекций] / Д.М. Васильков. — Минск : БГУ, 2011. — 203 с.

6. Глибовець А.М. Сучасні технології та програмне забезпечення мобільного зв'язку / А.М. Глибовець // Наукові праці [Чорноморського державного університету імені Петра Могили]. Сер. : Комп'ютерні технології. — 2008. — Т.90, Вип. 77. — С. 223-234.

7. Голованов Н.Н. Геометрическое моделирование / Н.Н. Голованов. — М. : Издательство Физико-математической литературы, 2002. — 472 с.

8. Грабченко А.І. Теорія 3D моделювання : [навчальний посібник] / А.І. Грабченко, В.Л. Доброскок. — Х. : НТУ «ХПІ», 2009. — 230 с.

9. Грибунин В.Г. Цифровая стеганография / В.Г. Грибунин, И.Н. Оков, И.В. Туринцев. — М. : СОЛОН-Пресс, 2002. — 272 с.

10. Дунаев В.В. HTML, скрипты и стили : [3-е издание] / В.В. Дунаев. — СПб. : БХВ-Петербург, 2011. — 816 с.

11. Задирака В.К. Анализ стойкости криптографических и стеганографических систем на основе общей теории оптимальных алгоритмов / В.К. Задирака, А.М. Кудин // Journal of Qafgaz University. — 2010. — №30. — С. 49–58.

12. Золотавкін Є.А. Методи та засоби підвищення стеганографічної стійкості захисту інформації до пасивних атак : дис. канд. техн. наук : 05.13.21 / Золотавкін Євген Анатолійович — В., 2010. — 194 с.

13. Казиев В. Введение в моделирование объектов, процессов и явлений [Електронний ресурс] / В. Казиев. — 2006. — Режим доступу до ресурсу: http://www.intuit.ru/studies/courses/108/108/lecture/3161.

14. Карпінець В.В. Аналіз стійкості до зловмисних атак методу вбудовування цифрових водяних знаків у векторні зображення / В.В. Карпінець, Ю.Є. Яремчук // Вісн. Вінниц. політехн. інституту — 2011. — № 4. — С. 154-159.

15. Карпінець В.В. Вирішення проблеми захисту авторських прав векторних зображень в інформаційно-комунікаційних системах / В.В. Карпінець, Ю.Є. Яремчук, Д.О. Іванішина // Науково-технічний журнал «Захист інформації». — 2011. — №4 (53) — С. 21–28.

16. Карпінець В.В. Забезпечення захисту векторних зображень від атак спрямованих на видалення цифрових водяних знаків / В.В. Карпінець, Ю.Є. Яремчук // // Вісник Східноукраїнського національного університету імені Володимира Даля. — 2013. — № 15 (1). — С. 62-68.

17. Карпінець В.В. Зменшення відхилень координат точок внаслідок вбудовування цифрових водяних знаків у векторні зображення / В.В. Карпінець, Ю.Є. Яремчук // Прав., нормат. та метрол. забезп. системи захисту інформації в Україні. — 2010. — Вип. 2. — С. 69-78.

18. Карпінець В.В. Методи захисту векторних зображень цифровими водяними знаками : [монографія] // В.В. Карпінець, Ю.Є. Яремчук. — 2013. — 156 с.

19. Карпінець В.В. Оцінювання впливу цифрових водяних знаків на якість векторних зображень та забезпечення стійкості до зловмисних атак / В.В. Карпінець, Ю.Є. Яремчук, К.В. Безпалий // Сучасна спеціальна техніка. — 2013. — №2. — С. 44-51.

20. Кінзерявий О.М. Дослідження стійкості методу шаблонного приховування інформації у векторні зображення / В.Ю. Ковтун, О.М. Кінзерявий // Безпека інформації у інформаційно-телекомунікаційних системах : сімнадцята міжнародна науково-практична конференція. — К. : Державна служба спеціального зв'язку та захисту інформації України, 2015. — С. 54-55.

21. Кінзерявий О.М. Експериментальне дослідження методу побітового приховування даних у векторні зображення / О.М. Кінзерявий, В.Ю. Ковтун // науковий журнал «Безпека інформації». — 2014. — Т.20, №1. — С. 66-70.

22. Кінзерявий О.М. Експериментальне дослідження стійкості методу побітового приховування даних відносно атак на основі афінних перетворень / О.М. Кінзерявий, В.Ю. Ковтун // науково-практичний журнал «Захист інформації». — 2014. — Т.16, №4. — С. 304-311.

23. Кінзерявий О.М. Експериментальне дослідження методу шаблонного приховування даних в структуру векторних зображень / О.М. Кінзерявий, В.М. Кінзерявий // Авіа-2015 : дванадцята міжнародна науково-технічна конференція. — К., 2015. — С. 2.1-2.4.

24. Кінзерявий О.М. Експериментальне дослідження стійкості методу побітового приховування даних у векторні зображення відносно афінних перетворень / О.М. Кінзерявий, В.М. Кінзерявий // Актуальні питання забезпечення кібернетичної безпеки та захисту інформації. — К. : Видавництво Європейського університету, 2015. — С. 48-52.

25. Кінзерявий О.М. Метод шаблонного приховування даних у векторні зображення / О.М. Кінзерявий, В.М. Кінзерявий // Інфокомунікації - сучасність та майбутнє : четверта міжнародна науково-практична конференція. — О. : ОНАЗ, 2014. — Ч.4. — С. 48-52.

26. Кінзерявий О.М. Метод шаблонного приховування даних у векторні зображення / О.М. Кінзерявий, В.Ю. Ковтун, О.Л. Стокіпний // науково-практичний журнал «Захист інформації». — 2014. — Т.16, №2. — С. 139-146.

27. Кінзерявий О.М. Нові ефективні алгоритми шифрування інформації / В.М. Кінзерявий, С.О. Гнатюк, О.М. Кінзерявий // науково-практичний журнал «Захист інформації». — 2012. — Т.14, №4. — С. 132-142.

28. Кінзерявий О.М. Побітове приховування даних у SVG зображення на основі кривих Без'є / О.М. Кінзерявий // Безпека інформаційних технологій (ITSEC) : четверта міжнародна науково-технічна конференція. — К. : НАУ, 2014. — С. 53-54.

29. Кінзерявий О.М. Побітовий метод приховування у векторні зображення стійкий до афінних перетворень / О.М. Кінзерявий, Б.С. Дорошенко // Механізми управління безпекою підприємств в сучасних умовах господарювання. — К. : Видавництво Європейського університету, 2013. — С. 91-94.

30. Кінзерявий О.М. Систематизація сучасних методів комп'ютерної стеганографії / О.М. Кінзерявий, В.Ю. Ковтун, С.О. Гнатюк // науковий журнал «Безпека інформації». — 2013. — №3. — С. 209-217.

31. Кінзерявий О.М. Стеганографічний метод приховування даних у векторних зображеннях / О.М. Кінзерявий // Інфокомунікації - сучасність та майбутнє : третя міжнародна науково-практична конференція молодих вчених — О. : ОНАЗ, 2013. — Ч.3. — С. 160-162.

32. Кінзерявий О.М. Шаблонний метод приховування даних у векторні зображення на основі розбиття кривих Без'є / О.М. Кінзерявий // Захист інформації і безпека інформаційних систем : третя міжнародна науково-технічна конференція. — Л. : Українська академія друк., 2014. — С. 86-87.

33. Кларк Д. Кодирование с исправлением ошибок в системах цифровой связи / Д. Кларк, Д. Кейн. — М. : Радио и связь, 1987. — 392 с.

34. Кобозева А.А. Связь свойств стеганографического алгоритма и используемой им области контейнера для погружения секретной информации / А.А. Кобозева // Искусств. интеллект. — 2007. — № 4. — С. 531-538.

35. Комп'ютерне моделювання систем та процесів. Методи обчислень : [навчальний посібник] / [Р.Н. Квєтний, І.В. Богач, О.Р. Бойко та ін.] // за заг. ред. Р.Н. Квєтного. — В. : ВНТУ, 2012. — 193 с.

36. Конахович Г.Ф. Компьютерная стеганография. Теория и практика / Г.Ф. Конахович, А.Ю. Пузыренко. — К. : МК-Пресс, 2006. — 288 с.

37. Копитова О.М. Курс лекцій з комп'ютерної графіки : [для студентів всіх форм навчання] / О.М. Копитова. — Д. : ДонНТУ, 2011. — 81 с.

38. Кузнецов О.О. Стеганография : [навчальний посібник] / О.О. Кузнецов, С.П. Євсеєв, О.Г. Король. — Х. : Вид. ХНЕУ, 2011. — 232 с.

39. Кунву Ли Основы САПР (CAO/CAM/CAE) / Ли Кунву. — СПб. : Питер, 2004. — 560 с.

40. Липкин И.А. Статистическая радиотехника. Теория информации и кодирования / И.А. Липкин. — М. : Вузовская книга, 2002. — 232 с.

41. Лукічов В.В. Методи та засоби стеганографічного захисту інформації в комп'ютерних системах і мережах на основі вейвлет-перетворень : дис. канд. техн. наук : 05.13.21 / Лукічов Віталій Володимирович — В., 2010. — 204 с.

42. Маракова И.И. Проблематика и перспективы развития методов сокрытия информации / И.И. Маракова, А.С. Сафронов // Пр. Одес. політехн. ун-ту. — 2003. — Вип. 1. — С. 184-188.

43. Маценко В.Г. Комп'ютерна графіка : [навчальний посібник] / В.Г. Маценко. — Ч. : Рута, 2009. — 343 с.

44. Неуймин Я.Г. Модели в науке и технике. История, теория, практика / Я.Г. Неуймин. — Л. : Наука, Ленинград. отд., 1984. — 190 с.

45. Огляд стеганографічних методів перетворення інформації в зображеннях / І.В. Бабич, С.А. Паламарчук, Н.А. Паламарчук, В.В. Овсянніков // Захист інформації. — 2012. — № 1. — С. 18-24.

46. Основи комп'ютерної стеганографії : [навч. посіб. для студентів і аспірантів] / В.О. Хорошко, О.Д. Азаров, М.Є. Шелест, Ю.Є. Яремчук. — В. : ВДТУ, 2003. — 143 с.

47. Пелещишин А.М. Загрози інформаційної безпеки держави в соціальних мережах / А.М. Пелещишин, Р.В. Гумінський // Наука і техніка Повітряних Сил Збройних Сил України. — 2013. — № 2. — С. 192-199.

48. Письменный перевод. Рекомендации переводчику, заказчику и редактору / редактор Н.К. Дупленский. — М. : Р. Валент, 2013. — 164 с.

49. Понарин Я.П. Аффинная и проективная геометрия / Я.П. Понарин. — М. : МЦНМО, 2009. — 288 с.

50. Потресов С. GPRS и EGPRS (EDGE) : Мифы и реальность в изложении «из первых рук» [Електронний ресурс] / С. Потресов. — 2006. — Режим доступу до ресурсу: http://www.mobile-review.com/articles/2005/erics-qa.shtml.

51. Про дипломатичну службу : Закон України від 20.09.2001 р. №2728-III [Електронний ресурс]. — Відомості Верховної Ради України, 2002. — №5. — 29 с. — Режим доступу до ресурсу : http://zakon2.rada.gov.ua/laws/show/2728-14/print1415536861200877.

52. Про затвердження Інструкції про порядок забезпечення комплексної безпеки дипломатичних установ України за кордоном : наказ МЗС України від 11.07.1995 р. № 1041 [Електронний ресурс]. — Режим доступу до ресурсу : http://zakon2.rada.gov.ua/laws/show/z0338-95/parao12#o12.

53. Про затвердження Переліку відомостей, які містять службову інформацію в системі органів дипломатичної служби України : наказ МЗС України від 19.08.2013 р. № 207 [Електронний ресурс]. — Режим доступу до ресурсу : http://mfa.gov.ua/ua/legal-acts/2092-nakaz-ministerstva-zakordonnih-sprav-ukrajini-vid-19082013--207-pro-zatverdzhennya-pereliku-vidomostej-jaki-mistyaty-sluzhbovu-informaciju-v-sistemi-organiv-diplomatichnoji-sluzhbi-ukrajini.

54. Про затвердження Положення про Адміністрацію Державної служби спеціального зв'язку та захисту інформації України : постанова КМ

України від 03.09.2014 р. № 411 [Електронний ресурс]. — Режим доступу до ресурсу : http://zakon2.rada.gov.ua/laws/show/411-2014-%D0%BF.

55. Про затвердження Положення про порядок взаємодії органів державної влади при здійсненні міжнародно-правового захисту радіочастотного ресурсу України : наказ Адміністрації Державної служби спеціального зв'язку та захисту інформації України, Міністерства оборони України від 25.06.2013 р. № 338/435 [Електронний ресурс]. — Режим доступу до ресурсу : http://zakon4.rada.gov.ua/laws/show/z1192-13.

56. Різуненко А.О. Теорія та практика цифрової обробки зображень : [монографія] / А.О. Різуненко. — Полтава : РВВ ПУСКУ, 2009. — 195 с.

57. Роджерс Д. Математические основы машинной графики : [пер. с англ.] / Д. Роджерс, Дж. Адамс. — М. : Мир, 2001. — 604 с.

58. Рябко Б.Я. Основы современной криптографии и стеганографии / Б.Я. Рябко, А.Н. Фионов. — М. : Горячая линия – Телеком, 2010. — 232 с.

59. Смірнов О.А. Методи та засоби цифрової стеганографії з використанням складних дискретних сигналів для захисту інформаційних ресурсів: дис. доктора технічних наук : 21.05.01 / Смірнов Олексій Анатолійович. – К., 2013. – 467 с.

60. Сорока Н.И. Теория передачи информации : [конспект лекций для студентов специальности 1-53 01 07 «Информационные технологии и управление в технических системах»] / Н.И. Сорока, Г.А. Кривинченко. — Минск : БГУИР, 2005. — 301 с.

61. Соснин Н.В. Компьютерная графика. Математические основы. : [учеб. пособие] / Н.В. Соснин. — Красноярск : ИПК СФУ, 2008. — 138 с.

62. Стеганография, цифровые водяные знаки и стеганоанализ : [монография] / А.В. Аграновский, А.В. Балакин, В.Г. Грибунин, С.А. Сапожников. — М. : Вузовская книга, 2009. — 220 с.

63. Стеганографічний метод приховування даних у векторних зображеннях / О.М. Кінзерявий, В.Ю. Ковтун, С.О. Гнатюк, В.М. Кінзерявий //

науковий журнал «Вісник Інженерної академії України». — 2013. — №3-4. — С. 66-68.

64. Сучасні стеганографічні методи захисту інформації / О.І. Стасюк, С.О. Гнатюк, Н.І. Довгич, М.С. Літош // Захист інформації. — 2011. — №1 (50). — С. 56–63.

65. Тарасов Д.О. Класифікація та аналіз безкоштовних програмних засобів стеганографії / Д.О. Тарасов, А.С. Мельник, М.М. Голобородько // Інформаційні системи та мережі. Вісник НУ “Львівська політехніка” №673. — Л., 2010. — С. 365-374.

66. Хорошко В.А. Введение в компьютерную стеганографию / В.А. Хорошко, М.Е. Шелест. — К. : НАУ, 2002. — 140 с.

67. Хорошко В.А. Методы и средства защиты информации / В.А. Хорошко, А.А. Чекатков. — К. : Юниор, 2003. — 501 с.

68. Швидченко И.В. Стойкие криптостеганографические алгоритмы / И.В. Швидченко // Искусств. интеллект. — 2008. — № 4. — С. 282-290.

69. Шелест М.Е. Цифровая стеганография и ее возможности // Захист інформації. — 1999. — №1. — С. 11-19.

70. Шишкина А.Р. Инструменты контроля в виртуальном пространстве и социально-политические трансформации в Арабских странах / А.Р. Шишкина // Вісник Одеського національного університету. Соціологія і політичні науки. — 2013. — Т.18, Вип. 2(3). — С. 131-138.

71. Шишкина А.Р. Интернет-цензура и «арабская весна» / А.Р. Шишкина // Неприкосновенный запас. Дебаты о политике и культуре. — 2014. — №1 (93). — С. 144-155.

72. Шнайер Б. Практическая криптография / Н. Фергюсон, Б. Шнайер. — М. : Вильямс, 2005. — 425 с.

73. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си / Б. Шнайер. — М. : Триумф, 2002. — 816 с.

74. Яглом И.М. Идеи и методы аффинной и проективной геометрии : [часть I] // И.М. Яглом, В.Г. Ашкинузе. — М. : Учпедгиз, 1962. — 248 с.

75. Cohen E. Discrete B-splines and subdivision techniques in computer-aided geometric design and computer graphics // E. Cohen, T. Lyche, R. Riesenfeld // Computer Graphics and Image Process. — 1980. — 14(2). — pp. 87-111.

76. Dailey D. Building Web Applications with SVG / D. Dailey, J. Frost, D. Strazzullo. — O'Reilly Media, 2012. — 268 p.

77. Depositphotos [Електронний ресурс]. — Режим доступу до ресурсу : http://ru.depositphotos.com/. — Назва з екрана.

78. Digital geographical map watermarking using polyline interpolation / K. Park, K. Kim, H. Kang, S. Han // Proc. of the IEEE Pacific Rim Conference on Multimedia. — 2002. — pp. 58-65.

79. Digital watermarking and steganography / [I. Cox, M. Miller, J. Bloom та ін.]. — 2007. — 558 с.

80. Farin G. Curves and Surfaces for CAGD : [5th Edition] / G. Farin. — Morgan Kaufmann, 2001. — 520 p.

81. Graphicstock [Електронний ресурс]. — Режим доступу до ресурсу : http://www.graphicstock.com/. — Назва з екрана.

82. Hegde S. Potential of SVG for a cartographic interface to a route optimization model for the transport of hazardous material / S. Hegde. — Enschede, 2004. — 82 p.

83. Historical trends in the usage of image file formats for websites [Електронний ресурс]. — 2014. — Режим доступу до ресурсу: http://w3techs.com/technologies/history_overview/image_format/all.

84. Illustration watermarks for vector graphics / H. Sonnet, T. Isenberg, J. Dittmann, T. Strothotte // Proc. of the 11th Pacific Conference on Computer Graphics and Applications. — 2003. — pp. 8-10.

85. Kang H. A vector watermarking using the generalized square mask / H. Kang // Proc. of the International Conference on Information Technology : Coding and Computing. — Las Vegas (USA), 2001. — pp. 234-236.

86. Kim J. Robust Vector Digital Watermarking Using Angles and a Random Table / J. Kim // Advances in Information Sciences and Service Sciences. — 2010. — vol.2. — pp. 79 - 90.

87. Kinzeryavyy O. Method of template hiding data in vector images structure / O. Kinzeryavyy, V. Kinzeryavyy // The sixth world congress «Aviation in the XXI-st century» : «Safety in Aviation and Space Technologies». — К., 2014. — V.1. — С. 568-572.

88. Kipper G. Investigator's Guide to Steganography / G. Kipper. — Boca Raton : Auerbach Publications, 2004. — 240 p.

89. Kitamura I. Copyright protection of vector map using digital watermarking method based on discrete Fourier transform / I. Kitamura, S. Kanai, T. Kishinami // Proc. of the IEEE 2001 International Symposium on Geoscience and Remote Sensing. — 2001. — vol. 3. — pp. 9-13.

90. Kutter M. The watermark copy attack / M. Kutter, S. Voloshvnovskiy, A. Herrigel // Security and Watermarking of Multimedia Content II. — San Jose (USA), 2000. — vol. 3971. — 10 p.

91. Li Y. A blind watermarking of vector graphics images / Y. Li, L. Xu // Proc. the Fifth International Conference on Computational Intelligence and Multimedia Applications. — 2003. — pp. 27-30.

92. Liang B. A Vector Maps Watermarking Algorithm Based On DCT Domain / B. Liang, J. Rong, C. Wang // ISPRS Congr. — 2010. — vol.38(1). — pp. 118-121.

93. Mishra A.R. Cellular technologies for emerging markets : 2G, 3G, and beyond / A.R. Mishra. — 2010. — 326 p.

94. Nikolaidis N. Fourier descriptors watermarking of vector graphics images / N. Nikolaidis, I. Pitas, V. Solachidis // Proc. of the International Conference on Image Processing. — 2000. — vol.3. — pp. 10-13.

95. Nikolaidis N. Watermarking of sets of polygonal lines using fusion techniques / N. Nikolaidis, I. Pitas, A. Giannoula // Proc. of the 2002 IEEE International Conference on Multimedia and Expo. — 2002. — vol.2. — pp. 26-29.

96. Niu X. A survey of digital vector map watermarking / X. Niu, C. Shao, X. Wang // International Journal of Innovative Computing, Information and Control ICIC International. — 2006. — vol.2, №6. — pp. 1301-1316.

97. Niu X. GIS Watermarking. Hiding Data in 2D Vector Maps / X. Niu, C. Shao, X. Wang // Intelligent Multimedia Data Hiding : Studies in Computational Intelligence. — 2007. — vol.58. — pp. 123-155.

98. Ohbuchi R. Robust watermarking of vector digital maps / R. Ohbuchi, H. Ueda, S. Endoh // Proc. of the IEEE International Conference on Multimedia and Expo. — Lausanne (Switzerland), 2002. — vol.1. — pp.577-580.

99. Ohbuchi R. Watermarking 2D vector maps in the mesh-spectral domain / R. Ohbuchi, H. Ueda, S. Endoh. — 2003. — pp. 216-228.

100. Saily M. GSM/EDGE : evolution and performance / M. Saily, G. Sebire, E. Riddington. — 2010. — 504 p.

101. Samoa M. A Scheme of digital watermarking for geographical map data / M. Samoa, Y. Matsuura, Y. Takashima // Proc. of the Symposium on Cryptography and Information Security. — Okinama (Japan), 2000. — pp. 26-28.

102. Scalable Vector Graphics (SVG) 1.1 (Second Edition) [Електронний ресурс] / [E. Dahlström, P. Dengler, A. Grasso та ін.]. — 2011. — Режим доступу до ресурсу: http://www.w3.org/TR/2011/REC-SVG11-20110816/.

103. Schulz G. A high capacity watermarking system for digital maps / G. Schulz, M. Voigt // Proc. of the 2004 Multimedia and Security Workshop on Multimedia and Security. — Magdeburg (Germany), 2004. — pp. 180-186.

104. Solachidis V. Watermarking polygonal lines using Fourier descriptors / V. Solachidis, N. Nikolaidis, I. Pitas // Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing. — Istanbul (Turkey), 2000. — vol.4. — pp. 1955-1958.

105. Sun J. A Reversible Digital Watermarking Algorithm for Vector Maps / J. Sun, G. Zhang, A. Yao, J. Wu // International Journal of Network Security. — 2014. — vol.16, №1. — pp. 40-45.

106. SVG (basic support) [Електронний ресурс]. — Режим доступу до ресурсу: http://caniuse.com/#feat=svg. — Назва з екрана.

107. The Mobile Economy 2015 [Електронний ресурс] // GSMA Intelligence. — 2015. — Режим доступу до ресурсу: http://www.gsmamobileeconomy.com/GSMA_Global_Mobile_Economy_Report_2015.pdf.

108. Vector.me [Електронний ресурс]. — Режим доступу до ресурсу : http://vector.me/. — Назва з екрана.

109. Watermarking road maps against crop and merge attacks / J. Kai, Q. Kenny, H. Yan, M. Xiaobin // Proceedings of the first ACM workshop on Information hiding and multimedia security. — 2013. — pp. 221-230.

Додаток А. Документи, що підтверджують впровадження результатів дисертації

ЗАТВЕРДЖУЮ:

В.о. ректора
Національного авіаційного
університету

_____ В. Харченко

"20" 06 2015 р.

АКТ

впровадження результатів дисертаційної роботи Кінзерявого Олексія Миколайовича «Стеганографічні методи приховування даних у векторні зображення, стійкі до активних атак на основі афінних перетворень» на здобуття кандидата технічних наук у навчальний процес Національного авіаційного університету.

Комісія у складі: голова – д.т.н., професор, завідувач кафедри безпеки інформаційних технологій (БІТ) О.Г. Корченко, доцент кафедри БІТ В.П. Щербина та к.т.н., доцент кафедри БІТ С.О. Гнатюк склали даний акт про те, що результати дисертаційної роботи Кінзерявого Олексія Миколайовича «Стеганографічні методи приховування даних у векторні зображення, стійкі до активних атак на основі афінних перетворень» впроваджено у навчальний процес та використовуються на кафедрі БІТ у 2014-2015 навчальному році при викладанні таких дисциплін: «Інформаційна безпека держави», «Основи криптографічного захисту інформації» та «Безпека інформації в інформаційно-комунікаційних системах».

№ з/п	*Назва роботи, що впроваджується*	*Форма впровадження*	*Ефективність від впровадження*
	1	*2*	*3*
1.	Стеганографічні методи приховування інформації	Лекція *(«Інформаційна безпека держави»)*	Ознайомлення студентів з сучасними стеганографічними методами приховування інформації у зображення
2.	Побітовий та шаблонний методи приховування інформації у векторні зображення	Лекція *(«Основи криптографічного захисту інформації»)*	Ознайомлення студентів з стеганографічними методами приховування інформації у векторні зображення
3.	Структурна модель процесу прихованої передачі інформації резервним каналом зв'язку. Методи приховування інформації у векторні зображення	Лекція *(«Безпека інформації в інформаційно-комунікаційних системах»)*	Ознайомлення студентів з структурною моделлю процесу прихованої передачі інформації резервним каналом зв'язку та методами приховування інформації у векторні зображення

Голова комісії,
завідувач кафедри БІТ
д.т.н., професор О. Корченко

Члени комісії:
доцент кафедри БІТ В. Щербина
к.т.н., доцент, доцент кафедри БІТ С. Гнатюк

САЙФЕР
Системи захисту інформації

Адреса: 04107, Київ, вул. Нагірна, 25
Тел./Факс: (044) 484-46-17, 484-46-12, 483-03-22
E-mail: sales@cipher.kiev.ua
http://www.elpay.com; http://www.cipher.kiev.ua

№ 018/15 від 23.06.2015р

АКТ
про впровадження результатів дисертаційної роботи
Кінзерявого Олексія Миколайовича

Комісія у складі голови – директора товариства з обмеженою відповідальністю «Сайфер ЛТД», кандидата технічних наук Боровікова О.М., членів комісії – провідного розробника Бойко С.Т., провідного розробника Прокоповича Л.Ю., складено цей акт про те, що при розробці перспективних бібліотек стеганографічних перетворень, реалізовано такі результати наукових досліджень Кінзерявого Олексія Миколайовича:

1. Побітовий та шаблонний алгоритми приховування інформації у векторні зображення, що призначені для вбудовування секретного повідомлення у криві Без'є третього ступеня SVG зображень. Дані алгоритми візуально не спотворюють зображення та, за допомогою властивості афінно-інваріантності кривих Без'є, забезпечуються стійкість до атак на основі афінних перетворень.

Запропоновані, у результаті виконання наукових досліджень Кінзерявого Олексія Миколайовича, програмні реалізації, дозволяють практично реалізувати розроблені стеганографічні алгоритми та забезпечують високу швидкість вбудовування/вилучення інформації.

Голова комісії
Директор ТОВ «Сайфер ЛТД», к.т.н. О.М. Боровіков

Члени комісії:

Провідний розробник С.Т. Бойко

Провідний розробник Л.Ю. Прокопович

ТОВ «КАСКАД ГРУП УКРАЇНА»

Україна, м. Київ, 04080, вул. Вікентія Хвойки, 18/14. оф 231

тел. (044) 425 29 69
тел./факс (044) 425 29 69
e-mail: office@cascade-group.com.ua

LLC «CASCADE GROUP UKRAINE».

Vikentiy Xvoiky st. 18/14, of. 231
Kiev 04080, Ukraine

phone. (044) 425 29 69
phone. (044) 425 29 69
e-mail: office@cascade-group.com.ua

19.02.2015р.

АКТ

впровадження результатів дисертаційної роботи
Кінзерявого Олексія Миколайовича за темою «Стеганографічні методи приховування даних у векторні зображення, стійкі до активних атак на основі афінних перетворень» у діяльність товариства з обмеженою відповідальністю (ТОВ) «Каскад Груп Укрїна»

Даний акт складено про те, що результати дисертаційної роботи Кінзерявого Олексія Миколайовича впроваджено та використано у діяльності ТОВ «Каскад Груп Україна».

Під час написання дисертації Кінзерявий О.М. розробив ряд консольних програмних засобів побітового та шаблонного методів приховування інформації у векторні зображення. Дані програмні засоби реалізовані на мові програмування C++ в середовищі Microsoft Visual Studio 2012.

Програмні засоби призначені для підвищення ефективності захисту електронних інформаційних ресурсів шляхом приховування їх у SVG зображення. Серед можливостей даних програм слід відмітити швидкісне приховування/відтворення електронних ресурсів та високу стійкість розроблених стеганографічних алгоритмів до атак на основі афінних перетворень.

У діяльності нашої компанії дані засоби використано для підвищення ефективності захисту обміну даних між розподіленими інформаційними базами даних.

Директор І.М. Кучмар

Додаток Б. Результати експериментального дослідження стійкості алгоритмів StegoBIT і StegoTEMPL до афінних перетворень

В додатку наведені середні результати стійкості розроблених алгоримів вбудовування інформації у криві Без'є 30-ти довільних векторних зображень до афінних перетворень, при використанні різних стеганоключах та параметрах приховування.

При вбудовуванні інформації використовувалися наступні пари стеганоключів:

–за алгоритмом StegoBIT: а) $\Delta t = 2 \cdot 10^{-3}$; б) $\Delta t = 1 \cdot 10^{-3}$; в) $\Delta t = 5 \cdot 10^{-4}$;

–за алгоритмом StegoTEMPL таблиці співвідношень представленні в табл. Б.1 при $l = 4$, $k = 16$.

Таблиця Б.1

Таблиці співвідношень значень елементів шаблону з різними кроками побудови кривої Без'є

Індекс k	Шаблон TV_l^k, біт	Крок зміни $T\Delta t^k$
1	0000	0,009
2	0001	0,0085
3	0010	0,008
4	0011	0,0076
5	0100	0,007
6	0101	0,0065
7	0110	0,006
8	0111	0,0055
9	1000	0,005
10	1001	0,0045
11	1010	0,004
12	1011	0,0035
13	1100	0,003
14	1101	0,0025
15	1110	0,002
16	1111	0,0015

а)

Індекс k	Шаблон TV_l^k, біт	Крок зміни $T\Delta t^k$
1	0000	0,007
2	0001	0,0066
3	0010	0,0062
4	0011	0,0058
5	0100	0,0054
6	0101	0,005
7	0110	0,0046
8	0111	0,0042
9	1000	0,0038
10	1001	0,0034
11	1010	0,003
12	1011	0,0026
13	1100	0,0022
14	1101	0,0018
15	1110	0,0014
16	1111	0,001

б)

Індекс k	Шаблон TV_l^k, біт	Крок зміни $T\Delta t^k$
1	0000	0,001
2	0001	0,0017
3	0010	0,0024
4	0011	0,0031
5	0100	0,0038
6	0101	0,0045
7	0110	0,0052
8	0111	0,0059
9	1000	0,0066
10	1001	0,0073
11	1010	0,008
12	1011	0,0087
13	1100	0,0094
14	1101	0,0101
15	1110	0,0108
16	1111	0,0115

в)

Приховування інформації у обрані SVG зображення для кожного експерименту здійснювалося за наступними параметрами:

Експеримент 1. $V_1 = 3$, $V_2 \geq 1$, $V_3 = 40$, $V_4 = 5$, $V_5 = 2 \cdot 10^{-5}$, стеганоключем *а* (для побітового алгоритму) та стеганоключем *а* представленого в табл. Б.1 (для шаблонного алгоритму).

Експеримент 2. $V_1 = 3$, $V_2 \geq 1$, $V_3 = 40$, $V_4 = 5$, $V_5 = 2 \cdot 10^{-5}$, стеганоключем *б* (для побітового алгоритму) та стеганоключем *б* представленого в табл. Б.1 (для шаблонного алгоритму).

Експеримент 3. $V_1 = 3$, $V_1 = 3 V_2 \geq 1$, $V_3 = 40$, $V_4 = 5$, $V_5 = 2 \cdot 10^{-5}$, стеганоключем *в* (для побітового алгоритму) та стеганоключем *в* представленого в табл. Б.1 (для шаблонного алгоритму). $V_1 = 3$

Експеримент 4. , $V_2 \geq 1$, $V_3 = 40$, $V_4 = 6$, $V_5 = 2 \cdot 10^{-5}$, стеганоключем *а* (для побітового алгоритму) та стеганоключем *а* представленого в табл. Б.1 (для шаблонного алгоритму).

Експеримент 5. , $V_2 \geq 1$, $V_3 = 40$, $V_4 = 6$, $V_5 = 2 \cdot 10^{-5}$, стеганоключем *б* (для побітового алгоритму) та стеганоключем *б* представленого в табл. Б.1 (для шаблонного алгоритму).

Експеримент 6. $V_1 = 3$, $V_2 \geq 1$, $V_3 = 40$, $V_4 = 6$, $V_5 = 2 \cdot 10^{-5}$, стеганоключем *в* (для побітового алгоритму) та стеганоключем *в* представленого в табл. Б.1 (для шаблонного алгоритму).

Експеримент 7. $V_1 = 3,\ V_2 \geq 1,\ V_3 = 40,\ V_4 = 5,\ V_5 = 4 \cdot 10^{-5}$, стеганоключем *а* (для побітового алгоритму) та стеганоключем *а* представленого в табл. Б.1 (для шаблонного алгоритму).

Експеримент 8. $V_1 = 3,\ V_2 \geq 1,\ V_3 = 40,\ V_4 = 5,\ V_5 = 4 \cdot 10^{-5}$, стеганоключем *б* (для побітового алгоритму) та стеганоключем *б* представленого в табл. Б.1 (для шаблонного алгоритму).

Експеримент 9. $V_1 = 3,\ V_2 \geq 1,\ V_3 = 40,\ V_4 = 5,\ V_5 = 4 \cdot 10^{-5}$, стеганоключем *в* (для побітового алгоритму) та стеганоключем *в* представленого в табл. Б.1 (для шаблонного алгоритму).

Експеримент 10. $V_1 = 3,\ V_2 \geq 1,\ V_3 = 40,\ V_4 = 6,\ V_5 = 4 \cdot 10^{-5}$, стеганоключем *а* (для побітового алгоритму) та стеганоключем *а* представленого в табл. Б.1 (для шаблонного алгоритму).

Експеримент 11. $V_1 = 3,\ V_2 \geq 1,\ V_3 = 40,\ V_4 = 6,\ V_5 = 4 \cdot 10^{-5}$, стеганоключем *б* (для побітового алгоритму) та стеганоключем *б* представленого в табл. Б.1 (для шаблонного алгоритму).

Експеримент 12. $V_1 = 3,\ V_2 \geq 1,\ V_3 = 40,\ V_4 = 6,\ V_5 = 4 \cdot 10^{-5}$, стеганоключем *в* (для побітового алгоритму) та стеганоключем *в* представленого в табл. Б.1 (для шаблонного алгоритму).

Експеримент 13. $V_1 = 3,\ V_2 \geq 1,\ V_3 = 60,\ V_4 = 5,\ V_5 = 2 \cdot 10^{-5}$, стеганоключем *а* (для побітового алгоритму) та стеганоключем *а* представленого в табл. Б.1 (для шаблонного алгоритму).

Експеримент 14. $V_1 = 3,\ V_2 \geq 1,\ V_3 = 60,\ V_4 = 5,\ V_5 = 2 \cdot 10^{-5}$, стеганоключем *б* (для побітового алгоритму) та стеганоключем *б* представленого в табл. Б.1 (для шаблонного алгоритму).

Експеримент 15. $V_1 = 3,\ V_2 \geq 1,\ V_3 = 60,\ V_4 = 5,\ V_5 = 2 \cdot 10^{-5}$, стеганоключем *в* (для побітового алгоритму) та стеганоключем *в* представленого в табл. Б.1 (для шаблонного алгоритму).

Експеримент 16. $V_1 = 3,\ V_2 \geq 1,\ V_3 = 60,\ V_4 = 6,\ V_5 = 2 \cdot 10^{-5}$, стеганоключем *а* (для побітового алгоритму) та стеганоключем *а* представленого в табл. Б.1 (для шаблонного алгоритму).

Експеримент 17. $V_1 = 3,\ V_2 \geq 1,\ V_3 = 60,\ V_4 = 6,\ V_5 = 2 \cdot 10^{-5}$, стеганоключем *б* (для побітового алгоритму) та стеганоключем *б* представленого в табл. Б.1 (для шаблонного алгоритму).

Експеримент 18. $V_1 = 3,\ V_2 \geq 1,\ V_3 = 60,\ V_4 = 6,\ V_5 = 2 \cdot 10^{-5}$, стеганоключем *в* (для побітового алгоритму) та стеганоключем *в* представленого в табл. Б.1 (для шаблонного алгоритму).

Експеримент 19. $V_1 = 3,\ V_2 \geq 1,\ V_3 = 60,\ V_4 = 5,\ V_5 = 4 \cdot 10^{-5}$, стеганоключем *а* (для побітового алгоритму) та стеганоключем *а* представленого в табл. Б.1 (для шаблонного алгоритму).

Експеримент 20. $V_1 = 3,\ V_2 \geq 1,\ V_3 = 60,\ V_4 = 5,\ V_5 = 4 \cdot 10^{-5}$, стеганоключем *б* (для побітового алгоритму) та стеганоключем *б* представленого в табл. Б.1 (для шаблонного алгоритму).

Експеримент 21. $V_1 = 3,\ V_2 \geq 1,\ V_3 = 60,\ V_4 = 5,\ V_5 = 4 \cdot 10^{-5}$, стеганоключем *в* (для побітового алгоритму) та стеганоключем *в* представленого в табл. Б.1 (для шаблонного алгоритму).

Експеримент 22. $V_1 = 3,\ V_2 \geq 1,\ V_3 = 60,\ V_4 = 6,\ V_5 = 4 \cdot 10^{-5}$, стеганоключем *а* (для побітового алгоритму) та стеганоключем *а* представленого в табл. Б.1 (для шаблонного алгоритму).

Експеримент 23. $V_1 = 3,\ V_2 \geq 1,\ V_3 = 60,\ V_4 = 6,\ V_5 = 4 \cdot 10^{-5}$, стеганоключем *б* (для побітового алгоритму) та стеганоключем *б* представленого в табл. Б.1 (для шаблонного алгоритму).

Експеримент 24. $V_1 = 3,\ V_2 \geq 1,\ V_3 = 60,\ V_4 = 6,\ V_5 = 4 \cdot 10^{-5}$, стеганоключем *в* (для побітового алгоритму) та стеганоключем *в* представленого в табл. Б.1 (для шаблонного алгоритму).

Експеримент 25. $V_1 = 3,\ V_2 \geq 1,\ V_3 = 80,\ V_4 = 5,\ V_5 = 2 \cdot 10^{-5}$, стеганоключем *а* (для побітового алгоритму) та стеганоключем *а* представленого в табл. Б.1 (для шаблонного алгоритму).

Експеримент 26. $V_1 = 3,\ V_2 \geq 1,\ V_3 = 80,\ V_4 = 5,\ V_5 = 2 \cdot 10^{-5}$, стеганоключем *б* (для побітового алгоритму) та стеганоключем *б* представленого в табл. Б.1 (для шаблонного алгоритму).

Експеримент 27. $V_1 = 3,\ V_2 \geq 1,\ V_3 = 80,\ V_4 = 5,\ V_5 = 2 \cdot 10^{-5}$, стеганоключем *в* (для побітового алгоритму) та стеганоключем *в* представленого в табл. Б.1 (для шаблонного алгоритму).

Експеримент 28. $V_1 = 3,\ V_2 \geq 1,\ V_3 = 80,\ V_4 = 6,\ V_5 = 2 \cdot 10^{-5}$, стеганоключем *а* (для побітового алгоритму) та стеганоключем *а* представленого в табл. Б.1 (для шаблонного алгоритму).

Експеримент 29. $V_1 = 3,\ V_2 \geq 1,\ V_3 = 80,\ V_4 = 6,\ V_5 = 2 \cdot 10^{-5}$, стеганоключем *б* (для побітового алгоритму) та стеганоключем *б* представленого в табл. Б.1 (для шаблонного алгоритму).

Експеримент 30. $V_1 = 3,\ V_2 \geq 1,\ V_3 = 80,\ V_4 = 6,\ V_5 = 2 \cdot 10^{-5}$, стеганоключем *в* (для побітового алгоритму) та стеганоключем *в* представленого в табл. Б.1 (для шаблонного алгоритму).

Експеримент 31. $V_1 = 3,\ V_2 \geq 1,\ V_3 = 80,\ V_4 = 5,\ V_5 = 4 \cdot 10^{-5}$, стеганоключем *а* (для побітового алгоритму) та стеганоключем *а* представленого в табл. Б.1 (для шаблонного алгоритму).

Експеримент 32. $V_1 = 3,\ V_2 \geq 1,\ V_3 = 80,\ V_4 = 5,\ V_5 = 4 \cdot 10^{-5}$, стеганоключем *б* (для побітового алгоритму) та стеганоключем *б* представленого в табл. Б.1 (для шаблонного алгоритму).

Експеримент 33. $V_1 = 3,\ V_2 \geq 1,\ V_3 = 80,\ V_4 = 5,\ V_5 = 4 \cdot 10^{-5}$, стеганоключем *в* (для побітового алгоритму) та стеганоключем *в* представленого в табл. Б.1 (для шаблонного алгоритму).

Експеримент 34. $V_1 = 3,\ V_2 \geq 1,\ V_3 = 80,\ V_4 = 6,\ V_5 = 4 \cdot 10^{-5}$, стеганоключем *а* (для побітового алгоритму) та стеганоключем *а* представленого в табл. Б.1 (для шаблонного алгоритму).

Експеримент 35. $V_1 = 3,\ V_2 \geq 1,\ V_3 = 80,\ V_4 = 6,\ V_5 = 4 \cdot 10^{-5}$, стеганоключем *б* (для побітового алгоритму) та стеганоключем *б* представленого в табл. Б.1 (для шаблонного алгоритму).

Експеримент 36. $V_1 = 3,\ V_2 \geq 1,\ V_3 = 80,\ V_4 = 6,\ V_5 = 4 \cdot 10^{-5}$, стеганоключем *в* (для побітового алгоритму) та стеганоключем *в* представленого в табл. Б.1 (для шаблонного алгоритму).

Результати стійкості алгоритмів StegoBIT та StegoTEMPL до атаки перенесення представлені в табл. Б.2-Б.4 та табл. Б.5-Б.7 відповідно.

Таблиця Б.2

Результати стійкості алгоритму StegoBIT до атаки перенесення (експерименти 1-12)

Номер перетворення	**Експеримент**											
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
0	0	1,13	1,33	0	0	0	0	3,43	24,13	0	0	0
4	0	0,11	1,33	0,13	0	0	0	2,46	24,48	0	0	0

8	0	0,23	1,70	0	0	0	0	2,63	24,13	0	0,03	0
12	0	0,08	1,35	0	0	0	0	2,47	24,13	0	0	0
16	0	0	2,09	0	0,05	0	0	2,30	24,43	0	0	0
20	0	0	1,33	0	0	0	0	2,30	24,13	0	0	0
24	0,08	0	1,67	0,16	0	0	0	2,30	24,43	0	0	0
28	0	0,07	1,33	0	0	0	0	2,30	24,24	0	0	0
32	0	0	1,55	0	0,08	0	0	2,30	24,13	0	0	0
36	0	0	1,33	0	0	0	0,30	2,30	24,37	0	0	0
40	0	0	1,33	0	0	0	0	2,30	24,13	0	0	0
44	0	0,43	1,37	0	0	0	0	2,30	24,13	0	0	0
48	0	0	1,63	0	0	0	0	2,30	24,13	0	0	0
52	0	0,29	1,33	0	0	0	0,20	2,30	24,28	0	0,20	0,19
56	0,58	0	1,62	0	0	0	0	2,30	24,13	0	0	0
60	0	0,11	1,47	0,02	0	0	0,26	2,30	24,40	0	0	0
64	0	0	1,42	0	0	0	0	2,63	24,23	0	0	0
68	0	0	1,33	0	0	0	0,32	2,46	24,13	0	0	0
72	0,13	0	1,33	0	0	0	0,07	2,30	24,13	0,08	0	0,29
76	0	0,05	1,48	0	0	0,18	0	2,30	24,13	0	0	0
80	0	0	1,33	0	0	0	0	2,37	24,13	0	0	0
84	0	0,07	1,80	0	0,04	0	0	2,30	24,13	0	0	0
88	0,12	0,26	1,52	0,29	0	0	0,05	2,34	24,13	0	0	0
92	0,02	0	1,39	0	0	0,27	0	2,30	24,13	0	0	0
96	0,22	0	1,33	0	0	0	0,09	2,30	24,19	0	0	0
100	0	0,09	1,33	0	0	0	0	3,02	24,20	0	0	0

Таблиця Б.3

Результати стійкості алгоритму StegoBIT до атаки перенесення (експерименти 13-24)

Номер перетворення	**Експеримент**											
	13	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
0	5,77	0	0	0	0	0	6,50	16,83	0	0	0	0
4	6,15	2,32	0,41	0	0,02	0	7,02	14,63	0	0	0	0
8	5,92	2,50	0	0	0	0	6,50	15,17	0,53	0	0	0
12	5,77	2,41	0,83	0,36	0	0	6,50	14,63	0	0	0	0
16	6,05	1,19	0	0	0	0	6,50	15,08	0,50	0	0	0
20	6,30	0,67	0	0	0	0	6,50	14,63	0,51	0	0	0
24	6,57	0,15	0,88	0	0	0,53	6,50	15,17	0,50	0	0	0
28	5,77	0,64	0	0	0	0	6,50	14,63	0	0,50	0	0
32	5,77	2,76	0,38	0	0	0	6,50	14,63	0	0	0	0
36	6,45	2,75	0	0	0	0	6,50	14,63	0	0	0	0
40	6,13	4,91	0,23	0	0	0	6,50	14,63	0	0	0	0
44	5,77	5,33	0,33	0	0	0	6,50	14,63	0	0	0	0
48	6,22	4,11	0	0	0	0	6,50	14,63	0	0	0	0
52	6,03	3,00	0,48	0	0,27	0	6,50	14,63	0	0	0	0
56	7,40	2,00	0,03	0	0	0	6,99	14,63	0	0	0	0
60	6,79	2,48	0,95	0	0	0	6,50	14,63	1,06	0	0	0
64	5,77	4,24	0	0	0	0	6,50	15,16	0	0	0,52	0
68	5,77	6,65	0,40	0	0	0	6,50	14,63	0	0	0	0
72	5,77	5,11	0	0	0	0	6,50	14,63	0	0	0	0
76	5,77	4,20	0,12	0,09	0	0	6,50	14,63	0	0	0	0
80	6,00	6,09	0	0	0,38	0	6,50	14,63	0,53	0	0	0
84	5,77	5,17	0	0	0	0	7,00	14,63	0	0	0	0
88	5,77	4,04	0	0	0	0	6,93	15,17	0	0	0	0
92	5,87	4,71	0,38	0	0	0	7,01	14,63	0,49	0	0	0
96	5,77	5,25	0	0	0	0	6,50	15,16	0	0	0	0
100	5,98	4,68	0,15	0,43	0	0,35	7,03	15,70	0	0	0	0

Таблиця Б.4

Результати стійкості алгоритму StegoBIT до атаки перенесення (експерименти 25-36)

Номер перетворення	**Експеримент**											
	25	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**	**36**
0	123,60	10,20	1,63	31,27	0	0	223,07	51,70	45,60	10,77	0	0
4	140,85	10,51	1,73	48,57	0	0	226,58	52,44	46,04	24,29	0	0
8	140,77	10,52	1,73	48,47	0	0	226,03	51,79	46,38	24,07	0	0
12	141,81	10,20	2,04	48,47	0	0	226,03	51,70	45,60	24,07	0	0
16	140,77	10,20	2,21	48,47	0	0	226,68	51,80	45,89	24,07	0	0
20	140,94	10,20	1,73	48,47	0	0	225,38	51,70	45,60	24,26	0	0
24	141,20	10,20	1,73	48,47	0	0	225,58	51,70	45,60	24,07	0,13	0
28	142,23	10,20	1,73	48,47	0	0	226,03	52,06	46,02	24,07	0	0
32	140,97	10,20	2,07	48,47	0	0	226,27	51,70	44,94	24,07	0	0
36	141,03	10,64	1,82	48,47	0	0	226,61	51,83	45,60	24,35	0	0
40	141,03	10,23	1,73	48,47	0	0	226,69	51,70	45,60	24,07	0	0
44	141,49	10,20	1,73	48,47	0	0	226,03	52,39	45,60	24,07	0	0
48	141,21	10,20	1,73	48,48	0	0	226,35	51,70	45,60	24,07	0	0,40
52	140,77	10,20	2,00	48,47	0	0	226,03	51,70	45,77	24,07	0	0
56	140,79	10,20	2,20	48,47	0,15	0	226,85	51,86	45,60	24,07	0,32	0

60	141,36	10,20	1,73	48,87	0	0	226,03	52,29	45,60	24,07	0	0
64	140,77	10,20	1,73	48,69	0	0	226,85	51,70	45,68	24,07	0	0
68	140,77	10,55	2,19	48,47	0	0,07	226,86	51,70	45,60	24,07	0	0
72	140,77	10,20	1,73	48,47	0	0	226,32	52,36	45,60	24,07	0	0
76	141,29	10,28	1,73	48,47	0	0	226,55	51,70	45,60	24,07	0	0
80	141,26	10,20	1,73	48,47	0	0	226,03	51,99	45,60	24,07	0	0
84	141,29	10,20	1,73	48,47	0	0,44	226,43	51,70	45,87	24,07	0	0
88	141,47	10,29	1,73	48,47	0	0	226,26	52,06	46,06	24,07	0	0,42
92	141,12	10,20	1,73	48,47	0	0	226,26	51,70	45,60	24,07	0	0
96	141,41	10,64	1,98	48,47	0	0	226,08	51,70	46,50	24,07	0	0
100	140,77	10,43	1,73	48,47	0	0	226,70	51,51	45,64	24,18	0	0

Таблиця Б.5

Результати стійкості алгоритму StegoTEMPL до атаки перенесення (експерименти 1-12)

Номер перетворення	**Експеримент**											
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
0	1,17	8,97	1,23	0	0	0	15,97	27,77	6,10	0	0	0
4	1,17	9,20	0	0	0	0	15,97	27,77	6,20	0	0	0
8	1,17	9,63	0	0	0	0	15,97	27,77	6,20	0	0	0
12	1,17	8,97	0	0	0	0	16,28	27,77	6,53	0,29	0	0
16	1,17	8,97	0	0	0	0	15,93	27,77	6,20	0	0	0
20	1,50	8,97	0	0	0	0	15,97	27,77	6,20	0	0	0
24	1,17	8,97	0	0	0	0	15,97	27,77	6,20	0	0,33	0
28	1,17	8,97	0,33	0	0	0	15,97	27,77	6,20	0	0	0
32	1,17	9,28	0,32	0	0	0	15,97	27,77	6,20	0	0	0
36	1,17	8,97	0	0	0	0	15,97	27,77	6,20	0	0	0
40	1,17	8,97	0	0	0	0	15,97	27,77	6,20	0	0	0,33
44	1,17	8,97	0	0	0	0	15,97	27,77	6,20	0	0	0
48	1,17	8,97	0	0	0	0	16,29	27,77	6,20	0	0	0
52	1,17	8,97	0	0	0	0	15,97	27,77	6,20	0	0	0
56	1,17	8,97	0,33	0	0	0	16,29	27,77	6,20	0	0	0
60	1,17	8,97	0,29	0	0	0	15,97	27,77	6,20	0	0	0
64	1,17	8,97	0	0	0	0	15,97	27,77	6,20	0	0	0
68	1,50	8,97	0	0	0	0	15,97	27,77	6,20	0	0	0
72	1,17	9,62	0	0,33	0	0	16,62	27,77	6,20	0	0	0
76	1,49	8,97	0	0	0	0	16,27	27,77	6,20	0	0	0
80	1,17	8,97	0	0	0	0	15,97	27,77	6,20	0	0	0
84	1,17	8,97	0	0	0	0	15,97	27,77	6,20	0	0	0
88	1,17	8,97	0	0	0	0	15,97	27,77	6,20	0	0	0
92	1,17	9,63	0,33	0	0	0	15,97	27,77	6,20	0	0	0
96	1,17	9,63	0	0	0	0	15,97	28,08	6,53	0	0	0
100	1,17	8,97	0	0	0	0	15,97	28,08	6,20	0	0	0

Таблиця Б.6

Результати стійкості алгоритму StegoTEMPL до атаки перенесення (експерименти 13-24)

Номер перетворення	**Експеримент**											
	13	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
0	6,50	16,83	0	0	0	0	63,10	74,17	20,80	0	0	0
4	7,02	14,63	0	0	0	0	62,90	71,51	20,28	0	0	0
8	6,50	15,17	0,53	0	0	0	62,90	72,13	20,80	0	0	0
12	6,50	14,63	0	0	0	0	62,90	71,56	20,80	0	0	0
16	6,50	15,08	0,50	0	0	0	62,90	72,03	20,28	0	0	0
20	6,50	14,63	0,51	0	0	0	62,93	71,51	21,33	0	0	0
24	6,50	15,17	0,50	0	0	0	62,95	72,03	20,80	0	0	0
28	6,50	14,63	0	0,50	0	0	62,90	71,51	20,28	0	0	0
32	6,50	14,63	0	0	0	0	63,68	71,51	20,80	0	0	0,53
36	6,50	14,63	0	0	0	0	62,90	72,03	20,28	0	0	0
40	6,50	14,63	0	0	0	0	63,39	72,03	21,28	0	0	0
44	6,50	14,63	0	0	0	0	62,90	71,51	20,80	0	0	0
48	6,50	14,63	0	0	0	0	62,90	72,03	20,28	0	0	0
52	6,50	14,63	0	0	0	0	62,90	71,51	20,80	0	0	0
56	6,99	14,63	0	0	0	0	62,90	72,03	20,80	0	0	0
60	6,50	14,63	1,06	0	0	0	63,18	71,51	20,28	0	0	0
64	6,50	15,16	0	0	0,52	0	63,41	72,03	20,80	0	0	0
68	6,50	14,63	0	0	0	0	63,43	72,03	21,33	0	0	0
72	6,50	14,63	0	0	0	0	62,97	72,03	21,33	0	0	0
76	6,50	14,63	0	0	0	0	62,90	72,03	20,76	0	0	0
80	6,50	14,63	0,53	0	0	0	62,90	72,03	20,80	0	0	0
84	7,00	14,63	0	0	0	0	63,43	72,03	20,80	0	0	0
88	6,93	15,17	0	0	0	0	62,90	72,56	20,80	0	0	0
92	7,01	14,63	0,49	0	0	0	63,42	72,03	21,32	0	0	0
96	6,50	15,16	0	0	0	0	62,90	72,56	20,80	0	0	0
100	7,03	15,70	0	0	0	0	62,90	72,03	20,80	0	0	0

Таблиця Б.7

Результати стійкості алгоритму StegoTEMPL до атаки перенесення (експерименти 25-36)

Номер перетворення	Експеримент											
	25	26	27	28	29	30	31	32	33	34	35	36
0	19,37	24,03	0	0	0	0	125,53	113,67	35,33	0	0	0
4	19,37	21,43	0	0	0	0	125,87	110,03	35,27	0	0	0
8	19,37	22,04	0	0	0	0	125,72	110,05	35,36	0	0	0
12	19,37	21,43	1,27	0	0	0	125,19	109,40	34,68	0	0	0
16	19,37	21,43	0,63	0	0	0	125,84	109,41	35,33	0	0	0
20	19,37	21,43	0	0	0	0	126,48	109,40	34,68	0	0	0
24	19,37	21,43	0	0	0	0	125,86	109,40	34,68	0	0	0
28	19,37	22,10	1,27	0	0	0	125,20	109,31	34,68	0	0	0
32	19,37	21,43	0	0	0	0	125,20	109,92	34,68	0	0	0
36	19,37	21,43	0	0	0	0	125,83	109,40	35,33	0,67	0	0
40	19,37	22,08	0	0	0	0	125,20	108,78	34,68	0	0	0
44	19,37	21,43	0	0	0	0	125,20	109,40	35,84	0	0	0
48	19,37	22,08	0	0	0	0	125,20	109,40	35,33	0	0	0
52	19,37	21,38	0	0	0	0	125,84	109,40	34,68	0	0	0
56	19,37	21,43	0,63	0	0	0	125,20	109,40	35,33	0	0	0
60	20,02	21,43	1,28	0	0	0	125,20	109,90	35,33	0	0	0
64	20,03	21,43	0	0	0	0	125,20	109,40	34,68	0	0	0
68	19,37	21,43	0	0	0	0	125,20	109,40	34,68	0	0	0
72	19,37	21,43	0	0	0	0	124,71	109,43	35,98	0	0	0
76	19,37	21,43	0,66	0	0	0	125,20	109,40	33,59	0	0	0
80	20	21,43	0	0	0	0	125,20	110,06	36,00	0	0	0
84	19,37	21,43	0	0	0,65	0	125,80	110,06	35,99	0	0	0
88	19,37	21,43	0	0	0	0	125,20	109,40	35,05	0	0	0
92	19,37	21,43	0	0	0	0	125,86	109,40	34,68	0	0	0
96	19,37	21,43	0,52	0	0	0	125,20	109,40	34,68	0	0	0
100	19,37	21,43	0,63	0,64	0	0	125,20	109,40	35,33	0	0	0

Результати стійкості алгоритмів StegoBIT та StegoTEMPL до атаки повороту представлені в табл. Б.8-Б.10 та табл. Б.11-Б.13 відповідно.

Таблиця Б.8

Результати стійкості алгоритму StegoBIT до атаки повороту (експерименти 1-12)

Номер перетворення	Експеримент											
	1	2	3	4	5	6	7	8	9	10	11	12
0	0	1,13	1,33	0	0	0	0	3,43	24,13	0	0	0
10	0,09	0	0,49	0	0,11	0,03	0,01	2,35	23,56	0	0,11	0,03
20	0,04	0,20	0,17	0	0	0	0,04	4,27	24,78	0	0	0
30	0,11	0,18	0,61	0	0	0	0,11	2,27	23,14	0	0	0
40	0,05	0,08	1,96	0	0,01	0	0,13	1,57	22,43	0	0,01	0
50	0	0,49	3,71	0	0	0	0	2,34	25,04	0	0	0
60	0,03	0,16	4,24	0,02	0,04	0	0,03	2,16	24,23	0,02	0,04	0
70	0,14	0,05	4,97	0	0	0	0,12	3,13	24,60	0	0	0
80	0,37	0,08	5,12	0,06	0	0,01	0,37	4,15	24,09	0,06	0	0,01
90	0,21	0,49	6,26	0	0	0,09	0,21	4,18	24,83	0	0	0,09
100	0,14	0,12	4,10	0	0	0	0,14	3,16	24,82	0	0	0
110	0,25	0,14	3,28	0	0,04	0,12	0,25	4,16	26,21	0	0,04	0,12
120	0,37	0,16	5,72	0,04	0,04	0	0,29	3,02	26,75	0,04	0,04	0
130	0,22	0,07	7,30	0,07	0,22	0,02	0,12	2,11	27,19	0,07	0,22	0,02
140	0,04	0,35	7,69	0	0,06	0	0,04	2,79	29,42	0	0,06	0
150	0,22	0,27	7,23	0,02	0	0,12	0,22	2,72	29,23	0,02	0	0,12
160	0,35	0,41	7,63	0	0,01	0	0,22	3,22	29,64	0	0,01	0
170	0,25	0,14	8,86	0	0	0	0,25	5,00	31,08	0	0	0
180	0,38	0,33	9,98	0,07	0,11	0,09	0,26	4,95	29,77	0,07	0,11	0,09
190	0,18	0,23	8,41	0	0	0	0,14	5,73	27,78	0	0	0
200	0,22	0,10	9,22	0	0	0,14	0,22	5,60	29,62	0	0	0,14
210	0,20	0,41	11,39	0	0	0,16	0,20	3,94	31,94	0	0	0,16
220	0,34	0,20	11,68	0	0,22	0	0,42	3,44	34,09	0	0,22	0
230	0,07	0,30	10,69	0	0,06	0	0,07	4,49	35,18	0	0,06	0
240	0,19	0,19	10,59	0	0	0	0,96	3,03	35,02	0	0	0
250	0,12	0,27	10,40	0	0,01	0	0,24	4,23	34,87	0	0,01	0
260	0,12	0,13	14,01	0,04	0	0,01	0,12	5,74	34,07	0,04	0	0,01
270	0,68	0,21	14,03	0,18	0,09	0,04	0,07	6,28	32,17	0,01	0	0,04
280	0,55	0,19	10,83	0,15	0,43	0	0,17	7,70	33,04	0,03	0	0
290	0,41	0,21	11,54	0,15	2,21	1,07	0,41	7,95	35,97	0	0,08	0,02
300	0,82	0,69	13,73	0,22	3,84	2,20	0,36	5,67	38,42	0	0	0,02
310	0,87	1,56	17,12	1,30	4,92	1,87	0,38	5,81	38,89	0,11	0	0
320	1,50	2,21	17,45	2,36	2,72	2,34	0,24	5,65	39,44	0	0	0,04
330	1,85	3,08	16,93	2,17	3,48	1,56	1,37	4,51	39,85	0,02	0,10	0
340	5,52	3,33	19,86	5,52	3,97	1,88	1,26	5,15	40,40	0	0,07	0
350	10,27	2,52	18,69	7,77	3,24	5,46	0,33	7,41	38,93	0	0	0,02
360	12,98	4,41	19,74	8,00	6,31	7,89	0,13	8,23	38,42	0,07	0,11	0,09

Таблиця Б.9

Результати стійкості алгоритму StegoBIT до атаки повороту (експерименти 13-24)

Номер перетворення	Експеримент											
	13	14	15	16	17	18	19	20	21	22	23	24
0	5,77	0	7,90	0	0	0	45,37	11,53	34,43	0	0	0
10	3,77	2,32	7,03	0,35	0	0,15	40,86	12,48	36,24	0,09	0	0,15
20	6,70	2,50	5,58	5,88	0	0,11	41,65	14,61	36,41	0	0	0,11
30	12,89	2,41	4,48	14,45	0	0,11	42,28	16,09	37,62	0	0	0,11
40	24,66	1,19	5,00	26,48	0,06	0	41,61	14,85	34,15	0,34	0,06	0
50	37,66	0,67	7,06	33,61	0	0	41,06	15,69	32,40	0,92	0	0
60	47,85	0,15	6,95	43,23	0	0,06	40,19	16,11	32,13	2,06	0	0,06
70	58,35	0,64	8,27	51,40	0,22	0	43,98	13,98	32,94	3,33	0,22	0
80	66,52	2,76	9,97	58,65	0	0	48,00	13,62	33,61	8,28	0	0
90	73,39	2,75	10,74	68,23	0,20	0	55,00	12,32	39,22	12,25	0,20	0
100	84,45	4,91	9,98	84,45	0	0	55,70	13,43	42,36	14,54	0	0
110	99,94	5,33	7,93	105,70	0	0,14	62,18	14,94	42,01	22,33	0	0,14
120	126,18	4,11	6,36	116,43	0	0,18	68,43	18,52	42,47	26,48	0	0,18
130	142,35	3,00	7,07	133,57	0	0	74,48	19,25	35,94	31,13	0	0
140	152,24	2,00	9,06	149,25	0	0	80,81	18,51	38,52	37,56	0	0
150	165,58	2,48	9,60	166,17	0	0	88,30	20,19	39,11	46,58	0	0
160	181,51	4,24	10,23	175,70	0,11	0	94,41	18,96	42,05	51,55	0,11	0
170	199,44	6,65	10,14	184,83	0	0	103,55	16,47	43,05	52,81	0	0
180	214,72	5,11	12,22	195,90	0	0	108,81	16,53	47,51	61,08	0	0
190	231,78	4,20	10,75	215,52	0	0,14	117,26	16,07	51,37	71,39	0	0,14
200	243,60	6,09	10,88	232,27	0	0	121,56	16,81	51,14	79,12	0	0
210	257,81	5,17	10,46	243,68	0,06	0	126,06	19,10	48,54	81,94	0,06	0
220	273,33	4,04	10,65	248,49	0	0	134,56	19,75	46,26	89,26	0	0
230	280,24	4,71	12,29	263,52	0	0,10	143,23	20,94	45,79	96,36	0	0,10
240	301,77	5,25	12,04	282,32	0	0,06	152,26	21,96	51,01	102,49	0	0,06
250	320,48	4,68	13,00	294,69	0,67	0,18	164,79	21,49	50,56	108,23	0,28	0,18
260	329,65	8,47	12,76	305,68	0,35	0	170,79	20,37	55,04	111,37	0	0
270	338,99	6,53	17,73	318,25	0,97	0	174,43	21,49	60,58	114,58	0,20	0
280	355,40	5,17	15,11	338,04	1,88	0,97	175,94	20,90	64,33	127,06	0	0
290	366,78	7,56	15,88	352,30	2,10	3,08	183,09	20,12	62,64	134,76	0	0
300	380,26	6,33	17,21	360,79	4,05	4,10	192,04	21,96	57,39	142,61	0	0
310	393,68	6,46	16,93	370,44	4,19	3,15	195,72	23,33	56,90	145,44	0	0
320	401,47	8,30	16,96	382,85	4,35	3,94	202,17	23,63	54,85	151,05	0,11	0
330	408,52	8,94	19,28	390,02	8,13	3,57	211,33	25,07	55,21	159,37	0	0
340	413,96	11,58	20,92	400,12	9,58	8,51	213,12	23,70	59,80	163,85	0,11	0,13
350	430,58	13,16	28,97	410,13	12,78	11,71	225,18	26,23	65,31	171,07	0,10	0
360	443,93	13,66	37,25	431,91	13,13	14,55	234,79	28,11	63,57	180,20	0	0,20

Таблиця Б.10

Результати стійкості алгоритму StegoBIT до атаки повороту (експерименти 25-36)

Номер перетворення	Експеримент											
	25	26	27	28	29	30	31	32	33	34	35	36
0	123,60	10,20	1,63	31,27	0	0	223,07	51,70	45,60	10,77	0	0
10	222,81	8,55	2,40	152,41	0	0,08	256,98	48,12	47,56	64,30	0	0,08
20	335,29	8,06	2,53	272,90	0	0	308,27	42,58	50,27	123,04	0	0
30	380	8,81	2,89	343,59	0	0,22	348,65	45,72	51,87	169,30	0	0,22
40	423,93	8,49	3,13	378,84	0,05	0	376,67	41,71	48,99	206,18	0,05	0
50	446,94	9,25	2,01	394,53	0	0	395,03	44,39	49,41	229,46	0	0
60	471,07	10,89	4,12	420,14	0	0	405,68	42,54	50,62	258,06	0	0
70	492,42	8,49	6,94	445,07	0	0	414,10	40,99	47,91	278,51	0	0
80	508,43	8,31	6,36	462,85	0	0,12	421,67	43,92	45,35	293,59	0	0,12
90	519,40	8,94	6,93	478,22	0,14	0	436,37	46,03	50,18	315,33	0,14	0
100	536,69	10,56	4,84	488,14	0,10	0,20	467,70	48,90	52,57	335,47	0,10	0,20
110	558,81	10,27	3,96	517,24	0	0	494,90	47,33	60,32	366,03	0	0
120	574,58	9,85	4,89	538,84	0	0	510,20	50,01	60,25	396,94	0	0
130	585,18	10,74	5,20	551,87	0,09	0	530,03	40,53	63,59	413,45	0	0
140	600,68	12,43	5,71	556,19	0,43	0	533,36	41,99	65,95	419,04	0	0
150	611,56	12,96	6,68	568,27	0,26	0,12	535,64	47,75	64,33	430,55	0	0,12
160	625,30	8,64	7,32	576,00	0,11	0,01	548,37	51,87	62,27	436,70	0	0,01
170	635,72	8,68	6,67	582,79	0,11	0,09	555,53	55,81	61,07	449,70	0	0,09
180	637,79	9,87	8,38	591,17	0	0	565,47	53,50	61,43	465,08	0	0
190	643,78	12,13	7,15	600,65	0,21	0	574,89	49,98	58,73	473,15	0	0
200	654,42	13,28	6,31	612,58	1,49	0,03	591,20	50,98	60,24	484,89	0	0,03
210	660,89	18,60	7,73	623,70	1,61	0	602,42	55,24	65,15	495,09	0,01	0
220	670,56	21,93	9,35	630,41	3,44	0	610,81	47,49	72,66	506,38	0	0
230	676,55	26,85	11,17	636,98	5,31	0	614,15	47,19	70,75	512,25	0,24	0
240	684,05	29,72	8,07	642,10	5,74	0,13	611,03	56,20	68,14	517,80	0,11	0,13
250	693,90	26,52	9,37	645,94	5,51	0,50	622,27	61,41	68,17	525,17	0,01	0,50
260	701,46	24,61	10,57	648,15	10,10	1,50	629,23	60,65	71,02	530,83	0	0
270	701,93	25,02	14,01	653,60	12,03	2,48	631,28	62,90	67,84	538,52	0,04	0,18
280	704,78	32,92	14,19	663,11	11,88	1,06	636,83	60,35	67,91	545,65	0,22	0,14
290	715,85	41,30	13,05	671,05	20,95	1,69	647,91	60	66,37	551,24	0	0
300	718,24	49,88	16,36	677,34	23,88	1,91	651,63	56,07	70,80	557,23	0	0,05

310	721,72	53,06	18,65	681,31	28,07	3,40	660,68	60,04	77,19	560,49	0	0
320	728,14	56,81	21,24	686,07	33,03	4,04	665,25	63,48	81,31	566,48	0	0,39
330	738,72	56,42	24,85	689,16	39,62	2,03	660,94	66,51	76,57	567,93	0,01	0
340	744,02	64,97	28,26	691,26	44,67	6,65	664,45	65,34	77,79	570,56	0	0
350	745,85	70,95	29,35	698,73	56,53	9,15	667,78	70,02	77,45	574,89	0,21	0,09
360	746,37	84,35	33,48	709,70	66,69	15,11	670,73	73,20	77,16	578,97	0	0

Таблиця Б.11

Результати стійкості алгоритму StegoTEMPL до атаки повороту (експерименти 1-12)

Номер перетворення	**Експеримент**											
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
0	1,17	8,97	1,23	0	0	0	15,97	27,77	6,10	0	0	0
10	1,00	7,46	0	0	0	0	17,03	26,38	6,69	0	0	0
20	0,23	5,65	0,39	0	0	0	17,34	29,03	7,37	0	0	0
30	1,39	5,88	0,26	0,13	0	0	21,05	29,50	9,63	0,13	0	0
40	5,23	10,37	0,12	0	0,13	0,12	22,23	30,42	10,23	0	0,13	0,12
50	8,14	9,78	0,24	0	0	0	21,55	31,65	10,95	0	0	0
60	8,71	8,88	0,39	0	0	0	19,84	30,90	8,14	0	0	0
70	7,79	10,53	1,40	0	0	0	17,96	30,78	9,46	0	0	0
80	6,59	10,27	0,26	0	0	0,13	18,69	29,46	9,81	0	0	0,13
90	5,23	10,36	0,13	0	0,25	0	19,25	30,74	9,64	0	0,25	0
100	3,60	11,47	0,26	0	0,12	0,13	19,21	35,80	10,02	0	0,12	0,13
110	6,00	14,03	0,26	0	0	0	20,60	36,57	9,12	0	0	0
120	6,68	13,27	0,13	0	0	0	23,74	38,07	10,04	0	0	0
130	11,20	15,16	0	0	0	0	28,07	41,91	11,30	0	0	0
140	13,00	14,57	0,79	0	0,13	0	25,66	38,04	12,34	0	0,13	0
150	12,71	14,72	2,65	0	0	0	24,89	39,55	12,03	0	0	0
160	12,89	14,91	1,92	0	0	0	24,65	39,37	11,45	0	0	0
170	11,47	13,37	1,17	0,13	0	0	25,10	38,69	11,70	0,13	0	0
180	10,60	13,00	0,59	0	0,12	0,12	24,18	42,50	12,45	0	0,12	0,12
190	11,70	14,61	0,48	0	0	0,13	24,35	49,16	10,53	0	0	0,13
200	12,81	17,73	1,00	0	0	0	26,34	50,32	11,43	0	0	0
210	14,04	15,38	0,50	0	0	0	29,21	49,07	12,51	0	0	0
220	15,18	17,35	3,00	0	0,13	0,12	30,08	48,61	12,60	0	0,13	0,12
230	14,97	17,75	5,06	0	1,07	0	29,60	50,59	12,66	0	0,13	0
240	14,68	19,03	6,15	0	1,33	0	30,02	51,30	13,91	0	0	0
250	14,67	16,30	5,29	0	0,27	0	28,45	50,42	16,44	0	0	0
260	16,25	15,00	4,06	0	0	0,13	29,54	52,49	14,69	0	0	0,13
270	16,85	16,25	2,33	0,12	0,27	0,12	30,21	54,46	14,65	0,12	0	0,12
280	17,95	20,42	3,64	0	0	0	31,48	53,31	12,56	0	0	0
290	18,12	19,50	3,86	0	0	1,37	33,46	54,11	13,92	0	0	0
300	18,51	19,27	6,70	0,25	0,48	2,13	35,09	58,27	15,32	0,13	0,12	0
310	20,07	22,31	9,10	0,36	2,27	2,57	36,61	60,82	15,62	0,13	0	0
320	19,90	23,50	10,94	2,32	4,53	4,87	38,19	57,33	12,71	0,12	0	0
330	20,73	25,32	10,49	6,24	6,28	5,07	34,66	59,14	15,59	0,13	0	0
340	21,62	25,40	13,41	5,58	8,89	4,21	32,74	61,39	18,48	0	0	0
350	22,24	22,84	13,11	6,19	9,08	5,26	37,07	61,00	15,49	0	0	0
360	27,70	29,57	12,59	8,04	8,62	5,67	40,20	60,87	14,40	0	0	0,12

Таблиця Б.12

Результати стійкості алгоритму StegoTEMPL до атаки повороту (експерименти 13-24)

Номер перетворення	**Експеримент**											
	13	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
0	6,50	16,83	0	0	0	0	63,10	74,17	20,80	0	0	0
10	7,42	13,43	0,06	0	0	0	59,95	69,64	18,03	0	0	0
20	7,02	14,04	1,43	0	0	0,20	59,71	66,68	16,22	0	0	0,20
30	7,61	14,06	2,14	0	0	0	63,54	69,73	17,72	0	0	0
40	8,85	15,01	1,68	0	0	0	62,47	71,19	19,09	0	0	0
50	11,93	15,10	2,11	0	0	0	64,76	72,08	16,33	0	0	0
60	14,90	17,29	2,07	0	0	0,21	66,79	70,38	19,50	0	0	0,21
70	14,72	19,39	1,32	0,21	0	0	62,81	72,25	16,90	0,21	0	0
80	13,91	20,17	2,50	0,21	0,21	0	65,00	75,10	20,15	0,21	0,21	0
90	14,14	20,90	1,54	0	0	0	67,19	74,15	21,88	0	0	0
100	12,51	22,30	1,76	0	0	0	67,77	76,67	19,39	0	0	0
110	8,65	24,37	2,39	0	0	0	64,44	79,99	16,88	0	0	0
120	11,68	23,88	1,37	0	0	0	69,32	77,63	18,16	0	0	0
130	11,93	21,16	2,84	0	0	0	69,57	78,78	21,47	0	0	0
140	15,68	21,97	4,18	0	0	0	71,88	75,91	21,12	0	0	0
150	17,55	25,86	2,36	0	0	0	70,32	79,73	20,75	0	0	0
160	19,45	24,12	4,43	0	0	0,21	70,63	83,83	17,65	0	0	0,21
170	19,37	22,69	6,93	0	0,21	0	75,77	87,26	21,13	0	0,21	0
180	18,28	23,14	5,10	0	0,20	0	76,02	87,99	21,73	0	0,20	0
190	17,41	25,31	4,77	0,21	0	0	73,34	90,77	23,41	0,21	0	0
200	16,27	27,74	6,32	0	0	0	76,02	90,71	22,67	0	0	0
210	17,78	28,08	6,56	0	0	0,20	81,81	89,70	21,89	0	0	0,20

220	21,45	23,53	7,36	0	0	0	79,14	88,47	23,21	0	0	0
230	25,76	27,54	7,15	0	0	0	79,21	90,29	23,36	0	0	0
240	24,07	29,87	6,16	0,21	0	0	77,53	94,87	24,80	0,21	0	0
250	24,89	26,87	9,21	1,22	0	0	79,67	96,27	21,67	0,20	0	0
260	28,48	29,55	8,93	2,24	0	0,21	79,94	95,98	22,73	0,21	0	0,21
270	25,30	29,65	11,33	1,63	1,21	1,05	82,70	98,70	25,86	0	0	0
280	26,07	30,89	10,96	0	2,20	0	85,92	99,41	25,39	0	0	0
290	28,62	33,51	8,27	1,07	4,03	1,03	87,40	102,43	24,95	0	0	0
300	31,45	33,16	10,26	3,62	3,23	1,84	91,00	102,16	28,44	0	0	0
310	32,71	35,52	8,99	10,09	2,01	0,84	88,96	102,26	28,34	0,21	0	0
320	38,13	41,24	10,17	12,82	1,42	0	90,53	98,54	27,22	0	0	0
330	37,97	43,36	13,53	8,89	4,64	1,65	90,20	104,21	27,42	0	0	0
340	43,72	46,46	21,03	10,68	8,95	5,82	90,48	108,16	25,16	0,20	0	0,21
350	47,82	49,38	27,40	17,81	7,69	11,47	90,62	103,30	28,56	0	0	0,21
360	48,69	55,42	27,47	22,25	9,37	15,35	96,28	105,53	28,76	0	0	0

Таблиця Б.13

Результати стійкості алгоритму StegoTEMPL до атаки повороту (експерименти 25-36)

Номер перетворення	**Експеримент**											
	25	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**	**36**
0	19,37	24,03	0	0	0	0	125,53	113,67	35,33	0	0	0
10	21,89	16,38	0,52	0	0	0	128,35	105,35	32,75	0	0	0
20	16,41	15,62	0,26	0	0	0	129,48	101,54	25,88	0	0	0
30	17,16	20,11	0,95	0	0	0	128,68	108,53	32,88	0	0	0
40	20,51	19,63	1,21	0	0	0,26	124,96	110,50	35,01	0	0	0,26
50	34,08	24,08	0,31	0	0,26	0	119,99	111,18	35,46	0	0,26	0
60	39,14	26,93	1,07	0	0	0	122,17	106,47	30,18	0	0	0
70	33,81	32,77	0,79	0	0	0	124,26	106,76	36,28	0	0	0
80	33,88	32,69	0,27	0,27	0	0	123,15	114,24	40,40	0,27	0	0
90	34,22	39,61	0,26	0	0	0	130,75	113,42	41,47	0	0	0
100	29,30	33,84	1,77	0	0	0	136,52	113,33	38,15	0	0	0
110	24,76	27,86	1,06	0	0	0	133,44	110,43	37,69	0	0	0
120	28,34	32,97	0,79	0	0	0,26	130,66	118,28	38,93	0	0	0,26
130	36,06	36,06	1,56	0	0	0,26	133,05	124,08	38,52	0	0	0,26
140	45,12	37,00	6,10	0	0	0	122,88	122,52	43,31	0	0	0
150	49,96	40,29	7,18	0	0	0	128,18	121,01	43,07	0	0	0
160	48,79	37,83	3,52	0	0,26	0	127,67	119,05	43,80	0	0,26	0
170	49,75	39,36	1,26	0	0	0	133,92	122,54	48,20	0	0	0
180	49,18	43,23	2,22	2,05	0	0	143,93	119,20	44,86	0	0	0
190	41,48	40,49	2,20	2,83	0	0	147,05	117,95	40	0	0	0
200	52,33	40,60	5,20	3,63	0	0	145,46	117,54	34,99	0	0	0
210	60,05	41,78	9,24	0,52	0	0,78	139,78	127,72	38,30	0	0	0
220	61,02	43,22	10,23	4,92	0	1,56	140,50	132,95	44,75	0	0	0
230	66,96	43,90	10,09	11,05	0	1,30	142,87	135,48	50,97	0	0	0
240	74,48	46,99	9,13	9,29	0	2,08	139,54	132,88	44,77	0	0	0
250	78,65	48,40	11,21	7,17	0,79	1,30	140,44	132,40	45,48	0	0	0
260	81,46	48,24	8,68	14,36	5,24	0	149,78	125,99	44,07	0	0	0
270	87,78	50,34	10,27	19,71	2,63	0,26	156,25	125,19	46,53	0	0	0
280	96,63	50,67	9,53	31,04	4,12	3,88	159,20	125,68	47,90	0	0	0
290	107,36	52,83	15,31	41,84	6,16	2,60	153,82	128,18	42,24	0	0	0
300	102,78	55,65	15,79	51,14	10,88	2,60	147,37	138,25	46,60	0,50	0	0
310	106,59	57,28	17,95	53,57	11,27	4,47	145,80	146,15	52,91	0	0	0
320	113,87	58,87	18,33	59,81	11,44	10,35	152,08	141,56	56,42	0	0	0
330	128,04	68,57	24,88	64,88	16,24	15,03	149,71	141,80	51,72	0	0	0
340	139,63	67,88	30,74	87,97	16,32	15,88	152,26	136,85	48,25	0	0	0
350	159,86	72,96	35,37	95,42	18,71	20,07	162,22	138,33	47,69	0	0,43	0
360	176,04	79,58	41,39	105,35	27,85	27,81	167,16	134,68	48,14	0,77	0,26	0,25

Результати стійкості алгоритмів StegoBIT та StegoTEMPL до атаки майже повороту представлені в табл. Б.14-Б.16 та табл. Б.17-Б.19 відповідно.

Таблиця Б.14

Результати стійкості алгоритму StegoBIT до атаки майже повороту (експерименти 1-12)

Номер перетворення	**Експеримент**											
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
0	0	1,13	1,33	0	0	0	0	3,43	24,13	0	0	0
10	0,03	0,20	1,12	0	0,11	0	0,03	2,57	23,32	0	0,11	0
20	0	0,13	0,46	0,12	0	0	0	3,84	23,48	0,12	0	0
30	0,22	0,18	0,28	0	0	0	0,22	1,90	23,79	0	0	0
40	0,21	0,12	2,27	0	0	0	0,21	1,84	23,69	0	0	0
50	0,43	0	3,45	0	0	0	0,43	2,63	24,81	0	0	0
60	0,01	0,33	4,46	0	0	0	0,01	1,91	24,72	0	0	0
70	0,11	0,39	4,75	0	0	0,12	0,11	3,62	24,15	0	0	0,12
80	0,20	0,14	6,22	0	0	0	0,20	4,40	22,85	0	0	0
90	0,13	0,15	5,36	0	0	0	0,13	3,69	23,14	0	0	0

100	0,28	0,20	4,00	0	0	0	0,28	3,45	24,19	0	0	0
110	0,02	0	3,62	0	0	0,16	0,02	4,95	26,77	0	0	0,16
120	0,14	0,07	5,31	0,01	0	0	0,14	3,39	25,79	0,01	0	0
130	0,16	0,30	4,53	0	0	0	0,16	3,59	25,34	0	0	0
140	0,09	0,28	6,78	0	0	0	0,09	2,89	26,25	0	0	0
150	0,23	0,26	8,81	0	0	0	0,23	2,97	25,65	0	0	0
160	0,08	0,10	8,74	0,02	0	0,07	0,08	4,78	24,59	0,02	0	0,07
170	0,24	0,14	8,33	0	0	0	0,24	6,50	24,39	0	0	0
180	0,32	0,19	8,15	0	0	0,01	0,32	5,38	25,15	0	0	0,01
190	0,37	0,03	9,48	0,03	0,01	0	0,37	3,64	24,27	0,03	0,01	0
200	0,25	0,43	9,87	0	0	0	0,25	5,52	26,58	0	0	0
210	0	0,11	6,84	0	0	0	0	5,04	27,06	0	0	0
220	0,09	0,50	6,17	0,06	0	0	0,09	5,35	30,64	0,06	0	0
230	0,20	0,09	6,92	0	0	0	0,20	3,81	29,37	0	0	0
240	0,10	0,12	9,20	0	0,03	0	0,10	4,21	27,88	0	0,03	0
250	0,35	0,15	9,54	0	0	0,11	0,35	6,02	27,16	0	0	0,11
260	0,28	0,03	8,60	0,12	0	0,09	0,28	6,30	27,51	0,12	0	0,09
270	0,48	0,29	8,65	0,09	0	0	0,48	5,68	26,55	0,09	0	0
280	0,17	0,11	9,43	0	0	0	0,17	5,76	27,61	0	0	0
290	0,14	0,32	8,56	0	0,07	0	0,14	5,47	27,70	0	0,07	0
300	0,51	0,14	8,62	0	0,03	0,10	0,51	5,91	28,46	0	0,03	0,10
310	0,08	0,14	8,31	0,10	0	0	0,08	5,64	30,55	0,10	0	0
320	0,28	0,25	9,26	0	0,12	0	0,28	4,79	31,95	0	0,12	0
330	0,19	0,16	10,16	0	0	0,04	0,19	5,18	32,64	0	0	0,04
340	0,34	0,22	9,67	0,11	0	0	0,34	6,99	31,15	0,11	0	0
350	0	0,20	9,36	0	0	0	0	6,65	29,97	0	0	0
360	0,15	0,37	9,95	0	0,09	0	0,15	7,46	28,90	0	0,09	0

Таблиця Б.15

Результати стійкості алгоритму StegoBIT до атаки майже повороту (експерименти 13-24)

Номер перетворення	**Експеримент**											
	13	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
0	5,77	0	7,90	0	0	0	45,37	11,53	34,43	0	0	0
10	1,95	0,14	6,87	0,11	0	0,20	40,86	12,46	36,37	0	0	0,20
20	4,63	2,28	6,44	4,42	0	0	38,51	14,49	35,99	0	0	0
30	12,88	2,26	5,30	11,90	0	0	41,36	16,02	40,39	0,24	0	0
40	29,12	1,47	5,68	22,85	0,15	0	40,52	12,83	33,85	1,85	0,15	0
50	38,64	0,39	6,45	35,06	0	0,14	38,86	13,56	31,56	3,70	0	0,14
60	50,80	1,93	6,79	44,80	0	0	39,18	13,80	31,92	7,98	0	0
70	60,91	0,67	6,35	61,30	0,20	0,17	47,12	13,99	30,48	11,39	0,20	0,17
80	73,05	0,57	8,10	68,20	0	0	53,17	13,22	36,91	12,88	0	0
90	83,03	1,83	8,69	79,40	0	0	60,42	14,24	38,92	16,88	0	0
100	91,35	3,90	10,12	87,61	0,37	0	64,70	14,79	42,66	14,22	0,37	0
110	99,77	2,19	9,73	94,26	0	0,21	65,92	17,97	44,48	16,95	0	0,21
120	112,31	2,21	10,27	105,07	0,12	0	61,68	18,40	47,30	20,10	0,12	0
130	113,70	2,07	9,30	113,70	0,09	0	59,20	15,87	42,27	23,95	0,09	0
140	117,36	2,28	9,82	117,49	0	0	59,07	14,71	37,49	24,98	0	0
150	124,41	1,60	10,68	120,16	0	0,14	67,55	15,80	35,08	29,81	0	0,14
160	129,83	0,19	10,96	128,68	0	0	76,98	16,07	35,24	31,23	0	0
170	134,06	0,77	10,72	134,48	0	0	80,86	16,96	41,82	32,16	0	0
180	137,55	3,99	11,33	137,65	0	0	81,48	17,15	44,79	37,28	0	0
190	145,43	3,78	13,32	140,93	0	0	79,80	16,04	48,62	41,26	0	0
200	149,12	2,38	13,00	149,33	0	0,09	82,05	18,57	46,65	41,78	0	0,09
210	150,58	3,97	11,84	154,96	0,11	0	79,90	17,05	47,80	46,14	0,11	0
220	150,39	2,97	11,01	165,50	0	0	82,67	16,44	46,19	46,56	0	0
230	157,55	2,45	11,89	172,54	0	0,11	82,74	16,16	41,19	50,13	0	0,11
240	165,80	2,29	12,77	176,47	0	0	85,85	15,56	41,45	51,33	0	0
250	176,47	2,88	14,28	180,75	0	0,19	91,32	17,08	43,43	51,71	0	0,19
260	184,02	0,78	12,91	188,49	0	0	94,50	15,72	45,93	51,50	0	0
270	190,08	2,34	12,65	188,06	0,11	0	99,75	17,73	49,22	52,03	0,11	0
280	186,39	1,62	13,78	192,67	0	0,15	103,65	18,45	51,35	54,19	0	0,15
290	191,53	3,34	14,35	195,17	0,16	0,18	98,95	16,80	53,31	54,07	0,16	0,18
300	190,34	3,15	13,62	201,85	0	0	98,59	17,13	49,80	57,75	0	0
310	193,87	2,74	13,27	203,94	0,21	0	99,63	17,55	48,85	59,37	0,21	0
320	198,96	3,31	12,86	207,97	0	0	100,29	16,01	48,71	64,31	0	0
330	207,71	4,30	13,93	209,82	0	0,31	102,92	15,80	48,08	69,45	0	0,31
340	213,56	3,45	14,51	215,33	0	0	107,20	15,64	47,89	71,84	0	0
350	221,66	2,74	13,12	219,24	0	0	116,20	18,22	50,55	78,61	0	0
360	226,56	4,13	13,63	227,38	0,18	0	118,41	18,98	56,14	77,69	0,18	0

Таблиця Б.16

Результати стійкості алгоритму StegoBIT до атаки майже повороту (експерименти 25-36)

Номер перетворення	**Експеримент**											
	25	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**	**36**
0	123,60	10,20	1,63	31,27	0	0	223,07	51,70	45,60	10,77	0	0

10	232,46	7,99	2,35	155,51	0	0	257,24	48,25	46,50	64,89	0	0
20	339,13	5,83	2,74	268,93	0,20	0	311,67	45,57	47,92	125,57	0,20	0
30	384,82	9,35	2,81	338,51	0	0	349,30	48,48	51,48	174,40	0	0
40	426,42	10,75	2,21	378,46	0	0,09	372,94	46,84	49,60	210,88	0	0,09
50	456,39	9,27	2,24	408,47	0	0	400,80	44,54	53,76	243,38	0	0
60	466,19	11,16	3,92	429,10	0,16	0,09	411,86	42,42	53,11	266,55	0,16	0,09
70	489,38	11,15	5,13	445,92	0,12	0,20	421,20	43,35	50,19	282,69	0,12	0,20
80	508,60	11,41	4,68	466,80	0	0	429,23	49,24	53,38	305,91	0	0
90	525,28	13,35	4,19	481,73	0	0	443,87	55,03	50,68	320,40	0	0
100	530,71	9,71	5,24	490,10	0	0	456,55	52,98	49,45	332,67	0	0
110	544,71	12,63	4,34	507,62	0	0,25	474,26	51,75	51,99	351,26	0	0,25
120	554,64	11,41	4,17	514,91	0,12	0	490,90	52,76	53,10	358,44	0,12	0
130	570,07	12,61	4,86	521,76	0	0	502,69	46,04	55,58	368,53	0	0
140	573,21	13,39	5,39	526,84	0	0,25	505,95	48,18	57,17	370,16	0	0,25
150	576,13	13,15	7,27	535,37	0	0	502,14	46,15	63,65	382,22	0	0
160	584,63	14,82	6,79	537,91	0,18	0	508,38	54,25	58,49	387,31	0,18	0
170	595,08	12,92	6,37	537,62	0	0,01	521,59	58,76	57,72	394,36	0	0,01
180	602,59	10,85	6,73	546,33	0	0	526,72	59,93	57,34	400,30	0	0
190	601,26	12,67	8,47	542,16	0	0	533,09	59,59	59,06	397,16	0	0
200	608,85	13,27	8,00	557,63	0,22	0,01	537,44	57,35	60,69	409,77	0,22	0,01
210	616,82	10,51	5,96	569,38	0,23	0	545,06	57,92	61,38	417,62	0,23	0
220	620,90	11,43	5,10	570,50	0	0	549,54	58,23	59,49	429,50	0	0
230	620	14,18	4,99	577,03	0	0	552,19	57,15	62,77	433,04	0	0
240	623,72	13,35	5,21	583,13	0,69	0	553,95	55,96	63,19	438,89	0,38	0
250	626,77	13,41	6,20	588,95	0,62	0	548,57	58,04	62,98	445,94	0	0
260	634,90	13,32	5,96	588,65	0,62	0,03	548,79	61,03	58,30	452,70	0	0,03
270	634,32	12,58	7,83	589,14	2,17	0	555,44	57,11	61,92	453,40	0	0
280	634,92	11,42	8,87	592,38	3,10	0,28	561,02	56,99	61,29	453,93	0	0,28
290	640,46	11,80	7,23	593,14	2,87	0	569,48	58,93	63,13	462,05	0	0
300	642,13	12,13	6,13	591,21	2,65	0,14	574,74	58,60	60,98	463,84	0	0,14
310	641,89	12,87	7,77	593,07	1,96	0	577,56	65,10	66,38	455,60	0	0
320	643,90	13,05	7,10	607,86	2,59	0,06	578,94	60,72	66,21	465,36	0	0,06
330	645,00	12,16	5,69	607,63	2,52	0	575,68	58,70	62,65	469,47	0,18	0
340	653,84	15,04	8,32	610,86	1,55	0,03	575,76	62,68	62,48	473,67	0	0,03
350	661,85	16,31	8,34	618,58	1,58	0	585,67	61,52	64,22	481,46	0,17	0
360	662,45	15,44	9,55	621,01	2,17	0	591,64	68,02	65,48	485,50	0	0

Таблиця Б.17

Результати стійкості алгоритму StegoTEMPL до атаки майже повороту (експерименти 1-12)

Номер перетворення	**Експеримент**											
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
0	1,17	8,97	1,23	0	0	0	15,97	27,77	6,10	0	0	0
10	0,91	8,24	0,26	0	0	0	16,55	27,01	6,75	0	0	0
20	0,61	6,51	0,13	0	0	0	17,58	29,04	7,22	0	0	0
30	2,82	6,98	0,39	0,12	0	0	21,32	31,43	9,42	0,12	0	0
40	8,01	10,77	0,39	0	0	0	23,05	30,63	9,98	0	0	0
50	7,30	10,64	0,26	0,24	0,11	0,11	22,01	32,30	11,04	0,24	0,11	0,11
60	7,34	9,96	0	0	0,13	0	20,63	31,41	7,90	0	0,13	0
70	8,72	11,99	1,46	0	0,13	0	18,66	33,09	9,18	0	0,13	0
80	6,83	11,26	0,76	0	0	0	18,26	30,91	9,78	0	0	0
90	6,67	10,70	0,13	0	0	0	19,71	29,26	10,94	0	0	0
100	5,66	10,77	0	0	0	0	21,03	30,68	10,15	0	0	0
110	6,15	11,96	0,12	0	0,12	0	22,21	33,74	7,83	0	0,12	0
120	6,91	13,33	0,13	0	0	0,13	22,92	33,29	10,30	0	0	0,13
130	7,26	12,81	0,26	0	0	0	24,30	34,38	11,21	0	0	0
140	8,77	12,57	1,65	0	0	0	26,80	39,05	11,89	0	0	0
150	10,81	14,00	1,59	0	0	0	27,28	38,96	11,78	0	0	0
160	11,00	15,01	2,64	0,11	0,13	0	23,03	38,43	12,18	0,11	0,13	0
170	10,74	14,26	1,48	0	0	0	24,43	36,93	12,61	0	0	0
180	8,92	12,53	0,33	0	0,25	0	20,41	35,94	12,20	0	0,25	0
190	9,99	12,22	0,83	0	0	0	19,79	35,70	10,42	0	0	0
200	10,17	12,24	1,28	0	0	0,13	22,57	37,44	10,46	0	0	0,13
210	8,63	12,32	0,89	0	0	0	24,90	40,17	12,07	0	0	0
220	10,23	12,32	1,63	0	0	0	25,77	41,45	12,42	0	0	0
230	9,46	14,68	3,33	0	0	0	29,87	43,26	11,04	0	0	0
240	11,41	16,14	3,84	0	0	0	30,10	41,99	13,03	0	0	0
250	10,96	17,03	4,47	0,23	0	0	28,02	39,74	13,75	0,23	0	0
260	11,46	16,83	3,50	0	0	0,13	27,18	42,82	12,55	0	0	0,13
270	10,58	16,12	2,14	0	0	0,13	24,68	42,52	12,09	0	0	0,13
280	11,18	16,16	2,37	0	0	0	24,63	43,01	11,16	0	0	0
290	8,94	16,07	0,50	0	0	0	24,94	42,06	9,81	0	0	0
300	9,96	14,29	1,91	0	0	0	28,28	43,86	11,57	0	0	0
310	12,46	14,14	3,09	0,13	0	0	30,50	47,24	13,07	0,13	0	0
320	13,19	16,10	5,66	0	0	0	30,56	47,44	15,30	0	0	0
330	12,28	15,08	5,79	0	0	0	30,23	46,87	14,87	0	0	0
340	12,05	16,35	4,92	0	0	0	28,22	47,24	14,02	0	0	0

350	11,38	16,09	4,27	0,12	0,13	0	28,39	46,21	13,54	0,12	0,13	0
360	11,77	16,48	3,14	0	0	0	27,07	46,62	12,30	0	0	0

Таблиця Б.18

Результати стійкості алгоритму StegoTEMPL до атаки майже повороту (експерименти 13-24)

Номер перетворення	**Експеримент**											
	13	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
0	6,50	16,83	0	0	0	0	63,10	74,17	20,80	0	0	0
10	7,63	14,35	0,28	0	0	0	60,45	70,14	18,03	0	0	0
20	8,00	15,08	0,91	0	0	0	60,62	68,27	16,26	0	0	0
30	7,81	13,22	0,43	0	0	0,21	61,91	73,22	17,49	0	0	0,21
40	13,09	13,00	0,57	0	0	0	63,16	69,46	16,42	0	0	0
50	13,87	14,41	0,34	0,21	0,21	0,21	69,84	68,07	16,21	0,21	0,21	0,21
60	15,27	16,44	0,77	0	0	0	70,15	67,45	17,91	0	0	0
70	13,87	15,95	1,57	0	0	0	67,60	74,31	16,96	0	0	0
80	15,65	17,13	1,72	0,20	0	0	68,30	74,70	19,14	0,20	0	0
90	13,38	15,26	0,87	0	0	0	68,85	81,84	23,47	0	0	0
100	14,29	15,19	0,96	0	0	0	66,60	80,70	23,92	0	0	0
110	12,68	16,75	1,74	0	0	0	66,71	79,79	21,01	0	0	0
120	15,38	16,14	1,10	0	0	0	66,29	79,69	19,02	0	0	0
130	15,58	15,72	2,80	0	0	0	68,12	75,91	20,68	0	0	0
140	15,67	18,23	3,87	0	0	0	72,42	79,71	18,72	0	0	0
150	16,08	21,61	1,30	0,21	0	0	73,09	77,20	18,91	0,21	0	0
160	19,21	21,37	4,49	0	0,21	0	73,10	80,46	19,01	0	0,21	0
170	19,09	20,23	1,08	0	0	0	73,51	82,99	20	0	0	0
180	20,64	22,77	1,50	0	0,21	0	75,00	88,54	22,61	0	0,21	0
190	21,49	21,59	1,99	0	0,61	0	74,24	86,94	23,59	0	0,61	0
200	19,73	21,21	1,88	0	0	0	72,95	85,97	20,17	0	0	0
210	16,74	20,80	1,30	0	0	0	70,49	89,19	20,20	0	0	0
220	18,39	21,80	2,14	0	0	0	70,31	82,07	22,11	0	0	0
230	17,77	23,92	2,95	0	0	0	77,24	84,38	20,32	0	0	0
240	17,79	23,99	1,78	0	0	0	74,41	82,56	20,43	0,21	0	0
250	23,15	23,07	5,14	0	0,21	0	75,89	82,94	21,96	0	0,21	0
260	26,43	23,12	4,11	0	0	0,21	80,91	88,62	22,81	0	0	0,21
270	26,24	23,05	2,27	0	0	0	79,22	92,36	24,33	0	0	0
280	20,73	23,09	2,73	0	0	0	81,79	92,52	22,02	0	0	0
290	21,22	21,06	2,34	0	0	0,21	75,36	89,89	22,14	0	0	0,21
300	21,17	22,26	2,65	0	0	0	77,86	88,61	21,70	0	0	0
310	21,17	23,81	2,89	0	0	0	74,85	89,81	24,87	0	0	0
320	24,89	26,72	4,78	0	0	0	76,73	93,78	23,60	0	0	0
330	27,08	25,12	5,70	0	0	0	78,04	91,54	23,72	0	0	0
340	25,11	28,54	5,70	0,21	0	0	80,66	87,40	22,61	0,21	0	0
350	25,05	25,10	3,56	0	0	0	88,72	90,99	24,93	0	0	0
360	25,90	27,13	6,05	0	0,21	0	84,29	93,29	23,08	0	0,21	0

Таблиця Б.19

Результати стійкості алгоритму StegoTEMPL до атаки майже повороту (експерименти 25-36)

Номер перетворення	**Експеримент**											
	25	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**	**36**
0	19,37	24,03	0	0	0	0	125,53	113,67	35,33	0	0	0
10	24,17	21,92	0,27	0	0	0	127,44	107,00	30,54	0	0	0
20	16,62	15,41	0,53	0	0	0,27	133,65	107,07	27,74	0	0	0,27
30	16,10	16,08	0,95	0	0,26	0	131,58	116,70	34,64	0	0,26	0
40	21,12	16,93	0,18	0	0	0	121,21	110,13	33,08	0	0	0
50	29,77	19,12	0,77	0	0	0	119,23	107,21	32,31	0	0	0
60	31,74	27,51	1,06	0	0	0	117,93	101,26	33,40	0	0	0
70	32,88	30,05	0,90	0	0	0,24	120,95	106,26	32,99	0	0	0,24
80	30,82	30,10	0,21	0	0	0	125,19	111,68	34,85	0	0	0
90	30,93	27,28	0,26	0	0	0	126,37	114,94	36,52	0	0	0
100	28,99	28,13	0,52	0	0	0	132,24	114,86	33,49	0	0	0
110	26,47	26,17	0,53	0,26	0	0	137,39	119,63	33,68	0,26	0	0
120	25,16	26,73	2,01	0	0	0	134,98	119,04	38,29	0	0	0
130	29,22	25,96	1,48	0	0	0	133,42	118,80	37,35	0	0	0
140	38,98	31,10	0,31	0	0,21	0	126,81	118,24	34,97	0	0,21	0
150	39,90	29,35	2,45	0	0,26	0	126,94	110,76	37,09	0	0,26	0
160	35,74	31,07	1,73	0,78	0	0	132,13	114,26	38,00	0,78	0	0
170	33,53	32,15	0,25	0	0	0	131,80	119,76	39,04	0	0	0
180	33,43	35,34	1,74	0,26	0	0	135,09	119,46	42,77	0,26	0	0
190	35,15	30,55	0,84	0	0,26	0	132,21	124,20	38,93	0	0,26	0
200	33,50	28,34	1,48	0	0	0	138,67	128,89	33,94	0	0	0
210	34,62	32,22	3,28	0	0	0	142,33	123,61	38,57	0	0	0
220	39,82	33,37	3,20	0	0	0	136,92	122,93	38,38	0	0	0
230	38,31	35,92	3,96	0	0	0	129,08	126,60	41,00	0	0	0
240	37,04	38,48	4,19	0	0	0	127,10	126,77	38,11	0	0	0
250	36,17	41,33	3,15	0	0	0	128,28	123,18	41,82	0	0	0

260	34,52	40,16	2,68	0	0	0	131,06	126,97	39,01	0	0	0
270	41,78	42,87	4,96	0	0	0	134,52	122,85	44,62	0	0	0
280	46,84	41,00	5,25	0,27	0	0	136,69	128,36	39,89	0,27	0	0
290	48,77	39,34	3,11	0,26	0,27	0	142,79	129,81	38,14	0	0,27	0
300	51,01	38,92	4,74	0	0	0,26	146,12	127,98	45,90	0	0	0,26
310	48,82	37,65	4,03	0	0	0,48	139,85	124,56	45,83	0	0	0,48
320	43,60	39,79	8,99	0	0	0	129,60	129,80	44,97	0	0	0
330	43,73	42,41	5,72	0	0	0	129,76	127,58	43,70	0	0	0
340	41,16	44,21	6,66	0,26	0	0,26	132,52	129,16	41,32	0,26	0	0,26
350	45,57	45,86	6,64	0,27	0	0	134,20	134,51	40,44	0,27	0	0
360	50,38	48,77	6,11	0	0	0	138,68	129,60	44,49	0	0	0

Результати стійкості алгоритмів StegoBIT та StegoTEMPL до атаки пропорційного масштабування представлені в табл. Б.20-Б.22 та табл. Б.23-Б.25 відповідно.

Таблиця Б.20

Результати стійкості алгоритму StegoBIT до атаки пропорційного масштабування (експерименти 1-12)

Номер перетворення	**Експеримент**											
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
	Стиснення											
0	0	0,51	1,77	0	0	0	0	2,99	24,52	0	0	0
5	0,15	0,03	3,66	0	0,24	0	0,15	3,29	25,94	0	0,24	0
10	0,13	0,14	5,77	0	0,15	0	0,13	6,80	29,40	0	0,15	0
15	0,29	0	7,09	0	0	0	0,34	9,33	31,16	0	0	0
20	0,25	0,63	9,57	0	0	0	0,25	13,12	33,21	0	0	0
25	0,31	0,23	10,99	0	0	0	0,32	14,57	38,45	0	0	0
30	0,25	0,02	11,37	0,23	0	0,15	0,83	15,87	45,58	0,23	0	0,15
35	0,15	0,25	13,96	0	0,25	0	1,49	17,51	49,95	0	0,25	0
40	0,44	0	16,67	0	0	0	1,77	18,07	56,52	0	0	0
45	0,35	0,24	17,11	0	0	0,23	2,88	19,99	64,83	0	0	0,23
50	0	1,20	20,75	0	0	0,21	4,35	23,24	70,15	0	0	0,21
55	0,15	1,58	21,35	0	0	0	5,29	24,50	77,03	0	0	0
60	0,31	2,53	23,19	0	0	0	6,41	25,98	90,66	0	0	0
65	0,29	3,73	25,79	0	0	0	8,73	28,01	104,45	0	0	0
70	0,15	5,24	27,09	0	0	0	10,07	30,93	116,73	0	0	0
75	0,03	9,11	29,06	0,05	0	0	12,39	34,33	133,05	0,05	0	0
80	0	12,37	30,99	0	0	0,18	12,70	38,93	144,33	0	0	0,18
85	0,24	13,83	38,16	0	0	0	15,41	45,05	155,95	0	0	0
90	0	14,42	42,45	0	0,16	0	17,19	49,71	171,23	0	0,16	0
95	0,11	14,56	49,65	0,05	0	0	19,98	58,44	181,64	0,05	0	0
	Розширення											
0	0	0,51	1,77	0	0	0	0	2,99	24,52	0	0	0
5	0,44	0,11	1,29	0	0	0	0,44	2,35	22,00	0	0,11	0
10	0,27	0	1,07	0	0	0	0,27	1,39	20,38	0	0	0
15	0,30	0	0,14	0	0	0	0,30	1,20	18,67	0	0	0
20	0	0	0,17	0	0	0	0	1,16	14,76	0	0	0
25	0,11	0,36	0,24	0	0	0	0,37	1,24	13,89	0	0	0
30	0,40	0,23	0,41	0	0	0	0,54	0,23	12,72	0	0	0
35	0,76	0,19	0,41	0	0	0	0,76	0,19	12,09	0	0	0
40	0,12	0	0,23	0,19	0	0	0,17	0	11,32	0,19	0	0
45	0,21	0,01	0,33	0	0	0	0,47	0,17	9,51	0	0	0
50	0,53	0,57	0,14	0	0	0	0,53	0,57	7,91	0	0	0
55	0,29	0,25	0,26	0	0	0	0,39	0,37	7,11	0	0	0
60	0,45	0,48	0	0	0	0	0,51	0,58	5,31	0	0	0
65	0,63	0,11	0,57	0,19	0	0	0,75	0,25	4,40	0,19	0	0
70	0,66	0,35	0,33	0	0	0,21	0,82	0,37	3,21	0	0	0,21
75	0,56	0	0,05	0,23	0	0	0,61	0	2,32	0,23	0	0
80	0,47	0,31	0,36	0	0,11	0	0,62	0,31	1,33	0	0,11	0
85	0,18	0,23	0,18	0,09	0	0	0,18	0,23	0,92	0,09	0	0
90	0,17	0,15	0	0	0	0	0,17	0,15	0	0	0	0
95	0,41	0,35	0,16	0,15	0	0,14	0,45	0,45	0,30	0,15	0	0,14
100	0,33	0,59	0,20	0,09	0	0	0,33	0,59	0,48	0,09	0,17	0

Таблиця Б.21

Результати стійкості алгоритму StegoBIT до атаки пропорційного масштабування (експерименти 13-24)

Номер перетворення	**Експеримент**											
	13	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
	Стиснення											
0	6,34	0	8,09	0	0	0	47,21	11,61	38,59	0	0	0
5	6,53	3,29	9,00	0,20	0	0	54,09	12,60	44,69	0	0	0
10	11,87	4,49	9,63	2,21	0,02	0	63,65	14,44	50,97	0	0,02	0
15	16,30	4,13	11,50	5,92	0	0	66,61	20,16	61,31	0,16	0	0

20	18,49	4,41	12,28	7,17	0	0,23	67,45	25,41	70,29	0,52	0	0,23
25	22,83	4,32	13,97	9,60	0	0	71,79	31,67	77,87	0,34	0	0
30	24,78	5,97	15,55	10,77	0	0	81,85	38,77	80,61	0,36	0	0
35	35,77	4,13	17,37	15,05	0	0	91,33	41,35	90,03	0	0	0
40	42,54	5,09	19,13	13,79	0	0,17	98,47	46,61	98,95	0,17	0	0,17
45	47,09	7,67	22,67	13,67	0	0,01	102,30	47,97	105,74	0	0	0,01
50	58,44	9,06	28,55	18,09	0	0	110,57	54,65	111,15	0,76	0	0
55	64,40	9,85	33,10	19,89	0	0	126,31	62,28	118,39	1,21	0	0
60	64,40	9,81	35,10	22,55	0	0	136,20	70,23	126,22	0,35	0	0
65	72,67	11,74	42,25	26,86	0	0	144,95	78,72	133,57	1,03	0	0
70	76,93	14,04	47,11	30,26	0	0	153,91	88,40	148,80	0,91	0	0
75	83,60	18,52	56,81	30,59	0	0	166,20	101,96	160,97	1,69	0	0
80	87,77	25,19	65,17	29,45	0	0	180,71	112,83	171,81	1,70	0	0
85	91,02	31,43	72,65	31,60	0	0	195,99	115,62	183,43	3,05	0	0
90	101,36	39,43	79,71	31,75	0,12	0,09	217,55	127,11	196,11	2,53	0,12	0,09
95	107,39	43,65	87,20	31,78	0	0,03	230,26	143,69	211,84	2,21	0	0,03
	Розширення											
0	6,34	0	8,09	0	0	0	47,21	11,61	38,59	0	0	0
5	4,25	0,43	7,09	0	0	0	44,07	9,92	28,89	0	0	0
10	3,22	0,17	5,10	1,74	0	0	38,36	9,64	27,21	0	0	0
15	6,31	0,71	2,10	4,25	0	0	32,71	9,42	23,89	0	0	0
20	10,25	0,27	2,74	9,76	0	0	28,67	8,44	21,62	0,17	0	0
25	20,11	0,36	2,27	16,31	0,14	0	25,76	8,22	20,16	0,21	0,14	0
30	27,80	0	1,43	29,68	0	0	19,88	6,13	18,62	0,85	0	0
35	32,35	0,21	0,89	35,96	0,21	0,34	17,76	6,35	16,65	1,87	0,21	0,34
40	42,19	0,80	0,46	48,50	0	0	18,45	5,33	15,13	3,41	0	0
45	54,53	0	0,05	64,35	0	0	18,34	4,13	13,11	6,79	0	0
50	63,87	0,35	0	75,59	0	0	14,69	4,49	12,70	6,99	0	0
55	77,15	0,43	0,63	86,94	0,31	0	15,82	5,20	13,26	10,03	0,31	0
60	85,13	0,27	0,07	96,86	0	0	20,34	4,50	11,29	14,29	0	0
65	98,48	0	0,71	111,18	0	0	28,82	1,67	10,64	22,57	0	0
70	110,27	0,52	1,04	114,04	0	0,31	30,88	0,53	9,06	28,31	0	0,31
75	121,93	0,31	1,29	126,70	0,35	0	36,74	0,31	8,07	35,56	0,35	0
80	138,29	0,35	0	143,67	0	0,17	38,55	0,35	5,15	39,83	0	0,17
85	157,74	0,44	0,32	159,47	0,19	0	41,86	0,81	4,55	46,47	0,19	0
90	170,73	0,49	0,84	174,11	0	0	49,29	0,49	4,33	55,56	0	0
95	183,31	1,13	0,07	187,44	0	0	54,60	1,13	1,93	62,51	0,27	0
100	196,45	0,33	0,23	198,19	0	0	63,61	0,46	0,23	69,67	0,17	0

Таблиця Б.22

Результати стійкості алгоритму StegoBIT до атаки пропорційного масштабування (експерименти 25-36)

Номер перетворення	**Експеримент**											
	25	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**	**36**
	Стиснення											
0	168,95	10,20	2,19	88,25	0	0	241,00	51,70	46,97	36,25	0	0
5	253,18	10,21	2,66	173,86	0	0	273,11	54,01	53,76	74,03	0	0
10	298,89	12,57	5,37	231,22	0,02	0	295,87	59,29	61,12	104,11	0,02	0
15	340,89	15,39	7,16	268,59	0	0	321,08	69,33	71,37	121,99	0	0
20	367,18	16,14	8,36	293,35	0	0	345,93	77,15	81,43	146,30	0	0
25	380,16	17,29	10,37	303,15	0	0	361,96	90,72	89,29	164,27	0	0
30	398,79	18,47	12,67	320,92	0,35	0	383,84	104,13	99,43	171,38	0,35	0
35	415,09	19,68	17,59	326,62	0,53	0	407,45	109,75	105,83	176,97	0,53	0
40	426,77	21,93	23,07	341,11	0	0	419,25	118,57	112,65	188,53	0	0
45	444,25	24,06	28,37	347,64	0,35	0,07	441,99	126,12	122,32	189,91	0,35	0,07
50	459,23	27,11	36,76	359,73	0	0,15	456,95	137,09	133,12	196,47	0	0,15
55	472,47	28,47	42,77	370,65	0	0	481,79	146,73	146,23	205,63	0	0
60	482,31	33,20	49,06	375,25	0	0	501,27	154,35	158,04	206,66	0	0
65	490,80	46,06	54,33	380,13	0	0	514,36	172,25	175,97	220,29	0	0
70	510,33	54,27	60,61	376,82	0	0	540,56	187,69	192,55	216,60	0	0
75	524,19	60,49	70,38	378,27	0	0	564,10	208,23	204,49	221,69	0	0
80	528,10	71,00	84,50	378,55	0	0	577,59	218,52	215,90	224,03	0	0
85	531,23	83,01	92,13	383,42	0	0	606,05	228,11	224,22	228,51	0,51	0
90	546,12	90,05	98,08	383,53	0	0,84	619,84	240,98	236,83	229,19	2,53	0,84
95	551,72	102,12	106,58	386,99	0	0	636,68	253,51	255,86	240,91	2,53	0
	Розширення											
0	168,95	10,20	2,19	88,25	0	0	241,00	51,70	46,97	36,25	0	0
5	176,61	9,28	1,92	117,07	0	0,23	232,80	45,62	42,03	44,58	0,11	0,23
10	249,79	6,62	0,33	205,11	0	0	251,05	31,75	39,19	83,60	0	0
15	311,73	3,47	0	261,57	0	0	262,40	26,69	34,96	115,19	0	0
20	342,03	3,44	0,21	309,27	0	0	287,89	21,39	28,97	149,85	0	0
25	375,99	2,60	0	347,76	0	0	309,09	21,07	23,05	184,84	0	0
30	400,23	3,25	0,85	376,58	0,07	0	317,51	21,72	19,05	210,35	0,21	0
35	422,67	2,75	0,39	407,77	0,10	0	323,14	21,21	14,05	234,18	0,10	0
40	440,79	3,08	1,14	432,09	0	0	341,09	18,92	12,87	255,69	0	0

45	455,41	3,60	0,42	462,15	0,02	0,42	353,36	15,73	9,09	288,90	0,02	0,42
50	471,25	3,01	0,18	485,69	0	0	356,32	13,54	7,61	317,04	0	0
55	493,03	3,44	0	495,15	0,31	0	366,73	14,09	7,19	334,45	0,31	0
60	516,97	3,60	2,35	506,46	0	0	378,73	12,01	8,01	356,97	0	0
65	526,54	2,60	0	522,24	0	0	396,01	10,47	4,49	377,23	0	0
70	538,57	3,23	0,44	531,70	0	0	418,45	9,99	4,34	392,01	0	0
75	552,69	3,32	1,71	543,94	0	0	435,89	7,34	5,03	407,37	0	0
80	566,66	3,79	0,27	561,44	0	0	442,63	5,35	1,63	418,11	0	0
85	575,67	3,41	1,20	568,89	0	0,53	455,67	3,41	1,78	428,53	0	0,53
90	585,07	4,23	0,61	580,25	0	0	458,86	3,09	0,61	438,01	0	0
95	594,38	4,29	0,33	595,14	0	0	472,61	3,35	0,33	459,21	0	0
100	603,75	4,82	0,22	606,75	0	0	483,01	3,90	0,22	469,78	0,17	0

Таблиця Б.23

Результати стійкості алгоритму StegoTEMPL до атаки пропорційного масштабування (експерименти 1-12)

Номер перетворення	**Експеримент**											
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
	Стиснення											
0	2,19	10,35	0,51	0	0	0	18,07	30,28	9,35	0	0	0
5	4,11	9,45	0	0	0	0	19,62	34,17	10,14	0	0	0
10	6,41	10,82	0,77	0	0	0	22,16	36,47	11,03	0	0	0
15	9,89	10,93	0,26	0	0	0	28,11	41,41	10,62	0	0	0
20	11,43	10,26	0,32	0	0,24	0	36,38	44,57	10,67	0	0,24	0
25	12,03	10,99	0,45	0	0	0	39,82	44,82	12,37	0	0	0
30	14,02	12,62	2,41	0	0	0	42,69	47,87	16,11	0	0	0
35	13,95	12,43	3,23	0	0	0	47,40	52,99	15,28	0	0	0
40	12,81	15,63	4,99	0	0	0	52,64	63,11	18,33	0	0	0
45	12,83	20,11	7,55	0	0	0	55,52	66,47	23,31	0	0	0,25
50	14,43	21,18	8,48	0	0	0	56,05	73,43	27,85	0	0	0
55	15,89	27,69	7,87	0	0	0	61,97	81,17	30,44	0	0	0
60	17,58	34,40	8,29	0	0	0	70,13	88,91	33,81	0	0	0
65	18,39	37,73	9,39	0	0	0	78,31	98,85	35,87	0	0	0
70	23,87	39,66	10,74	0	0	0	82,42	104,18	42,08	0	0	0
75	28,39	43,37	11,49	0	0	0	93,01	107,62	42,93	0	0	0
80	35,45	44,85	12,10	0	0	0	104,86	117,04	46,94	0	0	0
85	39,56	50,97	12,73	0	0,26	0	112,65	126,01	49,81	0	0,26	0
90	43,43	52,47	15,15	0	0	0	122,69	129,21	58,58	0	0	0
95	49,75	59,89	19,67	0	0	0	125,77	137,59	70,07	0	0	0
	Розширення											
0	2,19	10,35	0,51	0	0	0	18,07	30,28	9,35	0	0	0
5	1,00	8,88	0,52	0	0	0	14,73	25,88	8,28	0	0	0
10	0,37	7,60	0,53	0	0	0	13,34	19,78	9,13	0	0	0
15	0,24	5,19	0,78	0	0	0,26	12,45	16,26	9,45	0	0	0,26
20	0,33	5,43	0	0	0	0	10,44	15,52	7,84	0,26	0	0
25	0,87	2,59	0,26	0	0	0	10,23	11,27	5,92	0	0	0
30	0,24	0,78	0,26	0	0	0	11,59	12,11	4,63	0	0	0
35	0,26	0	0,26	0	0	0	11,43	10,15	2,94	0	0	0,26
40	0,75	1,05	0,25	0	0,27	0	9,58	11,77	0,79	0	0,27	0
45	0,26	0,27	0,26	0,25	0	0	9,71	11,29	0,77	0,25	0	0,26
50	1,56	0	0,75	0	0	0,19	12,11	10,51	1,01	0	0	0,19
55	0,26	0,53	0,51	0	0	0	9,89	9,84	1,53	0	0	0
60	0,51	0	0,52	0	0	0	7,81	10,49	0,77	0	0	0
65	0,27	0,53	1,29	0	0	0,25	3,54	12,16	1,81	0	0	0,51
70	0,52	0,26	1,55	0,26	0	0	2,33	13,26	1,55	0,26	0	0
75	0	0,77	0,53	0	0	0	1,49	12,37	1,05	0	0	0
80	0	0,25	1,02	0	0	0	1,07	10,89	1,02	0	0	0
85	0,26	0,52	0,26	0	0,52	0	0,89	8,93	0,52	0	0,52	0
90	0,26	0,75	0,25	0	0	0,26	1,26	6,55	1,05	0	0	0,26
95	1,54	0,77	0,97	0	0,26	0,25	2,07	5,32	1,49	0	0,26	0,48
100	0,50	0,26	0,49	0	0	0	0,76	4,06	0,49	0	0	0

Таблиця Б.24

Результати стійкості алгоритму StegoTEMPL до атаки пропорційного масштабування (експерименти 13-24)

Номер перетворення	**Експеримент**											
	13	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
	Стиснення											
0	6,93	15,39	1,63	0	0	0	67,27	74,85	22,85	0	0	0
5	9,98	15,21	0,39	0	0	0	76,29	86,91	24,94	0	0	0
10	15,83	23,55	3,07	0	0	0	86,10	95,28	25,43	0	0	0
15	21,53	25,63	1,92	0,42	0	0	92,95	102,45	24,94	0,42	0	0
20	24,93	25,07	3,95	0	0	0,42	100,35	109,64	28,14	0	0	0,42
25	28,70	24,85	4,61	0	0	0	107,73	113,69	29,38	0	0	0,41

30	31,69	28,32	6,75	0	0	0	118,84	124,99	34,01	0	0	0
35	36,11	35,67	8,04	0	0	0	130,29	139,49	44,22	0	0	0
40	37,71	42,98	11,66	0	0	0	144,52	142,46	51,30	0,42	0	0
45	40,89	50,41	14,59	0	0	0	152,11	149,42	57,24	0	0	0
50	45,78	52,09	14,13	0	0	0	161,38	162,84	61,64	0	0	0
55	52,55	63,55	17,43	0	0	0	169,86	176,83	71,59	0	0	0
60	59,07	70,95	20,46	0	0	0	179,15	187,45	72,69	0	0	0
65	64,57	80,37	22,84	0	0	0	196,15	200,06	81,35	0	0	0
70	74,05	85,21	24,59	0	0	0	208,47	204,28	93,49	0	0	0
75	81,37	88,61	25,01	0	0	0	222,49	215,41	96,66	0	0	0
80	97,89	97,04	26,89	0	0,39	0	238,46	233,05	104,06	0	0,39	0
85	107,31	107,92	30,25	0	0	0	255,41	243,49	112,79	0	0	0
90	113,67	117,53	34,62	0	0	0	268,61	267,58	122,10	0	0	0
95	123,69	129,36	39,03	0	0	0,43	283,75	283,10	132,59	0	0	0,43
	Розширення											
0	6,93	15,39	1,63	0	0	0	67,27	74,85	22,85	0	0	0
5	6,21	15,23	0	0	0	0	54,95	69,04	17,87	0	0	0
10	5,63	14,13	0,83	0	0	0,42	48,67	64,29	17,61	0	0,42	0,42
15	4,39	12,07	1,26	0	0	0	45,77	53,73	18,41	0	0	0
20	7,15	10,24	2,11	0	0	0,41	45,61	47,91	17,07	0,42	0	0,41
25	6,32	7,90	2,07	0	0	0	43,10	40,49	13,11	0	0	0
30	5,88	7,90	0,83	0	0	0	41,29	34,26	10,99	0	0	0
35	5,77	8,59	0,42	0	0	0	35,06	31,73	10,85	0	0	0,42
40	6,71	6,73	1,21	0	0	0	29,05	29,19	9,77	0	0	0
45	6,32	4,62	1,67	0	0	0	25,81	26,69	6,20	0	0	0
50	2,94	4,71	0,81	0	0	0	20,65	27,45	4,64	0	0	0
55	1,26	4,49	1,24	0	0	0	18,23	26,15	5,69	0	0,42	0
60	0,81	1,27	0,85	0	0,42	0	14,25	25,09	2,62	0	0,42	0
65	0,85	1,23	2,86	0,43	0	0,40	13,73	20,83	3,31	0,43	0,42	0,82
70	1,63	0,42	1,68	0	0	0	10,44	16,33	2,52	0	0	0
75	1,67	2,50	1,67	0	0,41	0	8,59	18,99	2,09	0	0,41	0
80	0	0	0,42	0	0	0	7,30	16,59	1,27	0	0	0
85	1,25	1,26	1,24	0	0,42	0	7,03	15,19	1,66	0	0,42	0
90	0,84	0,83	1,25	0,42	0	0	5,72	10,99	2,11	0,42	0	0
95	1,64	0,42	0,85	0	0	0,43	7,21	8,93	1,69	0	0	0,43
100	0,43	0,42	0,83	0	0	0	6,39	7,30	1,66	0,43	0	0

Таблиця Б.25

Результати стійкості алгоритму StegoTEMPL до атаки пропорційного масштабування (експерименти 25-36)

Номер перетворення	**Експеримент**											
	25	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**	**36**
	Стиснення											
0	23,79	22,46	1,04	0	0	0	132,77	114,61	37,02	0	0	0
5	25,45	30,45	0	0	0	0	143,85	119,33	43,52	0	0	0
10	28,92	39,98	3,64	0	0	0	159,77	124,27	48,51	0	0	0,53
15	43,08	40,57	1,55	0,53	0	0	173,47	137,27	52,01	0,53	0	0
20	45,55	41,38	1,75	0	1,05	0	190,15	154,57	52,31	0	1,05	0
25	51,77	45,19	3,89	0	0	0	207,17	161,81	63,38	0	0	0
30	61,10	49,35	5,55	0	0	0	220,01	182,04	72,95	0	0	0
35	69,25	55,92	6,94	0	0	0	232,03	197,23	85,15	0	0	0
40	73,03	62,62	10,53	0	0	0,53	251,73	212,57	99,21	0	0	0,53
45	82,29	79,79	15,93	0	0,52	0	266,33	227,04	102,67	0	0,52	0
50	100,03	89,70	20,44	0	0	1,06	284,61	238,54	119,70	0	0	1,58
55	111,17	93,53	24,99	0	0	0	299,85	256,13	134,41	0	0	0
60	118,13	107,73	31,27	0	0	0	316,13	271,56	151,43	0	0	0
65	130,12	117,09	38,41	0	0	0	329,68	286,24	164,08	0	0	0
70	137,85	122,95	38,39	0	1,07	0	345,23	303,71	173,21	0	1,07	0
75	145,65	127,49	42,45	0	0	0	360,81	310,06	174,13	0	0	0
80	161,24	140,60	48,87	0	0	0	377,97	334,09	184,55	0	0	0
85	183,49	162,31	54,54	0	0	0	405,83	353,18	202,97	0	0	0
90	199,19	168,71	55,31	0	0	0	429,55	371,08	219,85	0	0	0
95	214,70	180,05	68,05	0	0	0	458,65	395,97	231,11	0	0	0
	Розширення											
0	23,79	22,46	1,04	0	0	0	132,77	114,61	37,02	0	0	0
5	18,92	21,63	0,53	0,53	0	0,41	121,57	104,15	30,94	0,53	0	0,41
10	14,40	14,89	1,05	0,51	0	0	111,29	94,80	25,19	0,51	0	0
15	11,06	11,63	2,11	0	0	0	99,97	84,68	25,77	0	0	0
20	8,97	6,33	1,06	0,53	0	0	93,17	78,23	21,14	1,05	0	0,53
25	4,50	4,79	0,53	0	0	0	87,54	71,15	15,96	0	0	0
30	1,63	2,59	1,55	0	0	0	73,25	61,29	11,50	0	0	0
35	0,45	0	1,05	0	0	0	63,01	56,13	10,10	0	0	0,53
40	0,95	1,58	2,11	0	0,43	0,53	54,51	50,91	8,50	0	0,43	0,53
45	0,97	0,50	1,54	0	0	0	49,67	44,23	4,51	0	0	0
50	0,96	0	1,05	0	0	0	43,94	37,59	4,09	0	0	0

55	0,95	0	2,11	0	0	0	39,19	38,20	4,37	0	0	0
60	1,32	2,63	2,57	0	0,53	0	27,31	38,76	3,28	0	1,05	0
65	0,10	0,91	2,11	0	0	0	24,37	32,64	2,80	0	0	0,53
70	0,53	1,57	3,67	0	0	0	21,93	26,03	4,31	0	0	0,53
75	1,03	2,10	2,09	0	0	0	19,93	25,94	2,62	0	0	0
80	1,55	1,53	1,57	0,53	0	0	15,03	20,74	4,18	0,53	0	0
85	1,59	2,08	1,06	0	0	0	9,48	15,29	2,62	0	0	0
90	0,53	2,51	1,56	0	0,53	0	5,32	12,37	3,66	0	0,52	0
95	1,05	1,05	2,61	0	0	0	6,31	8,29	3,67	0	0	1,05
100	2,61	2,94	1,99	0,53	0	0,52	7,37	5,65	2,50	1,04	0	0,52

Результати стійкості алгоритмів StegoBIT та StegoTEMPL до атаки непропорційного масштабування за віссю абсцис представлені в табл. Б.26-Б.28 та табл. Б.29-Б.31 відповідно.

Таблиця Б.26

Результати стійкості алгоритму StegoBIT до атаки непропорційного масштабування за віссю абсцис (експерименти 1-12)

Номер перетворення	**Експеримент**											
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
	Стиснення											
0	0	0,51	1,56	0	0	0	0	2,99	24,41	0	0	0
5	0,15	0,03	1,55	0	0,24	0	0,15	3,23	25,52	0	0,24	0
10	0,02	0,14	1,99	0	0,15	0	0,02	4,84	25,73	0	0,15	0
15	0,29	0	2,60	0	0	0	0,29	5,79	26,01	0	0	0
20	0,25	0,63	4,31	0	0	0	0,25	9,27	27,01	0	0	0
25	0,31	0,21	5,09	0	0	0	0,32	9,49	28,72	0	0	0
30	0	0,02	5,31	0,23	0	0,15	0,05	10,72	31,23	0,23	0	0,15
35	0,15	0,25	6,92	0	0,25	0	0,15	10,99	36,07	0	0,25	0
40	0,44	0	8,62	0	0	0	0,44	10,77	38,81	0	0	0
45	0,35	0	9,57	0	0	0,23	0,35	11,66	39,83	0	0	0,23
50	0	0	10,51	0	0	0	0	13,71	42,16	0	0	0
55	0,15	0,19	10,42	0	0	0	0,15	14,11	44,06	0	0	0
60	0,15	0,37	11,26	0	0	0	0,15	14,45	46,13	0	0	0
65	0,29	0,11	11,91	0	0	0	1,31	14,47	48,05	0	0	0
70	0,15	0,50	11,90	0	0	0	2,37	14,55	50,04	0	0	0
75	0,03	0,59	11,76	0,05	0	0	2,65	15,49	53,62	0,05	0	0
80	0	1,82	12,11	0	0	0,07	2,77	16,33	56,17	0	0	0,07
85	0,24	2,61	13,28	0	0	0	3,50	17,70	60,69	0	0	0
90	0	2,55	14,97	0	0,16	0	3,69	19,29	63,70	0	0,16	0
95	0,11	2,55	18,41	0,05	0	0	4,93	20,14	67,92	0,05	0	0
	Розширення											
0	0	0,51	1,56	0	0	0	0	2,99	24,41	0	0	0
5	0,15	0	1,29	0	0	0	0,15	2,30	23,02	0	0	0
10	0,27	0	1,07	0	0	0	0,27	2,08	22,08	0	0	0
15	0,27	0	0,14	0	0	0	0,27	1,20	20,45	0	0	0
20	0	0	0,17	0	0	0	0	1,20	17,39	0	0	0
25	0	0,35	0,24	0	0	0	0,27	1,55	17,06	0	0	0
30	0,40	0,03	0,41	0	0	0	0,54	0,99	15,89	0	0	0
35	0,27	0,19	0,23	0	0	0	0,27	0,19	15,09	0	0	0
40	0,12	0	0,23	0,19	0	0	0,17	0	14,16	0,19	0	0
45	0	0,01	0,27	0	0	0	0,25	0,17	13,39	0	0	0
50	0,31	0,57	0,07	0	0	0	0,31	0,57	12,47	0	0	0
55	0,19	0,25	0,26	0	0	0	0,19	0,25	12,46	0	0	0
60	0,15	0,29	0	0	0	0	0,15	0,29	11,96	0	0	0
65	0,63	0	0,38	0,19	0	0	0,64	0	12,17	0,19	0	0
70	0,34	0,30	0,33	0	0	0,21	0,50	0,31	12,03	0	0	0,21
75	0,32	0	0,05	0	0	0	0,32	0	11,81	0	0	0
80	0,33	0,18	0,17	0	0	0	0,47	0,18	11,57	0	0	0
85	0,18	0,19	0,18	0	0	0	0,18	0,19	10,98	0	0	0
90	0	0,15	0	0	0	0	0	0,15	9,97	0	0	0
95	0,25	0,35	0,16	0,15	0	0,14	0,28	0,45	9,87	0,15	0	0,14
100	0,19	0,42	0,20	0,09	0	0	0,19	0,42	8,67	0,09	0	0

Таблиця Б.27

Результати стійкості алгоритму StegoBIT до атаки непропорційного масштабування за віссю абсцис (експерименти 13-24)

Номер перетворення	**Експеримент**											
	13	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
	Стиснення											
0	6,34	0	8,23	0	0	0	46,79	11,53	35,34	0	0	0
5	5,94	2,25	9,34	0,20	0	0	47,50	12,11	39,45	0	0	0
10	7,09	2,45	8,85	1,15	0,02	0	47,50	11,89	44,17	0	0,02	0
15	9,46	2,10	9,47	2,71	0	0	48,03	14,44	47,17	0,16	0	0
20	9,14	2,38	9,77	4,54	0	0,23	47,86	18,06	52,56	0,52	0	0,23

25	11,09	2,29	10,05	5,43	0	0	49,61	22,18	55,50	0,34	0	0
30	12,60	3,65	9,83	6,44	0	0	55,47	27,88	56,34	0,36	0	0
35	21,22	2,10	10,07	9,19	0	0	57,45	27,67	60	0	0	0
40	22,90	2,41	10,35	8,39	0	0,17	61,08	30,27	63,47	0,17	0	0,17
45	23,15	2,10	12,01	7,43	0	0,01	62,03	31,24	64,76	0	0	0,01
50	25,89	2,67	13,99	10,63	0	0	64,74	34,75	68,18	0	0	0
55	23,39	2,31	13,97	11,55	0	0	69,37	38,61	71,53	0,87	0	0
60	25,97	2,27	16,39	12,33	0	0	73,97	43,04	76,55	0,35	0	0
65	25,26	2,38	18,46	16,23	0	0	75,74	45,21	77,48	0,33	0	0
70	25,40	2,10	19,41	16,28	0	0	77,15	48,58	80,29	0,16	0	0
75	27,33	2,10	21,53	16,30	0	0	81,47	51,73	83,38	0,17	0	0
80	30,15	3,71	22,92	16,25	0	0	83,35	52,19	86,73	0,17	0	0
85	29,49	4,70	24,30	17,16	0	0	86,49	52,53	89,84	0,77	0	0
90	28,95	8,12	27,95	17,61	0,12	0,09	87,49	55,95	94,79	0,95	0,12	0,09
95	28,71	9,81	28,77	16,97	0	0	88,37	63,97	103,04	0,93	0	0
	Розширення											
0	6,34	0	8,23	0	0	0	46,79	11,53	35,34	0	0	0
5	4,08	0,32	7,90	0	0	0	43,93	11,85	32,45	0	0	0
10	2,01	0	8,33	1,56	0	0	41,63	11,53	32,37	0	0	0
15	4,83	0	6,86	3,20	0	0	37,91	11,28	29,85	0	0	0
20	5,33	0,27	6,45	5,23	0	0	35,37	10,54	28,36	0	0	0
25	10,20	0,36	6,42	10,17	0,14	0	35,29	10,63	25,44	0,21	0,14	0
30	14,17	0	6,17	17,75	0	0	30,45	10,27	25,08	0,67	0	0
35	17,33	0,21	4,95	23,45	0,21	0,34	26,41	8,88	23,46	0,87	0,21	0,34
40	26,59	0,80	4,26	27,56	0	0	24,33	9,07	20,50	2,31	0	0
45	33,63	0	4,13	38,85	0	0	22,35	8,27	20,28	3,81	0	0
50	38,82	0,35	4,13	47,24	0	0	20,84	8,62	20,27	5,21	0	0
55	44,62	0,02	4,75	54,51	0	0	21,97	8,76	20,85	7,47	0	0
60	48,93	0,27	4,21	62,68	0	0	24,48	8,53	20,07	9,30	0	0
65	58,95	0	3,21	73,16	0	0	29,27	7,84	19,59	14,74	0	0
70	65,12	0,52	3,14	75,61	0	0,31	30,16	6,70	18,41	18,88	0	0,31
75	75,71	0,31	2,92	85,01	0,35	0	33,56	6,47	18,13	19,97	0,35	0
80	86,31	0,35	2,10	91,37	0	0,17	37,77	6,52	16,17	22,89	0	0,17
85	98,19	0,44	2,10	101,25	0	0	39,28	6,88	16,08	26,79	0	0
90	105,92	0,49	2,94	109,31	0	0	40,35	4,62	16,74	32,66	0	0
95	113,93	0,73	2,17	116,12	0	0	44,51	4,87	15,86	35,75	0,27	0
100	123,66	0,15	2,10	122,90	0	0	45,61	4,42	15,49	40,95	0	0

Таблиця Б.28

Результати стійкості алгоритму StegoBIT до атаки
непропорційного масштабування за віссю абсцис (експерименти 25-36)

Номер перетворення	**Експеримент**											
	25	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**	**36**
	Стиснення											
0	148,26	10,20	1,91	60,81	0	0	232,24	51,70	45,88	24,22	0	0
5	201,62	10,21	1,63	115,37	0	0	246,31	51,70	49,57	51,09	0	0
10	230,67	12,57	2,25	147,64	0,02	0	260,83	55,56	53,81	64,31	0,02	0
15	259,36	15,27	2,67	175,93	0	0	273,41	59,45	58,41	77,20	0	0
20	276,98	16,14	3,55	193,86	0	0	293,81	63,27	64,29	92,69	0	0
25	288,95	16,25	4,51	202,74	0	0	307,31	70,33	65,65	104,42	0	0
30	301,23	15,87	5,20	221,33	0,35	0	314,97	74,30	70,17	113,79	0,35	0
35	312,01	16,01	7,73	220,51	0,53	0	329,86	78,11	75,38	120,23	0,53	0
40	317,77	16,24	7,63	224,61	0	0	337,01	79,31	78,19	122,99	0	0
45	322,83	16,15	8,24	231,69	0,35	0,07	348,55	82,27	79,61	118,59	0,35	0,07
50	336,10	16,07	12,17	245,03	0	0	358,16	87,37	83,43	120,70	0	0
55	345,67	15,80	13,02	248,53	0	0	365,75	91,94	84,84	124,18	0	0
60	354,95	15,80	14,39	258,63	0	0	379,96	92,97	92,98	128,73	0	0
65	350,90	18,85	15,89	256,58	0	0	385,47	98,28	100,63	132,98	0	0
70	366,36	23,53	19,71	247,35	0	0	400,91	102,89	103,90	123,07	0	0
75	375,25	24,92	22,12	249,83	0	0	408,57	105,03	107,97	131,02	0	0
80	377,27	29,53	27,07	259,21	0	0	408,95	105,06	113,92	138,42	0	0
85	377,85	32,22	32,83	262,67	0	0	420,12	109,17	116,72	141,15	0	0
90	385,64	31,56	34,14	274,03	0	0,84	423,56	112,91	118,53	136,79	0	0,84
95	384,58	33,26	36,08	273,19	0	0	426,65	115,85	123,00	142,16	0	0
	Розширення											
0	148,26	10,20	1,91	60,81	0	0	232,24	51,70	45,88	24,22	0	0
5	148,49	8,97	2,05	84,78	0	0,23	227,94	48,91	45,63	32,73	0	0,23
10	200,02	6,62	1,63	142,28	0	0	241,61	41,84	43,32	59,39	0	0
15	241,03	5,20	1,53	176,19	0	0	249,30	40,80	41,09	78,73	0	0
20	264,07	5,52	0,82	218,63	0	0	259,43	38,99	37,67	102,11	0	0
25	290,14	5,20	0	245,83	0	0	273,69	38,67	35,91	120,98	0	0
30	321,25	5,85	0,33	259,60	0,07	0	281,61	39,32	34,55	138,28	0,21	0
35	340,49	5,35	0,39	284,81	0,10	0	291,69	35,17	30,95	155,73	0,10	0
40	356,11	5,55	0,73	315,60	0	0	302,27	32,79	27,29	176,63	0	0
45	368,27	6,20	0,42	347,96	0,02	0,42	312,45	30,77	24,23	208,79	0,02	0,42

50	372,40	5,61	0,18	353,26	0	0	309,70	28,47	23,51	219,67	0	0
55	382,56	5,20	0	368,01	0	0	315,63	28,07	22,97	236,03	0	0
60	406,50	5,79	2,23	383,42	0	0	323,57	24,49	22,17	246,19	0	0
65	426,69	5,20	0	398,23	0	0	336,75	18,33	18,82	257,61	0	0
70	436,64	5,83	0,44	401,27	0	0	355,01	17,85	18,93	272,80	0	0
75	450,65	5,73	1,71	415,27	0	0	371,97	15,95	20,18	284,34	0	0
80	468,54	6,34	0,27	433,89	0	0	380,72	16,51	16,95	290,81	0	0
85	484,88	5,94	0,72	440,31	0	0,53	393,55	16,11	16,93	301,84	0	0,53
90	492,31	6,83	0,61	452,10	0	0	394,71	15,86	16,69	319,79	0	0
95	497,12	6,89	0,33	468,26	0	0	402,74	16,09	16,27	327,68	0	0
100	498,24	7,07	0,22	473,89	0	0	404,89	15,65	15,88	336,45	0	0

Таблиця Б.29

Результати стійкості алгоритму StegoTEMPL до атаки непропорційного масштабування за віссю абсцис (експерименти 1-12)

Номер перетворення	**Експеримент**											
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
	Стиснення											
0	1,47	9,59	0,51	0	0	0	16,89	28,26	6,63	0	0	0
5	2,95	8,81	0	0	0	0	16,39	29,73	7,53	0	0	0
10	3,89	8,45	0,26	0	0	0	16,99	33,04	10,25	0	0	0
15	5,48	8,25	0	0	0	0	20,04	33,52	9,77	0	0	0
20	6,35	8,36	0,32	0	0,24	0	23,31	35,09	9,54	0	0,24	0
25	5,97	9,94	0,27	0	0	0	24,31	34,50	10,61	0	0	0
30	7,43	10,94	0,91	0	0	0	25,62	36,64	13,15	0	0	0
35	8,73	10,61	0,76	0	0	0	29,94	38,79	11,73	0	0	0
40	9,10	12,06	2,35	0	0	0	31,25	42,66	12,71	0	0	0
45	9,07	12,93	3,33	0	0	0	32,13	42,49	13,16	0	0	0
50	9,13	11,65	4,53	0	0	0	31,45	42,75	14,57	0	0	0
55	10,53	12,27	4,25	0	0	0	34,43	46,72	14,05	0	0	0
60	12,31	13,05	4,62	0	0	0	37,59	50,55	14,31	0	0	0
65	12,32	13,89	6,15	0	0	0	39,18	52,87	15,75	0	0	0
70	11,61	17,07	7,09	0	0	0	42,99	56,67	20,32	0	0	0
75	11,22	21,49	8,57	0	0	0	45,87	60,67	22,71	0	0	0
80	11,45	21,59	8,63	0	0	0	47,60	66,43	23,65	0	0	0
85	14,77	26,98	8,63	0	0,26	0	51,02	66,99	24,76	0	0,26	0
90	17,61	27,57	8,98	0	0	0	55,08	65,95	28,88	0	0	0
95	19,89	29,76	9,38	0	0	0	61,29	66,68	31,53	0	0	0
	Розширення											
0	1,47	9,59	0,51	0	0	0	16,89	28,26	6,63	0	0	0
5	1,00	8,82	0,52	0	0	0	14,95	26,51	8,15	0	0	0
10	0,43	8,69	0	0	0	0	14,67	21,76	7,58	0	0	0
15	0,24	7,12	0,78	0	0	0,26	13,85	21,11	9,19	0	0	0,26
20	0,07	7,59	0	0	0	0	12,02	22,95	7,55	0	0	0
25	0,61	5,56	0	0	0	0	12,03	20,73	7,28	0	0	0
30	0,31	3,18	0,26	0	0	0	11,06	19,45	7,53	0	0	0
35	0,15	2,23	0	0	0	0	10,55	18,20	5,70	0	0	0
40	0,25	3,02	0,25	0	0,27	0	10,53	18,87	4,35	0	0,27	0
45	0,26	2,50	0,26	0,25	0	0	11,11	18,75	4,34	0,25	0	0,26
50	1,03	2,23	0,49	0	0	0	12,77	17,65	4,05	0	0	0
55	0,26	2,23	0,25	0	0	0	12,47	16,15	4,00	0	0	0
60	0,51	2,07	0,26	0	0	0	12,35	15,67	4,41	0	0	0
65	0	2,23	0,25	0	0	0,25	11,48	17,60	3,40	0	0	0,25
70	0,26	2,23	0,77	0,26	0	0	11,66	18,53	3,41	0,26	0	0
75	0	2,74	0,27	0	0	0	10,73	18,91	3,16	0	0	0
80	0	2,22	0,51	0	0	0	10,29	17,95	3,15	0	0	0
85	0	2,23	0,26	0	0,26	0	9,60	17,33	2,89	0	0,26	0
90	0,26	2,49	0	0	0	0	9,24	16,13	2,89	0	0	0
95	0,77	2,36	0,97	0	0,26	0,25	8,53	15,00	3,90	0	0,26	0,48
100	0,50	1,73	0,23	0	0	0	6,99	12,51	3,03	0	0	0

Таблиця Б.30

Результати стійкості алгоритму StegoTEMPL до атаки непропорційного масштабування за віссю абсцис (експерименти 13-24)

Номер перетворення	**Експеримент**											
	13	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
	Стиснення											
0	7,33	16,88	0,82	0	0	0	66,85	74,29	20,61	0	0	0
5	8,69	16,29	0	0	0	0	71,14	78,74	21,64	0	0	0
10	11,76	20,36	1,67	0	0	0	74,35	85,55	23,37	0	0	0
15	15,53	20,51	0,40	0,42	0	0	80,05	90,06	21,83	0,42	0	0
20	15,71	21,47	1,25	0	0	0,42	83,27	91,74	23,46	0	0	0,42
25	16,77	22,56	0,41	0	0	0	84,50	90,84	25,85	0	0	0,41
30	20,21	22,25	1,05	0	0	0	89,89	93,79	25,91	0	0	0

35	20,28	26,57	3,54	0	0	0	97,66	97,80	27,01	0	0	0
40	22,73	26,86	4,63	0	0	0	105,34	100,38	29,80	0,42	0	0
45	24,45	27,19	7,57	0	0	0	105,87	103,43	29,97	0	0	0
50	23,60	26,93	7,63	0	0	0	107,93	108,19	31,11	0	0	0
55	26,95	29,51	6,75	0	0	0	108,92	112,73	32,69	0	0	0
60	29,77	32,23	6,67	0	0	0	110,90	115,27	35,59	0	0	0
65	29,82	34,96	6,56	0	0	0	112,55	118,21	41,26	0	0	0
70	32,03	36,27	6,52	0	0	0	118,46	121,43	46,42	0	0	0
75	33,08	39,53	6,07	0	0	0	124,52	124,82	47,32	0	0	0
80	38,70	41,52	6,07	0	0,39	0	129,63	131,31	52,11	0	0,39	0
85	39,69	44,36	8,25	0	0	0	133,97	135,16	54,51	0	0	0
90	43,29	50,47	10,04	0	0	0	138,18	142,21	58,60	0	0	0
95	46,81	59,26	12,96	0	0	0,43	144,63	149,07	65,65	0	0	0,43
	Розширення											
0	7,33	16,88	0,82	0	0	0	66,85	74,29	20,61	0	0	0
5	6,91	17,05	0	0	0	0	57,97	71,62	19,49	0	0	0
10	6,52	16,69	0,83	0	0	0,42	55,49	67,99	18,74	0	0,42	0,42
15	5,02	16,53	0,84	0	0	0	53,59	62,49	17,92	0	0	0
20	5,59	14,75	1,68	0	0	0,41	51,59	59,87	16,27	0	0	0,41
25	5,21	14,10	1,23	0	0	0	49,94	55,14	14,17	0	0	0
30	6,03	14,09	0,83	0	0	0	48,19	52,79	12,73	0	0	0
35	5,17	14,44	0	0	0	0	43,58	49,69	11,49	0	0	0
40	5,59	12,82	0,81	0	0	0	37,97	46,32	12,15	0	0	0
45	6,43	10,29	0,84	0	0	0	36,67	45,59	10,83	0	0	0
50	3,77	11,76	0,81	0	0	0	31,32	46,33	8,96	0	0	0
55	2,88	12,66	1,24	0	0	0	29,23	45,55	9,13	0	0,42	0
60	3,22	10,69	0,43	0	0	0	30,13	42,77	8,80	0	0	0
65	3,22	9,66	2,02	0	0	0,40	30,38	42,21	9,89	0	0,42	0,40
70	5,29	8,89	0,42	0	0	0	30,70	39,26	8,71	0	0	0
75	5,59	10,06	0,83	0	0,41	0	29,60	40,02	8,69	0	0,41	0
80	4,33	7,09	0	0	0	0	24,21	38,04	8,29	0	0	0
85	4,73	8,19	0,41	0	0,42	0	23,65	39,21	8,06	0	0,42	0
90	4,74	7,35	0,42	0,42	0	0	21,51	39,23	8,36	0,42	0	0
95	5,52	7,00	0,85	0	0	0,43	21,67	38,78	7,43	0	0	0,43
100	4,73	7,00	0,83	0	0	0	20,01	39,09	7,23	0,43	0	0

Таблиця Б.31

Результати стійкості алгоритму StegoTEMPL до атаки
непропорційного масштабування за віссю абсцис (експерименти 25-36)

Номер перетворення	**Експеримент**											
	25	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**	**36**
	Стиснення											
0	22,86	22,72	0,52	0	0	0	131,37	111,94	34,27	0	0	0
5	24,13	25,46	0	0	0	0	137,57	115,43	36,24	0	0	0
10	27,63	31,93	2,59	0	0	0	141,70	116,49	42,77	0	0	0
15	37,92	32,28	1,55	0,53	0	0	148,27	122,73	47,29	0,53	0	0
20	40,30	33,37	1,03	0	0,53	0	156,77	131,86	46,66	0	0,53	0
25	42,47	34,20	1,03	0	0	0	163,97	135,44	48,45	0	0	0
30	46,53	35,42	1,38	0	0	0	170,90	143,59	50,03	0	0	0
35	52,37	38,75	3,30	0	0	0	178,83	145,13	53,33	0	0	0
40	54,35	42,09	3,44	0	0	0,53	182,14	150,99	55,13	0	0	0,53
45	58,97	45,33	4,87	0	0,52	0	184,07	162,07	57,24	0	0,52	0
50	64,98	45,00	8,09	0	0	0,53	186,93	163,54	66,65	0	0	1,05
55	66,45	45,31	8,16	0	0	0	193,66	167,21	70,96	0	0	0
60	67,15	48,13	7,03	0	0	0	196,85	175,36	77,33	0	0	0
65	69,53	50,07	9,34	0	0	0	199,76	177,13	82,18	0	0	0
70	72,16	55,08	7,57	0	0,53	0	206,30	178,39	89,24	0	0,53	0
75	71,98	59,05	10,49	0	0	0	217,87	181,65	91,24	0	0	0
80	74,03	65,39	17,83	0	0	0	223,67	189,53	92,30	0	0	0
85	78,34	73,17	19,37	0	0	0	233,77	194,71	95,74	0	0	0
90	78,66	77,83	20,33	0	0	0	232,99	202,09	100,18	0	0	0
95	81,83	80,15	27,37	0	0	0	245,05	205,89	109,11	0	0	0
	Розширення											
0	22,86	22,72	0,52	0	0	0	131,37	111,94	34,27	0	0	0
5	19,07	21,83	0,53	0,53	0	0,41	123,96	104,17	32,46	0,53	0	0,41
10	16,62	17,65	1,05	0,51	0	0	118,16	102,12	30,50	0,51	0	0
15	15,05	15,94	0,52	0	0	0	117,83	98,89	27,52	0	0	0
20	14,51	14,85	1,06	0,53	0	0	115,35	97,21	26,31	0,53	0	0,53
25	12,52	15,50	0,53	0	0	0	110,07	96,73	24,05	0	0	0
30	10	11,89	1,04	0	0	0	102,38	94,21	20,79	0	0	0
35	8,83	8,83	0	0	0	0	92,56	89,82	16,81	0	0	0
40	10,27	8,83	1,59	0	0,43	0,53	87,06	78,31	16,77	0	0,43	0,53
45	9,40	8,27	1,02	0	0	0	85,65	69,51	14,74	0	0	0
50	9,39	7,77	0	0	0	0	80,10	66,06	13,58	0	0	0
55	9,94	7,77	1,58	0	0	0	76,72	66,83	14,14	0	0	0

60	8,87	9,34	2,05	0	0	0	68,58	67,55	14,69	0	0,53	0
65	8,27	8,21	1,05	0	0	0	67,22	66,32	13,68	0	0	0
70	8,59	9,16	2,09	0	0	0	66,26	64,46	14,67	0	0	0,53
75	9,06	9,11	1,04	0	0	0	63,43	65,21	11,98	0	0	0
80	8,58	8,55	1,04	0,53	0	0	61,33	63,85	11,03	0,53	0	0
85	8,53	8,06	0,53	0	0	0	58,03	61,83	8,40	0	0	0
90	7,97	9,51	1,05	0	0,53	0	55,89	60,11	8,92	0	0,52	0
95	7,97	8,05	1,57	0	0	0	54,22	57,23	9,04	0	0	0,53
100	9,53	9,95	1,46	0,53	0	0,52	52,31	56,19	8,43	1,04	0	0,52

Результати стійкості алгоритмів StegoBIT та StegoTEMPL до атаки непропорційного масштабування за віссю абсцис представлені в табл. Б.32-Б.34 та табл. Б.35-Б.37 відповідно.

Таблиця Б.32

Результати стійкості алгоритму StegoBIT до атаки непропорційного масштабування за віссю ординат (експерименти 1-12)

Номер перетворення	**Експеримент**											
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
	Стиснення											
0	0	1,13	1,54	0	0	0	0	3,43	24,24	0	0	0
5	0	1,13	2,65	0	0	0	0	3,49	24,55	0	0	0
10	0,11	1,13	4,01	0	0	0	0,11	4,43	26,53	0	0	0
15	0	1,13	4,23	0	0	0	0,05	6,40	27,93	0	0	0
20	0	1,13	4,56	0	0	0	0	6,80	28,15	0	0	0
25	0	1,15	6,12	0	0	0	0	6,81	29,39	0	0	0
30	0,25	1,13	6,30	0	0	0	0,25	7,73	33,18	0	0	0
35	0	1,13	6,51	0	0	0	0	9,89	34,06	0	0	0
40	0	1,13	6,70	0	0	0	0	11,05	36,52	0	0	0
45	0	1,13	6,97	0	0	0	1,20	12,40	40,77	0	0	0
50	0	1,13	7,77	0	0	0,21	2,00	12,53	43,37	0	0	0,21
55	0	1,13	8,11	0	0	0	2,53	12,53	45,03	0	0	0
60	0,16	1,13	8,99	0	0	0	3,20	12,61	48,56	0	0	0
65	0	1,67	10,14	0	0	0	3,97	13,07	51,98	0	0	0
70	0	1,93	10,33	0	0	0	4,63	15,22	57,81	0	0	0
75	0	2,49	11,18	0	0	0	5,59	16,85	64,87	0	0	0
80	0	2,96	11,47	0	0	0,11	6,07	18,95	69,33	0	0	0,11
85	0	2,85	12,63	0	0	0	6,31	21,52	72,08	0	0	0
90	0	3,18	13,97	0	0	0	6,51	22,00	74,62	0	0	0
95	0	3,39	17,24	0	0	0	7,73	23,95	77,45	0	0	0
	Розширення											
0	0	1,13	1,54	0	0	0	0	3,43	24,24	0	0	0
5	0,29	1,24	1,33	0	0	0	0,29	3,49	23,11	0	0,11	0
10	0	1,13	1,33	0	0	0	0	3,35	21,47	0	0	0
15	0,03	1,13	1,33	0	0	0	0,03	3,30	21,39	0	0	0
20	0	1,13	0	0	0	0	0	2,68	20,53	0	0	0
25	0,11	1,15	0	0	0	0	0,11	2,03	20	0	0	0
30	0	1,33	0	0	0	0	0	1,33	19,75	0	0	0
35	0,49	1,13	0,18	0	0	0	0,49	1,13	18,47	0	0	0
40	0	1,13	0	0	0	0	0	1,13	17,66	0	0	0
45	0,21	1,13	0,05	0	0	0	0,22	1,13	15,38	0	0	0
50	0,21	1,13	0,07	0	0	0	0,21	1,13	14,71	0	0	0
55	0,11	1,13	0	0	0	0	0,20	1,25	14,59	0	0	0
60	0,31	1,32	0	0	0	0	0,36	1,42	14,51	0	0	0
65	0	1,24	0,19	0	0	0	0,11	1,38	14,37	0	0	0
70	0,32	1,19	0	0	0	0	0,32	1,19	13,20	0	0	0
75	0,24	1,13	0	0,23	0	0	0,29	1,13	13,11	0,23	0	0
80	0,15	1,27	0,19	0	0,11	0	0,15	1,27	12,66	0	0,11	0
85	0	1,17	0	0,09	0	0	0	1,17	11,72	0,09	0	0
90	0,17	1,13	0	0	0	0	0,17	1,13	10,41	0	0	0
95	0,16	1,13	0	0	0	0	0,17	1,13	9,84	0	0	0
100	0,15	1,31	0	0	0	0	0,15	1,31	10,01	0	0,17	0

Таблиця Б.33

Результати стійкості алгоритму StegoBIT до атаки непропорційного масштабування за віссю ординат (експерименти 13-24)

Номер перетворення	**Експеримент**											
	13	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
	Стиснення											
0	5,77	0	7,92	0	0	0	45,79	11,61	37,69	0	0	0
5	6,36	1,04	8,23	0	0	0	51,55	11,86	38,98	0	0	0
10	10,73	2,03	9,51	1,05	0	0	59,49	13,09	41,55	0	0	0
15	12,61	2,03	10,25	3,37	0	0	61,91	16,44	46,79	0	0	0
20	15,27	2,03	10,80	2,80	0	0	62,11	18,61	49,52	0	0	0
25	17,27	2,03	10,47	4,17	0	0	62,68	20,73	50,16	0	0	0

30	16,92	2,31	11,42	4,33	0	0	64,43	21,01	51,77	0	0	0
35	19,54	2,03	11,90	6,19	0	0	65,39	20,83	55,81	0	0	0
40	23,47	2,38	13,13	5,72	0	0	66,03	21,73	60,41	0	0	0
45	25,40	4,65	14,07	6,25	0	0	66,03	22,55	61,77	0	0	0
50	31,44	6,12	14,07	8,55	0	0	69,15	23,67	63,31	0,76	0	0
55	33,59	6,17	15,54	10,03	0	0	73,13	25,62	66,38	0,35	0	0
60	30,47	6,17	16,39	12,46	0	0	72,93	26,13	67,16	0	0	0
65	35,71	6,37	17,57	13,55	0	0	79,59	30,49	69,23	0,86	0	0
70	36,28	8,28	20,95	17,56	0	0	81,37	34,14	74,93	0,75	0	0
75	36,39	9,73	24,98	17,53	0	0	86,03	39,55	77,89	1,51	0	0
80	36,97	11,43	26,93	16,62	0	0	90,87	45,53	82,43	1,53	0	0
85	40,14	11,51	27,39	18,15	0	0	95,78	47,19	85,35	2,28	0	0
90	45,15	11,53	30,35	18,27	0	0	101,50	51,03	86,79	1,58	0	0
95	44,44	12,60	32,20	18,85	0	0,03	104,95	54,83	91,64	1,28	0	0,03
	Розширення											
0	5,77	0	7,92	0	0	0	45,79	11,61	37,69	0	0	0
5	5,94	0,11	7,09	0	0	0	45,51	9,60	30,87	0	0	0
10	6,97	0,17	4,99	0,18	0	0	42,10	9,64	29,01	0	0	0
15	7,45	0,71	3,67	1,05	0	0	39,36	10,14	27,24	0	0	0
20	11,01	0	3,71	4,71	0	0	36,22	8,65	27,09	0,17	0	0
25	17,01	0	2,02	6,48	0	0	29,60	7,86	26,51	0	0	0
30	22,89	0	1,20	13,89	0	0	26,93	6,13	25,55	0,17	0	0
35	23,59	0	0,89	14,97	0	0	24,41	6,13	23,82	1,00	0	0
40	25,55	0	0,33	25,15	0	0	25,33	4,53	21,45	1,10	0	0
45	34,29	0	0,05	34,67	0	0	26,28	4,13	18,52	2,97	0	0
50	40,41	0	0,26	40,08	0	0	23,26	4,13	16,52	1,77	0	0
55	48,04	0,41	0,79	46,69	0,31	0	21,50	4,71	16,31	3,08	0,31	0
60	53,77	0	0	51,99	0	0	24,94	4,23	16,11	5,50	0	0
65	61,35	0	0	59,09	0	0	28,57	2,10	15,92	9,40	0	0
70	69,65	0	0	63,31	0	0	29,21	2,10	14,54	11,58	0	0
75	77,21	0	0,47	73,04	0	0	32,24	2,10	11,97	18,63	0	0
80	86,09	0	0	80,95	0	0	30,17	2,10	9,31	20,40	0	0
85	98,73	0	0,32	88,67	0,19	0	31,14	2,19	8,52	24,09	0,19	0
90	110,63	0	0	101,44	0	0	37,71	2,10	8,13	27,70	0	0
95	118,51	0,40	0	111,84	0	0	40,30	2,50	7,07	32,23	0	0
100	125,73	0,17	0,23	121,59	0	0	42,59	2,27	6,92	35,03	0,17	0

Таблиця Б.34

Результати стійкості алгоритму StegoBIT до атаки
непропорційного масштабування за віссю ординат (експерименти 25-36)

Номер перетворення	**Експеримент**											
	25	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**	**36**
	Стиснення											
0	151,93	10,20	1,91	65,22	0	0	235,32	51,70	46,69	27,69	0	0
5	198,49	10,20	2,58	118,35	0	0	253,81	53,91	47,05	49,09	0	0
10	227,75	10,20	4,23	165,17	0	0	269,51	55,20	49,49	76,55	0	0
15	257,65	10,33	5,54	187,21	0	0	284,53	61,02	53,03	84,87	0	0
20	277,09	10,20	5,98	207,89	0	0	294,95	65,42	55,21	99,07	0	0
25	288,23	11,24	5,78	220,87	0	0	304,59	67,56	57,65	111,93	0	0
30	288,87	12,80	7,45	228,51	0	0	311,79	70,48	61,49	115,33	0	0
35	297,96	12,80	10,36	235,67	0	0	317,21	71,67	65,83	116,17	0	0
40	297,19	13,23	10,68	242,21	0	0	324,23	75,58	72,43	118,87	0	0
45	314,74	15,44	11,35	251,64	0	0	337,42	80,91	74,62	126,98	0	0
50	330,13	15,45	13,01	258,07	0	0,15	345,02	83,18	79,56	134,65	0	0,15
55	344,36	15,47	15,71	263,82	0	0	359,75	87,67	87,04	139,37	0	0
60	351,67	15,47	16,40	265,33	0	0	367,57	94,07	89,89	132,90	0	0
65	358,01	17,51	17,41	270,21	0	0	378,49	98,83	90,02	140,99	0	0
70	368,77	18,05	21,07	277,07	0	0	384,77	102,20	95,49	148,45	0	0
75	385,30	23,05	24,35	278,66	0	0	391,41	109,17	100,48	149,67	0	0
80	377,73	27,47	27,59	275,22	0	0	393,17	113,06	103,43	148,35	0	0
85	380,21	29,47	29,44	279,97	0	0	406,91	117,45	103,83	153,03	0	0
90	390,05	29,67	34,69	271,53	0	0	411,46	124,09	105,70	152,65	0	0
95	390,78	29,68	36,03	269,70	0	0	414,96	126,20	108,60	159,43	0	0
	Розширення											
0	151,93	10,20	1,91	65,22	0	0	235,32	51,70	46,69	27,69	0	0
5	155,82	10,51	1,50	79,41	0	0	228,69	48,41	43,36	31,66	0,11	0
10	197,31	10,20	0,33	137,66	0	0	238,63	43,55	42,83	55,61	0	0
15	240,61	6,11	0	179,81	0	0	242,64	39,49	40,84	74,39	0	0
20	271,27	6,23	0	213,72	0	0	264,29	33,87	38,27	99,35	0	0
25	292,01	5,23	0	247,27	0	0	278,60	33,87	34,75	126,21	0	0
30	307,78	5,23	0,52	269,67	0	0	285,85	32,41	31,54	145,19	0	0
35	328,77	5,23	0	298,72	0	0	289,39	31,43	30,90	164,69	0	0
40	339,03	5,36	0,41	315,87	0	0	302,09	30,66	29,71	178,69	0	0
45	354,12	5,23	0	337,49	0	0	304,91	29,05	24,53	193,91	0	0
50	378,23	5,23	0	363,90	0	0	307,57	28,83	23,62	222,44	0	0

55	401,85	6,07	0	371,11	0,31	0	318,50	29,26	23,35	231,19	0,31	0
60	414,62	5,65	0,12	383,50	0	0	329,71	29,25	23,43	242,92	0	0
65	422,13	5,23	0	396,61	0	0	338,09	26,14	21,51	260,08	0	0
70	432,09	4,71	0	407,53	0	0	353,97	26,13	17,35	272,14	0	0
75	440,26	3,32	0	421,87	0	0	361,73	26,33	16,63	291,85	0	0
80	452,28	2,65	0	441,48	0	0	366,93	25,15	13,05	300,66	0	0
85	459,67	2,67	0,48	447,74	0	0	380,51	23,09	13,45	312,37	0	0
90	470,74	2,60	0	453,67	0	0	387,05	21,00	12,97	320,29	0	0
95	490,05	2,60	0	472,67	0	0	397,48	18,47	12,57	341,50	0	0
100	491,80	2,95	0	489,35	0	0	399,61	18,82	11,12	358,04	0,17	0

Таблиця Б.35

Результати стійкості алгоритму StegoTEMPL до атаки непропорційного масштабування за віссю ординат (експерименти 1-12)

Номер перетворення	**Експеримент**											
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
	Стиснення											
0	1,89	9,63	1,23	0	0	0	16,65	29,15	7,19	0	0	0
5	2,33	9,19	1,23	0	0	0	18,45	30,66	8,71	0	0	0
10	3,27	9,81	1,74	0	0	0	20	31,93	8,48	0	0	0
15	6,23	10,87	1,49	0	0	0	21,11	37,13	8,52	0	0	0
20	8,89	10,40	1,23	0	0	0	25,32	39,79	8,00	0	0	0
25	10,40	9,64	1,23	0	0	0	27,05	41,16	8,83	0	0	0
30	11,38	10,35	1,49	0	0	0	30,21	45,07	9,84	0	0	0
35	10,26	10,31	1,57	0	0	0	32,03	45,31	9,63	0	0	0
40	9,36	11,12	2,07	0	0	0	34,54	48,86	11,02	0	0	0
45	8,65	11,62	2,92	0	0	0	36,03	50,65	13,32	0	0	0,25
50	10,39	13,74	3,92	0	0	0	38,59	55,37	14,13	0	0	0
55	10,69	15,30	3,65	0	0	0	38,19	60,59	14,21	0	0	0
60	10,95	16,81	3,80	0	0	0	38,73	64,27	15,41	0	0	0
65	11,45	17,11	4,53	0	0	0	41,21	67,94	18,74	0	0	0
70	12,24	17,37	5,30	0	0	0	45,29	68,13	22,41	0	0	0
75	14,61	20,95	4,82	0	0	0	49,91	68,33	24,98	0	0	0
80	18,17	22,43	4,37	0	0	0	55,96	70,49	24,69	0	0	0
85	19,47	23,46	4,40	0	0	0	59,71	73,75	25,87	0	0	0
90	19,96	27,23	5,33	0	0	0	66,77	74,90	28,12	0	0	0
95	20,85	30,84	6,75	0	0	0	69,68	81,78	28,64	0	0	0
	Розширення											
0	1,89	9,63	1,23	0	0	0	16,65	29,15	7,19	0	0	0
5	1,17	8,72	1,23	0	0	0	15,79	26,70	6,40	0	0	0
10	1,12	7,57	1,77	0	0	0	14,87	24,05	8,00	0	0	0
15	0,98	6,75	1,23	0	0	0	14,61	21,30	8,33	0	0	0
20	1,19	6,47	1,23	0	0	0	13,98	19,64	8,22	0,26	0	0
25	1,19	5,77	1,49	0	0	0	13,87	17,47	7,97	0	0	0
30	0,64	5,87	1,23	0	0	0	15,13	17,65	6,32	0	0	0
35	0,71	4,97	1,49	0	0	0	15,67	16,88	6,59	0	0	0,26
40	1,07	5,23	1,23	0	0	0	15,45	16,01	5,97	0	0	0
45	0,45	5,00	1,23	0	0	0	13,77	15,01	6,07	0	0	0
50	0,53	4,23	1,49	0	0	0,19	14,90	15,47	6,77	0	0	0,19
55	0	3,18	1,49	0	0	0	13,76	16,21	7,62	0	0	0
60	0	1,97	1,49	0	0	0	12,91	16,00	6,63	0	0	0
65	0,27	2,23	2,28	0	0	0	13,23	14,72	7,49	0	0	0,26
70	0,26	1,97	2,01	0	0	0	13,01	16,25	7,47	0	0	0
75	0	1,89	1,49	0	0	0	12,02	15,25	7,45	0	0	0
80	0	1,83	1,74	0	0	0	11,08	12,91	7,44	0	0	0
85	0,26	2,09	1,23	0	0,26	0	11,31	12,37	6,79	0	0,26	0
90	0	2,07	1,74	0	0	0,26	11,48	12,52	6,64	0	0	0,26
95	0,77	2,09	1,23	0	0	0	11,85	12,09	5,93	0	0	0
100	0	2,09	1,50	0	0	0	11,06	10,42	6,07	0	0	0

Таблиця Б.36

Результати стійкості алгоритму StegoTEMPL до атаки непропорційного масштабування за віссю ординат (експерименти 13-24)

Номер перетворення	**Експеримент**											
	13	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
	Стиснення											
0	6,11	15,34	0,81	0	0	0	63,03	74,19	22,77	0	0	0
5	7,35	14,69	0,34	0	0	0	68,18	80,14	24,62	0	0	0
10	11,39	18,46	1,25	0	0	0	74,18	82,37	22,87	0	0	0
15	16,41	21,63	1,35	0	0	0	76,13	86,35	23,53	0	0	0
20	15,52	21,43	2,37	0	0	0	80,01	91,35	24,92	0	0	0
25	15,45	20,92	2,98	0	0	0	82,42	93,36	26,41	0	0	0
30	17,06	22,91	5,06	0	0	0	84,31	94,34	27,05	0	0	0
35	17,47	24,03	5,15	0	0	0	83,62	99,23	30,07	0	0	0

40	19,77	26,69	7,61	0	0	0	90,51	100,85	32,77	0	0	0
45	21,02	30,83	8,48	0	0	0	97,65	104,52	37,34	0	0	0
50	22,83	34,93	8,22	0	0	0	103,50	109,27	38,07	0	0	0
55	23,56	35,99	9,19	0	0	0	104,70	116,23	38,93	0	0	0
60	27,20	35,93	9,58	0	0	0	107,54	117,50	40,31	0	0	0
65	26,89	35,77	8,31	0	0	0	110,86	119,30	42,54	0	0	0
70	31,15	37,32	10,11	0	0	0	117,46	121,46	46,83	0	0	0
75	36,61	39,46	11,63	0	0	0	119,77	128,37	49,98	0	0	0
80	42,82	42,41	13,06	0	0	0	124,96	130,65	54,62	0	0	0
85	48,03	44,95	13,99	0	0	0	130,73	132,84	58,43	0	0	0
90	49,49	47,15	16,84	0	0	0	139,62	135,47	56,41	0	0	0
95	51,15	47,83	18,16	0	0	0	142,57	138,89	57,94	0	0	0
	Розширення											
0	6,11	15,34	0,81	0	0	0	63,03	74,19	22,77	0	0	0
5	5,59	15,02	0	0	0	0	60,03	71,47	19,29	0	0	0
10	5,97	14,25	0	0	0	0	56,52	68,51	18,54	0	0	0
15	5,59	13,76	0,42	0	0	0	54,20	64,89	19,87	0	0	0
20	6,85	12,57	0,43	0	0	0	52,40	59,87	19,67	0,42	0	0
25	6,07	10,50	0,83	0	0	0	49,16	55,27	18,83	0	0	0
30	5,25	10,92	0	0	0	0	46,99	49,93	15,31	0	0	0
35	4,96	9,89	0,42	0	0	0	45,05	45,87	16,25	0	0	0,42
40	4,29	9,32	0,40	0	0	0	41,86	44,80	15,73	0	0	0
45	3,09	8,99	0,83	0	0	0	40,85	43,68	14,43	0	0	0
50	2,27	10,45	0	0	0	0	40,39	43,12	13,18	0	0	0
55	1,43	6,87	0	0	0	0	38,08	42,39	12,91	0	0	0
60	0,57	5,61	0,42	0	0,42	0	36,27	42,64	10,75	0	0,42	0
65	0,59	5,17	0,84	0,43	0	0	35,02	40,98	10,07	0,43	0	0,42
70	0,53	4,77	1,26	0	0	0	32,73	37,03	9,53	0	0	0
75	0,97	4,11	0,84	0	0	0	28,86	37,22	7,85	0	0	0
80	1,33	3,67	0,42	0	0	0	30,45	34,85	7,06	0	0	0
85	2,94	3,67	0,83	0	0	0	32,43	30,08	7,52	0	0	0
90	2,53	4,07	0,83	0	0	0	30,10	30,47	7,95	0,42	0	0
95	2,33	4,09	0	0	0	0	28,49	28,54	6,69	0	0	0
100	1,77	4,09	0	0	0	0	25,04	27,57	6,67	0	0	0

Таблиця Б.37

Результати стійкості алгоритму StegoTEMPL до атаки
непропорційного масштабування за віссю ординат (експерименти 25-36)

Номер перетворення	**Експеримент**											
	25	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**	**36**
	Стиснення											
0	20,29	23,75	0,52	0	0	0	127,29	116,11	37,21	0	0	0
5	20,31	29,13	0	0	0	0	133,18	117,63	42,71	0	0	0
10	19,47	33,42	1,05	0	0	0	142,85	120,63	45,78	0	0	0,53
15	27,19	35,56	0	0	0	0	149,37	128,05	46,41	0	0	0
20	29,27	35,33	0,17	0	0,52	0	148,77	135,37	46,23	0	0,52	0
25	29,75	34,87	0,17	0	0	0	155,60	137,93	51,93	0	0	0
30	31,68	35,08	0,69	0	0	0	160,15	140,89	57,66	0	0	0
35	33,75	36,81	1,30	0	0	0	166,49	146,49	61,74	0	0	0
40	37,93	40,03	2,96	0	0	0	170,93	148,60	71,19	0	0	0
45	41,00	48,04	4,53	0	0	0	174,79	152,44	74,98	0	0	0
50	44,28	56,02	7,56	0	0	0,53	176,83	155,41	77,33	0	0	0,53
55	49,10	60,43	8,58	0	0	0	179,31	162,02	82,05	0	0	0
60	54,94	63,72	9,71	0	0	0	186,01	167,22	84,67	0	0	0
65	63,04	62,33	11,75	0	0	0	196,31	168,41	88,25	0	0	0
70	67,55	62,88	12,16	0	0,53	0	208,47	174,29	95,73	0	0,53	0
75	71,17	66,92	14,13	0	0	0	213,34	175,12	96,83	0	0	0
80	74,75	72,06	17,43	0	0	0	223,72	179,51	97,20	0	0	0
85	77,01	71,71	19,29	0	0	0	229,59	181,46	98,81	0	0	0
90	83,15	71,65	20,03	0	0	0	234,93	182,02	102,27	0	0	0
95	84,70	73,20	21,71	0	0	0	235,89	187,43	105,32	0	0	0
	Розширення											
0	20,29	23,75	0,52	0	0	0	127,29	116,11	37,21	0	0	0
5	19,19	23,76	0	0	0	0	122,80	113,71	33,79	0	0	0
10	17,05	21,88	0	0	0	0	114,59	105,68	29,73	0	0	0
15	15,27	21,97	1,59	0	0	0	109,34	95,26	31,55	0	0	0
20	12,14	20,79	0	0	0	0	106,28	90,03	26,57	0,53	0	0
25	9,09	15,15	0	0	0	0	102,35	83,89	23,76	0	0	0
30	7,75	12,24	0,51	0	0	0	94,29	83,07	21,20	0	0	0
35	6,68	12,73	1,05	0	0	0	93,06	81,82	18,25	0	0	0,53
40	6,03	13,26	0,53	0	0	0	88,13	78,01	15,39	0	0	0
45	6,01	11,96	0,52	0	0	0	86,44	73,11	15,18	0	0	0
50	6,04	7,77	1,05	0	0	0	79,17	71,11	16,58	0	0	0
55	6,09	7,70	0,53	0	0	0	78,01	68,12	16,42	0	0	0
60	6,43	8,72	0,53	0	0,53	0	77,84	68,52	14,63	0	0,53	0

65	5,81	8,16	1,05	0	0	0	72,63	63,15	15,12	0	0	0,53
70	5,88	7,67	1,58	0	0	0	68,43	56,45	15,71	0	0	0
75	5,90	8,19	1,05	0	0	0	67,85	54,34	15,28	0	0	0
80	6,94	8,71	0,53	0	0	0	60,67	52,99	13,31	0	0	0
85	8,29	9,22	0,53	0	0	0	58,47	49,96	12,79	0	0	0
90	8,07	8,19	0,51	0	0	0	54,78	49,98	13,27	0	0	0
95	6,45	8,19	1,05	0	0	0	52,14	49,57	13,17	0	0	0,51
100	4,89	8,19	0,53	0	0	0	51,05	49,46	12,47	0	0	0

Результати стійкості алгоритмів StegoBIT та StegoTEMPL до атаки зсуву за віссю абсцис представлені в табл. Б.38-Б.40 та табл. Б.41-Б.42 відповідно.

Таблиця Б.38

Результати стійкості алгоритму StegoBIT до атаки зсуву за віссю абсцис (експерименти 1-12)

Номер перетворення	**Експеримент**											
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
0	0	1,13	1,33	0	0	0	0	3,43	24,13	0	0	0
4	0,33	0	1,43	0	0,23	0	0,33	2,30	23,96	0	0,23	0
8	0,12	0,23	1,35	0	0	0	0,12	2,82	23,88	0	0	0
12	0,12	0	2,81	0,19	0	0	0,12	4,83	24,21	0,19	0	0
16	0	0,02	4,13	0,03	0	0	0	5,48	24,82	0,03	0	0
20	0,02	0	4,81	0	0	0	0,02	6,94	25,29	0	0	0
24	0,18	0,20	5,92	0	0	0	0,18	8,27	26,13	0	0	0
28	0,24	0	7,49	0,08	0	0	0,24	9,10	27,65	0,08	0	0
32	0,09	0,30	8,02	0,09	0	0	0,09	9,40	29,53	0,09	0	0
36	0,41	0,27	9,43	0	0	0	0,41	9,37	32,56	0	0	0
40	0,77	0,18	8,95	0,22	0	0	0,04	10,84	33,77	0	0	0
44	4,66	1,95	12,89	6,80	0,90	1,02	0,68	10,93	35,39	0	0	0,11
48	27,58	12,03	32,19	27,30	13,37	23,43	0,60	11,71	36,88	0	0	0
52	73,72	56,63	88,28	71,71	57,21	74,63	1,23	11,13	38,14	0	0	0
56	134,02	121,03	152,83	140,43	131,18	136,99	1,77	11,38	40,48	0	0,23	0
60	212,77	204,81	229,06	231,52	212,75	215,46	1,54	11,49	41,13	0	0	0,33
64	313,91	296,17	338,44	327,99	306,70	317,83	2,05	11,78	42,64	0,32	0	0
68	402,80	392,25	441,62	419,97	402,28	412,83	1,55	12,14	45,09	0	0	0
72	480,86	470,41	515,74	495,76	485,17	487,89	2,74	12,39	47,04	0	0	0,18
76	545,93	544,15	578,50	551,51	552,47	544,93	1,51	14,32	47,31	0	0,43	0
80	602,89	591,14	618,79	599,93	603,32	592,37	3,58	14,11	47,97	0,45	0,02	0,60
84	650,46	630,63	659,51	645,02	644,50	634,33	5,00	15,83	50,99	4,58	0	0
88	685,82	664,21	689,31	687,31	687,47	673,33	9,44	18,41	54,88	10,71	2,12	3,18
92	721,40	692,42	718,74	720,78	720,75	707,24	25,21	24,43	65,96	21,03	10,01	16,59
96	753,64	725,01	738,60	748,51	742,32	735,48	44,22	40,19	90,16	38,03	22,66	37,84
100	775,68	749,20	758,34	770,17	761,71	759,57	69,89	64,23	116,54	62,42	45,49	64,32

Таблиця Б.39

Результати стійкості алгоритму StegoBIT до атаки зсуву за віссю абсцис (експерименти 13-24)

Номер перетворення	**Експеримент**											
	13	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
0	5,77	0	7,90	0	0	0	45,37	11,53	34,43	0	0	0
4	4,08	0	8,43	0	0	0	43,42	11,53	33,67	0	0	0
8	13,54	0,54	7,90	9,72	0	0	41,06	12,01	32,56	0,62	0	0
12	49,47	0	8,03	41,35	0	0	44,08	11,47	36,99	4,38	0	0
16	94,49	1,11	9,53	77,18	0	0	58,61	12,58	38,18	16,38	0	0
20	142,10	0,95	9,98	121,70	0,22	0	76,73	12,52	39,99	34,27	0,22	0
24	191,43	2,19	10,69	172,36	0	0	98,08	13,79	44,25	53,52	0	0
28	241,83	2,33	11,02	227,61	0	0,06	116,68	13,93	46,61	72,68	0	0,06
32	291,66	2,21	12,52	265,90	0	0,33	137,22	16,68	49,39	89,17	0	0,33
36	341,35	2,64	13,28	302,20	0	0	158,59	19,91	54,71	111,06	0	0
40	381,09	3,39	15,99	341,26	1,97	0,27	182,71	19,68	58,13	137,73	0	0,27
44	423,65	13,53	19,58	380,01	11,19	0	212,07	20	57,59	162,87	0	0
48	474,38	51,05	57,95	435,02	49,69	24,85	235,70	21,00	57,53	192,23	0	0
52	528,33	117,65	133,76	503,61	130,41	97,08	261,15	21,46	57,73	221,36	0	0
56	580,22	232,59	233,93	567,94	230,93	183,36	288,81	23,22	58,64	243,68	0	0
60	630,15	349,40	346,72	617,01	344,28	299,29	313,78	24,97	59,44	260,92	0	0,02
64	675,65	460,68	459,24	668,66	467,91	434,03	338,21	24,99	62,01	277,89	0	0
68	728,73	557,13	564,91	714,68	562,79	534,88	357,66	27,89	65,21	298,25	0,40	0
72	764,68	637,55	636,73	750,81	633,85	619,78	376,40	29,72	70,67	315,71	0	0
76	794,75	703,42	694,28	778,56	690,98	685,92	397,71	31,88	70,90	336,83	0,24	0
80	815,03	751,88	740,99	804,73	742,44	741,55	418,73	35,03	74,59	355,28	3,17	0
84	833,83	786,92	766,88	823,87	777,15	781,69	439,27	38,20	77,12	375,03	7,62	0
88	851,28	813,43	792,68	842,68	809,13	809,43	460,41	50,76	90,02	399,98	18,31	1,01
92	862,80	835,59	812,68	857,06	831,97	830,87	481,90	69,71	107,86	428,28	40,45	14,62
96	872,23	848,92	835,63	872,35	851,68	847,08	509,71	95,62	138,03	458,13	72,78	42,56
100	882,87	860,97	851,69	885,37	869,68	859,79	538,73	125,95	180,01	496,73	116,48	82,78

Таблиця Б.40

Результати стійкості алгоритму StegoBIT до атаки зсуву за віссю абсцис (експерименти 25-36)

Номер перетворення	Експеримент											
	25	26	27	28	29	30	31	32	33	34	35	36
0	123,60	10,20	1,63	31,27	0	0	223,07	51,70	45,60	10,77	0	0
4	174,60	10,76	2,21	100,31	0	0	236,70	50,28	46,00	42,28	0	0
8	320	10,49	2,19	259,80	0	0	289,86	49,36	46,17	130,22	0	0
12	412,57	10,20	1,86	367,97	0	0	366,24	49,72	50,18	224,88	0	0
16	472,59	10,74	3,07	435,06	0	0	412,88	52,90	53,58	298,35	0	0
20	514,76	12,16	3,10	489,13	0,22	0	453,83	55,66	50,35	353,62	0	0
24	552,44	16,10	4,36	527,34	1,18	0	487,74	56,73	52,64	391,42	0	0
28	580,57	18,55	8,09	556,73	5,16	0	513,23	60,40	53,85	422,91	0	0
32	597,52	27,20	10,76	587,78	14,70	0	530,03	65,42	57,18	457,16	0	0
36	620,31	42,31	10,43	611,67	28,00	0	556,61	69,46	61,33	482,53	0	0
40	634,16	65,53	13,12	631,88	57,44	0	577,73	72,47	67,08	507,38	0,43	0
44	656,62	113,85	21,57	658,48	97,39	9,28	597,94	77,18	70,38	522,61	0,87	0,54
48	683,79	179,76	56,25	682,82	177,73	51,78	613,30	77,32	73,68	537,87	2,13	0
52	709,43	269,18	136,03	709,69	264,39	135,72	625,00	82,03	75,10	552,86	4,12	0
56	737,69	369,79	256,33	736,33	357,58	247,43	636,09	86,87	81,43	566,57	7,07	0
60	766,85	460,23	374,48	761,93	455,53	383,96	644,86	94,32	86,50	580	14,50	0,02
64	804,14	542,83	484,53	795,18	552,27	502,84	656,67	97,66	86,76	592,72	18,87	0,16
68	835,18	617,82	596,88	823,79	635,52	600,59	664,37	105,28	89,04	605,09	23,13	0
72	855,93	688,30	662,54	841,71	699,55	669,71	674,66	113,45	87,54	616,64	37,23	0
76	869,75	735,28	712,08	855,84	747,23	716,83	680,45	122,85	88,27	628,10	50,83	0
80	883,72	773,47	755,41	867,87	786,10	759,83	689,62	145,78	91,41	643,53	72,25	0
84	895,53	803,17	794,12	881,12	814,43	794,38	697,69	170,76	96,21	659,51	87,36	3,54
88	906,23	826,14	824,63	890,02	836,93	820,11	709,31	191,93	104,43	670,33	119,02	18,13
92	910,38	847,22	844,17	897,43	849,67	841,75	717,53	228,40	118,04	681,09	168,32	38,38
96	915,98	864,59	858,84	902,20	861,48	857,68	733,30	256,84	148,44	690,29	203,03	73,12
100	920,79	876,32	870,30	908,02	874,93	874,02	746,73	305,54	187,66	704,40	246,51	115,36

Таблиця Б.41

Результати стійкості алгоритму StegoTEMPL до атаки зсуву за віссю абсцис (експерименти 1-12)

Номер перетворення	Експеримент											
	1	2	3	4	5	6	7	8	9	10	11	12
0	1,17	8,97	1,23	0	0	0	15,97	27,77	6,10	0	0	0
4	0,99	8,83	0	0	0	0	15,85	28,18	6,51	0	0	0
8	2,46	9,41	0	0	0	0	16,67	27,54	6,63	0	0	0
12	2,70	9,59	0,64	0	0	0	17,12	31,87	7,93	0	0	0
16	3,27	12,11	0,65	0	0	0	18,49	33,06	9,02	0	0	0
20	4,48	12,77	1,27	0	0,33	0	21,37	33,72	9,68	0	0,33	0
24	5,13	13,50	2,57	0	0	0	24,09	35,68	9,93	0	0	0
28	5,86	15,29	2,89	0	0	0,33	27,23	37,76	11,41	0	0	0,33
32	8,31	16,10	2,88	0	0	0	27,55	40,81	13,75	0	0	0
36	9,68	17,73	2,57	0	0,32	0	28,70	41,57	13,94	0	0	0
40	10,72	19,27	3,15	0	0	0	29,24	42,23	16,33	0	0	0
44	13,63	22,38	9,19	2,23	1,59	1,29	32,38	46,88	20,66	0	0	0
48	38,07	43,78	38,03	27,47	24,38	21,11	33,68	52,52	21,32	0	0	0
52	85,85	101,73	94,53	85,08	78,83	68,95	35,68	55,85	21,98	0	0	0
56	159,03	190,53	165,51	153,86	150,72	132,69	39,40	57,89	22,93	0	0	0
60	253,45	283,06	253,26	231,55	237,27	207,38	39,38	58,24	23,96	0	0	0
64	363,83	380,94	357,82	347,42	343,75	311,93	40,46	59,88	23,46	0,31	0	0
68	480,68	507,98	483,92	461,33	480,33	432,63	42,98	62,08	25,43	0	0	0
72	588,82	608,24	592,01	550,35	574,20	536,21	44,77	63,72	24,95	0	0	0,33
76	662,26	678,94	683,33	637,25	651,46	613,87	44,98	63,85	25,98	0	0	0
80	727,72	740,28	752,57	706,16	718,94	672,42	46,00	64,16	26,80	0,33	0	0
84	779,77	782,97	804,18	773,43	777,22	721,76	48,38	68,61	30,23	0,32	0,64	0,29
88	809,88	828,67	837,45	823,10	815,71	766,13	54,48	75,60	39,83	5,79	4,80	2,59
92	843,78	866,07	861,23	852,08	850,49	806,98	67,09	88,93	55,53	18,91	19,91	15,72
96	876,65	890,47	881,43	880,78	875,75	844,53	89,14	104,41	75,25	43,30	36,23	32,02
100	894,56	908,39	897,38	905,97	897,93	871,73	110,83	139,73	107,13	72,83	65,83	58,43

Таблиця Б.42

Результати стійкості алгоритму StegoTEMPL до атаки зсуву за віссю абсцис (експерименти 13-24)

Номер перетворення	Експеримент											
	13	14	15	16	17	18	19	20	21	22	23	24
0	6,50	16,83	0	0	0	0	63,10	74,17	20,80	0	0	0
4	7,09	16,83	1,05	0	0	0	61,98	73,31	21,31	0	0	0
8	8,75	17,22	0	0	0	0	60,12	74,10	21,77	0	0	0
12	10,74	18,67	0	0	0	0	60,02	74,43	18,43	0	0	0
16	12,80	20,75	0,48	0	0	0	60,73	77,03	18,85	0	0	0
20	16,76	21,28	2,87	0	0	1,01	63,48	78,66	20,75	0	0	1,01

24	20,18	23,23	2,03	0,53	0	0	66,63	81,11	20,23	0,53	0	0
28	19,44	25,85	2,56	0	0	0	64,47	82,63	22,33	0	0	0
32	20,03	26,83	4,11	0,51	0	0	67,06	83,83	25,06	0,51	0	0
36	19,84	26,57	3,09	0	0	0	69,05	89,23	25,88	0	0	0
40	24,01	26,76	2,56	1,55	0	0,52	72,15	92,63	25,43	0	0	0,52
44	37,40	33,66	6,63	15,73	2,62	7,73	73,93	94,31	26,54	0	0	0,53
48	84,87	79,85	32,16	61,75	24,18	40,80	75,93	99,29	30,43	0	0	0
52	162,75	161,02	112,99	171,84	91,55	111,27	78,68	100,26	31,76	0,51	0	0
56	286,22	283,73	230,32	290,49	221,69	224,08	83,03	102,19	34,67	0	0	0
60	420,56	404,21	365,80	412,00	392,93	341,45	88,21	102,63	36,29	0	0	0
64	549,08	526,79	509,30	549,21	537,16	483,73	93,87	106,87	38,92	0	0	0
68	691,43	653,16	652,29	675,53	648,08	633,97	94,80	109,03	40,98	0	0	0
72	790,11	739,26	765,71	781,36	731,78	734,25	98,29	110,89	40,97	0	0	0
76	842,19	800,88	817,21	833,56	815,57	805,42	98,81	111,72	41,53	1,03	0	0
80	883,25	864,53	862,05	883,22	861,00	857,86	104,83	112,18	42,40	3,13	0	0,53
84	920,58	902,57	903,96	916,50	887,90	896,28	115,58	115,68	46,13	11,18	1,58	4,61
88	932,84	934,19	927,43	942,34	910,31	936,55	129,12	124,00	58,75	23,82	8,22	13,92
92	944,12	949,76	940,91	956,44	937,32	951,03	150,98	150,76	76,02	46,13	18,50	32,54
96	950,42	958,04	953,01	963,56	955,94	960,70	181,55	182,59	100,12	93,98	39,78	58,61
100	955,13	966,59	961,36	970,91	965,84	969,31	221,75	218,77	151,74	154,90	73,02	99,80

Таблиця Б.43

Результати стійкості алгоритму StegoTEMPL до атаки зсуву за віссю абсцис (експерименти 25-36)

Номер перетворення	**Експеримент**											
	25	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**	**36**
0	19,37	24,03	0	0	0	0	125,53	113,67	35,33	0	0	0
4	21,83	20,17	0	0	0	0	125,79	108,93	34,68	0	0	0
8	19,51	21,63	0	0	0	0	129,68	110,93	33,98	0	0	0
12	20,63	23,19	0,64	0	0	0	130,23	112,25	32,46	0	0	0
16	25,34	24,68	0,67	0	0	0	131,49	112,98	31,49	0	0	0
20	29,92	27,63	0,67	0	0	0	133,97	114,82	36,08	0	0	0
24	44,95	31,95	3,95	0	0	0	136,52	117,61	36,94	0	0	0
28	54,31	33,56	2,67	0,58	0	0	144,05	123,50	36,08	0	0	0
32	72,76	37,48	6,47	6,22	0	1,30	149,68	123,01	39,33	0	0	0
36	96,32	45,70	14,19	21,18	0	3,26	149,57	124,84	39,13	0	0	0
40	137,38	56,64	23,31	63,03	4,53	16,20	149,38	124,17	42,42	0	0,64	0
44	190,33	97,48	53,41	154,33	32,52	51,37	154,73	124,70	48,26	0	0	0
48	272,33	150,47	111,15	260,54	88,73	107,66	166,45	134,73	49,60	0	0	0
52	396,73	259,24	204,69	388,09	209,93	203,09	172,65	136,88	52,59	0	0,65	0
56	514,34	397,87	345,47	507,88	341,86	359,01	183,53	142,63	55,52	1,15	0	0
60	610,34	541,03	478,40	603,18	478,16	510,93	192,23	137,02	55,48	4,93	0	0,65
64	712,61	681,54	616,45	713,83	604,36	662,58	201,43	139,92	62,16	10,78	0	2,60
68	832,31	812,31	750,53	804,06	745,07	765,78	217,71	142,51	70,18	19,28	0	2,60
72	898,23	871,93	842,35	854,32	834,00	845,38	228,57	149,08	72,51	27,08	0	3,92
76	929,94	905,36	883,95	904,05	886,78	891,12	250,32	154,70	78,70	51,19	2,57	12,29
80	956,43	932,79	920,43	937,36	919,88	927,13	283,69	173,96	83,13	89,89	7,14	24,63
84	966,60	949,52	940,98	952,13	943,61	947,30	306,44	198,88	103,14	138,22	24,62	42,25
88	970,37	960,28	958,58	964,72	955,78	962,49	333,58	213,89	130,31	186,83	46,13	63,75
92	976,93	971,93	964,85	972,13	967,45	972,34	371,63	232,26	167,63	230,09	66,78	90,97
96	978,08	978,23	971,63	980,29	975,57	978,34	411,37	269,80	199,32	294,18	121,22	134,05
100	981,27	982,67	978,95	988,33	978,91	979,63	465,18	323,86	247,56	368,85	187,85	188,54

Результати стійкості алгоритмів StegoBIT та StegoTEMPL до атаки зсуву за віссю ординат представлені в табл. Б.43-Б.45 та табл. Б.46-Б.48 відповідно.

Таблиця Б.43

Результати стійкості алгоритму StegoBIT до атаки зсуву за віссю ординат (експерименти 1-12)

Номер перетворення	**Експеримент**											
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
0	0	1,13	1,33	0	0	0	0	3,43	24,13	0	0	0
4	0	1,13	1,33	0	0	0	0	3,52	24,08	0	0	0
8	0	1,13	1,96	0	0	0	0	3,37	23,10	0	0	0
12	0	1,13	2,91	0	0	0	0	4,98	23,10	0	0	0
16	0	1,13	4,37	0	0	0	0	5,00	25,13	0	0	0
20	0	1,13	4,92	0	0	0	0	6,43	27,65	0	0	0
24	0	1,13	5,00	0	0	0	0	7,83	29,30	0	0	0
28	0	1,13	5,11	0	0	0	0	8,53	30,31	0	0	0
32	0	1,13	3,97	0	0	0	0	8,83	31,39	0	0	0
36	0,61	1,15	5,10	0,16	0	0	0,61	8,83	34,67	0,16	0	0
40	1,03	1,25	6,59	0	0	0	0	9,06	37,53	0	0	0
44	9,40	3,98	8,78	2,31	2,35	0,85	0	9,13	38,03	0	0,05	0
48	43,85	20,03	20,53	21,42	20,28	15,79	0	9,13	38,47	0	0	0
52	96,73	67,86	67,03	67,78	64,58	67,26	0	9,13	39,22	0,09	0	0
56	172,71	144,12	135,97	133,21	137,32	141,70	0,20	9,23	40,48	0	0	0

60	253,03	227,43	212,36	218,18	217,56	226,59	0	9,80	42,48	0	0	0
64	334,85	306,01	300,46	308,13	311,35	308,08	1,99	11,00	45,73	0	0	0
68	423,83	394,40	394,08	404,05	400,70	396,94	2,65	11,74	47,69	0	0	0
72	500,10	464,45	471,17	491,00	473,93	472,51	2,63	12,38	48,14	0	0	0
76	560,53	526,32	536,09	555,62	539,02	543,08	3,31	12,20	49,68	0	0	0
80	611,18	576,00	590,34	617,88	591,30	596,08	5,03	12,80	50,58	0	0	0
84	659,37	620,93	637,65	667,12	641,27	638,73	8,90	14,92	52,09	0,74	0,60	0
88	698,65	658,82	673,06	706,78	679,38	671,17	21,65	17,63	56,13	5,57	5,73	2,40
92	729,29	694,74	707,16	738,57	707,72	707,88	39,82	23,97	60,67	15,36	15,47	8,33
96	758,93	725,95	736,18	765,57	735,45	737,09	58,07	39,44	76,80	31,45	29,58	25,93
100	782,33	750,48	758,93	786,08	757,13	763,52	84,54	66,32	102,68	59,23	52,79	54,77

Таблиця Б.44

Результати стійкості алгоритму StegoBIT до атаки зсуву за віссю ординат (експерименти 13-24)

Номер перетворення	**Експеримент**											
	13	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
0	5,77	0	7,90	0	0	0	45,37	11,53	34,43	0	0	0
4	5,77	0	7,98	0	0	0	45,37	11,00	35,75	0	0	0
8	19,41	0,33	9,31	13,52	0	0	46,00	10,18	35,63	0,70	0	0
12	50,83	0	10,11	50,68	0	0	53,00	10,23	35,63	5,28	0	0
16	88,49	0	10,23	98,98	0	0	66,16	10,81	36,07	19,17	0	0
20	134,64	0	10,87	143,53	0	0	84,43	14,55	40,40	41,63	0	0
24	186,26	0	11,40	185,77	0	0	105,33	15,78	43,00	66,88	0	0
28	233,20	1,50	13,18	234,44	0	0	127,38	16,34	42,55	90,92	0	0
32	280,05	2,10	13,77	279,40	0	0	147,68	16,57	41,80	113,39	0	0
36	319,07	2,10	13,90	318,48	0	0	174,43	17,37	42,38	138,98	0	0
40	360,48	4,31	13,17	357,28	1,08	0	199,28	17,47	42,90	157,88	0	0
44	406,20	19,68	14,64	394,39	14,54	1,78	221,24	17,50	44,60	177,95	0	0
48	460,58	59,45	37,25	438,88	52,81	32,02	243,33	17,50	47,95	202,12	0	0
52	517,76	124,34	109,38	506,68	136,30	97,07	262,26	17,50	50,34	227,19	0	0
56	577,71	231,07	227,01	559,98	242,90	215,57	288,20	18,55	55,43	250,12	0	0
60	628,49	353,43	346,98	618,46	361,88	338,82	312,43	21,70	57,97	272,93	0,33	0
64	690,81	473,35	462,78	668,00	482,81	457,06	330,22	21,70	61,28	291,14	0,28	0
68	733,07	586,68	563,25	708,98	587,38	561,03	348,52	22,23	63,13	313,13	0	0
72	768,51	661,44	657,92	756,56	663,72	634,43	365,83	23,53	63,15	336,04	0	0
76	797,55	718,89	716,98	791,93	716,38	693,81	388,65	28,52	67,03	351,93	0,23	0
80	825,58	760,80	766,01	814,63	759,63	735,21	413,28	32,93	69,14	368,14	2,98	0
84	843,49	798,30	797,63	830,73	793,15	773,38	431,96	38,78	71,47	386,50	9,63	0
88	856,58	823,60	821,86	844,98	818,15	805,60	453,71	55,98	74,16	405,59	21,93	5,48
92	869,96	844,45	839,55	858,89	835,60	831,39	478,61	77,07	86,59	427,63	43,01	18,84
96	886,50	858,27	852,93	872,48	854,13	851,84	506,78	104,88	106,33	461,26	74,12	54,83
100	896,60	869,85	867,49	886,41	871,68	863,33	536,60	136,16	144,03	499,92	122,62	83,59

Таблиця Б.45

Результати стійкості алгоритму StegoBIT до атаки зсуву за віссю ординат (експерименти 25-36)

Номер перетворення	**Експеримент**											
	25	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**	**36**
0	123,60	10,20	1,63	31,27	0	0	223,07	51,70	45,60	10,77	0	0
4	168,67	10,20	1,20	95,36	0	0	234,62	50	43,63	39,30	0	0
8	306,83	10,20	1,97	255,53	0	0	299,00	47,33	46,98	125,37	0	0
12	402,01	10,20	3,48	359,55	0	0	371,53	49,16	48,98	226,23	0	0
16	452,70	10,32	7,48	422,08	0	0	430,86	52,80	51,38	299,34	0	0
20	502,44	9,97	9,14	468,09	0	0	470,88	53,33	53,40	342,77	0	0
24	541,43	12,53	9,44	502,28	0,63	0	494,66	55,28	53,48	382,58	0	0
28	569,16	17,79	9,55	535,63	2,13	0	519,83	57,53	58,98	413,37	0	0
32	597,84	29,61	10,36	566,83	11,23	0	542,02	63,15	60,33	439,32	0	0
36	616,46	45,88	13,53	587,63	26,06	0	561,25	65,76	60,83	461,08	0	0
40	642,98	68,10	13,53	618,53	45,31	0	576,09	65,35	63,62	481,35	0	0
44	663,87	108,41	15,34	643,30	78,34	9,93	593,49	66,62	66,37	497,55	0	0
48	685,74	172,15	51,22	669,43	144,32	55,80	611,71	68,11	69,22	516,02	1,27	0
52	707,95	265,79	139,98	695,55	241,65	159,28	627,08	72,33	70,10	533,38	2,08	0
56	732,36	373,75	256,48	721,40	385,59	292,32	642,79	74,65	71,55	548,86	3,20	0
60	761,83	476,26	394,13	750,78	497,94	426,74	655,71	79,71	74,41	564,59	8,09	0
64	786,30	568,73	503,58	779,74	597,97	530,01	665,33	83,70	75,13	576,23	15,32	0
68	812,65	661,42	595,30	804,63	664,23	616,95	673,54	95,01	77,28	590,54	22,24	0
72	832,28	719,09	667,35	824,70	721,28	689,94	683,63	106,76	78,54	601,41	32,60	0
76	853,15	763,11	707,12	840,78	765,31	732,44	695,85	116,51	81,19	613,53	43,18	0
80	865,97	793,76	748,53	856,97	799,51	769,53	704,43	129,08	82,80	626,30	51,73	0
84	876,73	819,70	784,26	866,85	827,27	807,23	711,68	149,30	84,73	638,61	66,98	4,12
88	887,55	840,48	806,63	876,28	842,05	832,02	719,54	174,94	90,18	651,64	97,08	18,70
92	896,29	855,53	829,72	889,32	859,95	853,03	724,16	208,54	108,40	666,13	130,96	39,43
96	904,24	872,22	846,90	896,51	874,31	869,61	734,43	244,98	145,63	678,97	172,78	78,17
100	910,76	885,48	861,53	902,13	886,27	881,80	746,56	292,63	189,57	692,11	223,78	139,96

Таблиця Б.46

Результати стійкості алгоритму StegoTEMPL до атаки зсуву за віссю ординат (експерименти 1-12)

Номер перетворення	Експеримент											
	1	2	3	4	5	6	7	8	9	10	11	12
0	1,17	8,97	1,23	0	0	0	15,97	27,77	6,10	0	0	0
4	2,07	8,27	1,23	0	0	0	15,97	28,53	7,17	0	0	0
8	2,56	8,84	1,23	0	0	0	17,12	29,38	8,43	0	0	0
12	3,30	11,15	1,23	0	0	0	18,74	31,54	8,73	0	0	0
16	3,88	11,87	1,57	0	0	0	20,77	35,23	9,30	0	0	0
20	6,19	11,87	1,23	0	0	0	21,52	37,58	9,13	0	0	0
24	5,93	11,97	1,23	0	0	0	23,22	39,48	9,33	0	0	0
28	6,37	12,48	1,23	0	0	0,33	23,79	45,23	9,77	0	0	0,33
32	7,91	14,17	2,86	0	0	0	26,27	47,45	9,97	0	0	0
36	8,99	16,47	3,47	0	0	0	27,13	48,83	11,38	0	0	0
40	11,30	17,78	3,71	0	0	0	29,71	49,64	12,43	0	0	0
44	14,64	20,13	6,05	1,60	1,57	0,97	30,90	51,23	12,96	0	0	0
48	41,83	47,28	33,18	23,05	25,79	24,62	33,09	53,22	13,85	0	0	0
52	103,61	101,25	94,94	72,08	85,05	77,31	35,52	57,26	15,22	0	0	0
56	176,78	181,96	172,08	135,15	173,43	172,20	36,54	58,46	15,67	0	0	0
60	265,76	278,18	279,95	239,03	271,95	271,75	38,07	59,48	17,06	0	0	0
64	377,80	390,63	392,50	349,50	370,53	389,56	40,18	61,47	20,19	0	0	0
68	506,15	516,70	510,27	479,38	490,57	506,30	41,27	62,54	24,39	0	0	0
72	602,58	609,68	598,13	589,55	595,21	589,20	44,99	64,88	25,94	0	0	0
76	684,86	694,21	682,35	661,87	688,18	662,91	46,77	66,97	28,71	0	0	0
80	741,53	755,78	735,02	716,03	740,33	724,51	48,98	71,74	30,20	0	0	0
84	799,90	804,16	785,80	764,27	781,52	775,07	50,76	74,79	31,44	0,32	0	0,32
88	840,84	841,41	826,27	816,28	817,86	813,48	57,24	79,24	37,23	3,87	4,73	3,23
92	873,23	875,24	855,70	857,55	842,02	849,35	67,54	96,08	51,13	13,83	17,65	18,07
96	893,74	902,27	889,62	877,06	864,57	882,89	88,48	124,98	77,95	38,14	39,84	38,83
100	915,87	918,43	910,24	889,77	889,68	902,67	118,73	145,92	111,86	63,68	68,08	66,66

Таблиця Б.47

Результати стійкості алгоритму StegoTEMPL до атаки зсуву за віссю ординат (експерименти 13-24)

Номер перетворення	Експеримент											
	13	14	15	16	17	18	19	20	21	22	23	24
0	6,50	16,83	0	0	0	0	63,10	74,17	20,80	0	0	0
4	7,57	14,78	0	0	0	0	62,55	71,98	16,88	0	0	0
8	9,65	17,88	0	0	0	0	63,25	72,38	19,97	0	0	0
12	9,03	19,38	0	0	0	0	66,83	74,33	20,86	0	0	0
16	11,49	19,95	0	0	0	0	71,80	78,08	20,32	0	0	0
20	13,75	20,17	0,15	0	0	0	71,44	81,68	20,28	0	0	0
24	13,52	21,32	0,93	0	0	0	73,40	81,06	21,47	0	0	0
28	13,68	22,89	4,27	0	0	0	74,87	81,23	21,50	0	0	0
32	14,60	25,40	4,79	0	0	0	74,45	86,07	22,61	0	0	0
36	17,28	27,26	6,80	0	0	0	74,83	87,94	24,08	0	0	0
40	19,03	30,68	7,74	3,17	0	0	75,83	89,35	25,88	0	0	0
44	26,83	31,22	12,94	10,89	4,18	2,60	75,42	91,38	26,19	0	0	0
48	74,14	57,98	39,96	52,64	43,59	36,19	78,65	92,16	27,72	0	0,53	0
52	167,92	146,17	119,36	153,12	134,04	133,51	80,86	93,57	29,75	0	0	0
56	281,90	268,89	254,08	264,19	254,50	246,26	84,66	96,03	30,85	0	0	0
60	417,47	394,84	378,74	397,57	386,87	385,20	87,03	97,33	32,90	0	0	0
64	552,20	529,23	514,11	549,10	529,23	514,82	89,73	97,78	35,95	0	0	0
68	672,08	677,57	659,47	688,26	660,49	634,83	91,82	103,24	38,79	0	0	0
72	766,18	757,22	745,85	769,58	759,43	732,41	94,50	104,60	41,04	0	0	0
76	838,01	828,46	802,60	844,29	822,36	824,07	100,44	107,53	42,53	2,13	0,51	0
80	898,73	877,93	853,93	881,00	870,53	873,36	107,37	112,03	45,01	4,70	0	0
84	923,61	901,79	899,86	906,28	909,68	909,03	106,87	116,08	47,29	9,37	1,06	0
88	939,53	924,32	934,94	930,88	928,58	937,50	112,26	126,28	52,07	18,07	6,30	7,29
92	955,24	941,23	955,45	954,20	944,55	946,40	136,53	139,72	71,98	37,03	24,65	29,58
96	964,28	956,16	970,76	968,43	962,11	953,84	166,93	167,63	96,90	79,28	61,26	60,99
100	973,68	970,90	978,65	981,18	971,79	961,32	227,08	216,00	144,00	135,55	105,96	113,82

Таблиця Б.48

Результати стійкості алгоритму StegoTEMPL до атаки зсуву за віссю ординат (експерименти 25-36)

Номер перетворення	Експеримент											
	25	26	27	28	29	30	31	32	33	34	35	36
0	19,37	24,03	0	0	0	0	125,53	113,67	35,33	0	0	0
4	21,06	24,57	0,50	0	0	0	126,76	113,51	33,28	0	0	0
8	24,97	28,02	0	0	0	0	133,24	112,49	33,96	0	0	0
12	28,08	32,19	0	0	0	0	130	115,94	35,03	0	0	0

16	34,35	33,53	0,64	0	0	0	130,73	120,07	33,80	0	0	0
20	37,89	35,13	0	0	0	0	134,43	121,99	33,77	0	0	0
24	43,08	36,38	1,22	0	0	0	137,79	121,25	35,54	0	0	0
28	46,33	45,77	4,77	2,51	0	0	138,92	126,78	37,92	0	0	0
32	51,02	53,73	8,64	10,22	0	0,67	138,58	128,98	40,50	0	0	0
36	89,45	54,41	12,99	29,19	0	3,33	138,31	128,16	43,13	0	0	0
40	143,81	69,81	24,13	85,57	5,79	17,01	140,64	129,42	44,71	0	0	0
44	206,25	97,87	63,73	154,83	24,32	48,83	143,63	133,11	46,81	0	0	0
48	311,42	174,96	131,71	239,86	86,01	136,09	149,73	136,21	50,19	0	0	0
52	428,85	267,42	247,50	353,38	195,75	248,43	152,77	139,64	54,32	0,62	0	0
56	548,21	409,16	389,35	473,25	318,49	391,81	162,04	144,96	56,18	5,73	0	0
60	655,40	531,97	492,44	610,76	472,80	515,68	169,41	147,21	58,80	10,23	0	0
64	753,37	651,30	629,67	710,91	600,81	644,03	180,31	148,58	59,99	15,48	0	2,66
68	835,11	771,76	756,03	813,59	711,97	764,98	196,98	149,99	61,74	22,67	0	2,67
72	902,48	850,43	826,41	875,15	795,14	847,97	227,56	147,63	62,25	48,73	0	5,30
76	929,03	895,20	884,05	911,32	858,55	887,88	241,54	158,29	69,93	70,83	3,18	13,78
80	949,48	927,95	922,48	935,18	893,68	919,59	264,81	174,93	80,28	106,90	6,22	29,39
84	966,15	957,28	941,97	948,97	928,82	946,97	297,33	196,10	98,31	141,95	12,06	41,77
88	976,20	971,14	958,10	966,12	946,53	961,41	323,31	223,16	125,90	183,88	36,68	73,73
92	978,08	979,33	969,05	970,42	963,49	972,95	375,48	261,93	157,93	220,73	66,77	118,64
96	980,92	983,44	981,43	978,24	971,01	980,95	420,93	300,27	203,08	275,46	102,43	159,17
100	983,47	986,45	983,38	983,21	979,20	984,74	475,44	346,67	261,02	326,30	171,18	221,71

Додаток В. Лістинги (коди) програмних засобів

1. Вихідний код ПЗ StegoInSVG-Bitwise

/// StegoBIT.h

```
#pragma once
class BITWISE
{
public:
	int BITWISE_HIDE(struct PARAM param);

	int BITWISE_SHOW(struct PARAM param);
};
```

/// StegoBIT.cpp

```
#include "stdafx.h"
#include <windows.h>
#include "StegoBIT.h"
#include "MATH.h"
#include <iostream>
#include <fstream>
#include <list>
#include <map>
#include <vector>
#include <string>
#include <bitset>

#define UPP 5

struct PARAM
{
	std::string file;
	std::string svg;
	std::string svgname;
	std::string rezult;
	int V1;
	int V2;
	int V3;
	int V4;
	int bit;
	std::string V5;
	std::string dt;
	std::string sp;
};

struct BIT_DATA
{
	int V1;
	int V2;
	int V3;
	int V4;
	int V5;
	iint bit;
	int UP;
	int NumPath;
	int NumBeze;

	std::string StartPoint;
	std::string SVG;
	std::string File;
	std::string Rezult;
	std::string DataBack;
};

struct BIT_INFO
{
	std::list<std::string> last_t;
	std::vector<int> ia;
	std::vector<int> ib;
	std::string dt;

	int Number;
	int SVGSize;
	int NumberPath;
	int NumberLine;
	int HideSize;
	int ZipSVGSize;

	std::list<std::string> FileStructure;
	std::map<int,std::map<int, std::map<int, std::map<int, bool>>>> Path;
};

int BIT_Round (double d)
{
	return static_cast<int> (d + 0.5);
}

std::string BIT_Rounding(std::string A, int V4)
```

```
{
	MATH math;

	if (A.find(".") != std::string::npos && A.substr(A.find(".") + 1).length() > V4)
	{
		std::string B;
		int sign = 0;

		switch(atoi(A.substr(A.find(".") + 1 + V4, 1).c_str()))
		{
		case 0:
		case 1:
		case 2:
		case 3:
		case 4:
			A = A.substr(0, A.find(".")) + "." + A.substr(A.find(".") + 1, V4);
			break;
		case 5:
		case 6:
		case 7:
		case 8:
		case 9:
			B = "0.";

			for (int i = 0; i < A.substr(A.find(".") + 1, V4).length() - 1; i++)
				B = B + "0";

			B = B + "1";

			if (A[0] == '-')
			{
				A = A.substr(1);
				sign = -1;
			}

			A = math.Add(A.substr(0, A.find(".")) + "." + A.substr(A.find(".") + 1, V4), B, V4);

			if (sign == -1)
				A = "-" + A;
		}
	}

	return A;
}

///HIDE DATA
void BIT_AlgorithmCrushing(std::list<std::string> &resultX, std::list<std::string> &resultY,
                                                        std::string &line, std::string t, BIT_DATA &data)
{
	MATH math;

	std::list<std::string>::iterator itX = resultX.begin(), itY = resultY.begin();
	std::string P01X, P01Y, P11X, P11Y, P21X, P21Y, P02X, P02Y, P12X, P12Y, P03X, P03Y;
	std::string sub_t = math.Sub("1", t, data.V4 + data.UP);

	line = line + " C";

	P01X = math.Mul(sub_t, itX->c_str(), data.V4 + data.UP);
	itX++;

	P01Y = math.Mul(sub_t, itY->c_str(), data.V4 + data.UP);
	itY++;

	P01X = math.Add(P01X, math.Mul(t, itX->c_str(), data.V4 + data.UP), data.V4 + data.UP);
	P01Y = math.Add(P01Y, math.Mul(t, itY->c_str(), data.V4 + data.UP), data.V4 + data.UP);

	if (P01X.substr(P01X.find(".") + 1).length() > data.V4)
		P01X = BIT_Rounding(P01X, data.V4);

	if (P01Y.substr(P01Y.find(".") + 1).length() > data.V4)
		P01Y = BIT_Rounding(P01Y, data.V4);

	line = line + P01X + "," + P01Y + " ";

	P11X = math.Mul(sub_t, itX->c_str(), data.V4 + data.UP);
	itX++;

	P11Y = math.Mul(sub_t, itY->c_str(), data.V4 + data.UP);
	itY++;

	P11X = math.Add(P11X, math.Mul(t, itX->c_str(), data.V4 + data.UP), data.V4 + data.UP);
	P11Y = math.Add(P11Y, math.Mul(t, itY->c_str(), data.V4 + data.UP), data.V4 + data.UP);

	P21X = math.Mul(sub_t, itX->c_str(), data.V4 + data.UP);
	itX++;

	P21Y = math.Mul(sub_t, itY->c_str(), data.V4 + data.UP);
	itY++;

	P21X = math.Add(P21X, math.Mul(t, itX->c_str(), data.V4 + data.UP), data.V4 + data.UP);
	P21Y = math.Add(P21Y, math.Mul(t, itY->c_str(), data.V4 + data.UP), data.V4 + data.UP);
```

```
	P02X = math.Mul(sub_t, P01X, data.V4 + data.UP);
	P02Y = math.Mul(sub_t, P01Y, data.V4 + data.UP);
	P02X = math.Add(P02X, math.Mul(t, P11X, data.V4 + data.UP), data.V4 + data.UP);
	P02Y = math.Add(P02Y, math.Mul(t, P11Y, data.V4 + data.UP), data.V4 + data.UP);

	if (P02X.substr(P02X.find(".") + 1).length() > data.V4)
		P02X = BIT_Rounding(P02X, data.V4);

	if (P02Y.substr(P02Y.find(".") + 1).length() > data.V4)
		P02Y = BIT_Rounding(P02Y, data.V4);

	line = line + P02X + "," + P02Y + " ";

	P12X = math.Mul(sub_t, P11X, data.V4 + data.UP);
	P12Y = math.Mul(sub_t, P11Y, data.V4 + data.UP);
	P12X = math.Add(P12X, math.Mul(t, P21X, data.V4 + data.UP), data.V4 + data.UP);
	P12Y = math.Add(P12Y, math.Mul(t, P21Y, data.V4 + data.UP), data.V4 + data.UP);

	P03X = math.Mul(sub_t, P02X, data.V4 + data.UP);
	P03Y = math.Mul(sub_t, P02Y, data.V4 + data.UP);
	P03X = math.Add(P03X, math.Mul(t, P12X, data.V4 + data.UP), data.V4 + data.UP);
	P03Y = math.Add(P03Y, math.Mul(t, P12Y, data.V4 + data.UP), data.V4 + data.UP);

	std::string X = itX->c_str();
	std::string Y = itY->c_str();

	resultX.clear();
	resultY.clear();

	if (P03X.substr(P03X.find(".") + 1).length() > data.V4)
		P03X = BIT_Rounding(P03X, data.V4);

	if (P03Y.substr(P03Y.find(".") + 1).length() > data.V4)
		P03Y = BIT_Rounding(P03Y, data.V4);

	line = line + P03X + "," + P03Y + " ";

	resultX.push_back(P03X);
	resultY.push_back(P03Y);

	resultX.push_back(P12X);
	resultY.push_back(P12Y);

	resultX.push_back(P21X);
	resultY.push_back(P21Y);

	resultX.push_back(X);
	resultY.push_back(Y);
}

int BIT_LineSplit(BIT_DATA &data, BIT_INFO &info, std::list<std::string> X, std::list<std::string> Y, std::string Data,
				std::string &line)
{
	MATH math;
	std::list<std::string> resultX;
	std::list<std::string> resultY;

	std::list<std::string>::iterator itX = X.begin();
	std::list<std::string>::iterator itY = Y.begin();

	for ( ;itX != X.end(); )
	{
		resultX.push_back(itX->c_str());
		resultY.push_back(itY->c_str());

		++itX;
		++itY;
	}

	int CLine = 0;
	std::string t = data.StartPoint;

	for (int i = 0; i < Data.length(); i++)
	{
		if (Data[i] == std::to_string(data.bit)[0])
		{
			BIT_AlgorithmCrushing(resultX, resultY, line, t, data);
			CLine++;
		}

		t = math.Add(t, info.dt, data.V4 + data.UP);
	}

	t = math.Sub(t, info.dt, data.V4 + data.UP);
	info.last_t.push_back(t.c_str());

	line = line + 'C';
	int et = 0;
	for (std::list<std::string>::iterator bX = resultX.begin(), bY = resultY.begin(); bX != resultX.end(); bX++, bY++, et++)
	{
		if (bX == resultX.begin() && bY == resultY.begin())
			continue;
```

```
                line = line + bX->c_str() + "," + bY->c_str();

                if (et < 3)
                        line = line + ' ';
        }
        CLine++;

        resultX.clear();
        resultY.clear();

        return CLine;
}

int BIT_HideInSVG(BIT_DATA &data, BIT_INFO &info, std::string &file, std::list<std::string> &svg)
{
        std::list<std::string> part, Data;
        std::vector<int> num;
        std::list<std::string>::iterator bt;
        std::list<std::string> AddLine, X, Y;
        int MaxLenght = 0, k = 0, l = 0, j = 0;
        bool NextPoint = false;

        if (data.V3 % 8 != 0)
                data.V3 = BIT_Round((double) data.V3 / 8) * 8;

        info.NumberLine = file.size() * 8 / data.V3;

        if (file.size() * 8 % data.V3 != 0)
                info.NumberLine++;

        for (int i = 0; i < info.NumberLine; i++)
        {
                if (file.size() * 8 >= data.V3)
                {
                        part.push_back(file.substr(0, data.V3 / 8));
                        file = file.substr(data.V3 / 8);

                        continue;
                }

                part.push_back(file);
        }

        file.clear();

        for (std::list<std::string>::iterator st = part.begin(); st != part.end(); )
        {
                std::string str;
                for (std::size_t i = 0; i < st->size(); i++)
                {
                        std::bitset<8> bit(st->c_str()[i]);
                        str = str + bit.to_string();
                }

                Data.push_back(str);

                ++st;
        }

        for (std::list<std::string>::iterator st = Data.begin(); st != Data.end(); )
        {
                if (MaxLenght < st->length())
                        MaxLenght = st->length();

                ++st;
        }

        info.NumberPath = info.Path.size();
        info.Number = 0;

        for (std::map<int,std::map<int, std::map<int, std::map<int, bool>>>>::iterator it = info.Path.begin(); it != info.Path.end(); )
        {
                for (std::map<int, std::map<int, bool>>::iterator st = it->second.begin()->second.begin(); st != it->second.begin()->second.end(); )
                {
                        if (st->second.begin()->second == true)
                        {
                                info.ia.push_back(it->second.begin()->first);
                                info.ib.push_back(st->first);
                        }

                        ++st;
                }

                info.Number = info.Number + it->second.begin()->second.size();

                ++it;
        }

        if (info.ia.size() < info.NumberLine)
                return -3;
```

```
bt = Data.begin();

for (std::list<std::string>::iterator itList = info.FileStructure.begin(); itList != info.FileStructure.end(); itList++, l++)
{
	if (l == info.ia[k] && k < info.NumberLine)
	{
		int s = info.ia[k];
		std::string str = itList->c_str();

		std::list<std::string> stm;
		std::string tmp = str.substr(3,str.length() - 4);
		int point = 0;
		while (tmp.find(' ', point) != std::string::npos)
		{
			stm.push_back(tmp.substr(0, tmp.find(' ', point)));
			tmp = tmp.substr(tmp.find(' ', point) + 1, tmp.length() - tmp.find(' ', point) - 1);
			point = tmp.find(' ');
		}

		stm.push_back(tmp);

		int numb = 0;

		while(k < info.NumberLine)
		{
			if (s != info.ia[k])
			{
				std::string st;
				int pi = 0, CLine = 0;
				for (std::list<std::string>::iterator it = stm.begin();it != stm.end(); )
				{
					if (it->find('C') != std::string::npos)
					{
						if (info.ib[k - 1] == CLine)
						{
							CLine++;
							++it;
							++it;
							++it;
							pi = pi + 3;

							for (int t = pi; t < stm.size(); t++)
							{
								st = st + it->c_str() + " ";
								++it;
							}
							break;
						}
						CLine++;
						pi = pi + 3;
						++it;
						++it;
						++it;
						continue;
					}
					pi++;
					++it;
				}

				file.clear();
				file = "d=\"";

				for (std::list<std::string>::iterator it = AddLine.begin(); it != AddLine.end(); )
				{
					file = file + " " + it->c_str();
					++it;
				}

				file = file + " " + st + "\"";

				svg.push_back(file);
				AddLine.clear();

				NextPoint = true;
				j = 0;
				break;
			}

			numb = 0;
			int CLine = 0, n = 0;
			bool ch = false;

			for (std::list<std::string>::iterator it = stm.begin();it != stm.end(); )
			{
				while (n < j && ch == false)
				{
					if (it->find('C') != std::string::npos)
					{
						n++;
						CLine++;
						++it;
						++it;
```

```
                                        ++it;
                                        numb = numb + 3;

                                        if (n == j)
                                                break;

                                        continue;
                                }

                                ++it;
                                numb++;
                        }

                        ch = true;

                        if (it->find('C') != std::string::npos)
                        {
                                if (info.ib[k] != CLine)
                                {
                                        AddLine.push_back(it->c_str());
                                        CLine++;
                                        numb = numb + 3;

                                        ++it;
                                        AddLine.push_back(it->c_str());
                                        ++it;
                                        AddLine.push_back(it->c_str());
                                        ++it;
                                        continue;
                                }

                                it--;
                                if (/*it->find('C') != std::string::npos || it->find('c') != std::string::npos*/
                                        it->find('M') != std::string::npos || it->find('m') != std::string::npos
                                        /*|| it->find('Z') != std::string::npos || it->find('z') != std::string::npos*/
                                        || it->find('L') != std::string::npos || it->find('l') != std::string::npos
                                        || it->find('H') != std::string::npos || it->find('h') != std::string::npos
                                        || it->find('V') != std::string::npos || it->find('v') != std::string::npos
                                        /*|| it->find('S') != std::string::npos || it->find('s') != std::string::npos*/
                                        /*|| it->find('Q') != std::string::npos || it->find('q') != std::string::npos*/
                                        || it->find('T') != std::string::npos || it->find('t') != std::string::npos
                                        /*|| it->find('A') != std::string::npos || it->find('a') != std::string::npos*/)
                                {
                                        X.push_back(it->substr(1,it->find(',')-1));
                                        Y.push_back(it->substr(it->find(',')+1,it->length() - it->find(',') - 1));
                                }
                                else
                                {
                                        X.push_back(it->substr(0,it->find(',')));
                                        Y.push_back(it->substr(it->find(',')+1,it->length() - it->find(',') - 1));
                                }

                                it++;

                                X.push_back(it->substr(1,it->find(',')-1));
                                Y.push_back(it->substr(it->find(',')+1,it->length() - it->find(',') - 1));
                                it++;

                                X.push_back(it->substr(0,it->find(',')));
                                Y.push_back(it->substr(it->find(',')+1,it->length() - it->find(',') - 1));
                                it++;

                                X.push_back(it->substr(0,it->find(',')));
                                Y.push_back(it->substr(it->find(',') + 1,it->length() - it->find(',') - 1));

                                std::string line;

                                num.push_back(BIT_LineSplit(data, info, X, Y, bt->c_str(), line));

                                AddLine.push_back(line);

                                line.clear();
                                X.clear();
                                Y.clear();

                                CLine++;
                                j = CLine;
                                numb = numb + 3;

                                break;
                        }

                        AddLine.push_back(it->c_str());
                        it++;
                        numb++;
                }

                ++bt;
                k++;
        }

        if (NextPoint == true)
```

```
                                {
                                        NextPoint = false;
                                        continue;
                                }

                                bool ch = false;

                                for (std::list<std::string>::iterator ip = stm.begin(); ip != stm.end(); )
                                {
                                        if (numb > 0 && ch == false)
                                        {
                                                numb--;
                                                ++ip;
                                                continue;
                                        }

                                        ch = true;

                                        AddLine.push_back(ip->c_str());

                                        ++ip;
                                }

                                for (int p = 0, i = 0, h = 0, ti = info.ia[0]; p < info.NumberLine; p++)
                                {
                                        if (ti != info.ia[p])
                                        {
                                                h = num[i];
                                                info.ib[p] = info.ib[p] + h;
                                                ti = info.ia[p];
                                                i++;
                                        }
                                        else
                                        {
                                                if (p == 0)
                                                {
                                                        h = num[i];
                                                        info.ib[p] = info.ib[p] + h;
                                                }
                                                else
                                                {
                                                        h = h + num[i] - 1;
                                                        info.ib[p] = info.ib[p] + h;
                                                }

                                                i++;
                                        }
                                }

                                file.clear();
                                file = "d=\"";

                                for (std::list<std::string>::iterator it = AddLine.begin(); it != AddLine.end(); )
                                {
                                        file = file + " " + it->c_str();
                                        ++it;
                                }

                                svg.push_back(file + "\"");
                        }
                        else
                        {
                                svg.push_back(itList->c_str());
                        }
                }

                return 1;
}

void BIT_HideAnalisParametrD(BIT_INFO &info, BIT_DATA &data,
                                                                        std::string line, int NumPath, int Size)
{
        std::string tmp = line.substr(3,line.length() - 4);
        tmp  = tmp.substr(0, tmp.size() - 1);

        std::list<std::string> str;
        int point = 0, Number = 0;
        while (tmp.find(' ', point) != std::string::npos)
        {
                str.push_back(tmp.substr(0, tmp.find(' ', point)));
                tmp = tmp.substr(tmp.find(' ', point) + 1, tmp.length() - tmp.find(' ', point) - 1);
                point = tmp.find(' ');
        }

        str.push_back(tmp);

        std::map<int, std::map<int, bool>> LineChek;
        std::map<int, std::map<int, std::map<int, bool>>> temp;
        std::map<int, bool> BezeParam;

        int i = 0, X[4], Y[4];
```

```
bool ch;

for (std::list<std::string>::iterator it = str.begin(); it != str.end(); )
{
	if (it->c_str()[0] == 'Z' || it->c_str()[0] == 'z')
	{
		++it;
		continue;
	}

	std::string tmpX, tmpY;

	switch (i)
	{
	case 0:
		if (it->c_str()[0] == 'C' || it->c_str()[0] == 'c'
		|| it->c_str()[0] == 'M' || it->c_str()[0] == 'm'
		//|| it->c_str()[0] == 'Z' || it->c_str()[0] == 'z'
		|| it->c_str()[0] == 'L' || it->c_str()[0] == 'l'
		|| it->c_str()[0] == 'H' || it->c_str()[0] == 'h'
		|| it->c_str()[0] == 'V' || it->c_str()[0] == 'v'
		|| it->c_str()[0] == 'S' || it->c_str()[0] == 's'
		|| it->c_str()[0] == 'Q' || it->c_str()[0] == 'q'
		|| it->c_str()[0] == 'T' || it->c_str()[0] == 't'
		|| it->c_str()[0] == 'A' || it->c_str()[0] == 'a')
		{
			tmpX = it->substr(1, it->find(',') - 1).c_str();
			tmpY = it->substr(it->find(',') + 1).c_str();

			X[i] = atoi(tmpX.substr(0, tmpX.find('.')).c_str());
			Y[i] = atoi(tmpY.substr(0, tmpY.find('.')).c_str());

			i++;
		}

		it++;

		break;
	case 1:
		if ((it->c_str()[0] == 'C' || it->c_str()[0] == 'c'))
		{
			tmpX = it->substr(1, it->find(',') - 1).c_str();
			tmpY = it->substr(it->find(',') + 1).c_str();

			X[i] = atoi(tmpX.substr(0, tmpX.find('.')).c_str());
			Y[i] = atoi(tmpY.substr(0, tmpY.find('.')).c_str());

			i++;
		}
		else
		{
			X[0] = X[1] = X[2] = X[3] = 0;
			Y[0] = Y[1] = Y[2] = Y[3] = 0;

			i = 0;

			if ((  //it->c_str()[0] == 'C' || it->c_str()[0] == 'c'
			   it->c_str()[0] == 'M' || it->c_str()[0] == 'm'
			|| it->c_str()[0] == 'Z' || it->c_str()[0] == 'z'
			|| it->c_str()[0] == 'L' || it->c_str()[0] == 'l'
			|| it->c_str()[0] == 'H' || it->c_str()[0] == 'h'
			|| it->c_str()[0] == 'V' || it->c_str()[0] == 'v'
			|| it->c_str()[0] == 'S' || it->c_str()[0] == 's'
			|| it->c_str()[0] == 'Q' || it->c_str()[0] == 'q'
			|| it->c_str()[0] == 'T' || it->c_str()[0] == 't'
			|| it->c_str()[0] == 'A' || it->c_str()[0] == 'a'))
			{
				tmpX = it->substr(1, it->find(',') - 1).c_str();
				tmpY = it->substr(it->find(',') + 1).c_str();

				X[i] = atoi(tmpX.substr(0, tmpX.find('.')).c_str());
				Y[i] = atoi(tmpY.substr(0, tmpY.find('.')).c_str());

				i++;
			}
		}

		it++;
		break;
	case 2:
		tmpX = it->substr(0, it->find(',') - 1).c_str();
		tmpY = it->substr(it->find(',') + 1).c_str();

		X[i] = atoi(tmpX.substr(0, tmpX.find('.')).c_str());
		Y[i] = atoi(tmpY.substr(0, tmpY.find('.')).c_str());

		i++;
		it++;

		break;
	case 3:
		tmpX = it->substr(0, it->find(',') - 1).c_str();
```

```
                                        tmpY = it->substr(it->find(',') + 1).c_str();

                                        X[i] = atoi(tmpX.substr(0, tmpX.find('.')).c_str());
                                        Y[i] = atoi(tmpY.substr(0, tmpY.find('.')).c_str());

                                        i++;

                                        ch = true;

                                        for (i = 0; i < 4; i++)
                                        {
                                                for (int j = i; j < 4; j++)
                                                {
                                                        if (i == j)
                                                                continue;

                                                        if ((int) sqrt((X[i] - X[j]) * (X[i] - X[j]) + (Y[i] - Y[j]) * (Y[i] - Y[j])) < data.V2)
                                                        {
                                                                ch = false;
                                                                i = 4;
                                                                break;
                                                        }
                                                }
                                        }

                                        if (ch == false)
                                        {
                                                BezeParam[2] = ch;
                                                LineChek[Number] = BezeParam;
                                                Number++;
                                        }
                                        else
                                        {
                                                BezeParam[2] = ch;
                                                LineChek[Number] = BezeParam;
                                                Number++;
                                        }

                                        temp[Size] = LineChek;
                                        info.Path[NumPath] = temp;

                                        X[0] = X[3];
                                        Y[0] = Y[3];

                                        X[1] = X[2] = X[3] = 0;
                                        Y[1] = Y[2] = Y[3] = 0;

                                        i = 1;

                                        it++;
                                        break;
                                }
                        }
}

int BIT_AnalisSVGHide(struct BIT_DATA &data, BIT_INFO &info)
{
        std::ifstream strfile(data.SVG, std::ios::in);

        if (strfile.is_open())
        {
                strfile.seekg (0, strfile.end);
                info.SVGSize = strfile.tellg();
                strfile.seekg (0);

                std::string otherLine, line;
                std::list<std::string> someTegPath;
                bool analisysTegPath = false;
                bool findParametr_d = false;
                bool someTegPathFind = false;
                int numberTegPath = 0;

                while (getline(strfile, line))
                {
                        if (line.find("<path") == std::string::npos && analisysTegPath == false && someTegPathFind == false)
                                info.FileStructure.push_back(line);
                        else
                        {
                                if (line.find(">") < line.find("<path") && line.find("<path") != std::string::npos && line.find(">") != std::string::npos ||
line.find("/>") < line.find("<path") && line.find("<path") != std::string::npos && line.find("/>") != std::string::npos)
                                {
                                        if (line.find("/>") < line.find("<path") && line.find("<path") != std::string::npos && line.find("/>") !=
std::string::npos)
                                        {
                                                otherLine = otherLine.substr(otherLine.find("<path"),otherLine.length() - otherLine.find("<path")) +
' ' + line.substr(0,line.find("/>")+2);
                                                someTegPath.push_back(otherLine);
                                                otherLine = line.substr(line.find("<path"),line.length() - line.find("/>"));
                                        }
                                        else
                                        {
                                                info.FileStructure.push_back(line.substr(0,line.find(">")+1));
```

```
                                        otherLine = line.substr(line.find("<path"),line.length() - line.find(">"));
                                }
                                someTegPathFind = true;
                                continue;
                        }

                        if (someTegPathFind = true)
                        {
                                otherLine = otherLine + ' ' + line;

                                if (otherLine.find("/>") != std::string::npos && otherLine.find("<path",otherLine.find("/>")) == std::string::npos)
                                {
                                        someTegPathFind = false;
                                        analisysTegPath = true;
                                        otherLine = otherLine.substr(otherLine.find("<path"),otherLine.length() - otherLine.find("<path"));
                                        someTegPath.push_back(otherLine.substr(0,otherLine.find("/>") + 2));
                                }
                        }

                        if (someTegPathFind == false && analisysTegPath == true)
                        {
                                for(std::list<std::string>::iterator number = someTegPath.begin(); number != someTegPath.end(); number++)
                                {
                                        std::string modeLine = number->c_str();
                                        std::size_t point = modeLine.find("\t");
                                        while (modeLine.find("\t") != std::string::npos)
                                        {
                                                modeLine = modeLine.substr(0,point) + modeLine.substr(point+1,modeLine.length());
                                                point = modeLine.find("\t");
                                        }

                                        info.FileStructure.push_back(modeLine.substr(modeLine.find("<path"), 5));
                                        point = 0;
                                        findParametr_d = true;

                                        while(findParametr_d == true)
                                        {
                                                int g = 1;
                                                if (modeLine.find(" d") != std::string::npos)
                                                {
                                                        if (point == 0)
                                                                point = modeLine.find(" d") + 1;
                                                        else
                                                                point = modeLine.find(" d", point + 2);

                                                        bool findParametr_d2 = true;

                                                        while(findParametr_d2 == true)
                                                        {
                                                                if (modeLine[point + g] == ' ' || modeLine[point + g] == '=')
                                                                {
                                                                        if(modeLine[point + g] == '=')
                                                                        {
                                                                                g++;
                                                                                bool findParametr_d3 = true;
                                                                                while(findParametr_d3 == true)
                                                                                {
                                                                                        if (modeLine[point + g]
== ' ' || modeLine[point + g] == '\"')
                                                                                        {

                if(modeLine[point + g] == '\"')
                                                                                                                {

                info.FileStructure.push_back(modeLine.substr(modeLine.find("<path")+6,point-modeLine.find("<path")-6));

                g++;

                std::string tmp = "d=\"";

                int t = 0, h = 0, l = 0;

                bool structureParametr_d = true;


                while (structureParametr_d == true)

                {

                if (modeLine[point + g] == ' ' || modeLine[point + g] == '\"')

                {

                        if (h > 0)

                                t++;


                        if(modeLine[point + g] == '\"')

                        {
```

```
                        tmp.push_back('\"');
                tmp = tmp.substr(0,3) + tmp.substr(4,tmp.length()-4);
                info.FileStructure.push_back(tmp);
                if (tmp.find("C") != std::string::npos)
                {
                        BIT_HideAnalisParametrD(info, data, tmp, numberTegPath, info.FileStructure.size() - 1);
                        numberTegPath++;
                }
                else
                        numberTegPath++;

                structureParametr_d = false;
        }
}
else
{
        if (isdigit(modeLine[point + g]) || modeLine[point + g] == '.' || modeLine[point + g] == ',' || modeLine[point + g] == '-' || isalpha(modeLine[point + g]))
        {
                if (isalpha(modeLine[point + g]))
                {
                        tmp.push_back(' ');
                        tmp.push_back(modeLine[point + g]);
                        t = 0;h = 0;
                }
                else
                {
                        if(isdigit(modeLine[point + g]))
                                h++;

                        if (modeLine[point + g] == '.' || modeLine[point + g] == ',')
                                h = t = 0;

                        if (t > 0)
                        {
                                tmp.push_back(' ');
                                tmp.push_back(modeLine[point + g]);

                                if (modeLine[point + g + 1] == ' ')
                                        tmp.push_back(' ');

                                t = h = 0;
                        }
                        else
                                tmp.push_back(modeLine[point + g]);
                }
        }
}
g++;
```

```
		}

		info.FileStructure.push_back(modeLine.substr(point + g + 1, modeLine.length() - point - g - (modeLine.length() - modeLine.find("/>") + 1)));

		info.FileStructure.push_back("/>");


		findParametr_d = findParametr_d2 = findParametr_d3 = false;
```

```
		findParametr_d2 = false;

		findParametr_d3 = false;
```

```
	}
}

int BITWISE::BITWISE_HIDE(PARAM param)
{
	BIT_DATA data;
	BIT_INFO info;

	data.V1		=	param.V1;
	data.V2		=	param.V2;
	data.V3		=	param.V3;
	data.V4		=	param.V4;
	data.bit	=	param.bit;
	info.dt		=	param.dt;

	data.StartPoint	=	param.sp;
	data.UP		=	UPP;
	data.NumPath	=	0;
	data.NumBeze	=	0;

	data.SVG	=	param.svg;
	data.File	=	param.file;

	data.Rezult	=	param.rezult + "\\Rezult.svg";
	data.DataBack	=	param.rezult + "\\DataBack.txt";

	std::string file, str;
	std::list<std::string> svg;
	int rez;

	FILE *f = fopen(data.File.c_str(), "rb");

	if (f == NULL)
	{
		fclose(f);
		return 0;
	}

	while(!feof(f))
	{
		char ch;
		fread(&ch, 1, 1, f);
		file = file + ch;
	}
```

```
		fclose(f);

		str = file = file.substr(0, file.size() - 1);

		info.HideSize = file.size();

		rez = BIT_AnalisSVGHide(data, info);

		if (rez != 1)
			return rez;

		rez = BIT_HideInSVG(data, info, file, svg);

		if (rez != 1)
			return rez;

		std::ofstream fileSVG(data.Rezult, std::ios::in | std::ios::trunc);

		for (std::list<std::string>::iterator st = svg.begin(); st != svg.end(); )
		{
			fileSVG << st->c_str() << std::endl;
			++st;
		}

		fileSVG.close();

		std::ofstream DATABACK;

		DATABACK.open(data.DataBack, std::ios::in | std::ios::trunc);

		DATABACK << data.V1 << '|' << data.V2 << '|' << data.V3 << '|' << data.V4 << '|'
			<< info.dt << '|' << data.StartPoint << std::endl;

		std::list<std::string>::iterator st = info.last_t.begin();
		for (int i = 0; i < info.last_t.size(); i++)
		{
			DATABACK << info.ia[i] << '|' << info.ib[i] << '|' << st->c_str() << std::endl;
			++st;
		}

		DATABACK.close();

		info.Path.clear();
		info.last_t.clear();
		info.FileStructure.clear();
		info.dt.clear();
		info.ia.clear();
		info.ib.clear();

		return 1;
}

///SHOW DATA
std::string BIT_AlgorithmCollection (BIT_DATA &data, BIT_INFO info, std::string &Rezult, std::list<std::string> &resultX,
					std::list<std::string> &resultY, std::string &t)
{
		MATH math;

		while (t != data.StartPoint)
		{
			std::list<std::string>::iterator itX = resultX.begin(), itY = resultY.begin();
			std::string P3X, P3Y, P21X, P21Y, P12X, P12Y, P02X, P02Y, P01X, P01Y, P0X, P0Y;
			std::string P03X, P03Y;
			std::string P1X, P1Y, P2X, P2Y, P11X, P11Y, P02X_2, P02Y_2, P02X_3, P02Y_3;
			std::string sub_t = math.Sub("1", t, data.V4 + data.UP);

			P3X = itX->c_str();
			P3Y = itY->c_str();

			itX++;
			itY++;

			P21X = itX->c_str();
			P21Y = itY->c_str();

			itX++;
			itY++;

			P12X = itX->c_str();
			P12Y = itY->c_str();

			itX++;
			itY++;

			P03X = itX->c_str();
			P03Y = itY->c_str();

			itX++;
			itY++;

			P02X = itX->c_str();
```

```
P02Y = itY->c_str();

if (P02X.substr(P02X.find(".") + 1).length() > data.V4)
	P02X = BIT_Rounding(P02X, data.V4);

if (P02Y.substr(P02Y.find(".") + 1).length() > data.V4)
	P02Y = BIT_Rounding(P02Y, data.V4);

itX++;
itY++;

P01X = itX->c_str();
P01Y = itY->c_str();

itX++;
itY++;

P0X = itX->c_str();
P0Y = itY->c_str();

P02X_2 = math.Div(math.Sub(P03X, math.Mul(t, P12X, data.V4 + data.UP), data.V4 + data.UP), sub_t, data.V4 + data.UP);
P02Y_2 = math.Div(math.Sub(P03Y, math.Mul(t, P12Y, data.V4 + data.UP), data.V4 + data.UP), sub_t, data.V4 + data.UP);

P02X_2 = BIT_Rounding(P02X_2, data.V4);
P02Y_2 = BIT_Rounding(P02Y_2, data.V4);

P11X = math.Div(math.Sub(P12X, math.Mul(t, P21X, data.V4 + data.UP), data.V4 + data.UP), sub_t, data.V4 + data.UP);
P11Y = math.Div(math.Sub(P12Y, math.Mul(t, P21Y, data.V4 + data.UP), data.V4 + data.UP), sub_t, data.V4 + data.UP);

P11X = BIT_Rounding(P11X, data.V4);
P11Y = BIT_Rounding(P11Y, data.V4);

P02X_3 = math.Add(math.Mul(sub_t, P01X, data.V4 + data.UP), math.Mul(t, P11X, data.V4 + data.UP), data.V4 + data.UP);
P02Y_3 = math.Add(math.Mul(sub_t, P01Y, data.V4 + data.UP), math.Mul(t, P11Y, data.V4 + data.UP), data.V4 + data.UP);

P02X_3 = BIT_Rounding(P02X_3, data.V4);
P02Y_3 = BIT_Rounding(P02Y_3, data.V4);

bool ch = false;

if (P02X.substr(0, P02X.find('.') - 1) == P02X_2.substr(0, P02X_2.find('.') - 1) && P02X.substr(0, P02X.find('.') - 1) == P02X_3.substr(0,
P02X_3.find('.') - 1))
{
	bool chX = false;
	bool chY = false;

	int len;

	len = data.V4 - P02X.substr(P02X.find('.') + 1).length();
	for (int i = 0; i < len; i++)
		P02X = P02X + "0";

	len = data.V4 - P02X_2.substr(P02X_2.find('.') + 1).length();
	for (int i = 0; i < len; i++)
		P02X_2 = P02X_2 + "0";

	len = data.V4 - P02X_3.substr(P02X_3.find('.') + 1).length();
	for (int i = 0; i < len; i++)
		P02X_3 = P02X_3 + "0";

	int iP02X = atoi(P02X.substr(P02X.find('.') + 1).c_str());
	int iP02X_2 = atoi(P02X_2.substr(P02X_2.find('.') + 1).c_str());
	int iP02X_3 = atoi(P02X_3.substr(P02X_3.find('.') + 1).c_str());

	if (abs(iP02X_2 - iP02X_3) <= data.V5 || abs(iP02X - iP02X_2) <= data.V5 || abs(iP02X - iP02X_3) <= data.V5)
		chX = true;

	if (chX == true)
	{
		int len;

		len = data.V4 - P02Y.substr(P02Y.find('.') + 1).length();
		for (int i = 0; i < len; i++)
			P02Y = P02Y + "0";

		len = data.V4 - P02Y_2.substr(P02Y_2.find('.') + 1).length();
		for (int i = 0; i < len; i++)
			P02Y_2 = P02Y_2 + "0";

		len = data.V4 - P02Y_3.substr(P02Y_3.find('.') + 1).length();
		for (int i = 0; i < len; i++)
			P02Y_3 = P02Y_3 + "0";

		int iP02Y = atoi(P02Y.substr(P02Y.find('.') + 1).c_str());
		int iP02Y_2 = atoi(P02Y_2.substr(P02Y_2.find('.') + 1).c_str());
		int iP02Y_3 = atoi(P02Y_3.substr(P02Y_3.find('.') + 1).c_str());

		if (abs(iP02Y_2 - iP02Y_3) <= data.V5 || abs(iP02Y - iP02Y_2) <= data.V5 || abs(iP02Y - iP02Y_3) <= data.V5)
			chY = true;

		if (chX == true && chY == true)
			ch = true;
```

```
                        }
                }

                if (ch == true)
                {
                        if (data.bit == 1)
                                Rezult = Rezult + "1";
                        else
                                Rezult = Rezult + "0";

                        resultX.clear();
                        resultY.clear();

                        P2X = math.Div(math.Sub(P21X, math.Mul(t, P3X, data.V4 + data.UP), data.V4 + data.UP), sub_t, data.V4 + data.UP);
                        P2Y = math.Div(math.Sub(P21Y, math.Mul(t, P3Y, data.V4 + data.UP), data.V4 + data.UP), sub_t, data.V4 + data.UP);

                        P11X = math.Div(math.Sub(P12X, math.Mul(t, P21X, data.V4 + data.UP), data.V4 + data.UP), sub_t, data.V4 + data.UP);
                        P11Y = math.Div(math.Sub(P12Y, math.Mul(t, P21Y, data.V4 + data.UP), data.V4 + data.UP), sub_t, data.V4 + data.UP);

                        P1X = math.Div(math.Sub(P11X, math.Mul(t, P2X, data.V4 + data.UP), data.V4 + data.UP), sub_t, data.V4 + data.UP);
                        P1Y = math.Div(math.Sub(P11Y, math.Mul(t, P2Y, data.V4 + data.UP), data.V4 + data.UP), sub_t, data.V4 + data.UP);

                        P2X = BIT_Rounding(P2X, data.V4);
                        P2Y = BIT_Rounding(P2Y, data.V4);

                        P1X = BIT_Rounding(P1X, data.V4);
                        P1Y = BIT_Rounding(P1Y, data.V4);

                        resultX.push_back(P3X);resultY.push_back(P3Y);
                        resultX.push_back(P2X);resultY.push_back(P2Y);
                        resultX.push_back(P1X);resultY.push_back(P1Y);
                        resultX.push_back(P0X);resultY.push_back(P0Y);

                        t = math.Sub(t, info.dt, data.V4 + data.UP);

                        return t;
                }

                if (data.bit == 1)
                        Rezult = Rezult + "0";
                else
                        Rezult = Rezult + "1";

                t = math.Sub(t, info.dt, data.V4 + data.UP);
        }

        return t;
}

bool BIT_AnalisLineSplit(BIT_DATA &data, BIT_INFO &info, std::string &Rezult, std::string line, int CNumb, int r, std::string st)
{
        std::list<std::string> str;
        std::string tmp = line.substr(3,line.length() - 4);
        int point = 0, i = 0;
        bool ph = false;

        while (tmp.find(' ', point) != std::string::npos)
        {
                str.push_back(tmp.substr(0, tmp.find(' ', point)));

                if (tmp.substr(0, tmp.find(' ', point)).find('C') != std::string::npos)
                {
                        if (ph == true)
                        {
                                ph = false;
                                break;
                        }
                        else
                                i++;
                }
                tmp = tmp.substr(tmp.find(' ', point) + 1, tmp.length() - tmp.find(' ', point) - 1);
                point = tmp.find(' ');
        }

        if (ph == true)
        {
                if (tmp.find(' ') != std::string::npos)
                        str.push_back(tmp.substr(0, tmp.find(' ', point)));
                else
                        str.push_back(tmp);
        }

        std::list<std::string> stb;
        std::list<std::string>::iterator it = str.begin();
        for (int i = 0; i < CNumb;)
        {
                if (it->find('C') != std::string::npos)
                {
                        stb.push_back(it->c_str());
                        ++it;
                        stb.push_back(it->c_str());
                        ++it;
```

```
                                stb.push_back(it->c_str());
                                ++it;

                                i++;
                                continue;
                        }

                        stb.push_back(it->c_str());
                        ++it;
                }

                bool lastLineBezie = false;

                it = stb.end();

                it--;

                std::string t = st;

                std::list<std::string> resultX, resultY;

                for (it; it != stb.begin();)
                {
                        if (it->find('Z') != std::string::npos || it->find('z') != std::string::npos)
                        {
                                --it;
                                continue;
                        }

                        if (it->find('C') != std::string::npos)
                                resultX.push_back(it->substr(1, it->find(',') - 1));
                        else
                                resultX.push_back(it->substr(0, it->find(',')));

                        resultY.push_back(it->substr(it->find(',') + 1,it->length() - it->find(',') - 1));

                        if (it->find('C') != std::string::npos)
                        {
                                if (lastLineBezie == false)
                                        lastLineBezie = true;
                                else
                                {
                                        it--;
                                        if (/*it->find('C') != std::string::npos || it->find('c') != std::string::npos*/
                                                it->find('M') != std::string::npos || it->find('m') != std::string::npos
                                                /*|| it->find('Z') != std::string::npos || it->find('z') != std::string::npos*/
                                                || it->find('L') != std::string::npos || it->find('l') != std::string::npos
                                                || it->find('H') != std::string::npos || it->find('h') != std::string::npos
                                                || it->find('V') != std::string::npos || it->find('v') != std::string::npos
                                                /*|| it->find('S') != std::string::npos || it->find('s') != std::string::npos*/
                                                /*|| it->find('Q') != std::string::npos || it->find('q') != std::string::npos*/
                                                || it->find('T') != std::string::npos || it->find('t') != std::string::npos
                                                /*|| it->find('A') != std::string::npos || it->find('a') != std::string::npos*/)
                                        {
                                                resultX.push_back(it->substr(1,it->find(',')-1));
                                                resultY.push_back(it->substr(it->find(',')+1,it->length() - it->find(',') - 1));
                                        }
                                        else
                                        {
                                                resultX.push_back(it->substr(0,it->find(',')));
                                                resultY.push_back(it->substr(it->find(',')+1,it->length() - it->find(',') - 1));
                                        }

                                        t = BIT_AlgorithmCollection(data, info, Rezult, resultX, resultY, t);

                                        if (it == stb.begin() || t == data.StartPoint)
                                                break;

                                        --it;
                                        continue;
                                }
                        }

                        --it;
                }

                for (int i = 0; i < Rezult.length() % 8; i++)
                        Rezult = Rezult + '0';

                return 1;
}

int BIT_ShowFromSVG(BIT_DATA &data, BIT_INFO &info, std::string &line)
{
                int NumBeze = 0;
                int i = 0, j = 0, k;
                std::list<std::string> Data, Rezult;
                std::list<std::string>::iterator st = info.last_t.begin();

                if (info.Path.size() == 0)
                        return -3;
```

```
	for (std::map<int,std::map<int, std::map<int, std::map<int, bool>>>>::iterator it = info.Path.begin(); it != info.Path.end(); )
	{
		NumBeze = NumBeze + it->second.begin()->second.size();
		++it;
	}

	if (NumBeze < info.NumberLine)
		return -3;

	for (std::list<std::string>::iterator itList = info.FileStructure.begin(); itList != info.FileStructure.end(); )
	{
		k = info.ia[j];

		if (i == info.ia[j])
		{
			int r;

			if (j == 0)
				r = 0;
			else
				r = info.ib[j - 1];

			std::string Rezult;

			if (!BIT_AnalisLineSplit(data, info, Rezult, itList->c_str(), info.ib[j], r, st->c_str()))
				return 0;

			Data.push_back(Rezult);

			j++;
			++st;
			info.NumberLine--;

			if(info.NumberLine == 0)
				break;

			if (k == info.ia[j])
				continue;
		}

		itList++;
		i++;
	}

	for (std::list<std::string>::iterator it = Data.begin(); it != Data.end(); )
	{
		std::string temp = it->c_str(), str;

		while (!temp.empty())
		{
			int pi = 0;
			for (int k = 0; k < temp.substr(0,8).length(); k++)
			{
				switch(temp.substr(0,8)[k])
				{
				case '1': pi = pi + pow(2, k); break;
				}
			}

			str = str + static_cast<char>(pi);

			temp = temp.substr(8);
		}

		Rezult.push_back(str);

		++it;
	}

	for (std::list<std::string>::iterator it = Rezult.begin(); it != Rezult.end(); )
	{
		for (int i = it->size() - 1; i >= 0; i--)
			line = line + it->c_str()[i];

		++it;
	}

	Data.clear();
	Rezult.clear();

	return 1;
}

void BIT_ShowAnalisParametrD(BIT_DATA &data, BIT_INFO &info, std::string line, int NumPath, int Size)
{
	std::string tmp = line.substr(3,line.length() - 4);
	std::list<std::string> str;
	int point = 0, k = 0, number = 0, i = 0;
	while (tmp.find(' ', point) != std::string::npos)
	{
		str.push_back(tmp.substr(0, tmp.find(' ', point)));
		tmp = tmp.substr(tmp.find(' ', point) + 1, tmp.length() - tmp.find(' ', point) - 1);
```

```
                point = tmp.find(' ');
        }

        str.push_back(tmp);

        std::map<int, std::map<int, bool>> lineBezierCheck;
        std::map<int, bool> lineBezierParametr;

        bool  check = false, firstcheck = false;
        for (std::list<std::string>::iterator it = str.begin();it != str.end(); it++, i++)
        {
                if ((it->find('C') != std::string::npos || it->find('c') != std::string::npos
                        || it->find('M') != std::string::npos || it->find('m') != std::string::npos
                        || it->find('Z') != std::string::npos || it->find('z') != std::string::npos
                        || it->find('L') != std::string::npos || it->find('l') != std::string::npos
                        || it->find('H') != std::string::npos || it->find('h') != std::string::npos
                        || it->find('V') != std::string::npos || it->find('v') != std::string::npos
                        || it->find('S') != std::string::npos || it->find('s') != std::string::npos
                        || it->find('Q') != std::string::npos || it->find('q') != std::string::npos
                        || it->find('T') != std::string::npos || it->find('t') != std::string::npos
                        || it->find('A') != std::string::npos || it->find('a') != std::string::npos) && check == true)
                {
                        if (k-1 == 2 && firstcheck == false)
                        {
                                lineBezierParametr[k-1] = true;
                                lineBezierCheck[number] = lineBezierParametr;
                                number++;
                        }
                        else
                        {
                                lineBezierParametr[k-1] = false;
                                lineBezierCheck[number] = lineBezierParametr;
                                number++;
                        }

                        firstcheck = false;
                        lineBezierParametr.clear();
                        k = 0;
                        if (it->find('C') != std::string::npos)
                                check = true;
                        else
                                check = false;
                }

                if(it->find('C') != std::string::npos && check == false)
                {
                        check = true;
                        if (it == str.begin())
                                firstcheck = true;
                }

                if (check == true && i != str.size() - 1)
                        k++;
        }

        std::map<int, std::map<int, std::map<int, bool>>> temp;
        temp[Size] = lineBezierCheck;

        info.Path[NumPath] = temp;
}

int BIT_AnalisSVGShow(BIT_DATA &data, BIT_INFO &info)
{
        std::ifstream strfile(data.SVG, std::ios::in);

        if (strfile.is_open())
        {
                strfile.seekg (0, strfile.end);
                info.SVGSize = strfile.tellg();
                strfile.seekg (0);

                std::string otherLine, line;
                std::list<std::string> someTegPath;
                bool analisysTegPath = false;
                bool findParametr_d = false;
                bool someTegPathFind = false;
                int numberTegPath = 0;

                while (getline(strfile,line))
                {
                        if (line.find("<path") == std::string::npos && analisysTegPath == false && someTegPathFind == false)
                                info.FileStructure.push_back(line);
                        else
                        {
                                if (line.find(">") < line.find("<path") && line.find("<path") != std::string::npos && line.find(">") != std::string::npos ||
line.find("/>") < line.find("<path") && line.find("<path") != std::string::npos && line.find("/>") != std::string::npos)
                                {
                                        if (line.find("/>") < line.find("<path") && line.find("<path") != std::string::npos && line.find("/>") !=
std::string::npos)
                                        {
                                                otherLine = otherLine.substr(otherLine.find("<path"),otherLine.length() - otherLine.find("<path")) +
' ' + line.substr(0,line.find("/>")+2);
```

```
                                someTegPath.push_back(otherLine);
                                otherLine = line.substr(line.find("<path"),line.length() - line.find("/>"));
                        }
                        else
                        {
                                info.FileStructure.push_back(line.substr(0,line.find(">")+1));
                                otherLine = line.substr(line.find("<path"),line.length() - line.find(">"));
                        }
                        someTegPathFind = true;
                        continue;
                }

                if (someTegPathFind = true)
                {
                        otherLine = otherLine + ' ' + line;

                        if (otherLine.find("/>") != std::string::npos && otherLine.find("<path",otherLine.find("/>")) == std::string::npos)
                        {
                                someTegPathFind = false;
                                analisysTegPath = true;
                                otherLine = otherLine.substr(otherLine.find("<path"),otherLine.length() - otherLine.find("<path"));
                                someTegPath.push_back(otherLine.substr(0,otherLine.find("/>") + 2));
                        }
                }

                if (someTegPathFind == false && analisysTegPath == true)
                {
                        for(std::list<std::string>::iterator number = someTegPath.begin(); number != someTegPath.end(); number++)
                        {
                                std::string modeLine = number->c_str();
                                std::size_t point = modeLine.find("\t");
                                while (modeLine.find("\t") != std::string::npos)
                                {
                                        modeLine = modeLine.substr(0,point) + modeLine.substr(point+1,modeLine.length());
                                        point = modeLine.find("\t");
                                }

                                info.FileStructure.push_back(modeLine.substr(modeLine.find("<path"), 5));
                                point = 0;
                                findParametr_d = true;

                                while(findParametr_d == true)
                                {
                                        int g = 1;
                                        if (modeLine.find(" d") != std::string::npos)
                                        {
                                                if (point == 0)
                                                        point = modeLine.find(" d") + 1;
                                                else
                                                        point = modeLine.find(" d", point + 2);

                                                bool findParametr_d2 = true;

                                                while(findParametr_d2 == true)
                                                {
                                                        if (modeLine[point + g] == ' ' || modeLine[point + g] == '=')
                                                        {
                                                                if(modeLine[point + g] == '=')
                                                                {
                                                                        g++;
                                                                        bool findParametr_d3 = true;
                                                                        while(findParametr_d3 == true)
                                                                        {
                                                                                if (modeLine[point + g]
== ' ' || modeLine[point + g] == \'"')
                                                                                {

                if(modeLine[point + g] == \'"')
                                                                                                {

                info.FileStructure.push_back(modeLine.substr(modeLine.find("<path")+6,point-modeLine.find("<path")-6));

                g++;

                std::string tmp = "d=\"";

                int t = 0, h = 0, l = 0;

                bool structureParametr_d = true;


                while (structureParametr_d == true)

                {

                if (modeLine[point + g] == ' ' || modeLine[point + g] == \'"')

                {

                        if (h > 0)

                                t++;
```

```
		if(modeLine[point + g] == '\"')
		{
			tmp.push_back('\"');
			tmp = tmp.substr(0,3) + tmp.substr(4,tmp.length()-4);
			info.FileStructure.push_back(tmp);
			if (tmp.find("C") != std::string::npos)
			{
				BIT_ShowAnalisParametrD(data, info, tmp, numberTegPath, info.FileStructure.size() - 1);
				numberTegPath++;
			}
			else
				numberTegPath++;

			structureParametr_d = false;
		}
	}
	else
	{
		if (isdigit(modeLine[point + g]) || modeLine[point + g] == '.' || modeLine[point + g] == ',' || modeLine[point + g] == '-' || isalpha(modeLine[point + g]))
		{
			if (isalpha(modeLine[point + g]))
			{
				tmp.push_back(' ');
				tmp.push_back(modeLine[point + g]);
				t = 0;h = 0;
			}
			else
			{
				if(isdigit(modeLine[point + g]))
					h++;

				if (modeLine[point + g] == '.' || modeLine[point + g] == ',')
					h = t = 0;

				if (t > 0)
				{
					tmp.push_back(' ');
					tmp.push_back(modeLine[point + g]);

					if (modeLine[point + g + 1] == ' ')
						tmp.push_back(' ');

					t = h = 0;
				}
				else
					tmp.push_back(modeLine[point + g]);
			}
		}
```

```
                }

        g++;

        }


        info.FileStructure.push_back(modeLine.substr(point + g + 1, modeLine.length() - point - g - (modeLine.length() - modeLine.find("/>") + 1)));

        info.FileStructure.push_back("/>");


        findParametr_d = findParametr_d2 = findParametr_d3 = false;
                                                                                                                                        }
                                                                                                                                        g++;
                                                                                                                                }
                                                                                                                                else
                                                                                                                                {

        findParametr_d2 = false;

        findParametr_d3 = false;
                                                                                                                                        }
                                                                                                                                }
                                                                                                                        }
                                                                                                                        g++;
                                                                                                                }
                                                                                                                else
                                                                                                                        findParametr_d2 = false;
                                                                                                        }
                                                                                                }
                                                                                        }
                                                                                otherLine.clear();
                                                                                someTegPath.clear();
                                                                                analisysTegPath = false;
                                                                                someTegPathFind = false;
                                                                        }
                                                                }
                                                        }

                        strfile.close();

                        if (info.Path.size() == 0)
                                return -2;
        }
        else
        {
                return -1;
        }

        return 1;
}

int BITWISE::BITWISE_SHOW(PARAM param)
{
        BIT_DATA data;
        BIT_INFO info;
        enum TYPE {FIRST, SECOND};
        TYPE type = FIRST;
        std::string str;
        int rez;

        data.V5                 = std::atoi(param.V5.substr(param.V5.length() - 1).c_str()) * 10;
        data.UP                 = UPP;
        data.DataBack           = param.file;
        data.Rezult             = param.rezult + "\\Rezult.txt";
        data.SVG                = param.svg;
        data.bit                = param.bit;

        std::ifstream DATABACK;

        DATABACK.open(data.DataBack, std::ios::in);

        if (!DATABACK.is_open())
                return -4;

        while (getline(DATABACK, str))
        {
                if (type == FIRST)
                {
                        data.V1         =       atoi(str.substr(0,str.find('|')).c_str());
                        str             =       str.substr(str.find('|') + 1);
                        data.V2         =       atoi(str.substr(0,str.find('|')).c_str());
                        str             =       str.substr(str.find('|') + 1);
                        data.V3         =       atoi(str.substr(0,str.find('|')).c_str());
                        str             =       str.substr(str.find('|') + 1);
                        data.V4         =       atoi(str.substr(0,str.find('|')).c_str());
                        str             =       str.substr(str.find('|') + 1);
                        info.dt         =       str.substr(0,str.find('|')).c_str();
                        str             =       str.substr(str.find('|') + 1);
```

```
                                data.StartPoint         =       str.substr(0,str.find('|')).c_str();
                                str.clear();

                                info.NumberLine         = 0;
                                type                    = SECOND;
                                continue;
                        }

                        info.ia.push_back(atoi(str.substr(0,str.find('|')).c_str()));
                        str                             = str.substr(str.find('|') + 1);
                        info.ib.push_back(atoi(str.substr(0,str.find('|')).c_str()));
                        str                             = str.substr(str.find('|') + 1);
                        info.last_t.push_back(str.substr(0,str.find('|')));
                        info.NumberLine++;
        }

        DATABACK.close();
        str.clear();

        rez = BIT_AnalisSVGShow(data, info);

        if (rez <= 0)
                return rez;

        rez = BIT_ShowFromSVG(data, info, str);

        if (rez <= 0)
                return rez;

        FILE *f = fopen(data.Rezult.c_str(), "wb");

        for (int it = 0; it < str.size(); it++)
        {
                        char ch = str.c_str()[it];
                        fwrite(&ch, 1, 1, f);
        }

        fclose(f);

        return 1;
}
```

2. Вихідний код ПЗ StegoInSVG-Template

/// StegoTEMPL.h

```
#pragma once
class TEMPLATE
{
public:
        int TEMPLATE_HIDE(struct PARAM param);

        int TEMPLATE_SHOW(struct PARAM param);
};
```

/// StegoTEMPL.cpp

```
#include "stdafx.h"
#include <windows.h>
#include "StegoTEMPL.h"
#include "MATH.h"
#include <iostream>
#include <fstream>
#include <vector>
#include <list>
#include <map>
#include <string>
#include <bitset>

#define UPP             5

struct PARAM
{
        std::string file;
        std::string svg;
        std::string svgname;
        std::string rezult;
        int V1;
        int V2;
        int V3;
        int V4;
        int K;
        std::string V5;
        std::string TV[16];
        std::string dt[16];
        std::string sp;
};

struct TEMP_DATA
{
        int                                                             V1;
        int                                                             V2;
        int                                                             V3;
```

```
	int V4;
	int V5;
	int L;
	int K;

	int UP;
	int NumPath;
	int NumBeze;

	std::string StartPoint;
	std::string SVG;
	std::string File;
	std::string Rezult;
	std::string DataBack;
};

struct TEMP_INFO
{
	std::list<std::string> last_t;
	std::vector<int> ia;
	std::vector<int> ib;
	std::vector<std::string> Tdt;
	std::vector<std::string> TV;

	int Number;
	double Time;
	int SVGSize;
	int NumberPath;
	int NumberLine;
	int HideSize;

	std::list<std::string> FileStructure;
	std::map<int,std::map<int, std::map<int, std::map<int, bool>>>> Path;
};

int TEMP_Round (double d)
{
	return static_cast<int> (d + 0.5);
}

std::string TEMP_Rounding(std::string A, int V4)
{
	MATH math;

	if (A.find(".") != std::string::npos && A.substr(A.find(".") + 1).length() > V4)
	{
		std::string B;
		int sign = 0;

		switch(atoi(A.substr(A.find(".") + 1 + V4, 1).c_str()))
		{
		case 0:
		case 1:
		case 2:
		case 3:
		case 4:
			A = A.substr(0, A.find(".")) + "." + A.substr(A.find(".") + 1, V4);
			break;
		case 5:
		case 6:
		case 7:
		case 8:
		case 9:
			B = "0.";

			for (int i = 0; i < A.substr(A.find(".") + 1, V4).length() - 1; i++)
				B = B + "0";

			B = B + "1";

			if (A[0] == '-')
			{
				A = A.substr(1);
				sign = -1;
			}

			A = math.Add(A.substr(0, A.find(".")) + "." + A.substr(A.find(".") + 1, V4), B, V4);

			if (sign == -1)
				A = "-" + A;
		}
	}

	return A;
}

///HIDE DATA
void TEMP_AlgorithmCrushing(std::list<std::string> &resultX, std::list<std::string> &resultY,
                            std::string &line, std::string t, TEMP_DATA &data)
{
	MATH math;
```

```
		std::list<std::string>::iterator itX = resultX.begin(), itY = resultY.begin();
		std::string P01X, P01Y, P11X, P11Y, P21X, P21Y, P02X, P02Y, P12X, P12Y, P03X, P03Y;
		std::string sub_t = math.Sub("1", t, data.V4 + data.UP);

		line = line + " C";

		P01X = math.Mul(sub_t, itX->c_str(), data.V4 + data.UP);
		itX++;

		P01Y = math.Mul(sub_t, itY->c_str(), data.V4 + data.UP);
		itY++;

		P01X = math.Add(P01X, math.Mul(t, itX->c_str(), data.V4 + data.UP), data.V4 + data.UP);
		P01Y = math.Add(P01Y, math.Mul(t, itY->c_str(), data.V4 + data.UP), data.V4 + data.UP);

		if (P01X.substr(P01X.find(".") + 1).length() > data.V4)
			P01X = TEMP_Rounding(P01X, data.V4);

		if (P01Y.substr(P01Y.find(".") + 1).length() > data.V4)
			P01Y = TEMP_Rounding(P01Y, data.V4);

		line = line + P01X + "," + P01Y + " ";

		P11X = math.Mul(sub_t, itX->c_str(), data.V4 + data.UP);
		itX++;

		P11Y = math.Mul(sub_t, itY->c_str(), data.V4 + data.UP);
		itY++;

		P11X = math.Add(P11X, math.Mul(t, itX->c_str(), data.V4 + data.UP), data.V4 + data.UP);
		P11Y = math.Add(P11Y, math.Mul(t, itY->c_str(), data.V4 + data.UP), data.V4 + data.UP);

		P21X = math.Mul(sub_t, itX->c_str(), data.V4 + data.UP);
		itX++;

		P21Y = math.Mul(sub_t, itY->c_str(), data.V4 + data.UP);
		itY++;

		P21X = math.Add(P21X, math.Mul(t, itX->c_str(), data.V4 + data.UP), data.V4 + data.UP);
		P21Y = math.Add(P21Y, math.Mul(t, itY->c_str(), data.V4 + data.UP), data.V4 + data.UP);

		P02X = math.Mul(sub_t, P01X, data.V4 + data.UP);
		P02Y = math.Mul(sub_t, P01Y, data.V4 + data.UP);
		P02X = math.Add(P02X, math.Mul(t, P11X, data.V4 + data.UP), data.V4 + data.UP);
		P02Y = math.Add(P02Y, math.Mul(t, P11Y, data.V4 + data.UP), data.V4 + data.UP);

		if (P02X.substr(P02X.find(".") + 1).length() > data.V4)
			P02X = TEMP_Rounding(P02X, data.V4);

		if (P02Y.substr(P02Y.find(".") + 1).length() > data.V4)
			P02Y = TEMP_Rounding(P02Y, data.V4);

		line = line + P02X + "," + P02Y + " ";

		P12X = math.Mul(sub_t, P11X, data.V4 + data.UP);
		P12Y = math.Mul(sub_t, P11Y, data.V4 + data.UP);
		P12X = math.Add(P12X, math.Mul(t, P21X, data.V4 + data.UP), data.V4 + data.UP);
		P12Y = math.Add(P12Y, math.Mul(t, P21Y, data.V4 + data.UP), data.V4 + data.UP);

		P03X = math.Mul(sub_t, P02X, data.V4 + data.UP);
		P03Y = math.Mul(sub_t, P02Y, data.V4 + data.UP);
		P03X = math.Add(P03X, math.Mul(t, P12X, data.V4 + data.UP), data.V4 + data.UP);
		P03Y = math.Add(P03Y, math.Mul(t, P12Y, data.V4 + data.UP), data.V4 + data.UP);

		std::string X = itX->c_str();
		std::string Y = itY->c_str();

		resultX.clear();
		resultY.clear();

		if (P03X.substr(P03X.find(".") + 1).length() > data.V4)
			P03X = TEMP_Rounding(P03X, data.V4);

		if (P03Y.substr(P03Y.find(".") + 1).length() > data.V4)
			P03Y = TEMP_Rounding(P03Y, data.V4);

		line = line + P03X + "," + P03Y + " ";

		resultX.push_back(P03X);
		resultY.push_back(P03Y);

		resultX.push_back(P12X);
		resultY.push_back(P12Y);

		resultX.push_back(P21X);
		resultY.push_back(P21Y);

		resultX.push_back(X);
		resultY.push_back(Y);
}

int TEMP_LineSplit(TEMP_DATA &data, TEMP_INFO &info, std::list<std::string> X, std::list<std::string> Y, std::string Data,
```

```
                                        std::string &line)
{
        MATH math;
        std::list<std::string> resultX;
        std::list<std::string> resultY;

        std::list<std::string>::iterator itX = X.begin();
        std::list<std::string>::iterator itY = Y.begin();

        for ( ;itX != X.end(); )
        {
                resultX.push_back(itX->c_str());
                resultY.push_back(itY->c_str());

                ++itX;
                ++itY;
        }

        int CLine = 0;
        std::string t = data.StartPoint;

        while (Data.length() > 0)
        {
                TEMP_AlgorithmCrushing(resultX, resultY, line, t, data);
                CLine++;

                for (int j = 0; j < data.K; j++)
                {
                        if (info.TV[j] == Data.substr(0, data.L))
                        {
                                t = math.Add(t, info.Tdt[j], data.V4 + data.UP);
                                break;
                        }
                }

                Data = Data.substr(data.L);
        }

        info.last_t.push_back(t.c_str());

        line = line + 'C';
        int et = 0;
        for (std::list<std::string>::iterator bX = resultX.begin(), bY = resultY.begin(); bX != resultX.end(); bX++, bY++, et++)
        {
                if (bX == resultX.begin() && bY == resultY.begin())
                        continue;

                line = line + bX->c_str() + "," + bY->c_str();

                if (et < 3)
                        line = line + ' ';
        }
        CLine++;

        resultX.clear();
        resultY.clear();

        return CLine;
}

int TEMP_HideInSVG(TEMP_DATA &data, TEMP_INFO &info, std::string &file, std::list<std::string> &svg)
{
        std::list<std::string> part, Data;
        std::vector<int> num;
        std::list<std::string>::iterator bt;
        std::list<std::string> AddLine, X, Y;
        int MaxLenght = 0, k = 0, l = 0, j = 0;
        std::string dt, step;
        std::vector<int> bit;
        bool NextPoint = false;

        if (data.V3 % 8 != 0)
                data.V3 = TEMP_Round((double) data.V3 / 8) * 8;

        info.NumberLine = file.size() * 8 / data.V3;

        if (file.size() * 8 % data.V3 != 0)
                info.NumberLine++;

        for (int i = 0; i < info.NumberLine; i++)
        {
                if (file.size() * 8 >= data.V3)
                {
                        part.push_back(file.substr(0, data.V3 / 8));
                        file = file.substr(data.V3 / 8);

                        continue;
                }

                part.push_back(file);
        }
```

```
file.clear();

for (std::list<std::string>::iterator st = part.begin(); st != part.end(); )
{
	std::string str;
	for (std::size_t i = 0; i < st->size(); i++)
	{
		std::bitset<8> bit(st->c_str()[i]);
		str = str + bit.to_string();
	}

	Data.push_back(str);

	++st;
}

for (std::list<std::string>::iterator st = Data.begin(); st != Data.end(); )
{
	if (MaxLenght < st->length())
		MaxLenght = st->length();

	++st;
}

info.NumberPath = info.Path.size();
info.Number = 0;

for (std::map<int,std::map<int, std::map<int, std::map<int, bool>>>>::iterator it = info.Path.begin(); it != info.Path.end(); )
{
	for (std::map<int, std::map<int, bool>>::iterator st = it->second.begin()->second.begin(); st != it->second.begin()->second.end(); )
	{
		if (st->second.begin()->second == true)
		{
			info.ia.push_back(it->second.begin()->first);
			info.ib.push_back(st->first);
		}

		++st;
	}

	info.Number = info.Number + it->second.begin()->second.size();

	++it;
}

if (info.ia.size() < info.NumberLine)
	return -3;

bt = Data.begin();

for (std::list<std::string>::iterator itList = info.FileStructure.begin(); itList != info.FileStructure.end(); itList++, l++)
{
	if (l == info.ia[k] && k < info.NumberLine)
	{
		int s = info.ia[k];
		std::string str = itList->c_str();

		std::list<std::string> stm;
		std::string tmp = str.substr(3,str.length() - 4);
		int point = 0;
		while (tmp.find(' ', point) != std::string::npos)
		{
			stm.push_back(tmp.substr(0, tmp.find(' ', point)));
			tmp = tmp.substr(tmp.find(' ', point) + 1, tmp.length() - tmp.find(' ', point) - 1);
			point = tmp.find(' ');
		}

		stm.push_back(tmp);

		int numb = 0;

		while(k < info.NumberLine)
		{
			if (s != info.ia[k])
			{
				std::string st;
				int pi = 0, CLine = 0;
				for (std::list<std::string>::iterator it = stm.begin();it != stm.end(); )
				{
					if (it->find('C') != std::string::npos)
					{
						if (info.ib[k - 1] == CLine)
						{
							CLine++;
							++it;
							++it;
							++it;
							pi = pi + 3;

							for (int t = pi; t < stm.size(); t++)
							{
								st = st + it->c_str() + " ";
```

```
                                                ++it;
                                        }
                                        break;
                                }
                                CLine++;
                                pi = pi + 3;
                                ++it;
                                ++it;
                                ++it;
                                continue;
                        }
                        pi++;
                        ++it;
                }

                file.clear();
                file = "d=\"";

                for (std::list<std::string>::iterator it = AddLine.begin(); it != AddLine.end(); )
                {
                        file = file + " " + it->c_str();
                        ++it;
                }

                file = file + " " + st + "\"";

                svg.push_back(file);
                AddLine.clear();

                NextPoint = true;
                j = 0;
                break;
        }

        numb = 0;
        int CLine = 0, n = 0;
        bool ch = false;

        for (std::list<std::string>::iterator it = stm.begin();it != stm.end(); )
        {
                while (n < j && ch == false)
                {
                        if (it->find('C') != std::string::npos)
                        {
                                n++;
                                CLine++;
                                ++it;
                                ++it;
                                ++it;
                                numb = numb + 3;

                                if (n == j)
                                        break;

                                continue;
                        }

                        ++it;
                        numb++;
                }

                ch = true;

                if (it->find('C') != std::string::npos)
                {
                        if (info.ib[k] != CLine)
                        {
                                AddLine.push_back(it->c_str());
                                CLine++;
                                numb = numb + 3;

                                ++it;
                                AddLine.push_back(it->c_str());
                                ++it;
                                AddLine.push_back(it->c_str());
                                ++it;
                                continue;
                        }

                        it--;
                        if (/*it->find('C') != std::string::npos || it->find('c') != std::string::npos*/
                                it->find('M') != std::string::npos || it->find('m') != std::string::npos
                                /*|| it->find('Z') != std::string::npos || it->find('z') != std::string::npos*/
                                || it->find('L') != std::string::npos || it->find('l') != std::string::npos
                                || it->find('H') != std::string::npos || it->find('h') != std::string::npos
                                || it->find('V') != std::string::npos || it->find('v') != std::string::npos
                                /*|| it->find('S') != std::string::npos || it->find('s') != std::string::npos*/
                                /*|| it->find('Q') != std::string::npos || it->find('q') != std::string::npos*/
                                || it->find('T') != std::string::npos || it->find('t') != std::string::npos
                                /*|| it->find('A') != std::string::npos || it->find('a') != std::string::npos*/)
                        {
                                X.push_back(it->substr(1,it->find(',')-1));
```

```
                                        Y.push_back(it->substr(it->find(',')+1,it->length() - it->find(',') - 1));
                                }
                                else
                                {
                                        X.push_back(it->substr(0,it->find(',')));
                                        Y.push_back(it->substr(it->find(',')+1,it->length() - it->find(',') - 1));
                                }

                                it++;

                                X.push_back(it->substr(1,it->find(',')-1));
                                Y.push_back(it->substr(it->find(',')+1,it->length() - it->find(',') - 1));
                                it++;

                                X.push_back(it->substr(0,it->find(',')));
                                Y.push_back(it->substr(it->find(',')+1,it->length() - it->find(',') - 1));
                                it++;

                                X.push_back(it->substr(0,it->find(',')));
                                Y.push_back(it->substr(it->find(',') + 1,it->length() - it->find(',') - 1));

                                std::string line;

                                num.push_back(TEMP_LineSplit(data, info, X, Y, bt->c_str(), line));

                                AddLine.push_back(line);

                                line.clear();
                                X.clear();
                                Y.clear();

                                CLine++;
                                j = CLine;
                                numb = numb + 3;

                                break;
                        }

                        AddLine.push_back(it->c_str());
                        it++;
                        numb++;
                }

                ++bt;
                k++;
        }

        if (NextPoint == true)
        {
                NextPoint = false;
                continue;
        }

        bool ch = false;

        for (std::list<std::string>::iterator ip = stm.begin(); ip != stm.end(); )
        {
                if (numb > 0 && ch == false)
                {
                        numb--;
                        ++ip;
                        continue;
                }

                ch = true;

                AddLine.push_back(ip->c_str());

                ++ip;
        }

        for (int p = 0, i = 0, h = 0, ti = info.ia[0]; p < info.NumberLine; p++)
        {
                if (ti != info.ia[p])
                {
                        h = num[i];
                        info.ib[p] = info.ib[p] + h;
                        ti = info.ia[p];
                        i++;
                }
                else
                {
                        if (p == 0)
                        {
                                h = num[i];
                                info.ib[p] = info.ib[p] + h;
                        }
                        else
                        {
                                h = h + num[i] - 1;
                                info.ib[p] = info.ib[p] + h;
                        }
```

```
                                                                i++;
                                                }
                                        }

                                        file.clear();
                                        file = "d=\"";

                                        for (std::list<std::string>::iterator it = AddLine.begin(); it != AddLine.end(); )
                                        {
                                                file = file + " " + it->c_str();
                                                ++it;
                                        }

                                        svg.push_back(file + "\"");
                                }
                                else
                                {
                                        svg.push_back(itList->c_str());
                                }
                        }

                        return 1;
}

void TEMP_HideAnalisParametrD(TEMP_INFO &info, TEMP_DATA &data,
                                                                        std::string line, int NumPath, int Size)
{
                        std::string tmp = line.substr(3,line.length() - 4);
                        tmp  = tmp.substr(0, tmp.size() - 1);

                        std::list<std::string> str;
                        int point = 0, Number = 0;
                        while (tmp.find(' ', point) != std::string::npos)
                        {
                                str.push_back(tmp.substr(0, tmp.find(' ', point)));
                                tmp = tmp.substr(tmp.find(' ', point) + 1, tmp.length() - tmp.find(' ', point) - 1);
                                point = tmp.find(' ');
                        }

                        str.push_back(tmp);

                        std::map<int, std::map<int, bool>> LineChek;
                        std::map<int, std::map<int, std::map<int, bool>>> temp;
                        std::map<int, bool> BezeParam;

                        int i = 0, X[4], Y[4];

                        bool ch;

                        for (std::list<std::string>::iterator it = str.begin(); it != str.end(); )
                        {
                                if (it->c_str()[0] == 'Z' || it->c_str()[0] == 'z')
                                {
                                        ++it;
                                        continue;
                                }

                                std::string tmpX, tmpY;

                                switch (i)
                                {
                                case 0:
                                        if (it->c_str()[0] == 'C' || it->c_str()[0] == 'c'
                                        || it->c_str()[0] == 'M' || it->c_str()[0] == 'm'
                                        //|| it->c_str()[0] == 'Z' || it->c_str()[0] == 'z'
                                        || it->c_str()[0] == 'L' || it->c_str()[0] == 'l'
                                        || it->c_str()[0] == 'H' || it->c_str()[0] == 'h'
                                        || it->c_str()[0] == 'V' || it->c_str()[0] == 'v'
                                        || it->c_str()[0] == 'S' || it->c_str()[0] == 's'
                                        || it->c_str()[0] == 'Q' || it->c_str()[0] == 'q'
                                        || it->c_str()[0] == 'T' || it->c_str()[0] == 't'
                                        || it->c_str()[0] == 'A' || it->c_str()[0] == 'a')
                                        {
                                                tmpX = it->substr(1, it->find(',') - 1).c_str();
                                                tmpY = it->substr(it->find(',') + 1).c_str();

                                                X[i] = atoi(tmpX.substr(0, tmpX.find('.')).c_str());
                                                Y[i] = atoi(tmpY.substr(0, tmpY.find('.')).c_str());

                                                i++;
                                        }

                                        it++;

                                        break;
                                case 1:
                                        if ((it->c_str()[0] == 'C' || it->c_str()[0] == 'c'))
                                        {
                                                tmpX = it->substr(1, it->find(',') - 1).c_str();
                                                tmpY = it->substr(it->find(',') + 1).c_str();
```

```
                X[i] = atoi(tmpX.substr(0, tmpX.find('.')).c_str());
                Y[i] = atoi(tmpY.substr(0, tmpY.find('.')).c_str());

                i++;
        }
        else
        {
                X[0] = X[1] = X[2] = X[3] = 0;
                Y[0] = Y[1] = Y[2] = Y[3] = 0;

                i = 0;

                if ((  //it->c_str()[0] == 'C' || it->c_str()[0] == 'c'
                   it->c_str()[0] == 'M' || it->c_str()[0] == 'm'
                || it->c_str()[0] == 'Z' || it->c_str()[0] == 'z'
                || it->c_str()[0] == 'L' || it->c_str()[0] == 'l'
                || it->c_str()[0] == 'H' || it->c_str()[0] == 'h'
                || it->c_str()[0] == 'V' || it->c_str()[0] == 'v'
                || it->c_str()[0] == 'S' || it->c_str()[0] == 's'
                || it->c_str()[0] == 'Q' || it->c_str()[0] == 'q'
                || it->c_str()[0] == 'T' || it->c_str()[0] == 't'
                || it->c_str()[0] == 'A' || it->c_str()[0] == 'a'))
                {
                        tmpX = it->substr(1, it->find(',') - 1).c_str();
                        tmpY = it->substr(it->find(',') + 1).c_str();

                        X[i] = atoi(tmpX.substr(0, tmpX.find('.')).c_str());
                        Y[i] = atoi(tmpY.substr(0, tmpY.find('.')).c_str());

                        i++;
                }
        }

        it++;
        break;
case 2:
        tmpX = it->substr(0, it->find(',') - 1).c_str();
        tmpY = it->substr(it->find(',') + 1).c_str();

        X[i] = atoi(tmpX.substr(0, tmpX.find('.')).c_str());
        Y[i] = atoi(tmpY.substr(0, tmpY.find('.')).c_str());

        i++;
        it++;

        break;
case 3:
        tmpX = it->substr(0, it->find(',') - 1).c_str();
        tmpY = it->substr(it->find(',') + 1).c_str();

        X[i] = atoi(tmpX.substr(0, tmpX.find('.')).c_str());
        Y[i] = atoi(tmpY.substr(0, tmpY.find('.')).c_str());

        i++;

        ch = true;

        for (i = 0; i < 4; i++)
        {
                for (int j = i; j < 4; j++)
                {
                        if (i == j)
                                continue;

                        if ((int) sqrt((X[i] - X[j]) * (X[i] - X[j]) + (Y[i] - Y[j]) * (Y[i] - Y[j])) < data.V2)
                        {
                                ch = false;
                                i = 4;
                                break;
                        }
                }
        }

        if (ch == false)
        {
                BezeParam[2] = ch;
                LineChek[Number] = BezeParam;
                Number++;
        }
        else
        {
                BezeParam[2] = ch;
                LineChek[Number] = BezeParam;
                Number++;
        }

        temp[Size] = LineChek;
        info.Path[NumPath] = temp;

        X[0] = X[3];
        Y[0] = Y[3];
```

```
                                        X[1] = X[2] = X[3] = 0;
                                        Y[1] = Y[2] = Y[3] = 0;

                                        i = 1;

                                        it++;
                                        break;
                                }
                        }
}

int TEMP_AnalisSVGHide(struct TEMP_DATA &data, TEMP_INFO &info)
{
        std::ifstream strfile(data.SVG, std::ios::in);

        if (strfile.is_open())
        {
                strfile.seekg (0, strfile.end);
                info.SVGSize = strfile.tellg();
                strfile.seekg (0);

                std::string otherLine, line;
                std::list<std::string> someTegPath;
                bool analisysTegPath = false;
                bool findParametr_d = false;
                bool someTegPathFind = false;
                int numberTegPath = 0;

                while (getline(strfile, line))
                {
                        if (line.find("<path") == std::string::npos && analisysTegPath == false && someTegPathFind == false)
                                info.FileStructure.push_back(line);
                        else
                        {
                                if (line.find(">") < line.find("<path") && line.find("<path") != std::string::npos && line.find(">") != std::string::npos ||
line.find("/>") < line.find("<path") && line.find("<path") != std::string::npos && line.find("/>") != std::string::npos)
                                {
                                        if (line.find("/>") < line.find("<path") && line.find("<path") != std::string::npos && line.find("/>") !=
std::string::npos)
                                        {
                                                otherLine = otherLine.substr(otherLine.find("<path"),otherLine.length() - otherLine.find("<path")) +
' ' + line.substr(0,line.find("/>")+2);
                                                someTegPath.push_back(otherLine);
                                                otherLine = line.substr(line.find("<path"),line.length() - line.find("/>"));
                                        }
                                        else
                                        {
                                                info.FileStructure.push_back(line.substr(0,line.find(">")+1));
                                                otherLine = line.substr(line.find("<path"),line.length() - line.find(">"));
                                        }
                                        someTegPathFind = true;
                                        continue;
                                }

                                if (someTegPathFind = true)
                                {
                                        otherLine = otherLine + ' ' + line;

                                        if (otherLine.find("/>") != std::string::npos && otherLine.find("<path",otherLine.find("/>")) == std::string::npos)
                                        {
                                                someTegPathFind = false;
                                                analisysTegPath = true;
                                                otherLine = otherLine.substr(otherLine.find("<path"),otherLine.length() - otherLine.find("<path"));
                                                someTegPath.push_back(otherLine.substr(0,otherLine.find("/>") + 2));
                                        }
                                }

                                if (someTegPathFind == false && analisysTegPath == true)
                                {
                                        for(std::list<std::string>::iterator number = someTegPath.begin(); number != someTegPath.end(); number++)
                                        {
                                                std::string modeLine = number->c_str();
                                                std::size_t point = modeLine.find("\t");
                                                while (modeLine.find("\t") != std::string::npos)
                                                {
                                                        modeLine = modeLine.substr(0,point) + modeLine.substr(point+1,modeLine.length());
                                                        point = modeLine.find("\t");
                                                }

                                                info.FileStructure.push_back(modeLine.substr(modeLine.find("<path"), 5));
                                                point = 0;
                                                findParametr_d = true;

                                                while(findParametr_d == true)
                                                {
                                                        int g = 1;
                                                        if (modeLine.find(" d") != std::string::npos)
                                                        {
                                                                if (point == 0)
                                                                        point = modeLine.find(" d") + 1;
                                                                else
                                                                        point = modeLine.find(" d", point + 2);
```

```
                                                                                bool findParametr_d2 = true;

                                                                                while(findParametr_d2 == true)
                                                                                {
                                                                                        if (modeLine[point + g] == ' ' || modeLine[point + g] == '=')
                                                                                        {
                                                                                                if(modeLine[point + g] == '=')
                                                                                                {
                                                                                                        g++;
                                                                                                        bool findParametr_d3 = true;
                                                                                                        while(findParametr_d3 == true)
                                                                                                        {
                                                                                                                if (modeLine[point + g]
== ' ' || modeLine[point + g] == '\"')
                                                                                                                {

        if(modeLine[point + g] == '\"')
                                                                                                                                {

        info.FileStructure.push_back(modeLine.substr(modeLine.find("<path")+6,point-modeLine.find("<path")-6));

        g++;

        std::string tmp = "d=\"";

        int t = 0, h = 0, l = 0;

        bool structureParametr_d = true;


        while (structureParametr_d == true)

        {

        if (modeLine[point + g] == ' ' || modeLine[point + g] == '\"')

        {

                if (h > 0)

                        t++;


                if(modeLine[point + g] == '\"')

                {

                        tmp.push_back('\"');

                        tmp = tmp.substr(0,3) + tmp.substr(4,tmp.length()-4);

                        info.FileStructure.push_back(tmp);

                        if (tmp.find("C") != std::string::npos)

                        {

                                TEMP_HideAnalisParametrD(info, data, tmp, numberTegPath, info.FileStructure.size() - 1);

                                numberTegPath++;

                        }

                        else

                                numberTegPath++;


                        structureParametr_d = false;

                }

        }

        else

        {

                if (isdigit(modeLine[point + g]) || modeLine[point + g] == '.' || modeLine[point + g] == ',' || modeLine[point + g] == '-' || isalpha(modeLine[point + g]))

                {

                        if (isalpha(modeLine[point + g]))

                        {

                                tmp.push_back(' ');

                                tmp.push_back(modeLine[point + g]);

                                t = 0;h = 0;
```

```
                                        }
                                else
                                {
                                        if(isdigit(modeLine[point + g]))
                                                h++;

                                        if (modeLine[point + g] == '.' || modeLine[point + g] == ',')
                                                h = t = 0;

                                        if (t > 0)
                                        {
                                                tmp.push_back(' ');
                                                tmp.push_back(modeLine[point + g]);

                                                if (modeLine[point + g + 1] == ' ')
                                                        tmp.push_back(' ');

                                                t = h = 0;
                                        }
                                        else
                                                tmp.push_back(modeLine[point + g]);
                                }
                        }
                }
                g++;
                }

                info.FileStructure.push_back(modeLine.substr(point + g + 1, modeLine.length() - point - g - (modeLine.length() - modeLine.find("/>") + 1)));
                info.FileStructure.push_back("/>");

                findParametr_d = findParametr_d2 = findParametr_d3 = false;
                                                                                                                                        }
                                                                                                                                        g++;
                                                                                                                                }
                                                                                                                                else
                                                                                                                                {

                findParametr_d2 = false;

                findParametr_d3 = false;
                                                                                                                                        }
                                                                                                                                }
                                                                                                                        }
                                                                                                                        g++;
                                                                                                                }
                                                                                                                else
                                                                                                                        findParametr_d2 = false;
                                                                                                        }
                                                                                                }
                                                                                        }
                                                                                }
                                                                                otherLine.clear();
                                                                                someTegPath.clear();
                                                                                analisysTegPath = false;
                                                                                someTegPathFind = false;
                                                                        }
                                                                }
                                                        }

                                                        strfile.close();

                                                        if (info.Path.size() == 0)
                                                                return -2;
                }
                else
                {
                        return -1;
```

```
	}

	return 1;
}

int TEMPLATE::TEMPLATE_HIDE(PARAM param)
{
	TEMP_DATA data;
	TEMP_INFO info;

	data.V1 = param.V1;
	data.V2 = param.V2;
	data.V3 = param.V3;
	data.V4 = param.V4;

	for (int i = 0; i < param.K; i++)
	{
		info.TV.push_back(param.TV[i]);
		info.Tdt.push_back(param.dt[i]);
	}

	data.L = std::sqrt(param.K);
	data.K = param.K;
	data.StartPoint = param.sp;
	data.UP = 5;
	data.NumPath = 0;
	data.NumBeze = 0;

	data.SVG = param.svg;
	data.File = param.file;

	data.Rezult = param.rezult + "\\Rezult.svg";
	data.DataBack = param.rezult + "\\DataBack.txt";

	std::string file, str;
	std::list<std::string> svg;
	int rez;

	FILE *f = fopen(data.File.c_str(), "rb");

	if (f == NULL)
	{
		fclose(f);
		return 0;
	}

	while(!feof(f))
	{
		char ch;
		fread(&ch, 1, 1, f);
		file = file + ch;
	}

	fclose(f);

	str = file = file.substr(0, file.size() - 1);

	info.HideSize = file.size();

	rez = TEMP_AnalisSVGHide(data, info);

	if (rez != 1)
		return rez;

	rez = TEMP_HideInSVG(data, info, file, svg);

	if (rez != 1)
		return rez;

	std::ofstream fileSVG(data.Rezult, std::ios::in | std::ios::trunc);

	for (std::list<std::string>::iterator st = svg.begin(); st != svg.end(); )
	{
		fileSVG << st->c_str() << std::endl;
		++st;
	}

	fileSVG.close();

	std::ofstream DATABACK;

	DATABACK.open(data.DataBack, std::ios::in | std::ios::trunc);

	DATABACK << data.V1 << '|' << data.V2<< '|' << data.V3 << '|' << data.V4 << '|' << data.StartPoint
		<< '|' << data.L << std::endl;

	for (int i = 0; i < data.K; i++)
		DATABACK << info.TV[i] << '|' << info.Tdt[i] << std::endl;

	std::list<std::string>::iterator st = info.last_t.begin();
	for (int i = 0; i < info.last_t.size(); i++)
	{
```

```
                DATABACK << info.ia[i] << '|' << info.ib[i] << '|' << st->c_str() << std::endl;
                ++st;
        }

        DATABACK.close();

        info.Path.clear();
        info.last_t.clear();
        info.FileStructure.clear();
        info.TV.clear();
        info.Tdt.clear();
        info.ia.clear();
        info.ib.clear();

        return 1;
}

///SHOW DATA
std::string TEMP_AlgorithmCollection (TEMP_DATA &data, TEMP_INFO info, std::string &Rezult, std::list<std::string> &resultX,
                                      std::list<std::string> &resultY, std::string t)
{
        MATH math;
        std::list<std::string>::iterator itX = resultX.begin(), itY = resultY.begin();
        std::string P3X, P3Y, P21X, P21Y, P12X, P12Y, P02X, P02Y, P01X, P01Y, P0X, P0Y;
        std::string P03X, P03Y;
        std::string P1X, P1Y, P2X, P2Y, P11X, P11Y, P02X_2, P02Y_2, P02X_3, P02Y_3;

        P3X = itX->c_str();
        P3Y = itY->c_str();

        itX++;
        itY++;

        P21X = itX->c_str();
        P21Y = itY->c_str();

        itX++;
        itY++;

        P12X = itX->c_str();
        P12Y = itY->c_str();

        itX++;
        itY++;

        P03X = itX->c_str();
        P03Y = itY->c_str();

        itX++;
        itY++;

        P02X = itX->c_str();
        P02Y = itY->c_str();

        itX++;
        itY++;

        P01X = itX->c_str();
        P01Y = itY->c_str();

        itX++;
        itY++;

        P0X = itX->c_str();
        P0Y = itY->c_str();

        for (int i = 0; i < 16; i++)
        {
                int k = data.V4 + data.UP;

                std::string tp = math.Sub(t, info.Tdt[i], k);
                std::string sub_t = math.Sub("1", tp, k);

                P02X_2 = math.Div(math.Sub(P03X, math.Mul(tp, P12X, k), k), sub_t, k);
                P02Y_2 = math.Div(math.Sub(P03Y, math.Mul(tp, P12Y, k), k), sub_t, k);

                P02X_2 = TEMP_Rounding(P02X_2, data.V4);
                P02Y_2 = TEMP_Rounding(P02Y_2, data.V4);

                P11X = math.Div(math.Sub(P12X, math.Mul(tp, P21X, k), k), sub_t, k);
                P11Y = math.Div(math.Sub(P12Y, math.Mul(tp, P21Y, k), k), sub_t, k);

                P11X = TEMP_Rounding(P11X, data.V4);
                P11Y = TEMP_Rounding(P11Y, data.V4);

                P02X_3 = math.Add(math.Mul(sub_t, P01X, k), math.Mul(tp, P11X, k), k);
                P02Y_3 = math.Add(math.Mul(sub_t, P01Y, k), math.Mul(tp, P11Y, k), k);

                P02X_3 = TEMP_Rounding(P02X_3, data.V4);
                P02Y_3 = TEMP_Rounding(P02Y_3, data.V4);

                bool ch = false;
```

```
                        bool chX = false;
                        bool chY = false;

                        if (P02X.substr(0, P02X.find('.') - 1) == P02X_2.substr(0, P02X_2.find('.') - 1) && P02X.substr(0, P02X.find('.') - 1) == P02X_3.substr(0,
P02X_3.find('.') - 1))
                        {
                                int len;

                                len = data.V4 - P02X.substr(P02X.find('.') + 1).length();
                                for (int i = 0; i < len; i++)
                                        P02X = P02X + "0";

                                len = data.V4 - P02X_2.substr(P02X_2.find('.') + 1).length();
                                for (int i = 0; i < len; i++)
                                        P02X_2 = P02X_2 + "0";

                                len = data.V4 - P02X_3.substr(P02X_3.find('.') + 1).length();
                                for (int i = 0; i < len; i++)
                                        P02X_3 = P02X_3 + "0";

                                int iP02X = atoi(P02X.substr(P02X.find('.') + 1).c_str());
                                int iP02X_2 = atoi(P02X_2.substr(P02X_2.find('.') + 1).c_str());
                                int iP02X_3 = atoi(P02X_3.substr(P02X_3.find('.') + 1).c_str());

                                if (abs(iP02X_2 - iP02X_3) <= data.V5 || abs(iP02X - iP02X_2) <= data.V5 || abs(iP02X - iP02X_3) <= data.V5)
                                        chX = true;

                                if (chX == true)
                                {
                                        int len;

                                        len = data.V4 - P02Y.substr(P02Y.find('.') + 1).length();
                                        for (int i = 0; i < len; i++)
                                                P02Y = P02Y + "0";

                                        len = data.V4 - P02Y_2.substr(P02Y_2.find('.') + 1).length();
                                        for (int i = 0; i < len; i++)
                                                P02Y_2 = P02Y_2 + "0";

                                        len = data.V4 - P02Y_3.substr(P02Y_3.find('.') + 1).length();
                                        for (int i = 0; i < len; i++)
                                                P02Y_3 = P02Y_3 + "0";

                                        int iP02Y = atoi(P02Y.substr(P02Y.find('.') + 1).c_str());
                                        int iP02Y_2 = atoi(P02Y_2.substr(P02Y_2.find('.') + 1).c_str());
                                        int iP02Y_3 = atoi(P02Y_3.substr(P02Y_3.find('.') + 1).c_str());

                                        if (abs(iP02Y_2 - iP02Y_3) <= data.V5 || abs(iP02Y - iP02Y_2) <= data.V5 || abs(iP02Y - iP02Y_3) <= data.V5)
                                                chY = true;
                                }
                        }

                        if (chX == true && chY == true)
                                ch = true;

                        if (ch == true)
                        {
                                Rezult = info.TV[i] + Rezult;

                                resultX.clear();
                                resultY.clear();

                                P2X = math.Div(math.Sub(P21X, math.Mul(tp, P3X, k), k), sub_t, k);
                                P2Y = math.Div(math.Sub(P21Y, math.Mul(tp, P3Y, k), k), sub_t, k);

                                P11X = math.Div(math.Sub(P12X, math.Mul(tp, P21X, k), k), sub_t, k);
                                P11Y = math.Div(math.Sub(P12Y, math.Mul(tp, P21Y, k), k), sub_t, k);

                                P1X = math.Div(math.Sub(P11X, math.Mul(tp, P2X, k), k), sub_t, k);
                                P1Y = math.Div(math.Sub(P11Y, math.Mul(tp, P2Y, k), k), sub_t, k);

                                P2X = TEMP_Rounding(P2X, data.V4);
                                P2Y = TEMP_Rounding(P2Y, data.V4);

                                P1X = TEMP_Rounding(P1X, data.V4);
                                P1Y = TEMP_Rounding(P1Y, data.V4);

                                resultX.push_back(P3X);resultY.push_back(P3Y);
                                resultX.push_back(P2X);resultY.push_back(P2Y);
                                resultX.push_back(P1X);resultY.push_back(P1Y);
                                resultX.push_back(P0X);resultY.push_back(P0Y);

                                return tp;
                        }
                }

                return data.StartPoint;
}

bool TEMP_AnalisLineSplit(TEMP_DATA &data, TEMP_INFO &info, std::string &Rezult, std::string line, int CNumb, int r, std::string st)
{
                std::list<std::string> str;
```

```
std::string tmp = line.substr(3,line.length() - 4);
int point = 0, i = 0;
bool ph = false;

while (tmp.find(' ', point) != std::string::npos)
{
	str.push_back(tmp.substr(0, tmp.find(' ', point)));

	if (tmp.substr(0, tmp.find(' ', point)).find('C') != std::string::npos)
	{
		if (ph == true)
		{
			ph = false;
			break;
		}
		else
			i++;
	}
	tmp = tmp.substr(tmp.find(' ', point) + 1, tmp.length() - tmp.find(' ', point) - 1);
	point = tmp.find(' ');
}

if (ph == true)
{
	if (tmp.find(' ') != std::string::npos)
		str.push_back(tmp.substr(0, tmp.find(' ', point)));
	else
		str.push_back(tmp);
}

std::list<std::string> stb;
std::list<std::string>::iterator it = str.begin();
for (int i = 0; i < CNumb;)
{
	if (it->find('C') != std::string::npos)
	{
		stb.push_back(it->c_str());
		++it;
		stb.push_back(it->c_str());
		++it;
		stb.push_back(it->c_str());
		++it;

		i++;
		continue;
	}

	stb.push_back(it->c_str());
	++it;
}

bool lastLineBezie = false;

it = stb.end();

it--;

std::string t = st;

std::list<std::string> resultX, resultY;

for (it; it != stb.begin();)
{
	if (it->find('Z') != std::string::npos || it->find('z') != std::string::npos)
	{
		--it;
		continue;
	}

	if (it->find('C') != std::string::npos)
		resultX.push_back(it->substr(1, it->find(',') - 1));
	else
		resultX.push_back(it->substr(0, it->find(',')));

	resultY.push_back(it->substr(it->find(',') + 1,it->length() - it->find(',') - 1));

	if (it->find('C') != std::string::npos)
	{
		if (lastLineBezie == false)
			lastLineBezie = true;
		else
		{
			it--;
			if (/*it->find('C') != std::string::npos || it->find('c') != std::string::npos*/
				it->find('M') != std::string::npos || it->find('m') != std::string::npos
				/*|| it->find('Z') != std::string::npos || it->find('z') != std::string::npos*/
				|| it->find('L') != std::string::npos || it->find('l') != std::string::npos
				|| it->find('H') != std::string::npos || it->find('h') != std::string::npos
				|| it->find('V') != std::string::npos || it->find('v') != std::string::npos
				/*|| it->find('S') != std::string::npos || it->find('s') != std::string::npos*/
				/*|| it->find('Q') != std::string::npos || it->find('q') != std::string::npos*/
				|| it->find('T') != std::string::npos || it->find('t') != std::string::npos
```

```
                                        /*|| it->find('A') != std::string::npos || it->find('a') != std::string::npos*/)
                                {
                                        resultX.push_back(it->substr(1,it->find(',')-1));
                                        resultY.push_back(it->substr(it->find(',')+1,it->length() - it->find(',') - 1));
                                }
                                else
                                {
                                        resultX.push_back(it->substr(0,it->find(',')));
                                        resultY.push_back(it->substr(it->find(',')+1,it->length() - it->find(',') - 1));
                                }

                                t = TEMP_AlgorithmCollection(data, info, Rezult, resultX, resultY, t);

                                if (it == stb.begin() || t == data.StartPoint)
                                        break;

                                --it;
                                continue;
                        }
                }

                --it;
        }

        return true;
}

int TEMP_ShowFromSVG(TEMP_DATA &data, TEMP_INFO &info, std::string &line)
{
        int NumBeze = 0;
        int i = 0, j = 0, k;
        std::list<std::string> Data, Rezult;
        std::list<std::string>::iterator st = info.last_t.begin();

        if (info.Path.size() == 0)
                return -3;

        for (std::map<int,std::map<int, std::map<int, std::map<int, bool>>>>::iterator it = info.Path.begin(); it != info.Path.end(); )
        {
                NumBeze = NumBeze + it->second.begin()->second.size();
                ++it;
        }

        if (NumBeze < info.NumberLine)
                return -3;

        for (std::list<std::string>::iterator itList = info.FileStructure.begin(); itList != info.FileStructure.end(); )
        {
                k = info.ia[j];

                if (i == info.ia[j])
                {
                        int r;

                        if (j == 0)
                                r = 0;
                        else
                                r = info.ib[j - 1];

                        std::string Rezult;

                        if (!TEMP_AnalisLineSplit(data, info, Rezult, itList->c_str(), info.ib[j], r, st->c_str()))
                                return false;

                        Data.push_back(Rezult);

                        j++;
                        ++st;
                        info.NumberLine--;

                        if(info.NumberLine == 0)
                                break;

                        if (k == info.ia[j])
                                continue;
                }

                itList++;
                i++;
        }

        for (std::list<std::string>::iterator it = Data.begin(); it != Data.end(); )
        {
                std::string temp = it->c_str(), str;

                if (temp.size() % 8 != 0)
                {
                        int j = temp.size() % 8;
                        for (int i = 0; i < j; i++)
                                temp = temp + '0';
                }
```

```
                while (!temp.empty())
                {
                        int pi = 0;

                        for (int st = temp.substr(0,8).length() - 1, kt = 0; st >= 0; st--, kt++)
                        {
                                switch(temp.substr(0,8)[kt])
                                {
                                case '1': pi = pi + pow(2, st); break;
                                }
                        }

                        str = str + static_cast<char>(pi);

                        temp = temp.substr(8);
                }

                line = line + str;

                ++it;
        }

        Data.clear();
        Rezult.clear();

        return 1;
}

void TEMP_ShowAnalisParametrD(TEMP_DATA &data, TEMP_INFO &info, std::string line, int NumPath, int Size)
{
        std::string tmp = line.substr(3,line.length() - 4);
        std::list<std::string> str;
        int point = 0, k = 0, number = 0, i = 0;
        while (tmp.find(' ', point) != std::string::npos)
        {
                str.push_back(tmp.substr(0, tmp.find(' ', point)));
                tmp = tmp.substr(tmp.find(' ', point) + 1, tmp.length() - tmp.find(' ', point) - 1);
                point = tmp.find(' ');
        }

        str.push_back(tmp);

        std::map<int, std::map<int, bool>> lineBezierCheck;
        std::map<int, bool> lineBezierParametr;

        bool  check = false, firstcheck = false;
        for (std::list<std::string>::iterator it = str.begin();it != str.end(); it++, i++)
        {
                if ((it->find('C') != std::string::npos || it->find('c') != std::string::npos
                        || it->find('M') != std::string::npos || it->find('m') != std::string::npos
                        || it->find('Z') != std::string::npos || it->find('z') != std::string::npos
                        || it->find('L') != std::string::npos || it->find('l') != std::string::npos
                        || it->find('H') != std::string::npos || it->find('h') != std::string::npos
                        || it->find('V') != std::string::npos || it->find('v') != std::string::npos
                        || it->find('S') != std::string::npos || it->find('s') != std::string::npos
                        || it->find('Q') != std::string::npos || it->find('q') != std::string::npos
                        || it->find('T') != std::string::npos || it->find('t') != std::string::npos
                        || it->find('A') != std::string::npos || it->find('a') != std::string::npos) && check == true)
                {
                        if (k-1 == 2 && firstcheck == false)
                        {
                                lineBezierParametr[k-1] = true;
                                lineBezierCheck[number] = lineBezierParametr;
                                number++;
                        }
                        else
                        {
                                lineBezierParametr[k-1] = false;
                                lineBezierCheck[number] = lineBezierParametr;
                                number++;
                        }

                        firstcheck = false;
                        lineBezierParametr.clear();
                        k = 0;
                        if (it->find('C') != std::string::npos)
                                check = true;
                        else
                                check = false;
                }

                if(it->find('C') != std::string::npos && check == false)
                {
                        check = true;
                        if (it == str.begin())
                                firstcheck = true;
                }

                if (check == true && i != str.size() - 1)
                        k++;
        }
```

```
		std::map<int, std::map<int, std::map<int, bool>>> temp;
		temp[Size] = lineBezierCheck;

		info.Path[NumPath] = temp;
}

int TEMP_AnalisSVGShow(TEMP_DATA &data, TEMP_INFO &info)
{
		std::ifstream strfile(data.SVG, std::ios::in);

		if (strfile.is_open())
		{
				strfile.seekg (0, strfile.end);
				info.SVGSize = strfile.tellg();
				strfile.seekg (0);

				std::string otherLine, line;
				std::list<std::string> someTegPath;
				bool analisysTegPath = false;
				bool findParametr_d = false;
				bool someTegPathFind = false;
				int numberTegPath = 0;

				while (getline(strfile,line))
				{
						if (line.find("<path") == std::string::npos && analisysTegPath == false && someTegPathFind == false)
								info.FileStructure.push_back(line);
						else
						{
								if (line.find(">") < line.find("<path") && line.find("<path") != std::string::npos && line.find(">") != std::string::npos ||
line.find("/>") < line.find("<path") && line.find("<path") != std::string::npos && line.find("/>") != std::string::npos)
								{
										if (line.find("/>") < line.find("<path") && line.find("<path") != std::string::npos && line.find("/>") !=
std::string::npos)
										{
												otherLine = otherLine.substr(otherLine.find("<path"),otherLine.length() - otherLine.find("<path")) +
' ' + line.substr(0,line.find("/>")+2);
												someTegPath.push_back(otherLine);
												otherLine = line.substr(line.find("<path"),line.length() - line.find("/>"));
										}
										else
										{
												info.FileStructure.push_back(line.substr(0,line.find(">")+1));
												otherLine = line.substr(line.find("<path"),line.length() - line.find(">"));
										}
										someTegPathFind = true;
										continue;
								}

								if (someTegPathFind = true)
								{
										otherLine = otherLine + ' ' + line;

										if (otherLine.find("/>") != std::string::npos && otherLine.find("<path",otherLine.find("/>")) == std::string::npos)
										{
												someTegPathFind = false;
												analisysTegPath = true;
												otherLine = otherLine.substr(otherLine.find("<path"),otherLine.length() - otherLine.find("<path"));
												someTegPath.push_back(otherLine.substr(0,otherLine.find("/>") + 2));
										}
								}

								if (someTegPathFind == false && analisysTegPath == true)
								{
										for(std::list<std::string>::iterator number = someTegPath.begin(); number != someTegPath.end(); number++)
										{
												std::string modeLine = number->c_str();
												std::size_t point = modeLine.find("\t");
												while (modeLine.find("\t") != std::string::npos)
												{
														modeLine = modeLine.substr(0,point) + modeLine.substr(point+1,modeLine.length());
														point = modeLine.find("\t");
												}

												info.FileStructure.push_back(modeLine.substr(modeLine.find("<path"), 5));
												point = 0;
												findParametr_d = true;

												while(findParametr_d == true)
												{
														int g = 1;
														if (modeLine.find(" d") != std::string::npos)
														{
																if (point == 0)
																		point = modeLine.find(" d") + 1;
																else
																		point = modeLine.find(" d", point + 2);

																bool findParametr_d2 = true;

																while(findParametr_d2 == true)
																{
```

```
if (modeLine[point + g] == ' ' || modeLine[point + g]
== '=')
{
	if(modeLine[point + g] == '=')
	{
		g++;
		bool findParametr_d3 = true;
		while(findParametr_d3 == true)
		{
			if (modeLine[point + g]
== ' ' || modeLine[point + g] == '\"')
			{
	if(modeLine[point + g] == '\"')
				{
	info.FileStructure.push_back(modeLine.substr(modeLine.find("<path")+6,point-modeLine.find("<path")-6));
	g++;
	std::string tmp = "d=\"";
	int t = 0, h = 0, l = 0;
	bool structureParametr_d = true;

	while (structureParametr_d == true)
	{
	if (modeLine[point + g] == ' ' || modeLine[point + g] == '\"')
	{
		if (h > 0)
			t++;

		if(modeLine[point + g] == '\"')
		{
			tmp.push_back('\"');
			tmp = tmp.substr(0,3) + tmp.substr(4,tmp.length()-4);
			info.FileStructure.push_back(tmp);
			if (tmp.find("C") != std::string::npos)
			{
				TEMP_ShowAnalisParametrD(data, info, tmp, numberTegPath, info.FileStructure.size() - 1);
				numberTegPath++;
			}
			else
				numberTegPath++;

			structureParametr_d = false;
		}
	}
	else
	{
		if (isdigit(modeLine[point + g]) || modeLine[point + g] == '.' || modeLine[point + g] == ',' || modeLine[point + g] == '-' || isalpha(modeLine[point + g]))
		{
			if (isalpha(modeLine[point + g]))
			{
				tmp.push_back(' ');
				tmp.push_back(modeLine[point + g]);
				t = 0;h = 0;
			}
			else
```

```
                                        {
                                        if(isdigit(modeLine[point + g]))
                                                h++;

                                        if (modeLine[point + g] == '.' || modeLine[point + g] == ',')
                                                h = t = 0;

                                        if (t > 0)
                                        {
                                                tmp.push_back(' ');
                                                tmp.push_back(modeLine[point + g]);

                                                if (modeLine[point + g + 1] == ' ')
                                                        tmp.push_back(' ');

                                                t = h = 0;
                                        }
                                        else
                                                tmp.push_back(modeLine[point + g]);
                                }
                        }
                }
                g++;
                }

                info.FileStructure.push_back(modeLine.substr(point + g + 1, modeLine.length() - point - g - (modeLine.length() - modeLine.find("/>") + 1)));
                info.FileStructure.push_back("/>");

                findParametr_d = findParametr_d2 = findParametr_d3 = false;
                                                                                                                                                }
                                                                                                                                                g++;
                                                                                                                                        }
                                                                                                                                        else
                                                                                                                                        {
                findParametr_d2 = false;
                findParametr_d3 = false;
                                                                                                                                                }
                                                                                                                                        }
                                                                                                                                }
                                                                                                                                g++;
                                                                                                                        }
                                                                                                                        else
                                                                                                                                findParametr_d2 = false;
                                                                                                                }
                                                                                                        }
                                                                                                }
                                                                                        }
                                                                                        otherLine.clear();
                                                                                        someTegPath.clear();
                                                                                        analisysTegPath = false;
                                                                                        someTegPathFind = false;
                                                                                }
                                                                        }
                                                                }

                                                                strfile.close();

                                                                if (info.Path.size() == 0)
                                                                        return -2;
                                                }
                                                else
                                                {
                                                                return -1;
                                                }

                                                return 1;
                                }
```

```
int TEMPLATE::TEMPLATE_SHOW(PARAM param)
{
	TEMP_DATA data;
	TEMP_INFO info;
	enum TYPE {FIRST, SECOND, THIRD};
	TYPE type = FIRST;
	std::string str;
	int rez;

	data.V5 = std::atoi(param.V5.substr(param.V5.length() - 1).c_str()) * 10;
	data.UP = UPP;
	data.DataBack = param.file;
	data.Rezult = param.rezult + "\\Rezult.txt";
	data.SVG = param.svg;

	std::ifstream DATABACK;

	DATABACK.open(data.DataBack, std::ios::in);

	if (!DATABACK.is_open())
		return -4;

	while (getline(DATABACK, str))
	{
		if (type == FIRST)
		{
			data.V1 = atoi(str.substr(0,str.find('|')).c_str());
			str = str.substr(str.find('|') + 1);
			data.V2 = atoi(str.substr(0,str.find('|')).c_str());
			str = str.substr(str.find('|') + 1);
			data.V3 = atoi(str.substr(0,str.find('|')).c_str());
			str = str.substr(str.find('|') + 1);
			data.V4 = atoi(str.substr(0,str.find('|')).c_str());
			str = str.substr(str.find('|') + 1);
			data.StartPoint = str.substr(0,str.find('|')).c_str();
			str = str.substr(str.find('|') + 1);
			data.L = atoi(str.substr(0,str.find('|')).c_str());
			data.K = pow(2, data.L);
			str.clear();

			info.NumberLine = 0;
			type = SECOND;
			continue;
		}

		if (type == SECOND)
		{
			info.TV.push_back(str.substr(0, str.find('|')).c_str());
			str = str.substr(str.find('|') + 1);
			info.Tdt.push_back(str.substr(0,str.find('|')).c_str());
			info.NumberLine++;
			str.clear();
			if (info.NumberLine == data.K)
			{
				type = THIRD;
				info.NumberLine = 0;
			}

			continue;
		}

		info.ia.push_back(atoi(str.substr(0,str.find('|')).c_str()));
		str = str.substr(str.find('|') + 1);
		info.ib.push_back(atoi(str.substr(0,str.find('|')).c_str()));
		str = str.substr(str.find('|') + 1);
		info.last_t.push_back(str.substr(0,str.find('|')));
		info.NumberLine++;
	}

	DATABACK.close();
	str.clear();

	rez = TEMP_AnalisSVGShow(data, info);

	if (rez <= 0)
		return rez;

	rez = TEMP_ShowFromSVG(data, info, str);

	if (rez <= 0)
		return rez;

	FILE *f = fopen(data.Rezult.c_str(), "wb");

	for (int it = 0; it < str.size(); it++)
	{
			char ch = str.c_str()[it];
			fwrite(&ch, 1, 1, f);
	}

	fclose(f);
```

```
	return 1;
}
```

3. Вихідний код процедури виконання математичних операцій алгоритмів StegoBIT та StegoTEMPL

/// MATH.h

```
#pragma once
#include <iostream>
class MATH
{
public:
	std::string NullBeginCheck(std::string A);
	std::string NullEndCheck(std::string A);

	std::string Add(std::string A, std::string B, int AC);
	std::string Sub(std::string A, std::string B, int AC);
	std::string Mul(std::string A, std::string B, int AC);
	std::string Div(std::string A, std::string B, int AC);
};
```

/// MATH.cpp

```
#include "stdafx.h"
#include "MATH.h"
#include <vector>
#include <string>

#define LENGHT 50

std::string MATH::NullBeginCheck (std::string A)
{
	if (A.find('.') != std::string::npos)
	{
		while(true)
		{
			if (atoi(A.substr(0,1).c_str()) > 0 || A[0] == '0' && A[1] == '.')
				break;
			else
				A = A.substr(1);
		}
	}
	else
	{
		while(true)
		{
			if (atoi(A.substr(0,1).c_str()) > 0 || A.length() == 1)
				break;
			else
				A = A.substr(1);
		}
	}

	return A;
}

std::string MATH::NullEndCheck (std::string A)
{
	if (A.length() > 1 && A.find('.') != std::string::npos)
	{
		for (int i = A.length() - 1; ; i--)
		{
			if (A[i] == '0' && A[i - 1] != '.')
			{
				A = A.substr(0, A.length() - 1);
				continue;
			}

			if (A[i] == '0' && A[i - 1] == '.')
				A = A.substr(0, A.length() - 2);

			break;
		}
	}

	return A;
}

std::string MATH::Add (std::string A, std::string B, int AC)
{
	std::string R;

	if (A[0] != '-' && B[0] == '-' || A[0] == '-' && B[0] != '-')
	{
		if (A[0] != '-')
			R = Sub(A, B.substr(1), AC);
		else
```

```
                        R = Sub(A, "-" + B, AC);
        return R;
}

bool sign = true;

if (A[0] == '-' && B[0] == '-')
        sign = false;

int Lenght = 0, LenghtA = 0, LenghtB = 0;

if (A.find('.') != std::string::npos)
        LenghtA = A.substr(A.find('.') + 1).length();

if (B.find('.') != std::string::npos)
        LenghtB = B.substr(B.find('.') + 1).length();

if (LenghtA > LenghtB)
        Lenght = LenghtA;
else
        Lenght = LenghtB;

std::vector<int> X(LENGHT), Y(LENGHT);
int Xi = 0, Yi = 0, Z = 0, C = 0;

if (LenghtA < Lenght)
        Xi = Lenght - LenghtA;

if (LenghtB < Lenght)
        Yi = Lenght - LenghtB;

for (int i = A.length() - 1; i >= 0; i--)
{
        switch(A[i])
        {
        case '0': X[Xi] = 0; Xi++; break;
        case '1': X[Xi] = 1; Xi++; break;
        case '2': X[Xi] = 2; Xi++; break;
        case '3': X[Xi] = 3; Xi++; break;
        case '4': X[Xi] = 4; Xi++; break;
        case '5': X[Xi] = 5; Xi++; break;
        case '6': X[Xi] = 6; Xi++; break;
        case '7': X[Xi] = 7; Xi++; break;
        case '8': X[Xi] = 8; Xi++; break;
        case '9': X[Xi] = 9; Xi++; break;
        default: continue;
        }
}

for (int i = B.length() - 1; i >= 0; i--)
{
        switch(B[i])
        {
        case '0': Y[Yi] = 0; Yi++; break;
        case '1': Y[Yi] = 1; Yi++; break;
        case '2': Y[Yi] = 2; Yi++; break;
        case '3': Y[Yi] = 3; Yi++; break;
        case '4': Y[Yi] = 4; Yi++; break;
        case '5': Y[Yi] = 5; Yi++; break;
        case '6': Y[Yi] = 6; Yi++; break;
        case '7': Y[Yi] = 7; Yi++; break;
        case '8': Y[Yi] = 8; Yi++; break;
        case '9': Y[Yi] = 9; Yi++; break;
        default: continue;
        }
}

if (Xi > Yi)
        Z = Xi;
else
        Z = Yi;

for (int i = 0; i < Z; i++)
{
        X[i] = X[i] + Y[i] + C;
        C = X[i] / 10;
        R = std::to_string(X[i] % 10) + R;
}

if (C > 0)
{
        R = std::to_string(C) + R;
        Z++;
}

if (Lenght > 0)
{
        R = R.substr(0, Z - Lenght) + '.' + R.substr(Z - Lenght);

        if (R.substr(R.find('.') + 1).length() > AC)
                R = R.substr(0, R.length() - (R.substr(R.find('.') + 1).length() - AC));
}
```

```
	R = NullEndCheck(R);

	if (sign == false)
		R = '-' + R;

	X.clear();
	Y.clear();

	return R;
}

std::string MATH::Sub (std::string A, std::string B, int AC)
{
	std::string R;

	if (A[0] != '-' && B[0] == '-' || A[0] == '-' && B[0] != '-')
	{
		if (A[0] != '-')
			R = Add(A, B.substr(1), AC);
		else
			R = Add(A, "-" + B, AC);
		return R;
	}

	int Lenght = 0, LenghtA = 0, LenghtB = 0, sign = 0;

	if (A.find('.') != std::string::npos)
		LenghtA = A.substr(A.find('.') + 1).length();

	if (B.find('.') != std::string::npos)
		LenghtB = B.substr(B.find('.') + 1).length();

	if (LenghtA > LenghtB)
		Lenght = LenghtA;
	else
		Lenght = LenghtB;

	std::vector<int> X(LENGHT), Y(LENGHT);
	int Xi = 0, Yi = 0, Z = 0, C = 0;

	if (LenghtA < Lenght)
		Xi = Lenght - LenghtA;

	if (LenghtB < Lenght)
		Yi = Lenght - LenghtB;

	for (int i = A.length() - 1; i >= 0; i--)
	{
		switch(A[i])
		{
		case '0': X[Xi] = 0; Xi++; break;
		case '1': X[Xi] = 1; Xi++; break;
		case '2': X[Xi] = 2; Xi++; break;
		case '3': X[Xi] = 3; Xi++; break;
		case '4': X[Xi] = 4; Xi++; break;
		case '5': X[Xi] = 5; Xi++; break;
		case '6': X[Xi] = 6; Xi++; break;
		case '7': X[Xi] = 7; Xi++; break;
		case '8': X[Xi] = 8; Xi++; break;
		case '9': X[Xi] = 9; Xi++; break;
		default: continue;
		}
	}

	for (int i = B.length() - 1; i >= 0; i--)
	{
		switch(B[i])
		{
		case '0': Y[Yi] = 0; Yi++; break;
		case '1': Y[Yi] = 1; Yi++; break;
		case '2': Y[Yi] = 2; Yi++; break;
		case '3': Y[Yi] = 3; Yi++; break;
		case '4': Y[Yi] = 4; Yi++; break;
		case '5': Y[Yi] = 5; Yi++; break;
		case '6': Y[Yi] = 6; Yi++; break;
		case '7': Y[Yi] = 7; Yi++; break;
		case '8': Y[Yi] = 8; Yi++; break;
		case '9': Y[Yi] = 9; Yi++; break;
		default: continue;
		}
	}

	if (Xi - Lenght > Yi - Lenght)
	{
		Z = Xi;
		sign = 1;
	}
	else
	{
		if (Xi - Lenght < Yi - Lenght)
		{
```

```
                Z = Yi;
                sign = -1;

                if (B[0] != '-')
                        B = '-' + B.substr(1);
                else
                        B = B.substr(1);
        }
        else
        {
                Z = Xi;

                for (int i = Z - 1; i >= 0; i--)
                {
                        if (X[i] == Y[i])
                                continue;

                        if (X[i] > Y[i])
                                sign = 1;
                        else
                        {
                                sign = -1;

                                if (B[0] != '-')
                                        B = '-' + B.substr(1);
                                else
                                        B = B.substr(1);
                        }

                        break;
                }

                if (sign == 0)
                        return std::to_string(0);
        }
}

if (sign == 1)
{
        for (int i = 0; i < Z; i++)
        {
                if (X[i] - C < Y[i])
                {
                        R = std::to_string(X[i] - C + 10 - Y[i]) + R;
                        C = 1;
                        continue;
                }

                R = std::to_string(X[i] - C - Y[i]) + R;
                C = 0;
        }
}
else
{
        for (int i = 0; i < Z; i++)
        {
                if (Y[i] - C < X[i])
                {
                        R = std::to_string(Y[i] - C + 10 - X[i]) + R;
                        C = 1;
                        continue;
                }

                R = std::to_string(Y[i] - C - X[i]) + R;
                C = 0;
        }
}

if (Lenght > 0)
{
        R = R.substr(0, Z - Lenght) + '.' + R.substr(Z - Lenght);

        if (R.substr(R.find('.') + 1).length() > AC)
                R = R.substr(0, R.length() - (R.substr(R.find('.') + 1).length() - AC));
}

R = NullBeginCheck(R);
R = NullEndCheck(R);

if (sign == 1)
{
        if(A[0] == '-')
                R = A.substr(0,1) + R;
}
else
{
        if(B[0] == '-')
                R = B.substr(0,1) + R;
}

X.clear();
Y.clear();
```

```
	return R;
}

std::string MATH::Mul (std::string A, std::string B, int AC)
{
	std::string R;

	bool sign = true;

	if (A[0] != '-' && B[0] == '-' || A[0] == '-' && B[0] != '-')
		sign = false;

	int LenghtA = 0, LenghtB = 0;

	if (A.find('.') != std::string::npos)
		LenghtA = A.substr(A.find('.') + 1).length();

	if (B.find('.') != std::string::npos)
		LenghtB = B.substr(B.find('.') + 1).length();

	std::vector<int> X(LENGHT), Y(LENGHT);
	int Xi = 0, Yi = 0, Z[LENGHT][LENGHT], D = 0, K = 0, C = 0;

	for (int i = 0; i < LENGHT; i++)
		for (int j = 0; j < LENGHT; j++)
			Z[i][j] = 0;

	for (int i = A.length() - 1; i >= 0; i--)
	{
		switch(A[i])
		{
		case '0': X[Xi] = 0; Xi++; break;
		case '1': X[Xi] = 1; Xi++; break;
		case '2': X[Xi] = 2; Xi++; break;
		case '3': X[Xi] = 3; Xi++; break;
		case '4': X[Xi] = 4; Xi++; break;
		case '5': X[Xi] = 5; Xi++; break;
		case '6': X[Xi] = 6; Xi++; break;
		case '7': X[Xi] = 7; Xi++; break;
		case '8': X[Xi] = 8; Xi++; break;
		case '9': X[Xi] = 9; Xi++; break;
		default: continue;
		}
	}

	for (int i = B.length() - 1; i >= 0; i--)
	{
		switch(B[i])
		{
		case '0': Y[Yi] = 0; Yi++; break;
		case '1': Y[Yi] = 1; Yi++; break;
		case '2': Y[Yi] = 2; Yi++; break;
		case '3': Y[Yi] = 3; Yi++; break;
		case '4': Y[Yi] = 4; Yi++; break;
		case '5': Y[Yi] = 5; Yi++; break;
		case '6': Y[Yi] = 6; Yi++; break;
		case '7': Y[Yi] = 7; Yi++; break;
		case '8': Y[Yi] = 8; Yi++; break;
		case '9': Y[Yi] = 9; Yi++; break;
		default: continue;
		}
	}

	for (int i = 0; i < Yi; i++, K++)
	{
		D = i;

		for (int j = 0; j < Xi; j++)
		{
			Z[i][D] = (X[j] * Y[i] + C) % 10;
			C = (X[j] * Y[i] + C) / 10;
			D++;
		}

		if (C > 0)
		{
			Z[i][D] = C;
			C = 0;
			D++;
		}
	}

	for (int i = 0; i < D; i++)
	{
		for (int j = 0; j < K; j++)
		{
			if (j == 0)
				continue;

			Z[0][i] = Z[0][i] + Z[j][i];
		}
```

```
                R = std::to_string((Z[0][i] + C) % 10) + R;
                C = (Z[0][i] + C) / 10;
        }

        while(C > 0)
        {
                R = std::to_string(C % 10) + R;
                C = C / 10;
        }

        if (LenghtA + LenghtB > 0)
        {
                R = R.substr(0, R.length() - (LenghtA + LenghtB)) + '.' + R.substr(R.length() - (LenghtA + LenghtB));

                if (R.substr(R.find('.') + 1).length() > AC)
                        R = R.substr(0, R.length() - (R.substr(R.find('.') + 1).length() - AC));
        }

        R = NullBeginCheck(R);
        R = NullEndCheck(R);

        if (sign == false && R != "0")
                R = '-' + R;

        X.clear();
        Y.clear();

        return R;
}

std::string MATH::Div (std::string A, std::string B, int AC)
{
        if (B == "0")
                return "NULL_ERROR";

        if (A == "0")
                return "0";

        std::string R;

        bool sign = true, AC_CHECK = true;

        A = NullEndCheck(A);
        B = NullEndCheck(B);

        if (A[0] == '-' && B[0] == '-')
        {
                A = A.substr(1);
                B = B.substr(1);
        }

        if (A[0] == '-' && B[0] != '-' || A[0] != '-' && B[0] == '-')
        {
                sign = false;

                if (A[0] == '-')
                        A = A.substr(1);

                if (B[0] == '-')
                        B = B.substr(1);
        }

        int Lenght = 0, LenghtA = 0, LenghtB = 0;

        if (A.find('.') != std::string::npos)
        {
                Lenght = A.substr(0, A.find('.')).length();
                LenghtA = A.substr(A.find('.') + 1).length();
        }
        else
                Lenght = A.length();

        if (B.find('.') != std::string::npos)
                LenghtB = B.substr(B.find('.') + 1).length();

        std::vector<int> X(LENGHT), Y(LENGHT), Z(LENGHT), T(LENGHT);
        int Xi = 0, Yi = 0, K = 0, Zi = 0, Ti = 0, NUL = 0;

        for (int i = 0; i < A.length(); i++)
        {
                if (AC_CHECK == true)
                {
                        if (A[i] == '0' || A[i] == '.')
                        {
                                if (A[i] == '0')
                                        NUL++;

                                continue;
                        }

                        AC_CHECK = false;
```

```
		}

		switch(A[i])
		{
		case '0': X[Xi] = 0; Xi++; break;
		case '1': X[Xi] = 1; Xi++; break;
		case '2': X[Xi] = 2; Xi++; break;
		case '3': X[Xi] = 3; Xi++; break;
		case '4': X[Xi] = 4; Xi++; break;
		case '5': X[Xi] = 5; Xi++; break;
		case '6': X[Xi] = 6; Xi++; break;
		case '7': X[Xi] = 7; Xi++; break;
		case '8': X[Xi] = 8; Xi++; break;
		case '9': X[Xi] = 9; Xi++; break;
		default: continue;
		}
}

AC_CHECK = true;

for (int i = 0; i < B.length(); i++)
{
		if (AC_CHECK == true)
		{
				if (B[i] == '0' || B[i] == '.')
						continue;

				AC_CHECK = false;
		}

		switch(B[i])
		{
		case '0': Y[Yi] = 0; Yi++; break;
		case '1': Y[Yi] = 1; Yi++; break;
		case '2': Y[Yi] = 2; Yi++; break;
		case '3': Y[Yi] = 3; Yi++; break;
		case '4': Y[Yi] = 4; Yi++; break;
		case '5': Y[Yi] = 5; Yi++; break;
		case '6': Y[Yi] = 6; Yi++; break;
		case '7': Y[Yi] = 7; Yi++; break;
		case '8': Y[Yi] = 8; Yi++; break;
		case '9': Y[Yi] = 9; Yi++; break;
		default: continue;
		}
}

int step = 0, M, C, N = 0;
bool zero = false, point = false;
AC_CHECK = false;

if (LenghtB >= 0)
{
		if (Lenght > 1)
				Lenght = Lenght + LenghtB;
		else
		{
				if(A[0] == '-')
				{
						if (Lenght == 1 && A[1] != '0')
								Lenght = Lenght + LenghtB;
						else
						{
								Lenght = 0;
								bool first = true;

								for (int i = 3, j = 0; j < LenghtB; i++, j++)
								{
										if (A[i] == '0' && first == true)
												continue;

										first = false;
										Lenght++;
								}

								if (Lenght == 0)
								{
										while (NUL - LenghtB > 0)
										{
												R = R + '0';

												if (AC_CHECK == true)
														K++;

												if (point == false)
												{
														R = R + '.';
														point = AC_CHECK = true;
												}

												NUL--;
										}
								}
```

```
                        }
                }
                else
                {
                        if (Lenght == 1 && A[0] != '0')
                                Lenght = Lenght + LenghtB;
                        else
                        {
                                Lenght = 0;
                                bool first = true;

                                for (int i = 2, j = 0; j < LenghtB; i++, j++)
                                {
                                        if (i < A.length() && A[i] == '0' && first == true)
                                                continue;

                                        first = false;
                                        Lenght++;
                                }

                                if (Lenght == 0)
                                {
                                        while (NUL - LenghtB > 0)
                                        {
                                                R = R + '0';

                                                if (AC_CHECK == true)
                                                        K++;

                                                if (point == false)
                                                {
                                                        R = R + '.';
                                                        point = AC_CHECK = true;
                                                }

                                                NUL--;
                                        }
                                }
                        }
                }
        }

        if (Lenght > Xi)
                Xi = Xi + (Lenght - Xi);
}

T[Ti] = X[N];
Ti++;
N++;

while (K < AC + 1)
{
        switch (step)
        {
        case 0:
                if (Ti < Yi)
                {
                        if (zero == true)
                                zero = false;
                        else
                        {
                                R = R + '0';
                                if (AC_CHECK == true)
                                        K++;
                        }

                        if (N == Lenght && point == false)
                        {
                                R = R + '.';
                                point = AC_CHECK = true;
                        }

                        if (N < Xi)
                        {
                                if (Ti == 0 && X[N] == 0)
                                        N++;
                                else
                                {
                                        T[Ti] = X[N];
                                        Ti++;
                                        N++;
                                }
                        }
                        else
                                Ti++;
                        continue;
                }

                for (int i = 0; i < Ti; i++)
                {
                        if (T[i] < Y[i])
                        {
```

```
                                        if (zero == true)
                                                zero = false;
                                        else
                                        {
                                                R = R + '0';
                                                if (AC_CHECK == true)
                                                        K++;
                                        }

                                        if (N == Lenght && point == false)
                                        {
                                                R = R + '.';
                                                point = AC_CHECK = true;
                                        }

                                        if (N < Xi)
                                        {
                                                T[Ti] = X[N];
                                                Ti++;
                                                N++;
                                        }
                                        else
                                                Ti++;
                                        step++;
                                        i = Ti;
                                        continue;
                                }

                                if (T[i] > Y[i])
                                {
                                        step++;
                                        i = Ti;
                                        continue;
                                }
                        }

                        if (step == 1)
                        {
                                if (Ti > Yi)
                                        M = (T[0] * 10 + T[1]) / Y[0];
                                else
                                        M = T[0] / Y[0];

                                continue;
                        }

                        if (N < Xi)
                        {
                                R = R + '1';
                                if (AC_CHECK == true)
                                        K++;

                                if (N == Lenght && point == false)
                                {
                                        R = R + '.';
                                        point = AC_CHECK = true;
                                }

                                zero = true;
                                T.clear();
                                T.resize(LENGHT);
                                Ti = 0;
                        }
                        else
                        {
                                if (R.length() > 0)
                                        R = R + '1';
                                else
                                        R = '1';

                                R = NullBeginCheck(R);
                                R = NullEndCheck(R);

                                if (sign == false)
                                        R = '-' + R;

                                return R;
                        }

                        break;
case 1:
                        Zi = C = 0;

                        for(int i = Yi - 1; i >= 0; i--, Zi++)
                        {
                                Z[Zi] = Y[i] * M + C;
                                C = 0;

                                if (Z[Zi] >= 10)
                                {
                                        C = Z[Zi] / 10;
                                        Z[Zi] = Z[Zi] % 10;
```

```
		}
}

while (C > 0)
{
		Z[Zi] = C;
		C = C / 10;
		Zi++;
}

if (Zi > Ti)
{
		M--;
		Z.clear();
		Z.resize(LENGHT);
		continue;
}

if (Ti == Zi)
{
		for (int i = 0, j = Zi - 1; i < Ti; i++, j--)
		{
			if (T[i] < Z[j])
			{
				M--;
				Z.clear();
				i = Ti;
				continue;
			}

			if (T[i] > Z[j])
			{
				i = Ti;
				continue;
			}
		}

		if (Z.empty())
		{
			Z.resize(LENGHT);
			continue;
		}
}

R = R + std::to_string(M);
if (AC_CHECK == true)
		K++;

for (int i = Ti - 1, j = 0; i >= 0; i--, j++)
{
		if (T[i] - C < Z[j])
		{
			T[i] = T[i] + 10 - C - Z[j];
			C = 1;

			continue;
		}

		T[i] = T[i] - C - Z[j];

		C = 0;

}

Z.clear();
Z.resize(LENGHT);
Zi = 0;

bool begin = true;

for (int i = 0; i < Ti; i++)
{
		if (begin == true && T[i] == 0)
			continue;

		begin = false;

		Z[Zi] = T[i];
		Zi++;
}

if (begin == true)
{
		T.clear();
		T.resize(LENGHT);
		Ti = 0;
}
else
{
		T.clear();
		T.resize(LENGHT);
		Ti = Zi;
```

```
                                        for (int i = 0; i < Zi; i++)
                                                T[i] = Z[i];

                                        Z.clear();
                                        Z.resize(LENGHT);
                                        Zi = 0;
                                }

                                step--;
                                zero = true;

                                break;
                        }
                }

                if (R.find('.') != std::string::npos && R.substr(R.find('.') + 1).length() > AC)
                        R = R.substr(0, R.length() - (R.substr(R.find('.') + 1).length() - AC));

                R = NullBeginCheck(R);
                R = NullEndCheck(R);

                if (sign == false)
                        R = '-' + R;

                X.clear();
                Y.clear();
                Z.clear();
                T.clear();

                return R;
}
```