МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний авіаційний університет

ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ, КОМП’ЮТЕРНОЇ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

КАФЕДРА КОМП’ЮТЕРИЗОВАНИХ СИСТЕМ УПРАВЛІННЯ

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач випускової кафедри

О.Є. Литвиненко

« » 20 р.

**ДИПЛОМНА РОБОТА**

**(пояснювальна записка)**

випускника освітньоГО СТУПЕНЯ магістра

ЗА СПЕЦІАЛІЗАЦІЄЮ «СИСТЕМНЕ ПРОГРАМУВАННЯ»

**Тема: «Програмний комплекс шифрування даних з використанням сплайнів»**

Виконавець: студент групи СП–224М Чередниченко Дмитро Олексійович

(студент, група, прізвище, ім’я, по батькові)

Керівник: кандидат технічних наук, доцент, Нечипорук Віталій Володимирович

(науковий ступінь, вчене звання, прізвище, ім’я, по батькові)

Нормоконтролер: Тупота Євгеній Вікторович

(підпис) (П.І.Б.)

Київ 2020

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра комп’ютеризованих систем управління

Спеціальність: 123 «Комп'ютерна інженерія»

(шифр, найменування)

Освітньо–професійна програма «Системне програмування»

ЗАТВЕРДЖУЮ

Завідувач кафедри

О. Є. Литвиненко

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2020 р.

**ЗАВДАННЯ**

**на виконання дипломної роботи**

Чередниченка Дмитра Олексійовича

(П.І.Б. випускника)

1. Тема роботи «Програмний комплекс шифрування даних з використанням сплайнів» затверджена наказом ректора від «27» липня 2019 р. № 1808/ст.

2. Термін виконання роботи: з «14» жовтня 2019 р. по «09» лютого 2020 р.

3. Вихідні дані роботи: завдання на дипломну роботу, документація для розроблення програмного комплексу.

4. Зміст пояснювальної записки: аналіз стану проблеми та постановка завдання дослідження, функції декомпозиції та реконструкції сплайн–вейвлетових перетворень, програмний комплекс для шифрування даних.

5. Перелік обов'язкового графічного (ілюстративного) матеріалу:

1) структура програмного модуля (схема структурна);

2) основні етапи роботи програмного модуля (схема алгоритму);

3) діаграма прецедентів;

4) діаграма класів;

5) діаграма послідовностей.

6. Календарний план–графік

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Завдання | Термін виконання | Підпис керівника |
| 1 | Ознайомлення з постановкою завдання дипломного проекту | 15.10.2019 – 17.10.2019 |  |
| 2 | Аналіз літературних джерел та інтернет ресурсів | 18.10.2019 –  20.10.2019 |  |
| 3 | Проектування архітектури програмного комплексу | 21.10.2019 –  12.11.2019 |  |
| 4 | Проектування інтерфейсу користувача | 13.11.2019 –  18.11.2019 |  |
| 5 | Програмування програмного комплексу | 19.11.2019 –  07.01.2020 |  |
| 6 | Підготовка пояснювальної записки | 08.01.2020 –  27.01.2020 |  |
| 7 | Підготовка графічного матеріалу | 28.01.2020 –  03.02.2020 |  |
| 8 | Захист дипломного проекту | 04.02.2020 |  |

7. Дата видачі завдання: « 14 » жовтня 2020 р.

Керівник дипломної роботи Нечипорук В.В.

(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання Чередниченко Д.О.

(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи на тему «Програмний комплекс шифрування даних з використанням сплайнів»: 88 с., 9 рис., 1 табл., 16 літературних джерел, 4 додатки.

БЛОЧНИЙ ШИФР, ВЕЙВЛЕТОВЕ ПЕРЕТВОРЕННЯ, КРИПТОАНАЛІЗ, СПЛАЙН, ДЕКОМПОЗИЦІЯ ТА РЕКОНСТРУКЦІЯ, *C#*.

Об’єкт дослідження – алгоритми шифрування на основі сплайн–вейвлетових перетворень.

Предмет дослідження – програмний комплекс шифрування даних з використанням сплайнів.

Мета дипломного проекту – розробити програмний комплекс шифрування даних для побудови криптографічно стійких алгоритмів.

Методи проектування – програмна документація про реалізацію поширених алгоритмів шифрування, методи використання сплайнів в обробці цифрових сигналів та інформації.

Результати дипломного проекту: теоретичні (методи та алгоритми) та практичні (програмна реалізація комплексу). Призначенні для використання при розробці програмного забезпечення, для забезпечення надійності даних. Та використання вже готової системи для забезпечення надійності файлам.

ЗМІСТ

[ВСТУП 6](#_Toc31077362)

[РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ ДОСЛІДЖЕННЯ 13](#_Toc31077363)

[1.1 Сучасне програмне забезпечення для шифрування даних 13](#_Toc31077364)

[1.2 Криптоаналіз блочних алгоритмів шифрування 21](#_Toc31077365)

[1.3 Постановка завдання дослідження 26](#_Toc31077366)

[1.4. Висновки до розділу 26](#_Toc31077367)

[РОЗДІЛ 2 АЛГОРИТМИ ШИФРУВАННЯ З ВИКОРИСТАННЯ СПЛАЙНІВ 27](#_Toc31077368)

[2.1. Використання сплайнів першого порядку 27](#_Toc31077369)

[2.2. Використання сплайнів другого порядку 29](#_Toc31077370)

[2.3. Використання сплайнів третього порядку 29](#_Toc31077371)

[2.4. Висновки до розділу 30](#_Toc31077372)

[РОЗДІЛ 3 ПРОГРАМНИЙ КОМПЛЕКС ШИФРУВАННЯ ДАНИХ 31](#_Toc31077373)

[3.1. Структура програми 31](#_Toc31077374)

[3.2. Функціональна модель 34](#_Toc31077375)

[3.3. Основні процедури та функції 37](#_Toc31077376)

[3.4. Керівництво користувача 38](#_Toc31077377)

[3.5 Висновки до розділу 41](#_Toc31077378)

[РОЗДІЛ 4 ПЕРЕВІРКА РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ 42](#_Toc31077379)

[4.1. Порівняння з поширеними алгоритмами 42](#_Toc31077380)

[4.2. Перевірка на криптографічну стійкість 44](#_Toc31077381)

[4.3. Висновки до розділу 44](#_Toc31077382)

[ВИСНОВКИ 45](#_Toc31077383)

[СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ 46](#_Toc31077384)

[ДОДАТОК А 48](#_Toc31077385)

[ДОДАТОК Б 84](#_Toc31077386)

[ДОДАТОК В 91](#_Toc31077387)

[ДОДАТОК Г 108](#_Toc31077388)

# ВСТУП

З кожним роком зростають обсяги інформацію, які потрібно передавати, обробляти та зберігати для своєчасного використання їх при вирішенні виробничих, наукових, економічних та інших завдань. Основними властивостями інформації як об’єкта захисту є конфіденційність (властивість інформації, яка полягає в тому, що інформація не може бути отримана неавторизованим користувачем), цілісність (означає неможливість модифікації неавторизованим користувачем) та доступність (властивість інформації бути отриманою авторизованим користувачем, за наявності у нього відповідних повноважень, в необхідний для нього час). Одним з методів забезпечення всіх цих властивостей є шифрування даних на різних етапах (при зберігання чи обміні).

Захист інформації від несанкціонованого доступу включає різноманітні аспекти не пов’язані безпосередньо з кодуванням і криптографією. Однак розробники програмного забезпечення повинні мати уявлення про загрози, які формуються ними ж при недотриманні вимог безпеки при розробці того чи іншого програмного забезпечення, наприклад, системи криптографічного захисту.

Сучасне програмне забезпечення, яке випускається навіть гігантами індустрії програмного забезпечення, характеризується помилками, що з’являються і зникають від версії до версії і непередбаченими можливостями, які зловмисники використовують в своїх цілях.

Програмне забезпечення включає два основних компоненти:

* операційні системи;
* прикладне програмне забезпечення.

Операційні системи є основною частиною програмного комплексу комп’ютера і при цьому виконують величезний набір проміжних операцій між прикладними програмами і апаратним забезпеченням. Природно, що помилки в них цікавлять зловмисників понад усе. Від рівня реалізації політики безпеки в кожній конкретній операційній системі багато в чому залежить і загальна безпека інформаційної системи. У більшості операційних систем є механізми ідентифікації користувача, які забезпечують той чи інший рівень захисту інформації. Основні методи захисту інформації від несанкціонованого доступу в операційній системі такі:

* захист інформації за допомогою матриці управління доступом та списків управління доступом;
* захист інформації за допомогою "паролів";
* захист інформації за допомогою криптографічних методів.

Недоліки двох перших методів полягають у тому, що "ключі" доступу зберігаються в самій системі. Це може призвести до того, що підготовлений недобросовісний користувач може їх розкрити i скористатись секретною інформацією. А при шифруванні ключ доступу не зберігається у системі. Користувач вводить його тільки тоді, коли зашифровує або розшифровує інформацію. Питання шифрування дешифрування інформації є предметом криптології.

Основною проблемою прикладного програмного забезпеченя є розробка його фірмами, які просто не в змозі виділити кошти на ретельну перевірку свого продукту. Помилки в прикладному програмному забезпеченні були і залишаються основним шляхом проникнення зловмисника як на сервери, так і на робочі станції. Об’єктивна причина цього – розробка подібного програмного забезпечення різними групами розробників, які просто не в змозі приділити належної уваги мережевій і локальній безпеці свого продукту. І якщо фірми–розробники операційних систем витрачають величезні суми на ретельні випробування поведінки їх програм в нестандартних ситуаціях, а також активно враховують багаторічний досвід своїх же помилок, то для невеликих фірм це просто не під силу, та і украй невигідно економічно.

Сплайн − функція, область визначення якої розбита на шматки, на кожному зі шматків функція є деяким поліномом. В задачах інтерполяції, інтерполяція сплайном краща, ніж інтерполяція многочленом, оскільки дає схожі результати навіть при менших степенях поліномів. Максимальний степінь поліномів в сплайні називається степенем сплайна. Різниця між степенем сплайна і його гладкістю називається дефектом сплайна.

Сплайни мають численні застосування як в математичній теорії, так і в прикладній математиці (зокрема, в різноманітних обчислювальних програмах). Зокрема, сплайни двох змінних інтенсивно використовуються для завдання поверхонь в різних системах комп'ютерного моделювання. Сплайни двох аргументів називають бі–сплайнами (наприклад, Бікубічеський сплайн), які є двовимірними сплайнами, що моделюють поверхні. Їх часто плутають з B–сплайнами (базисними сплайнами), які є одновимірними і в лінійній комбінації складають криві – каркас для «натягування» поверхонь. Також з базисних сплайнів можливо скласти тривимірну конструкцію для моделювання об'ємних тіл.

Початок розвитку теорії інтерполяції сплайнами та й сам термін сплайн відраховують з 1946 року зі статті Ізо Шонберга (*Isaac Jacob Schoenberg*). Особливо інтенсивний її розвиток відбувся в 50–70 роки, традиційною прикладною сферою використання інтерполяційних сплайнів стали в цей час системи автоматизованого проектування. Однак потенційні можливості сплайнів значно ширші ніж просто опис деяких кривих. В реальному світі велика кількість фізичних процесів за самою своєю природою є сплайнами. В механіці це деформація гнучкої пластини чи стержня, зафіксованих в окремих точках; траєкторія руху тіла, якщо сила, що діє на нього змінюється ступінчасто (траєкторія штучного космічного об’єкта з активними та інерційними відрізками руху, траєкторія руху літака при ступінчастій зміні тяги двигунів та зміні профілю крила і т. д.). В термодинаміці це теплообмін в стержні складеному з фрагментів з різною теплопередачею. В хімії — дифузія через шари різних речовин. В електриці — поширення електромагнітних полів через різнорідні середовища. Тобто сплайн не надумана математична абстракція, а в багатьох випадках він є розв’язанням диференційних рівнянь, які описують цілком реальні фізичні процеси.

Для визначення інтерполяційного сплайна з неперервною першою похідною достатньо розрахувати значення першої похідної у вузлах. Спосіб визначення похідних у вузлах сплайна визначає широку різноманітність інтерполяційних сплайнів. Часто похідні визначаються не як константи, а як деякі залежності від інтерпольованої функції та сітки інтерполяції.

Якщо значення першої похідної у вузлах розраховувати виходячи з умови неперервності другої похідної (вирішуючи систему з n лінійних рівнянь) то сплайн матиме дві неперервні похідні. Такий спосіб побудови сплайна, як і сам сплайн називають глобальним, оскільки при визначенні кожного з його коефіцієнтів враховується вся множина вузлів інтерполяції.

В інших випадках, для визначення окремого коефіцієнта, враховуються лише найближчі вузли інтерполяції і такі способи побудови, як і самі сплайни, називають локальними. Параметри фрагмента такого сплайна можна визначити незалежно від інших фрагментів.

Для сплайнів є характерними такі ознаки: сплайн складається з фрагментів — функцій одного класу, які різняться лише своїми параметрами; на сусідні фрагменти в точках стикування накладаються певні умови, що зводяться до неперервності значень та деяких перших похідних. Сплайни — напрямок прикладної математики, що інтенсивно розвивається.

В математиці, серії вейвлетів є подання квадратної інтегрованої (дійсні або комплексні значення) функції за певною ортонормованої серії, породженої вейвлетом. В даний час, вейвлет–перетворення є одним з найбільш популярних тимчасових частот–перетворювань.

Розглядають функцію (взяту будучи функцією від часу) в термінах коливань, локалізованих за часом і частоті.

Використовуються в обробці сигналів, нерідко замінюючи звичайне перетворення Фур'є в багатьох областях фізики, включаючи молекулярну динаміку, обчислення ab initio, астрофізику, локалізацію матриці щільності, сейсмічну геофізику, оптику, турбулентність, квантову механіку, обробку зображень, аналізи кров'яного тиску, пульсу і ЕКГ, аналіз ДНК, дослідження білків, дослідження клімату, загальну обробку сигналів, розпізнавання мови, комп'ютерну графіку, мультифрактальний аналіз та інші.

Вейвлет–аналіз застосовується для аналізу нестаціонарних медичних сигналів, в тому числі в електрогастроентерографіі.

З розвитком обчислювальної техніки почалося використання сплайнів в комп’ютерній графіці, моделюванні, а в даний час сплайни і вейвлети широко застосовуються для обробки числових потоків, що дозволяє виділяти характерні складові, економити ресурси обчислювальної системи і каналів передачі (час обробки і передачі, обсяги пам’яті для зберігання інформації і т. п.). Перераховані властивості зберігаються і для алгоритмів шифрування основаних на вейвлетном перетворенні просторів сплайнів. Використання ланцюжків вкладених просторів сплайнів дозволяє будувати вейвлетові перетворення (декомпозицію і реконструкцію) в дуже загальних умовах, в тому числі, з використанням нерівномірної сітки, остання може розглядатися як ключ в шифруванні вихідного числового потоку, при якому результати шифрування представлений вейвлетовим перетворенням. Оскільки постійно розглядаються нові спосіб шифрування, то побудова сплайн–вейвлетних перетворень і розробка на їх основі сплайн–вейвлетових криптосистем представляються актуальними. Однією з областей математичного моделювання традиційно є криптографія.

Криптографія – наука про методи забезпечення конфіденційності, цілісності даних, аутентифікації. Спочатку криптографія вивчала методи шифрування інформації – оборотного перетворення відкритого тексту на основі секретного алгоритму або ключа в шифрований текст. Традиційна криптографія утворює розділ симетричних криптосистем, в яких зашифровування і розшифрування проводиться з використанням одного і того ж секретного ключа. Крім цього розділу сучасна криптографія включає в себе асиметричні криптосистеми, системи електронного цифрового підпису, хеш–функції, управління ключами, отримання прихованої інформації, квантову криптографію.

Криптографічний алгоритм, або шифр, — математична формула, що описує процеси шифрування і розшифрування. Щоб зашифрувати відкритий текст, криптоалгоритм працює в сполученні з ключем — словом, числом або фразою. Те саме повідомлення одним алгоритмом, але різними ключами буде перетворюватися в різний шифротекст. Захищеність шифротексту цілком залежить від двох речей: стійкості криптоалгоритму і таємності ключа. Криптоалгоритм плюс усілякі ключі і протоколи, що приводять їх у дію, складають криптосистему.

Інформація, що може бути прочитана, осмислена і зрозуміла без яких–небудь спеціальних заходів, називається відкритим текстом (*plaintext*). Метод перетворення відкритого тексту таким чином, щоб сховати його суть, називається зашифруванням (*encryption*). Шифрування відкритого тексту приводить до його перетворення в незрозумілу абракадабру, іменовану шифротекстом (*ciphertext*). Шифрування дозволяє сховати інформацію від тих, для кого вона не призначається, попри те, що вони можуть бачити сам шифротекст. Протилежний процес перетворення шифротексту в його вихідний вид називається розшифруванням (*decryption*).

Сучасні дослідження симетричних алгоритмів шифрування зосереджено, в основному, навколо блочних та потокових алгоритмів шифрування та їхнього застосування. Блочний шифр подібний до поліалфавітного шифру Алберті: блочні шифри отримують фрагмент відкритого тексту та ключ, і видають на виході шифротекст такого самого розміру. Оскільки повідомлення зазвичай довші за один блок, потрібен деякий метод склеювання послідовних блоків. Було розроблено декілька методів, що відрізняються в різних аспектах. Вони є режимами дії блочних шифрів та мають обережно обиратись під час застосування блочного шифру в криптосистемі.

Шифри *Data Encryption Standard* (*DES*) та *Advanced Encryption Standard* (*AES*) є стандартами блочних шифрів, затверджених урядом США (однак, стандартизацію *DES* було скасовано після прийняття стандарту *AES*). Не зважаючи на те, що стандарт *DES* було визнано застарілим, він (та особливо його все ще дійсний варіант *triple DES*) залишається досить популярним; він використовується в багатьох випадках, від шифрування в банкоматах до забезпечення приватності електронного листування та безпечного доступу до віддалених терміналів. Було також розроблено багато інших шифрів різної якості. Багато з них було зламано.

Потокові шифри, на відміну від блочних, створюють ключ довільної довжини, що накладається на відкритий текст побітово або політерно. В потокових шифрах, потік шифротексту обчислюється на основі внутрішнього стану алгоритму, який змінюється протягом його дії. Зміна стану керується ключем, та, в деяких алгоритмах, ще і потоком відкритого тексту. *RC4* є прикладом добре відомого, та широко розповсюдженого потокового шифру.

На відміну від симетричних, асиметричні алгоритми шифрування використовують пару споріднених ключів — відкритий та секретний. При цьому, не зважаючи на пов'язаність відкритого та секретного ключа в парі, обчислення секретного ключа на основі відкритого вважається технічно неможливим.

В асиметричних криптосистемах, відкритий ключ може вільно розповсюджуватись, в той час як приватний ключ має зберігатись в таємниці. Зазвичай, відкритий ключ використовується для шифрування, в той час як приватний (секретний) ключ використовується для дешифрування.

# РОЗДІЛ 1

АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ ДОСЛІДЖЕННЯ

## 1.1 Сучасне програмне забезпечення для шифрування даних

*TrueCrypt* – комп'ютерна програма для шифрування на «льоту» для 32– і 64–розрядних операційних систем сімейств *Microsoft Windows NT* 5 і новіше (*GUI*–інтерфейс), *Linux* і *Mac OS X*. Дозволяє створювати зашифрований логічний (віртуальний) диск, що зберігається у вигляді файлу. За допомогою *TrueCrypt* також можна повністю зашифрувати розділ жорсткого диска або інший носій інформації, наприклад флеш–накопичувач *USB*. Всі збережені дані в томі *TrueCrypt* повністю шифруються, включаючи імена файлів і каталогів. Змонтований тому *TrueCrypt* подібний до звичайного логічного диску, тому з ним можна працювати за допомогою звичайних утиліт перевірки та дефрагментації файлової системи.

За допомогою *TrueCrypt* можна створювати зашифрований віртуальний диск:

* в файлі–контейнері, що дозволяє легко працювати з ним – переносити, копіювати (в тому числі на зовнішні пристрої як будь–який інший звичайний файл), перейменовувати або видаляти;
* до вигляді зашифрованого розділу диска, що робить роботу більш продуктивною і зручною, в версії 5.0 з'явилася можливість шифрувати системний розділ;
* шляхом повного шифрування вмісту пристрою, такого як флеш–накопичувач *USB* (дискети не підтримуються починаючи з версії 7.0).

У список підтримуваних *TrueCrypt* 6.2 алгоритмів шифрування входять *AES*, *Serpent* і *Twofish*. Попередні версії програми також підтримували алгоритми з розміром блоку 64 біта (*Triple DES*, *Blowfish*, *CAST5*) (включаючи версії 5.х, які могли відкривати, але не створювати розділи, захищені цими алгоритмами). Крім того, можливе використання каскадного шифрування різними шифрами, наприклад *AES* + *Twofish* + *Serpent*.

Всі алгоритми шифрування використовують режим *XTS*, який більш надійний для шифрування «на льоту», ніж *CBC* та *LRW*, що застосовувалися в попередніх версіях (работа с вже створеними шіфроконтейнерамі в цих форматах також можлива).

Програма дозволяє вибрати одну з трьох хеш–функцій: *HMAC–RIPEMD–160*, *HMAC–Whirlpool*, *HMAC–SHA–512* для генерування ключів шифрування, солі і ключа заголовка.

Для доступу до зашифрованих даних можна застосовувати пароль (ключову фразу), ключові файли (один або кілька) або їх комбінації. В якості ключових файлів можна використовувати будь–які доступні файли на локальних, мережевих, знімних дисках (при цьому використовуються перші 1 048 576 байт) і генерувати свої власні ключові файли.

Одна з примітних можливостей *TrueCrypt* – забезпечення двох рівнів правдоподібного заперечення наявності зашифрованих даних, необхідного в разі вимушеного відкриття пароля користувачем: створення прихованого томів, що дозволяє задати другий пароль (і набір ключових файлів) до звичайного того, щоб отримати доступ до своїх даних, які неможливо отримати доступ з основним паролем, при цьому прихований тому може мати будь–яку файлову систему і розташовується в невикористаний просторі основного томи.

Інші можливості *TrueCrypt*:

* Переносимість, що дозволяє запускати *TrueCrypt* без установки в операційній системі (необхідно мати права групи адміністраторів в *NT*).
* Підтримка створення зашифрованого динамічного файлу на дисках *NTFS*. Такі томи *TrueCrypt* збільшуються в розмірі по мірі накопичення нових даних аж до зазначеного максимального розміру. Проте це трохи зменшує продуктивність і безпеку системи.
* Шифрування системного фізичного або логічного диска для *Windows*–систем з передзавантажувальною аутентифікацією (*TrueCrypt* не здатний виконувати шифрування *GPT*–дисків, які встановлені в більшості сучасних комп'ютерів, тому перед шифруванням їх необхідно перетворити в *MBR* (*master boot record*). Така ж функціональність вбудована в *Windows Vista* (не всіх редакцій), *Windows 7* (Корпоративна і Максимальна), *Windows Server 2008*, *Windows 8.1* під ім'ям *BitLocker*. Однак *BitLocker* не пропонує функції правдоподібного заперечення і має програму з закритим кодом, що унеможливлює перевірку на наявність вразливостей або вбудованих лазівок для обходу захисту. Своя система шифрування – *FileVault* – вбудована в *Mac OS X*, починаючи з версії 2.0 вона вміє шифрувати весь диск. Як і *BitLocker*, *FileVault* не пропонує функції правдоподібного заперечення і має програму з закритим кодом що унеможливлює перевірку на наявність вразливостей або вбудованих лазівок для обходу захисту).
* Зміна паролів і ключових файлів для томи без втрати зашифрованих даних.
* Можливість резервного збереження і відновлення заголовків томів (1024 байт).
* Це може бути використано для відновлення заголовка пошкодженого файлу, щоб монтувати тому після помилки на апаратному рівні, в результаті якої пошкодився заголовок.
* Відновлення старого заголовка також скидає пароль томи на той, який діяв для колишнього заголовка.
* Можливість призначати комбінації клавіш для монтування / размонтирования розділів (в тому числі і швидкого від'єднання зі стиранням ключа в пам'яті, закриттям вікна і очищенням історії), відображення і приховування вікна (і значка) *TrueCrypt*.
* Можливість використовувати *TrueCrypt* на комп'ютері з правами звичайного користувача (в тому числі створювати файлові контейнери і працювати з ними), однак первісну установку програми повинен виконати адміністратор.

*BestCrypt* – платний пакет пропрієтарних програм для створення на жорсткому диску комп'ютера віртуального зашифрованого диска (контейнера) – одного або кількох. Віртуальний диск працює, як звичайний дисковий розділ. Шифрування і розшифрування йдуть у фоновому режимі, і користувач не помічає різниці в роботі зі звичайним і зашифрованим диском, який, при необхідності, можна перетворити в звичайний, але нечитаний, файл. У програми *BestCrypt* є опція створення «прихованого» контейнера, який криптографически гарантовано не можна визначити ніякими засобами.

У *BestCrypt* є можливість шифрування своп–файлу (файлу віртуальної пам'яті) – в ньому зберігаються тимчасові дані, в тому числі там можуть виявитися документи і паролі, тому його шифрування є корисною можливістю з точки зору захисту інформації. Також присутня функція шифрування системного диска з наступним завантаженням з нього. У комплект програми входить утиліта *BCWipe*, що дозволяє видаляти файли без можливості їх відновлення. Файли, що підлягають знищенню, затираються випадковими даними, також чистяться службові таблиці розміщення файлів і області вільного місця.

У плані збереження інформації *BestCrypt* – одна з кращих програм при роботі з зашифрованими контейнерами. Крім безпосередньої функції зберігання інформації в зашифрованому файлі–контейнері, ця програма має комплексом додаткових функцій, істотно розширюють її можливості. це:

* можливість створення прихованого контейнера в уже створеному явному контейнері;
* можливість зберігання контейнерів на будь–яких типах носіїв (як мережевих, так і локальних) з можливістю їх переміщення, копіювання, дублювання зі збереженням всіх захисних можливостей;
* можливість закриття контейнера по "гарячим" клавішах або після певного часу при неактивності користувача;
* можливість шифрування свап–файлу;
* функція захисту контейнера від випадкового видалення;
* утиліта гарантованого знищення даних.

Для шифрування даних в програмі використовується один з чотирьох запропонованих алгоритмів:

* алгоритм *Rijndael* (*AES*) в режимі зчеплення блоків шифротекста (*Cipher Block Chaining mode*);
* російський федеральний стандарт ГOСT 28147–89 в режимі зі зворотним зв'язком по шіфротекста (*Cipher Feedback mode*);
* *Blowfish* в режимі зчеплення блоків шифротекста (*Cipher Block Chaining mode*);
* *Twofish* в режимі зчеплення блоків шифротекста (*Cipher Block Chaining mode*).

Всі вони вважаються найбільш сильними шифрувальними алгоритмами з існуючих, і всі вони реалізовані з 256–бітовим ключем. При необхідності можна змінювати параметри шифрування: алгоритм і ключ.

Створення контейнерів здійснюється досить просто, але є одна маленька хитрість. Після завдання параметрів контейнера вам доведеться трохи посидіти перед екраном монітора, давлячи на будь–які клавіші клавіатури, поки в діалоговому вікні не заповниться смуга прогресу. Тільки після цього відбудеться власне створення контейнера. При створенні контейнерів треба враховувати, що максимальний розмір контейнера обмежений для різних файлових систем. Якщо для *NTFS* обмеження не встановлені, то для *FAT 32* це обмеження становить 4 Гб, а для *FAT 16* – 2 Гб. Також програма не дозволить вам використання паролів для контейнерів менше шести знаків. Ще однією особливістю можна назвати те, що програма при введенні пароля зчитує не символ, а коди натиснутих клавіш, тому 1, набрана на цифровій клавіатурі, не є еквівалентною 1, набраної на основній клавіатурі.

Однією з основних сильних сторін програми вважається можливість створення прихованих контейнерів в уже створеному контейнері. Процес створення прихованого контейнера аналогічний створенню звичайного контейнера, тільки необхідно спочатку демонтувати (відключити від системи) первинний контейнер. Кожен з контейнерів матиме власний пароль для доступу і може мати власні алгоритм і ключ для шифрування. Після монтування контейнерів кожен з них видно в системі як логічний диск під власною буквою. Тепер всі програми можуть працювати з даними на цих дисках в прозорому режимі. Саме монтування (підключення до системи) може здійснюватися як на вимогу, так і в автоматичному режимі.

Наявність флеш–карт великого розміру робить *BestCrypt* особливо зручним. Створивши такий зашифрований контейнер на флеш–карті, можна спокійно носити все конфіденційні дані з собою, не боячись за їх цілісність при втраті носія. Такий спосіб дозволяє не тримати конфіденційні дані в місцях, де немає гарантії захисту від несанкціонованого доступу до самого комп'ютера. У такому випадку дані краще прибирати в сейф.

Як відомо, прикладні програми можуть залишати за собою "неприбраними" різні тимчасові файли, за якими можна отримати хоча б частковий доступ до секретних даних. Крім того, існує чимало методів відновлення видалених файлів. *BestCrypt* зводить такий ризик до мінімуму за рахунок наявності утиліти зачистки вільного постранстве. Видаливши всі тимчасові файли, а також стали непотрібними робочі файли, можна запустити утиліту *BCWipe* (яку треба завантажити з сайту окремо) і надійно вичистити вільний простір.

Іншим потенційним джерелом втрати даних може послужити свап–файл. Його можна зашифрувати, *BestCrypt* може це зробити за допомогою спеціальної утиліти *CryptoSwap*, яка входить до складу програми. Таким чином, при правильному використанні всього інструментарію програми можна досить надійно захистити ваші дані від несанкціонованого доступу. Єдиним більш надійним способом в цьому випадку буде тільки шифрування системного диска цілком.

*VeraCrypt* – програмне забезпечення, що використовується для шифрування «на льоту». *VeraCrypt* – безкоштовний і відкритий проект, розпочатий в якості ФОРКОМ *TrueCrypt*. Запущено і по теперішній час підтримується Mounir Idrassi, засновником компанії *IDRIX*, після того як було оголошено про припинення підтримки програми *TrueCrypt*.

*VeraCrypt* може використовувати такі алгоритми шифрування: *AES*, *Serpent*, *Twofish*, *Camellia*, Коник, а також комбінації цих алгоритмів. Використовувані криптографічні хеш–функції: *RIPEMD–160*, *SHA–256*, *SHA–512*, Стрибог і *Whirlpool*. *VeraCrypt* використовує режим шифрування *XTS*.

Ключ заголовка і вторинний ключ заголовка для режиму *XTS* генеруються за допомогою алгоритму *PBKDF2* з використанням 512–бітної криптографічного солі, число ітерацій становить від 327 661 до 655 331, в залежності від використовуваної хеш–функції.

За твердженням розробників, в *VeraCrypt* реалізований ряд удосконалень в області безпеки в порівнянні з *TrueCrypt*.

У той час як *TrueCrypt* використовує 1000 ітерацій при генерації ключа, яким шифрується системний розділ при використанні алгоритму *PBKDF2–RIPEMD–160*, *VeraCrypt* використовує 327 661 ітерацію. Для стандартних шифрованих розділів на диску і файлових контейнерів *VeraCrypt* використовує 655 331 ітерацію для хеш–функції *RIPEMD–160* і 500 000 ітерацій для *SHA–2* і *Whirlpool*. Це істотно уповільнює *VeraCrypt* при відкритті зашифрованих розділів диска при їх монтуванні, але робить його не менше ніж в 10 (і не більше ніж в 300) раз більш стійким до атаки прямим перебором.

Виправлена ​​уразливість початкового завантажувача для *Windows*. Для режиму завантаження з зашифрованого розділу додана підтримка алгоритму *SHA–256* і виправлені проблеми з уразливістю *ShellExecute* для *Windows*.

Для *Linux* і *macOS* додана підтримка дисків з секторами більше 512 байт. Для Linux реалізована підтримка розділів, відформатованих під *NTFS*.

Зазначені вдосконалення привели до несумісності з форматом розділів *TrueCrypt*. Розробники *VeraCrypt* визнали старий формат *TrueCrypt* занадто вразливим для потенційної атаки і відмовилися від нього. Це одне з головних відмінностей між *VeraCrypt* і конкуруючим проектом–Форком *TrueCrypt* – *CipherShed*, який продовжує підтримувати старий формат. Починаючи з версії 1.0f, *VeraCrypt* може відкривати і перетворювати в свій формат розділи, відформатовані в форматі *TrueCrypt*. У версію 1.18а додана можливість шифрування дискових розділів, відформатованих в форматі *GPT*.

*VeraCrypt*, також як і *TrueCrypt*, підтримує можливість заперечується шифрування, дозволяючи створювати всередині зашифрованого томи ще один, «прихований тому». Крім того, версія *VeraCrypt* для *Windows* дозволяє створювати і виконувати прихований екземпляр операційної системи *Windows*, чиє наявність також можна правдоподібно заперечувати. У документації *VeraCrypt* перерахований ряд способів, якими ця можливість могла б виявитися скомпрометованої (наприклад, витік даних через тимчасові файли на незашифрованому диску) і можливі способи боротьби з цією проблемою.

*VeraCrypt* підтримує паралельну роботу в багатопроцесорних і багатоядерних системах. *VeraCrypt* вміє використовувати апаратне прискорення шифрування, доступне в процесорах, що мають набір інструкцій *AES–NI*.

Програма схильна до ряду потенційних атак, до яких відчутно і інше програмне забезпечення для шифрування дисків, наприклад *BitLocker*. Для пом'якшення цієї небезпеки розробники *VeraCrypt* надали користувачам ряд профілактичних рекомендацій. Деякі з цих проблем перераховані нижче.

*VeraCrypt* зберігає ключі в оперативній пам'яті у відкритому вигляді. Теоретично, зловмисник міг би отримати доступ до її вмісту за допомогою так званої атаки методом холодної перезавантаження, при якій атакуючий отримує фізичний доступ до вмісту модулів оперативної пам'яті комп'ютера після його виключення і за допомогою спеціального програмного забезпечення або обладнання відновлює їх старе вміст. Подібна атака була успішно застосована відносно розділів диска, зашифрованих *TrueCrypt*. Для протидії деяким атакам даної категорії, починаючи з версії 1.24 в *VeraCrypt* додано шифрування ключів в оперативній пам'яті, а також стирання ключів з пам'яті при перезавантаженні або завершення роботи системи.

*VeraCrypt* не забезпечує безпеки даних на комп'ютері, до якого атакуючий має фізичний доступ, як правило, таємний, в процесі роботи з зашифрованими даними. Ця вразливість відноситься не до випадку втрачених, конфіскованих або вкрадених комп'ютерів, а коли зловмисники мають можливість встановити на комп'ютер той чи інший вид шпигунської апаратури – апаратний кейлоггер, bus–master пристрій, що володіє прямим доступом до оперативної пам'яті, або якесь ще пристрій, призначене для вирішення подібних завдань.

*VeraCrypt* не забезпечує безпеки даних на комп'ютері з встановленим шкідливим ПЗ. Багато шкідливі програми цього типу містять в собі кейлоггери і можуть, зокрема, зчитувати вводяться з клавіатури паролі і передавати їх зловмисникам.

*VeraCrypt* не підтримує взаємодії з *Trusted Platform Module* (*TPM*), так як, на думку розробників, *TPM* не може бути використаний для вирішення завдань інформаційної безпеки, оскільки *TPM* не забезпечує ніякого захисту комп'ютера від зловмисника, який отримав фізичний доступ до комп'ютера і, наприклад, встановив апаратний кейлоггер.

## 1.2 Криптоаналіз блочних алгоритмів шифрування

Блочний шифр – різновид симетричного шифру, що оперує групами біт фіксованої довжини – блоками, характерний розмір яких змінюється в межах 64–256 біт. Якщо вихідний текст (або його залишок) менше розміру блоку, перед шифруванням його доповнюють. Фактично, блочний шифр являє собою підстановку на алфавіті блоків, яка, як наслідок, може бути моно– або поліалфавітних. Блочний шифр є важливою компонентою багатьох криптографічних протоколів і широко використовується для захисту даних, що передаються по мережі.

Від поточних шифрів робота блочного відрізняється обробкою біт групами, а не потоком. Симетричні системи мають перевагу над асиметричними в швидкості шифрування, що дозволяє їм залишатися актуальними, незважаючи на більш слабкий механізм передачі ключа (одержувач повинен знати секретний ключ, який необхідно передати по вже налагодженій зашифрованому каналу. У той же час, в асиметричних шифри відкритий ключ, необхідний для шифрування, можуть знати всі, і немає необхідності в передачі ключа шифрування).

До переваг блокових шифрів відносять схожість процедур шифрування і розшифрування, які, як правило, відрізняються лише порядком дій. Це спрощує створення пристроїв шифрування, так як дозволяє використовувати одні і ті ж блоки в ланцюгах шифрування і дешифрування. Гнучкість блокових шифрів дозволяє використовувати їх для побудови інших криптографічних примітивів: генератора псевдовипадкової послідовності, поточного шифру і криптографічних хешей.

Блоковий шифр складається з двох парних алгоритмів: шифрування та розшифрування. Обидва алгоритми можна представити у вигляді функцій. Функція шифрування *E* на вхід отримує блок даних *M* розміром *n* біт і ключ *K* розміром *k* біт і на виході віддає блок шифротекста *C* розміром *n* біт:

Функція розшифрування D на вхід отримує шифр C, ключ K і на виході віддає M: будучи, при цьому, зворотної до функції шифрування.

Піонерами в розробці блокових шифрів стали співробітники компанії *IBM* при роботі над шифром «*Lucifer*». Вони спроектували перші основи, які стали використовуватися при розробці наступних схем. При цьому слід враховувати, що новий шифр повинен бути не тільки стійким до всіх відомих видів атак, але і досить простий в реалізації.

Більшість блочних шифрів є ітеративними. Це означає, що даний шифр перетворює блоки відкритого тексту постійної довжини в блоки шифротекста тієї ж довжини за допомогою циклічно повторюваних оборотних функцій, відомих як раундові функції. Це пов'язано з простотою і швидкістю виконання як програмних, так і апаратних реалізацій. Зазвичай раундові функції використовують різні ключі, отримані з початкового ключа: , де Ci – значення блоку після i–го раунду, C0 = M – відкритий текст, Ki – ключ, який використовується в i–му раунді і отриманий з початкового ключа K.

Розмір блоку *n* – це фіксований параметр блокового шифру, зазвичай рівний 64 або 128 бітів, хоча деякі шифри допускають кілька різних значень. Довжина 64 біта була прийнятна до середини 90–х років, потім використовувалася довжина 128 біт, що приблизно відповідає розміру машинного слова і дозволяє ефективну реалізацію на більшості поширених обчислювальних платформ. Різні схеми шифрування дозволяють зашифровувати відкритий текст довільної довжини. Кожна має певні характеристики: ймовірність помилки, простота доступу, вразливість до атак. Станом на 2006 рік 80–бітний ключ здатний був запобігти атаці грубою силою.

*SP*–мережа (*Substitution–permutation network*) – один з найважливіших типів ітеративних блокових шифрів. Шифр на основі *SP*–мережі отримує на вхід блок і ключ і робить кілька чергуючихся раундів, що складаються з чергування стадій підстановки і стадій перестановки. Для досягнення безпеки досить одного S–блоку, але такий блок буде вимагати великого обсягу пам'яті. Тому використовуються маленькі *S*–блоки, змішані з *P*–блоками. Нелінійна стадія підстановки перемішує біти ключа з бітами відкритого тексту, створюючи конфуз Шеннона. Лінійна стадія перестановки розподіляє надмірність по всій структурі даних, породжуючи дифузію.

*S*–блок заміщає маленький блок вхідних біт на інший блок вихідних біт. Ця заміна повинна бути взаємно однозначною, щоб гарантувати оборотність. Призначення *S*–блоку полягає в нелінійному перетворенні, що перешкоджає проведенню лінійного криптоаналізу. Одним з властивостей *S*–блоку є лавинний ефект, тобто зміна одного біта на вході призводить до зміни всіх біт на виході. *P*–блок – перестановка всіх біт: блок отримує на вхід результат *S*–блоку, змінює місцями всі біти і подає результат *S*–блоку наступного раунду. Важливим якістю *P*–блоку є можливість розподілити виходи одного *S*–блоку між входами якомога більшого числа *S*–блоків. Для кожного раунду використовується свій, одержуваний з початкового, ключ. Подібний ключ називається раундовим. Він може бути отриманий діленням спочатку ключа на рівні частини, так і будь–яким перетворення всього ключа.

Мережа Фейстеля – це загальний метод перетворення довільної функції F в перестановку на безлічі блоків. Вона складається з циклічно повторюваних осередків – раундів. Усередині кожного раунду блок відкритого тексту поділяється на дві рівні частини. Раундова функція бере одну половину, перетворює її з використанням ключа Ki і об'єднує результат з другою половиною допомогою операції виключного АБО (*XOR*). Цей ключ задається початковим ключем *K* і різний для кожного раунду. Далі половинки міняються місцями (інакше буде перетворюватися тільки одна половина блоку) і подаються на наступний раунд. Перетворення мережі Фейстеля є оборотною операцією.

Для функції *F* існують певні вимоги:

* її робота повинна приводити до лавинному ефекту;
* повинна бути нелінійна по відношенню до операції *XOR*.

У разі невиконання першого вимоги, мережа буде схильна до диференціальним атакам (схожі повідомлення будуть мати схожі шифри). У другому випадку дії шифру лінійні і для злому достатньо рішення системи лінійних рівнянь. Подібна конструкція має істотну перевагу: процедури шифрування/дешифрування збігаються, тільки похідні від початкового ключі використовуються в зворотному порядку. Це означає, що одні й ті ж блоки можуть використовуватися як для шифрування, так і для розшифрування, що, безумовно, спрощує реалізацію шифру. Недолік схеми полягає в тому, що в кожному раунді обробляється тільки половина блоку, що призводить до необхідності збільшувати число раундів.

При побудові алгоритму враховують формування групи, в якій елементами є безліч блоків шифротекста при всіх ключах, а груповий операцією – композиція раундів шифрування. Якщо даний шифр утворює практично повну групу, не має сенсу застосовувати кратне шифрування.

Як і всі шифри, алгоритми яких відомі, блокові шифри піддаються криптографічним атакам. Мета атаки – розробити алгоритм злому більш ефективний, ніж повний перебір всіх можливих ключів. У разі знаходження такого рішення, атака вважається успішною. При цьому шифр є зламаним, якщо існують атака, що дозволяє провести злом за час, протягом якого інформація зберігає актуальність, і проведення подібної атаки вигідно зловмисникові.

Атака повним перебором. Завдяки такій характеристиці блочного шифру, як оборотність функції, його висновок стає відмінним від істинної випадкової послідовності внаслідок парадоксу днів народження. Ця особливість призводить до зниження безпеки шифру і необхідності брати до уваги розмір блоку. Таким чином, існує компроміс між великими, що знижують продуктивність шифру, і ненадійними маленькими блоками. Не менш важливу роль відіграє розмір ключа. Ранній шифр DES характеризувався розміром ключа в 56 біт, що, як показала практика, явно не достатньо для надійної пересилання даних. Саме атакою повним перебором вперше був розкритий DES. Більш сучасні алгоритми, такі як *AES* і ДСТУ ГОСТ 28147–89 мають розмір ключа в 128 біт і 256 біт відповідно, що робить безглуздим подібні атаки.

Диференціальний криптоаналіз. У 1990 році Елі Біхам і Аді Шамір визначили ідею диференціального криптоаналізу. За допомогою цього методу вдалося зламати шифр *DES*. Подібної атаці піддаються шифри з постійним *S*–блоком і шифрування в режимі кодової електронної книги. Даний метод працює з парами шифротекста, для яких відомо відмінність відповідних відкритих текстів, і розглядає еволюцію цих відмінностей. Поряд з лінійним є найпоширенішим при атаках на блоковий шифр.

Лінійний криптоаналіз. Лінійний криптоаналіз – метод розтину шифру, заснований на пошуку афінних наближень для роботи алгоритму. Розроблено японським математиком Міцуру Мацуї, першим застосував цю техніку для атаки на *DES і FEAL*. Метод заснований на застосуванні операції виключає АБО (*XOR*) до блокам відкритого тексту, шифротекста і до їх результату, що дозволяє отримати результат застосування *XOR* для бітів ключа. Структура *S*–блоку впливає на стійкість до лінійним атакам. Коли метод був розроблений, виявилося, що *DES* має слабкість до нього, так як ніхто не припускав подібних атак при його розробці.

Інтегральний криптоаналіз. Інтегральний криптоаналіз – вид атак, особливо застосовний до блоковий шифрів, побудованим на *SP*–мережі. На відміну від диференціального криптоаналізу, що використовує пару обраного відкритого тексту з фіксованим різницею, обчисленої за допомогою операції *XOR*, інтегральний криптоаналіз використовує безлічі відкритих текстів, в яких одні частини утримуються постійними, в той час як інші варіюються серед всіляких значень. Подібне безліч з необхідністю має суму по модулю 2 (*XOR*), що дорівнює 0, в той час, як відповідна сума шифротекста містить інформацію про операції шифру.

Атака на основі пов'язаних ключів. Криптографічний атака, в якій криптоаналітик може спостерігати за роботою алгоритму шифрування або розшифрування, що використовує кілька секретних ключів. Дана атака не є простим перебором всіх можливих значень ключа. Спочатку криптоаналітик нічого не знає про точному значенні ключів, але передбачається, що атакуючому відомо деяке математичне відношення, що зв'язує між собою ключі. Наприклад, співвідношення може бути просто значенням xor з відомою константою: або більш складний зв'язок виду, де – довільна функція, вибрана атакуючим. В реальному житті такі залежності можуть виникнути при збоях в апаратному забезпеченні або погано спроектованих протоколах безпеки.

## 1.3 Постановка завдання дослідження

Метою даного дослідження є розробка програмного комплексу шифрування даних з використанням сплайнів. Алгоритми шифрування повинні бути стійкими до атаки повним перебором (мінімальна довжина ключа не повинна бути менше 128 біт, а також наявність можливості використовувати ключі довжиною 512 біт та більше), диференціального та лінійного криптоаналізу (функції декомпозиції та реконструкції не повинні базуватися на використанні *S*–блоків).

У відповідності до сформульованої мети у роботі поставлено наступні завдання:

1. Проаналізувати сучасний стан розробки симетричних алгоритмів шифрування.
2. Розробити структуру програмного комплексу.
3. Окреслити формули декомпозиції та реконструкції.
4. Реалізувати алгоритми шифрування/дешифрування.
5. Розробити модуль шифрування даних.
6. Протестувати роботу розробленого програмного комплексу.

## 1.4. Висновки до розділу

У першому розділі проведено аналіз сучасних програмних комплексів для шифрування даних. Визначено критерії для розроблення алгоритмів шифрування, здатних протидіяти більшості методів криптоаналізу. Проаналізовано сучасний стан розробки симетричних алгоритмів шифрування. І виокремлено критерії до алгоритму, які дозволять зробити алгоритм стійким до більшості атак.

Завдання розроблюваного комплексу є реалізація та доповнення алгоритмів, які не мають широкого застосування, але мають високі показники криптографічної стійкості та надійності алгоритму. Поставлено за мету розробити програмний комплекс, який буде реалізовувати надійні алгоритми шифрування.

# РОЗДІЛ 2

АЛГОРИТМИ ШИФРУВАННЯ З ВИКОРИСТАННЯ СПЛАЙНІВ

## 2.1. Використання сплайнів першого порядку

Запропонована криптосистема відноситься до симетричних, блоковим, при шифрування і дешифрування використовується один ключ. Ключем є нерівномірна сітка *X*, порядок викидання вузлів з цієї сітки *γ*. Позначимо ключ = *(X, γ).* Кількість раундів − *K*. Всі обчислення проводяться в кінцевих полях, по модулю *N* (*N* − просте число). Сітка *X* є циклічно впорядкованим безліччю, її елементи − вузли *xj: X* = *{xj} j ∈ N0*, де *N0* − безліч натуральних чисел з нулем. Сітка X періодична, з періодом, рівним *P*, так що *xj = xj + P* для *∀j ∈ Z*.

Послідовність номерів викиданих вузлів *γ: γ = {γn} n∈ [0, ..., K–1],* де *K* − число раундів шифрування, *γn* − номер випадково обраного вузла xj.

Відкритим текстом є послідовність *C = {ci} i ∈ N0*. Вона розбивається на *L* блоків однакової довжини, | *CL | = M*, тут | *CL* | − кількість елементів в блоці. При необхідності послідовність {*ci} i = 0, ..., M–k* буде продовжуватися періодично, період буде рівним *M–k*, тут *M–k* – кількість в блоці елементів на k–му раунді.

У всіх формулах *i, j ∈ N0*. На кожному раунді з сітки викидається один вузол з номером *γk*, далі при шифруванні за формулами декомпозиції вважається послідовність *{cik } i∈J*, після проведення *K* раундів виходить шифротекст. При дешифрування текст відновлюється за допомогою формул реконструкції.

Опишемо процес шифрування. На першому кроці шифрування отримуємо відкритий текст *{Ci} i=0, … , M–1.* З допомогою ключа та формул декомпозиції проводиться перший раунд шифрування.

З первісної сітки викидаємо вузол .

Позначимо отриману сітку як , а її вузли відповідно символами

(2.1)

(2.2)

Для викинутого вузла введемо позначення .

Розглянемо формули декомпозиції для сплайнів першого порядку.

(2.3)

(2.4)

(2.5)

Проведемо циклічний зсув вправо елементів (на останньому раунді зсув не проводиться)

Після ­–раундів отримуємо послідовність вкладених сіток

та дві послідовності .

Послідовністьявляється шифротекстом.

Процес дешифрування. Він проводиться по аналогії з процесом шифрування. Використовується той самий ключ та формули реконструкції. Знаючи число раундів , шифротекст розбивається на дві послідовності

Беремо сітку та .

Випишемо формули реконструкції:

(2.6)

(2.7)

(2.8)

Проведемо циклічний зсув вліво елементів (на останньому раунді зсув не проводиться)

## 2.2. Використання сплайнів другого порядку

Етапи роботи алгоритму співпадаються з описаним вище. Головна відмінність полягає в формулах декомпозиції та реконструкції.

Випишемо формули декомпозиції:

(2.9)

(2.10)

(2.11)

(2.12)

Випишемо формули реконструкції:

(2.13)

(2.14)

(2.15)

(2.16)

## 2.3. Використання сплайнів третього порядку

Випишемо формули декомпозиції:

(2.17)

(2.18)

(2.19)

(2.20)

(2.21)

Випишемо формули реконструкції:

(2.22)

(2.23)

(2.24)

(2.25)

(2.26)

## 2.4. Висновки до розділу

# РОЗДІЛ 3

ПРОГРАМНИЙ КОМПЛЕКС ШИФРУВАННЯ ДАНИХ

## 3.1. Структура програми

Для написання програми використовувалася мова програмування *C#.* Розроблені класи *SplineFirst*, *SplineSecond*, *SplineThird* унаслідувані від розробленого класу *Spline* (Додаток А) мають подібну структуру, але відрізняються діями в методах. Клас *Spline* відповідає за операції в кінцевих полях та містить такі елементи:

* + поле *modDefault* зберігає число, по модулю котрого проходять розрахунки;
  + поле *division* зберігає масив елементів, який заміняює операцію ділення в кінцевих полях;
  + метод *Div* повертає обернене за модулем число;
  + метод *Mod* виконує операцію ділення по модулю.

Класи *SplineFirst*, *SplineSecond*, *SplineThird* відповідають за шифрування та дешифрування з використанням сплайнів першого, другого та третього порядків відповідно та містять такі елементи:

* + поле *k\_max* зберігає кількість раундів шифрування;
  + поле *y* зберігає порядок викидання елементів з сітки;
  + поле *eps* зберігає викинуті елементи з сітки;
  + поле *x* зберігає вкладені сітки;
  + метод *GenerateKey* створює ключ шифрування;
  + метод *CheckKey* перевіряє ключ шифрування;
  + метод *CreateGrid* заповнює поля *k\_max*, *eps, y*,  *x* відповідно до ключа;
  + метод *EncryptBlock* виконує шифрування по блокам;
  + метод *DecryptBlock* виконує дешифрування по блокам;
  + метод *Encrypt* виконує шифрування всього вхідного потоку;
  + метод *Decrypt* виконує дешифрування всього вхідного потоку.

Також програма має форму зі своїми обробниками подій (Додаток Б), які відповідають за відображення назв вхідних та вихідних файлів, ключа шифрування, тексту шифрування/дешифрування, повідомляють про успішне завершення процесу чи помилки під час виконання. Структура розробленого програмного комплексу зображена на рис. 3.1.



Рис. 3.1. Структура програмного комплексу

Структура представлена у вигляді 5 блоків «Графічний інтерфейс» використовується для зручного керування програмним забезпеченням, з його допомогою заповнюється вся необхідна інформація та повідомляється про процес виконання. «Файлова система» виконує операції з файлами, такі як читання та запис. «Модуль шифрування даних» отримує вхідні повідомлення по блокам та виконує операції шифрування чи дешифрування, та повертає отримані результати. «Модуль обробки даних» отримує вхідний потій, розбиває його на блоки, доповнює в разі необхідності та надсилає на шифрування/дешифрування в «Модуль шифрування даних», генерую та перевіряє ключі шифрування. «Модуль керування» реагує на запити з «Графічного інтерфейсу» перевіряє вхідні дані на правильність, комутує дані та команди між іншими модулями.

На рис. 3.2 зображено алгоритм роботи програмного комплексу.



Рис. 3.2. Алгоритм роботи програмного комплексу

## 3.2. Функціональна модель

Розроблений програмний модуль має наступні функціональні можливості:

* зашифровувати дані з файлу в файл;
* зашифровувати дані з текстового поля в файл;
* розшифровувати дані з файлу в файл;
* розшифровувати дані з файлу в текстове поле;
* шифрувати/дешифрувати дані трьома алгоритмами на основі сплайн–вейвлетових перетворень першого, другого та третього порядку;
* генерувати ключі шифрування;
* зберігати ключі в файл;
* відкривати ключі з файлу.

На рис. 3.3–3.5 відображені основні діаграми побудовані за допомогою мови *UML*, що використовувалися при розробці та тестуванні.

На діаграмі прецедентів зображено дії, які можуть виконувати об’єкти в огляді представленої системи. На діаграмі класів представлено властивості та функції, виконуванні об’єктами. На діаграмі послідовностей зображено основні кроки при роботі з розробленим програмним модулем в часовій залежності.

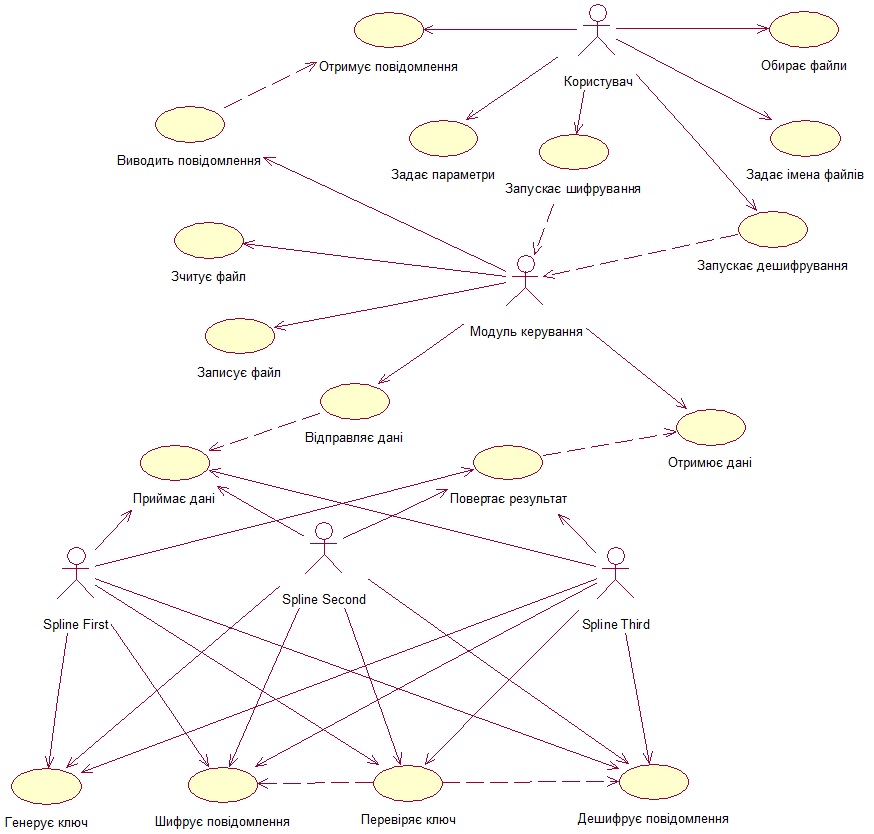


Рис. 3.3. Діаграма прецедентів

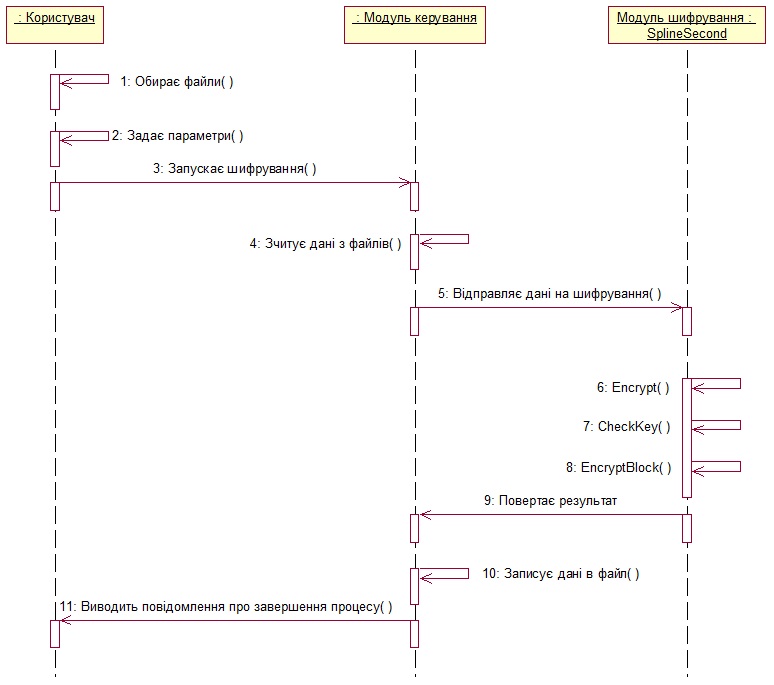


Рис. 3.4. Діаграма послідовностей

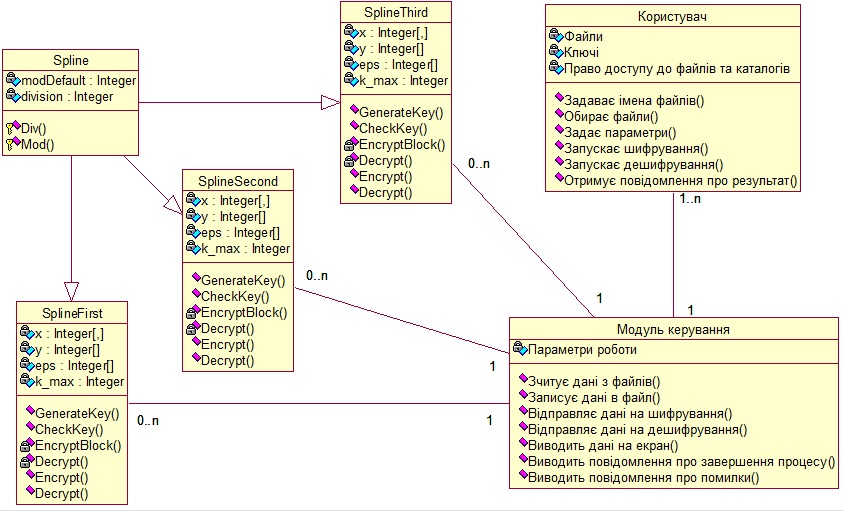


Рис. 3.5. Діаграма класів

## 3.3. Основні процедури та функції

*Div* виконує аналог ділення в кінцевих полях, а саме повертає обернене за модулем число.

*Mod* виконує операцію ділення по модулю, так як вбудована в мову *C#* відповідна функція, не виконує жодних операцій з від’ємними числами.

*GenerateKey* створює ключ шифрування, відповідно за розміром блока та алгоритмом шифрування. Перший байт ключа містить довжину блока в байтах, другий байт визначає кількість раундів шифрування та залежить від алгоритму, для сплайна першого порядку , для другого – , третього – , де – кількість раундів, – розмір блоку в байтах. Далі знаходиться сітка , розмірністю . А потім порядок викиданих вузлів γ розмірністю .

*CheckKey* перевіряє ключ шифрування на вище описані правила. А також перевіряє відповідність довжині блока шифрування.

*CreateGrid* проводить викидання вузлів сітки відповідно до порядок вказаного в ключі, та записує значення сітки на кожному раунді для пришвидшення роботи програми при шифруванні багатьох блоків з одним і тим самим ключем.

*EncryptBlock* виконує шифрування по блокам, використовуючи сформовану сітку та ключ. *DecryptBlock* виконує дешифрування по блокам, використовуючи сформовану сітку та ключ.

*Encrypt* виконує всі перевірки вхідних даних (ключа та інформації для шифрування дешифрування, розбиває на блоки вхідний потік, та надсилає дані блоками в *EncryptBlock*. *Decrypt* виконує всі перевірки вхідних даних (ключа та інформації для шифрування дешифрування, розбиває на блоки вхідний потік, та надсилає дані блоками в *DecryptBlock*.

## 3.4. Керівництво користувача

Вигляд розробленого програмного комплексу представлено на рис. 3.6 (при роботі з файлами) та рис. 3.7 (при роботі з текстом).



Рис. 3.6. Зображення форми програмного модуля для роботи з файлами

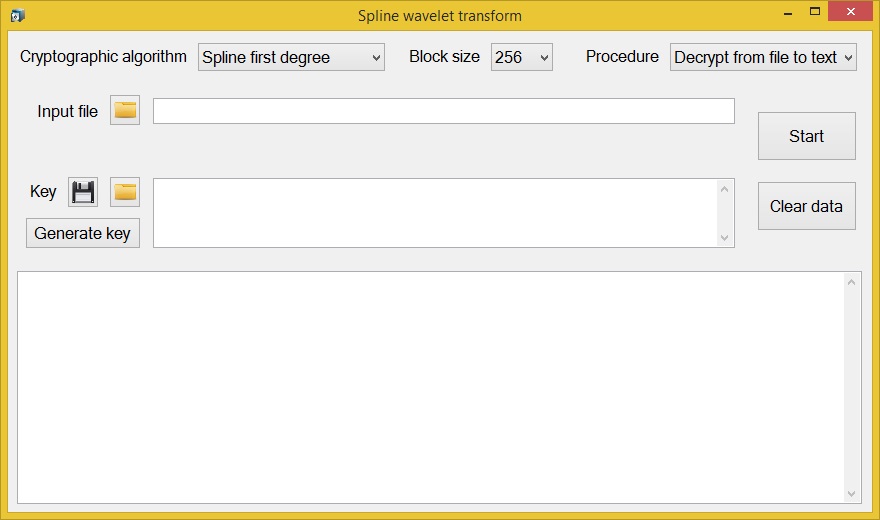


Рис. 3.7. Зображення форми програмного модуля для роботи з текстом

Далі буде описано дії при роботі з даною програмою. Спочатку обирається алгоритм шифрування один з запропонованих «Spline first degree», «Spine second degree», «Spine third degree», та розмір блоку шифрування, один з наведених 64, 128, 192, 256, 384, 512, 768, 1024 біт. Також необхідно вибрати між шифрувати чи дешифрувати та працювати з файлами чи текстом. Якщо обрано зашифрувати текст в файл, то повідомлення, яке потрібно зашифрувати вводиться в нижньому текстовому полі, а розташування вихідного файлу вводиться вручну в верхньому текстовому полі, або при натисненні на кнопку зліва від нього вибирається за допомогою діалогового вікна. Файл з зашифрованим вмістом пропонується зберігати в розширенні *spline*. При дешифруванні в текст, результат буде відображено в нижньому полі. Дані про вхідний файл вказуються, так само як і з вихідним файлом. В полі ключа відображується його значення в шістнадцятирічному форматі. Його можна редагувати вручну, зберігати в файл та відкривати з файлу. При генерації ключа враховується виставлені значення алгоритму та розміру блока, саме вони є вхідними даними для його генерації. Приклад заповнених полів представлено на рис. 3.7, а на рис. 3.8 зображено повідомлення про успішне завершення.

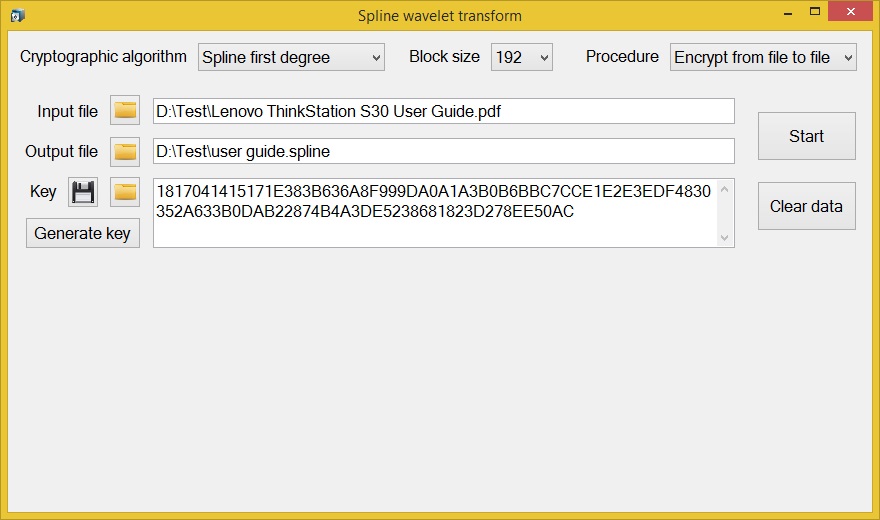


Рис. 3.7. Зображення форми програмного модуля при заповнених полях

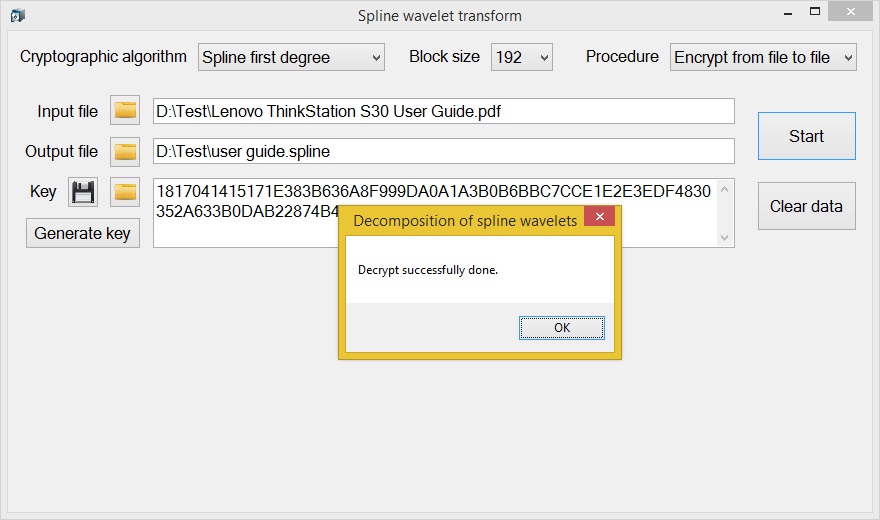


Рис. 3.8. Зображення повідомлення про успішне завершення процесу шифрування

## 3.5 Висновки до розділу

Розроблено структуру програмного комплексу, визначено його функціональні можливості з наведенням пояснень. Розглянуто структуру розроблених класів.

Реалізовано алгоритми шифрування/дешифрування з використанням сплайнів. Розроблено модуль шифрування даних, наведено алгоритм його роботи. Описано процедури та функції, які використовуються в розробленому програмному модулі. Наведено детальну, покрокову інструкцію з використання розробленого продукту.

Протестовано роботу розробленого програмного модуля. Виявлено некритичні помилки та описано способи їх вирішення.

# РОЗДІЛ 4

ПЕРЕВІРКА РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

## 4.1. Порівняння з поширеними алгоритмами

DES (Data Encryption Standard) - алгоритм для симетричного шифрування. Розмір блоку для DES дорівнює 64 бітам. В основі алгоритму лежить мережу Фейстеля з 16 циклами (раундами) і ключем, що має довжину 56 біт. Алгоритм використовує комбінацію нелінійних (S-блоки) і лінійних (перестановки E, IP, IP-1) перетворень.

Triple DES (3DES) – симетричний блоковий шифр, створений на основі алгоритму DES з метою усунення головного недоліку останнього – малої довжини ключа (56 біт), який може бути зламаний методом повного перебору ключа. Швидкість роботи 3DES в 3 рази нижче, ніж у DES, але криптостійкість набагато вище – час, необхідний для криптоаналізу 3DES, може бути в мільярд разів більше, ніж час, потрібний для розтину DES. 3DES використовується частіше, ніж DES, який легко зламуються за допомогою сучасних технологій. 3DES є простим способом усунення недоліків DES. Алгоритм 3DES побудований на основі DES, тому для його реалізації можливо використовувати програми, створені для DES. Офіційна назва алгоритму, що використовується в стандартах – TDEA або Triple DEA (англ. Triple Data Encryption Algorithm). Однак, термін «3DES» використовується більш широко постачальниками, користувачами і розробниками криптосистем.

Схема алгоритму 3DES має такий вигляд: , де – ключі для кожного DES–кроку, – вхідні дані, які потрібно шифрувати.

Порівняння швидкодії виконувалося шифруванням та дешифруванням файлу розміром 15109161 байт, кожен алгоритм з відповідною довжиною ключа чи блока виконував процес 100 разів з різними ключами. Код програми, що виконувала відповідні дії представлений в Додатку В. У таблицю занесені середні результати. Зміст журналу тестування можна побачити в Додатку Г.

Таблиця 4.1

Час виконання шифрування та дешифрування реалізованих алгоритмів

|  |  |  |
| --- | --- | --- |
| Алгоритм (розмір блоку, розмір ключа в бітах) | Час шифрування, мс | Час дешифрування, мс |
| AES (128, 128) | 125 | 126 |
| AES (128, 192) | 138 | 140 |
| AES (128, 256) | 154 | 155 |
| 3–DES (64, 128) | 811 | 813 |
| 3–DES (64, 192) | 810 | 810 |
| SplineFirst (128, 256) | 3674 | 3279 |
| SplineFirst (256, 512) | 4407 | 4013 |
| SplineFirst (512, 1024) | 6044 | 5848 |
| SplineFirst (1024, 2048) | 9048 | 9079 |
| SplineSecond (128, 255) | 6851 | 5132 |
| SplineSecond (256, 511) | 8461 | 6710 |
| SplineSecond (512, 1023) | 11519 | 9606 |
| SplineSecond (1024, 2047) | 17046 | 14704 |

Продовження таблиці 4.1

|  |  |  |
| --- | --- | --- |
| SplineThird (128, 254) | 7616 | 6071 |
| SplineThird (256, 510 | 9564 | 7726 |
| SplineThird (512, 1022) | 12688 | 10703 |
| SplineThird (1024, 2046) | 18248 | 15887 |

## 4.2. Перевірка на криптографічну стійкість

## 4.3. Висновки до розділу

# ВИСНОВКИ

# СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ДСТУ 3008-95 Структура і правила оформлення. Документація. Звіти у сфері науки і техніки.

2. Единая система конструкторской документации. Основные надписи: ГОСТ 2.104-2006. – Взамен ГОСТ 2.104-68; введ. 01.09.06. – М.: Стандартинформ, 2006. – 14 с.

3. Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения: ГОСТ 19.701-90. – Введ. 01.01.92. – М.: Изд-во стандартов, 1991. – 25 с.

4. Библиографическое описание документа. Общие требования и правила составления: ГОСТ 7.1-84. – Введ. 01.01.86. – М: Изд-во стандартов, 1985. – 72 с.

5. Таненбаум. Э. Современные операционные системы. 2-е изд. – СПб.: Питер, 2002. – 1040 с.

6. Троелсен. Э. С# и платформа .NET. Библиотека программиста. – СПб.: Питер, 2004. – 796 с.

7. Шилдт, Герберт C# 4.0: полное руководство.: Пер. с англ. – М.: ООО "И.Д. Вильямс", 2011. – 1056 с.

8. Тлумачний словник з інформатики / Г.Г. Півняк, Б.С. Бусигін, М.М. Дівізінюк та ін. – Д., Нац. гірнич. ун-т, 2010. – 495 с.

9. Міжнародний союз електрозв’язку; переклад – Юрій Пероганич. Словник термінів. Всесвітній саміт з питань інформаційного суспільства. Підсумкові документи – 84 с.

10. Феликс Воройский. Информатика. Энциклопедический словарь-справочник. – Litres, 2017-01-12. – 769 с. – ISBN 9785457966338.

11. Вербіцький О. В. Вступ до криптології. — Л. : ВНТЛ, 1998. — 248 с.

12. Шнайер Б. Прикладная криптография. — М. : Триумф, 2002. — 816 с.

13. Демьянович Ю. К., Левина А. Б.О вэйвлетных разложениях линейных пространств над произвольным полем и о некоторых приложениях // Математическое моделирование. 2008. Т. 20, No 11.С. 104–108.

14. Демьянович Ю. К., Левина А. Б.Вэйвлетные разложени яи шифрование // Методы вычислений / под ред. В. М. Рябова. СПб.: Изд-во С.-Петерб. ун-та, 2008. Вып. 22. С. 56–64.

15. Левина А. Б.Работа криптосистемы, основанной на формулах реконструкции и декомпозициина неравномерной сетке // Математические модели: теори яи приложени я/ под ред. М. К. Чиркова.СПб.: Изд-во С.-Петерб. ун-та, 2008. Вып. 9. С. 3–29.

16. Бойчено С В., Іванченко О. В. Положення про дипломні роботи (проекти) випускників Національного Авіаційного Університету – К.: НАУ, 2017. – 63 с.