КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет комп'ютерних наук та кібернетики

Кафедра теоретичної кібернетики

«До захисту допущено»

Завідувач кафедри

Ю. В. Крак _____

(підпис)

« __ »_____20__р.

Дипломна робота на здобуття ступеня бакалавра

За спеціальністю 122 Комп'ютерні науки

на тему:

КРИПТОГРАФІЧНІ АЛГОРИТМИ НА ОСНОВІ ПСЕВДОВИПАДКОВИХ ЧИСЕЛ

Виконав студент 4-го курсу

Бондарець Артем Павлович _____

(підпис)

Науковий керівник:

д.ф.-м.н., проф. Пашко А. О. _____

(підпис)

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

Студент _____

(підпис)

Київ – 2021

РЕФЕРАТ

Робота складається зі вступу, 4 розділів, висновків, списку використаних джерел (15 найменувань). Робота містить 28 рисунків. Загальний обсяг становить 41 сторінка, основний текст роботи викладено на 41 сторінці.

Ключові слова: АБСОЛЮТНО СТІЙКИЙ ШИФР, XOR-ШИФРУВАННЯ, ПСЕВДОВИПАДКОВІ ЧИСЛА, РІВНОМІРНИЙ РОЗПОДІЛ, ТЕСТИ NIST.

Об’єктом роботи є процес шифрування та дешифрування тексту. Метою дипломної роботи є створення програмного засобу для оцінки випадковості криптографічного ключа, а також шифрування тексту з його допомогою.

В якості засобу розроблення системи було обрано Sublime Text 3 та мову програмування Python. Інструментами розроблення були обрані бібліотека для наукових та інженерних розрахунків SciPy та binascii — модуль з методами для перетворення між двійковим представленням та всіма іншими.

У процесі виконання було вилучено з різних файлів та перевірено тестами NIST багато двійкових послідовностей. Було перевірено нульову гіпотезу про рівномірний розподіл цих двійкових послідовностей, перетворених у десяткові числа від 0 до 255. Потім ці двійкові послідовності було використано в якості ключа для шифрування та дешифрування текстів XOR-алгоритмом. Результати виявилися досить непоганими для стандартних псевдовипадкових генераторів мови Python.

Програмний продукт може застосовуватися в навчальному процесі студентами шкіл та університетів для визначення ступеня випадковості різних дискретних величин, а також як демонстрація роботи алгоритму шифру XOR. Якщо оформити додаток у вигляді веб-проекту з гарною візуалізацією, він стане мобільним та зручним для використання будь-ким.

ЗМІСТ

ВСТУП 4
РОЗДІЛ 1. ТЕОРЕТИЧНІ ВІДОМОСТІ 6
1.1 Основні поняття криптографії 6
1.2 Основні принципи захисту інформації 6
1.3 Класифікація шифрів 8
1.4 Гамування 9
1.5 Абсолютно стійкий шифр 9
1.6 Критерій хі-квадрат 10
1.7 Статистичні тести NIST 11
РОЗДІЛ 2. ХАРАКТЕРИСТИКИ ТА ВИМОГИ ДО СИСТЕМИ 16
2.1 Призначення системи 16
2.2 Вимоги до функцій, які виконуються системою 16
2.3 Технічні вимоги 17
РОЗДІЛ 3.РЕАЛІЗАЦІЯ СИСТЕМИ 18
3.1 Зчитування файлу та підготовка послідовності до тестування 18
3.2 Перевірка на рівномірний розподіл 20
3.3 Тести NIST 22
3.4 Кодування та декодування 24
РОЗДІЛ 4. ТЕСТУВАННЯ СИСТЕМИ 25
ВИСНОВКИ 39
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 40

ВСТУП

Оцінка сучасного стану об'єкта розробки. В останні часи спостерігається швидке зростання кількості користувачів глобальною мережею Інтернет. Також зростає кількість інформації, що передається через Всесвітню павутину. Коронавірус та пов'язаний з цим карантин ще сильніше пришвидшили цей процес. Дистанційне навчання в школах та університетах, віддалена робота великих ІТ-компаній, державних та фінансових установ – все це передбачає великі об'єми даних, що подорожують через Глобальну мережу. Постає величезна потреба в захисті цих даних, адже з ростом користувачів, зростає і кількість зловмисників, що бажають заволодіти цінною інформацією.

Надважливим є створення систем з безпечними каналами обміну даних. Удосконалення криптографічних алгоритмів є ключовим аспектом у запобіганні злочинів на просторах Всесвітньої павутини. Криптографічні системи відіграють вирішальну роль у захисті коштовної інформації.

Актуальність теми та підстави для її виконання. Зростаючий інтерес хакерів до злому захищених криптографічних систем, змушує їх удосконалювати засоби до здійснення неправомірних дій. Час від часу до новин потрапляють повідомлення про успішні дії зловмисників. У зв'язку з цим кожна компанія намагається створити власні засоби для протидії кіберзлочинності. В наслідок цього на ринку праці виникає великий дефіцит професіоналів, що спеціалізуються на криптографічних системах.

Важливу роль у захисті інформації відіграє удосконалення вже існуючих криптографічних систем, покращення ключів шифрування, а також ведеться робота над збільшенням випадковості стандартних генераторів псевдовипадкових чисел різних високорівневих мов програмування.

Одним із можливих варіантів використання криптографічних систем залишається шифрування тексту. Сьогодні майже весь документообіг перейшов в онлайн простір, повсякчас використовується електронна пошта.

Месенджери стали основним засобом спілкування. Всі повідомлення та документи, що транспортуються через Глобальну мережу повинні бути захищені.

Ця мета буде досягнута, коли текст зможе бути зашифрований за допомогою дійсно випадкових ключів, які не будуть піддаватися жодним закономірностям, що може дозволити злодіям заволодіти цінною інформацією.

Метою дипломної роботи є створення програмного засобу для шифрування тексту, а також для перевірки на випадковість різних ключів шифрування.

Для досягнення цієї мети поставлені наступні завдання:

- поглибити знання мови Python та опанувати необхідні бібліотеки;
- вилучити бінарний код різних файлів, для подальшого його тестування на випадковість та використання як ключа;
- протестувати ключ на випадковість;
- зашифрувати та розшифрувати текст;

Об’єкт розроблення – процес створення програми для шифрування та дешифрування тексту.

В якості **засобу розроблення** програмного засобу було обрано Sublime Text 3 – легкий та швидкий редактор коду з доступною мовою програмування Python, яке є безкоштовним та вільно поширюваним. Python це мова програмування високого рівня з підтримкою кількох парадигм програмування: об'єктно-орієнтованої, узагальненої та процедурної.

Можливі сфери застосування. Програмний продукт може застосовуватися в таких сферах: перевірка випадковості генераторів псевдовипадкових чисел, шифрування повідомлень.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1 Основні поняття криптографії

Криптографія – це наука, яка за допомогою математичних засобів вивчає конфіденційність та цілісність інформації. Виникнула за практичної потреби надсилати важливі дані захищеними способами. У Криптографії використовується математичний аналіз, теорія ймовірності, математична статистика та інші дисципліни.[1]

Криптографія – наука, що займається розробкою шифрів. При цьому, шифром називається такий метод або спосіб перетворення повідомлень, який забезпечує захист інформації в них від зловмисників[2].

Шифруванням називається процес застосування шифру до повідомлення, що має бути захищеним[2].

Ключем в криптографії називають змінюваний елемент шифру, який застосовується до шифрування конкретного повідомлення[2].

1.2 Основні принципи захисту інформації

Принцип рівномірної міцності захисту. Під час передавання інформації між власниками, вона може охоронятися різними методами в залежності від небезпек, які можуть статися. Таку послідовність вузлів охорони інформації прагне обійти зловмисник, вибравши найменш захищену зону для своїх неправомірних дій. Це дозволяє йому отримати несанкціонований доступ з найбільш малими зусиллями. Виходячи, з цього законні господарі інформації повинні детально продумати всі вузли, адже немає сенсу робити якусь ділянку на шляху передачі інформації дуже складною для злому, а інші незрівнянно простішими.[2]

Принцип доцільності захисту. Сьогодні в епоху надзвичайної діджиталізації та цифрофізації повсякденного життя, а разом з тим і

підвищеним попитом на технічне обладнання, дуже недешево обходяться пристрої та розробка програмного забезпечення як для захисту, так і для несанкціонований доступу до інформації. Тому перш ніж приступити до цього потрібно усвідомити дві речі. Перше, якою буде вигода в разі злому інформації? Чи отримає зловмисник вигоду від злому, чи понесе більше витрат? Друге, які будуть втрати для господаря інформації? Чи понесе він більше витрат від попадання його даних до рук злодіїв, чи витрати будуть більші на сам захист? Співставляючи відповіді на ці питання, потрібно знайти «золоту середину».[2]

Принцип використання ключа. Досить складно створити надійний шифр. Тому оптимальним є створення шифру, який можна використовувати відносно довго. Проте існує ризик, що зловмисник довідається шифр і безперешкодно отримає доступ до великої кількості інформації. Через це сьогодні застосовують ключі.[2]

Принцип Керкгоффса. В 1883 році голландський криптограф Огюст Керкгоффс виклав шість принципів проектування воєнних шифрів в своїй книзі «Воєнна криптографія»[3].

Дослівно вони звучать так:

1. Система немає розшифровуватися, хоча б на практиці, якщо вже це було зроблено математично.
2. Система не має бути секретною. Якщо вона опиниться в руках злочинців, то це не створить проблем.
3. Ключ до системи повинен мати можливість доволі легко поширюватися між учасниками процесу передачі даних. За потреби його можна перевизначити.
4. Систему можливо застосовувати при обміну даними через бездротові засоби.
5. Достатньо однієї людини для виконання сервісу. Також систему можна легко транспортувати.

6. І, найголовніше, система має бути нескладною. Нею повинно бути легко користуватися. А також, не потрібно мати високої кваліфікації.[4]

Суть принципу полягає в тому, що міцність криптосистеми не повинна покладатися на незнанні зловмисником принципів роботи алгоритму[5]. Брюс Шнайер впевнений, що принцип Керкгоффса можна застосувати не тільки до кодів і шифрів, а й для системи в цілому. Кожен секрет створює потенційну проблему. Іншими словами, секретність є основною проблемою. З іншого боку відкритість гарантує гнучкість системи.[3]

1.3 Класифікація шифрів

Шифри класифікують наступним чином:

- за методом шифрування – шифри заміни та шифри перестановки;
- за технологією шифрування – блокові шифри та потокові шифри;
- за особливостями ключів – симетричні шифри та асиметричні[2].

Рисунок 1.1 – Класифікація криптографічних алгоритмів[2]

1.4 Гамування

Своє місце в криптографії посіли шифри гамування. Вони є найефективнішими з точки зору стійкості і швидкості процедур шифрування та дешифрування[6]. Для шифрування і дешифрування використовуються елементарні арифметичні операції - відкрите/зашифроване повідомлення і гамма, представлені в числовому вигляді, сумуються одне з одним по модулю[6]. Сумування зазвичай виконується в якомусь скінченному полі. Наприклад, в полі Галуа *GF(2)* сумування приймає вигляд операції виключної диз'юнкції (XOR)[7].Частковим випадком цього шифрування є шифр Вернама, який використовувався військовими[2][8]. Не будучи шифрувальником, телеграфіст Гільберт Вернам не помилився, коли помітив важливу властивість свого шифру – кожний ключ повинен використовуватися тільки один раз[9]. Американський інженер, криптоаналітик та математик Клод Шеннон довів, що при певних властивостях гами методи шифрування гамування є абсолютно стійкими[10][7].

1.5 Абсолютно стійкий шифр

Абсолютно стійкий шифр. Доведення його існування є одним з найголовніших результатів роботи Клода Шеннона. Цей шифр повинен мати три властивості:

- одноразовість використання
- ключ повинен бути випадковим
- довжина ключа повинна бути не менша за довжину тексту[2]

У випадку недотримання хоча б одної властивості шифр не є абсолютно стійким[8].

Нехай X та Y – алфавіти відкритого і шифрованого текстів такі, що $|X| > 1, \quad |Y| > 1, \quad |Y| \geq |X|$. Шифрування задається ін’єктивним

відображенням $e_k: X \to Y$, дешифрування – відображенням $d_k: e_k(X) \to X, k \in K = \{1, 2, \dots, n\}$.

$E = \{e_k, k \in K\}, D = \{d_k, k \in K\}$ – набір правил шифрування і дешифрування.

Максимальне число $|E| = n$ можливих відображень дорівнює кількості розміщень з $|Y|$ по $|X|$ (число спрособів вибрати підмножину розміром $|X|$ в множині Y, враховуючи порядок елементів). Виникнення чергового символу $x \in X$, вибір ключа $k \in K$ (тобто вибір відображення e_k), отримання зашифрованого тексту $y \in Y$ реалізується з деякими вірогідностями.[11]

1.6 Критерій хі-квадрат

Критерій χ^2 (хі-квадрат, також відомий як критерій Пірсона) застосовується дуже часто в статистиці. В загальному випадку він використовується для перевірки нульової гіпотези чи підлягає випадкова величина певному теоретичному закону розподілу.[12]

Критерій можна використовувати для перевірки гіпотез виду $H_0: F_n(x) = F(x, \theta)$, де θ – відомий вектор параметрів теоретичного закону, а також для перевірки складних гіпотез виду $H_0: F_n(x) \in \{F(x, \theta), \theta \in \Theta\}$, коли оцінка $\hat{\theta}$ скалярного чи векторного параметру розподілу $F(x, \theta)$ рахується на тій же самій вибірці.[13]

Критерій передбачає, що ми працюємо з групованими даними. Тобто першим етапом в застосуванні цього критерію є розбиття області визначення випадкової величини: $x_{(0)}, x_{(1)} \dots x_{(n)}$.

Після того як ми розбили область визначення випадкової величини на інтервали ми повинні порахувати кількість спостережень, які потрапляють до кожного з інтервалів n_i

Після цього вираховуємо спостережувану частоту потраплянь в кожен з інтервалів, тобто відношення $\frac{n_i}{n}$.

Потім рахуємо теоретичну ймовірність потрапляння в кожен з інтервалів згідно з тим законом розподілу, з яким ми перевіряємо згоду, тобто функцію розподілу $P_i(\theta) = F(x_i, \theta) - F(x_{i-1}, \theta)$

Статистика критерію має вигляд $\chi_k^2 = n\sum_{i=1}^{k} \frac{(\frac{n_i}{n} - P_i(\theta))^2}{P_i(\theta)}$

1.7 Статистичні тести NIST

Статистичні тести NIST – пакет статистичних тестів розроблений Лабораторією інформаційних технологій, яка є головною дослідницькою організацією Національного інституту стандартів і технологій (NIST). Цей пакет складається з 15 статистичних тестів, використання яких ставить собі за мету перевірку випадковості двійкових послідовностей, які були згенеровані або апаратними, або програмними генераторами псевдовипадкових чисел. Ці тести базуються на різних статистичних властивостях, які можуть відноситися лише до випадкових послідовностей.[14]

Частотний побітовий тест. Природа даного тесту полягає у визначенні співвідношення між нулями і одиницями всієї двійкової послідовності. Ціль – вияснити, чи є число нулів та одиниць приблизно однаковим. Тест оцінює наскільки близький коефіцієнт одиниць до 0,5. Таким чином, число нулів і одиниць повинно бути приблизно однаковим. Під час тесту вираховується вірогідність; якщо її значення $p < 0{,}01$, то дана двійкова послідовність не є випадковою, інакше послідовність є випадковою. Всі наступні тести проводяться лише при позитивному проходженні даного тесту.[14]

Частотний блоковий тест. Цей тест виявляє частку одиниць всередині блоку довжиною m біт. Ціль – вияснити, чи справді частота повторів одиниць в блоці довжиною m біт дорівнює $m/2$, що є очевидним у випадку випадковості послідовності. Під час тесту також вираховується вірогідність; якщо її значення $p < 0{,}01$, то дана двійкова послідовність не є випадковою,

інакше послідовність є випадковою. Якщо вважати $m = 1$, то даний тест стає частотним тестом.[14]

Тест на послідовність однакових бітів. Задача цього тесту полягає у підрахунку числа рядів послідовності, де під словом «ряд» розуміється неперервна послідовність однакових бітів. Ряд довжиною k біт складається з k однакових бітів, починається і закінчується бітами, що містять протилежні значення. Суть даного тесту – визначити, чи дійсно кількість рядів, що складаються з одиниць та нулів з різними довжинами. Зокрема, визначається наскільки швидко чергуються нулі та одиниці в досліджуваній послідовності. Якщо вірогідність, що вирахувалася під час тесту, менша за 0,01, то дана послідовність не є випадковою. В іншому випадку вона є випадковою.[14]

Тест на найдовшу послідовність одиниць в блоці. Даний тест покликаний визначити найдовшу послідовність одиниць всередині блоку довжиною m біт. Ціль полягає у підтвердженні припущення щодо очікуваної довжини такої послідовності одиниць в абсолютно випадковій двійковій послідовності. Під час цього тесту, як і в попередніх тестах, вираховується вірогідність, яка повинна бути не менша за 0,01 для підтвердження випадковості досліджуваної послідовності. Можна також згадати той факт, що позитивне проходження першого тесту гарантує, що результати цього тесту будуть однакові у випадку дослідження нулів, а не одиниць.[14]

Тест рангів бінарних матриць. У цьому тесті рахуються ранги підматриць, що не пересікаються. Метою цього тесту є перевірка лінійної залежності підрядків фіксованої довжини, що утворюють початкову послідовність. Аналогічно до попередніх тестів, значення вірогідності очікується не менше 0,01 для позитивного проходження тесту.[14]

Спектральний тест. Даний тест проводиться для оцінки висоти пікових значень дискретного перетворення Фур'є. Ціль – виявити періодичні властивості вхідної послідовності, наприклад близьке розташування повторюваних ділянок. Підтвердження існування таких явищ прямо говорить про відхилення від випадковості досліджуваної послідовності. Ідея полягає у

тому, що число піків, які перевищують значення 95% по амплітуді, було значно більше 5%. Знову ж таки вірогідність повинна бути не менше 0,01.[14]

Тест на співпадіння шаблонів, що не перекриваються. У тесті підраховується кількість завчасно визначених шаблонів, знайдених у вхідній послідовності. Метою є виявлення генераторів псевдовипадкових чисел, що занадто часто генерують визначені шаблони, що не перекриваються. Так само, як і в наступному тесті, для пошуку конкретних шаблонів довжиною *m* біт використовується вікно також довжиною *m* біт. Якщо шаблон не вдалося знайти, то вікно переміщується на один біт. У випадку якщо вдалося виявити шаблон, то вікно переміщується на той біт, який іде одразу за знайденим шаблоном і пошук продовжується. Вірогідність очікується не менша за 0,01.

Тест на співпадіння шаблонів, що перекриваються. У даному тесті проводиться підрахунок завчасно визначених шаблонів, знайдених у вхідній послідовності. Як і в попередньому тесті для пошуку шаблонів довжиною *m* біт використовується вікно довжиною *m* біт. Різниця полягає лише у тому, що коли шаблон знайдено, то вікно зміщується на один біт, після чого пошук продовжується. Вірогідність повинна бути така, як і в попередньому тесті.

Універсальний статистичний тест Маурера. Тест виявляє кількість біт між однаковими шаблонами у послідовності. На меті – визначити чи може дана послідовність бути зжата без втрат інформації. Якщо це можливо зробити, то вона не є випадковою. Вірогідність залишається однаковою.[14]

Тест на лінійну складність. В основі цього тесту покладено принцип роботи лінійного регістру зсуву зі зворотним зв’язком. Ціль – вияснити, чи є послідовність достатньо складною, щоб вважатися випадковою. Випадкові послідовності характеризуються довгими лінійними регістрами зсуву зі зворотним зв’язком. Якщо ж такий регістр є коротким, то вважати дану послідовність випадковою не можна. Вірогідність повинна бути не менша за 0,01.[14]

Тест на періодичність. Тест підраховує частоти всіх можливий шаблонів довжини *m* біт, що перекриваються у вхідній послідовності бітів.

Тест покладений виявити чи дійсно кількість появ 2^m шаблонів довжиною m біт, що перекриваються, приблизно така ж, як і у випадку випадкової вхідної послідовності біт. Випадкова послідовність має властивість одноманітності, тобто кожен шаблон довжиною m біт, з'являється у послідовності з однаковою вірогідністю.[14]

Тест приблизної ентропії. Як і в тесті на періодичність в даному тесті акцент робиться на підрахунку частоти всіх можливих перекривань шаблонів довжиною m біт на всій довжині вхідної послідовності. Мета тесту – порівняти частоти перекриття двох послідовних блоків вхідної послідовності з довжинами m та $m+1$ з частотами перекривання аналогічних блоків у випадковій послідовності.[14]

Тест кумулятивних сум. Тест полягає в максимальному відхиленні при довільному проходженні, яке визначається кумулятивною сумою заданих (-1, +1) цифр в послідовності. Ціль даного тесту – визначити чи є кумулятивна сума часткових послідовностей, які трапляються у вхідній послідовності, занадто великою чи занадто маленькою в порівнянні з очікуваною поведінкою такої суми для випадкової послідовності. Таким чином, кумулятивна сума може розглядатися як довільне проходження. Для випадкової послідовності відхилення від довільного проходження повинно бути близьке до нуля. Для деяких типів послідовностей, які не є повною мірою випадковими, подібне відхилення від нуля при довільному проходженні буде доволі суттєвим.[14]

Тест на довільне відхилення. Даний тест підраховує кількість циклів, які мають строго k відвідувань при довільному проходженні кумулятивної суми. Довільне проходження кумулятивної суми починається з часткових сум після послідовності (0,1), яка перетворена у відповідну послідовність (-1,+1). Цикл довільного проходження складається із серії кроків одиничної довжини, які здійснюються в довільному порядку. Ціль даного тесту визначити чи відрізняється число потраплянь певного стану всередині циклу від аналогічного числа у випадку випадкової вхідної послідовності.

Інший тест на довільне відхилення. В цьому тесті підраховується загальне число входжень визначеного стану при довільному проходженні кумулятивної суми.[14]

РОЗДІЛ 2. ХАРАКТЕРИСТИКИ ТА ВИМОГИ ДО СИСТЕМИ

2.1 Призначення системи

Призначенням системи є дослідження ефективності використання mp3-файлів пісень різного жанру, шумів, псевдовипадкових генераторів мови Python в якості ключа для алгоритму XOR-шифрування, що має дати змогу якісно зашифрувати текст.

Система створюється з метою:

- перевірка дискретних випадкових величин на рівномірний розподіл
- тестування випадкової двійкової послідовності на випадковість тестами NIST
- шифрування та дешифрування вхідного тексту за допомогою XOR-алгоритму

2.2 Вимоги до функцій, які виконуються системою

Система повинна взяти частину бінарного представлення вхідного файлу та перетворити її до масиву однобайтовий чисел (від 0 до 255). Потім перевірити за критерієм хі-квадрат Пірсона нульову гіпотезу про рівномірний розподіл випадкової дискретної величини, тобто отриманого на попередньому кроці масиву. Система має відображати гістограму розподілу, а також статистику для критерію хі-квадрат Пірсона та порівнювати її з табличним значенням при рівні значимості $\alpha = 0{,}05$ по таблиці критичних значень розподілу хі-квадрат.

Наступним кроком є перетворення даного масиву до двійкової послідовності та тестування отриманої послідовності статистичними тестами NIST. Система має відображати результати проходження тестів.

Також система має пропонувати користувачеві ввести текст, який вона має зашифрувати за допомогою даного ключа, відобразити зашифрований текст та провести дешифрування.

2.3 Технічні вимоги

Операційна система Windows. Програма буде працювати для всіх операційних систем зі встановленою мовою програмування Python версії 3.0 і вище та всіма необхідними бібліотеками. Необхідні знання англійської мови для розуміння інтерфейсу та вміння користуватися командним рядком.

РОЗДІЛ 3.РЕАЛІЗАЦІЯ СИСТЕМИ

Систему було реалізовано мовою програмування Python за допомогою додаткових бібліотек SciPy, matplotlib, binascii та редактору коду Sublime Text 3. У системі в якості ключа для гамування було порівняно три варіанти. Перший – це звуковий файл з мелодією якогось певного жанру перетворений в двійкову послідовність. Другий – це звуковий файл з різними шумами перетворений в двійкову послідовність. Третій – це псевдовипадкова двійкова послідовність згенерована засобами мови програмування Python. Випадковість перших двох варіантів була протестована рівномірним розподілом байтів перетворених до десяткових чисел, а також бінарне зображення файлів протестоване статистичними тестами NIST. Також варто додати, що звукові файли були представлені у mp3 форматі.

3.1 Зчитування файлу та підготовка послідовності до тестування

Перш за все, якщо відкрити будь-який mp3-файл у текстовому редакторі коду, то побачимо наступну картину

```
4944 3303 0000 0005 2b7e 5449 5432 0000
001b 0000 01ff fe4a 0075 0073 0074 0020
0048 006f 006c 0064 0020 004f 006e 0054
5045 3100 0000 3b00 0001 fffe 5300 7400
6500 7600 6500 2000 4100 6f00 6b00 6900
2000 2600 2000 4c00 6f00 7500 6900 7300
2000 5400 6f00 6d00 6c00 6900 6e00 7300
6f00 6e00 5441 4c42 0000 002d 0000 01ff
fe4a 0075 0073 0074 0020 0048 006f 006c
0064 0020 004f 006e 0020 002d 0020 0053
0069 006e 0067 006c 0065 0054 5945 5200
0000 2b00 0001 fffe 3200 3000 3100 3600
2d00 3100 3200 2d00 3100 3200 5400 3000
3800 3a00 3000 3000 3a00 3000 3000 5a00
5443 4f4e 0000 000d 0000 01ff fe44 0061
006e 0063 0065 0054 5243 4b00 0000 0500
0001 fffe 3100 4150 4943 0001 50e3 0000
0069 6d61 6765 2f6a 7065 6700 0300 ffd8
ffe0 0010 4a46 4946 0001 0101 0048 0048
0000 ffdb 0043 000b 0808 0a08 070b 0a09
0a0d 0c0b 0d11 1c12 110f 0f11 2219 1a14
1c29 242b 2a28 2427 272d 3240 372d 303d
3027 2738 4c39 3d43 4548 4948 2b36 4f55
```

Рисунок 3.1 – вигляд байтів mp3-файлу

Файл відкрився у бінарному форматі, але двійкова послідовність зображена шістнадцятковими цифрами. Кожні дві цифри відповідають одному байту, адже значення коливаються від 0 (що зображений 00) до 255 (ff).

Ще цікавою особливістю є те, що mp3-файл на своєму початку містить різні метадані, тому початок файлу зазвичай виглядає наступним чином

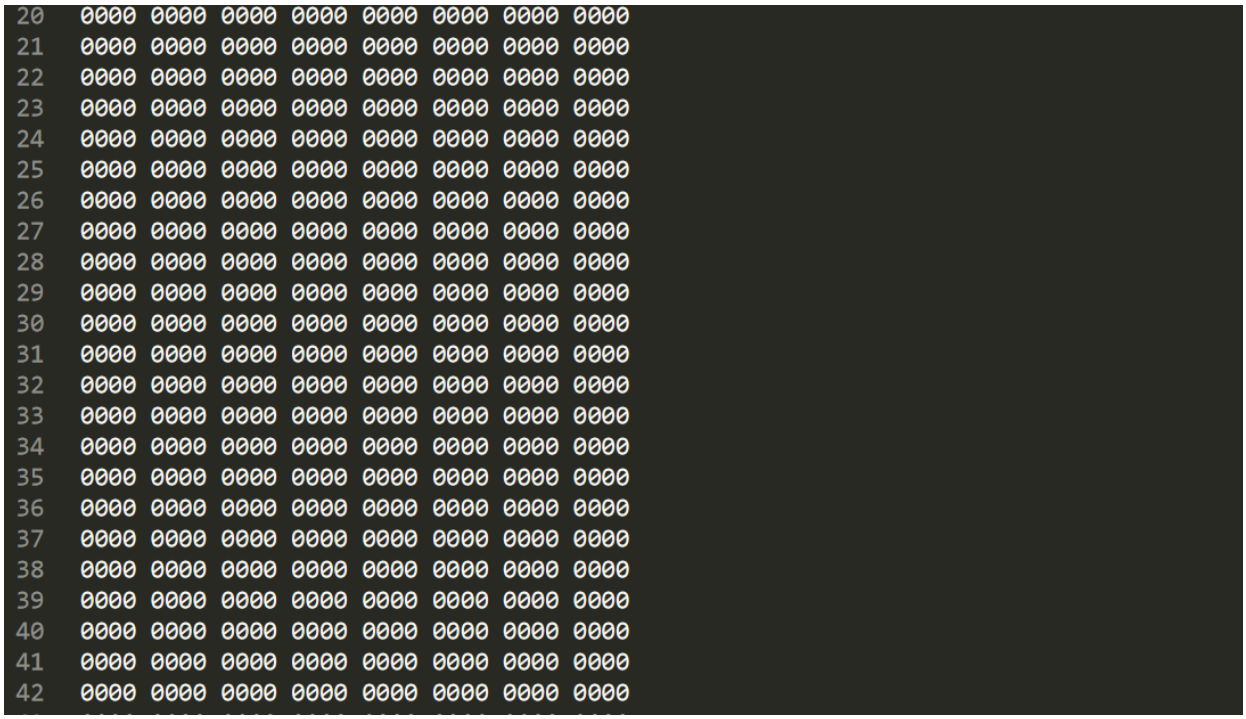

Рисунок 3.2 – Початок mp3-файлу

Така послідовність очевидно не є випадковою, тому у всіх наступних файлах під час їхнього зчитування буде відкинуто перші 160 тисяч байтів. Емпіричним шляхом було досліджено, що цього цілком достатньо.

Варто зауважити, що в даному випадку файли можуть бути будь якого формату, mp3-формат був вибраний саме для зручної класифікації за жанрами музики.

Наступним кроком буде зчитування файлу у бінарному форматі, потім перетворення отриманої двійкової послідовності у шістнадцяткову для подальшого зручного «нарізання» на байти. Це перетворення здійснюється за допомогою бібліотеки binascii, а саме функцією hexlify(content). Далі

відкинемо перші 320 тисяч шістнадцяткових цифр, що відповідають 160000 байтів. Отриману послідовність перетворимо у масив десяткових чисел від 0 до 255. Який далі і будемо перевіряти на рівномірний розподіл. Для цього кожні дві цифри шістнадцяткової послідовності перетворюватимемо в десяткове число(що і буде по суті відповідати одному байту). Утворимо масив довжиною 131072. Забігаючи наперед, можна сказати, що така довжина обумовлена подальшим тестуванням NIST, яке вимагає для успішного проходження тестів довжину двійкової послідовності мінімум 2^{20} бітів. Тому $2^{20}/2^3 = 2^{17}$. (255 у десятковому представленні буде двійковим числом довжиною послідовності 8).

3.2 Перевірка на рівномірний розподіл

Тепер отриманий масив перевіримо на рівномірний розподіл за допомогою критерію хі-квадрат. А також для наочності намалюємо гістограму засобами бібліотеки matplotlib. Для перевірки критерію використаємо формулу $\chi_k^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i}$, де O_i – спостережувані частоти, O_i – теоретичні частоти, k – кількість ступенів свободи (в нашому випадку рівне 256). Отримане значення порівняємо з таблицею критичних значень розподілу хі-квадрат при рівні значимості 0,05 (в нашому випадку табличне значення буде 294,3207).

Для прикладу порівняємо гістограми масиву випадкових чисел від 0 до 255, які очевидно повинні бути рівномірно розподілені і не зовсім випадкового масиву. Випадковий масив матиме вигляд:

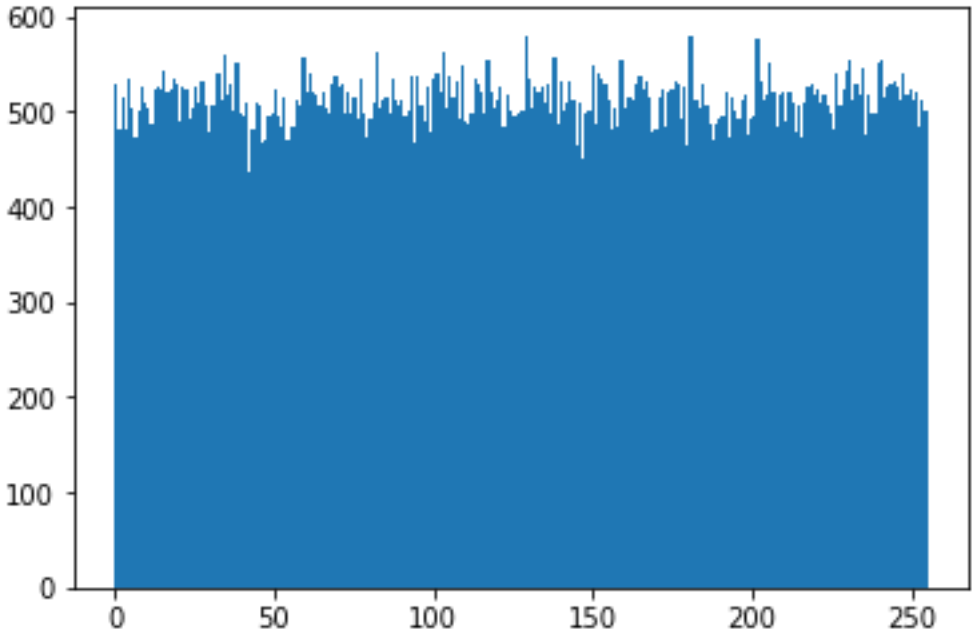

Рисунок 3.3 – Гістограма рівномірно розподіленої дискретної величини

В цьому випадку хі-квадрат критерій дорівнює 255.8672, що цілком менше за табличне значення 294,3207. З графіку видно, що частоти намагаються «вишикуватися» в одну лінію:

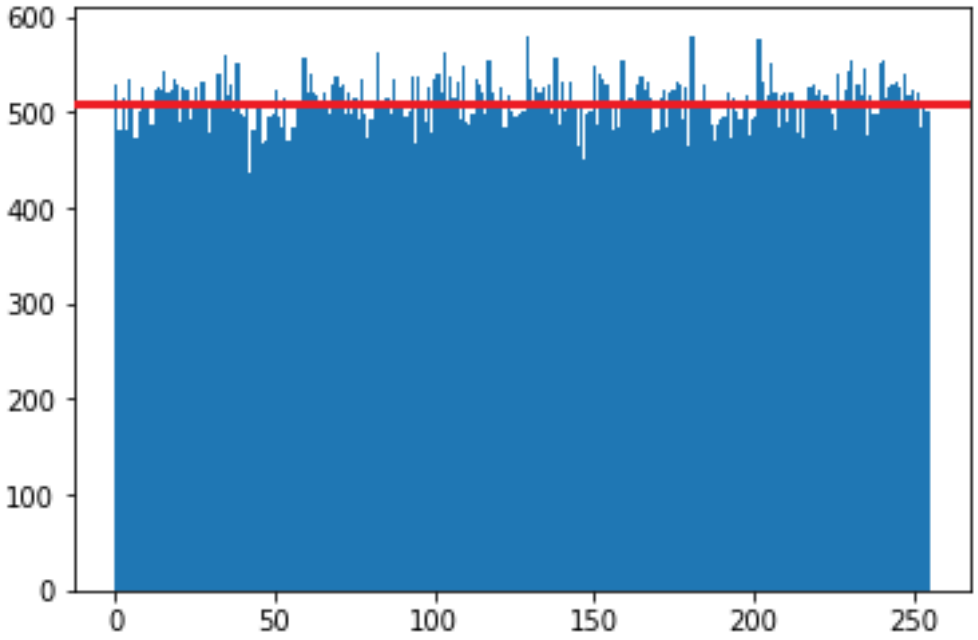

Рисунок 3.4 – Гістограма рівномірно розподіленої дискретної величини

І навпаки масив невипадкових чисел від 0 до 255 матиме вигляд:

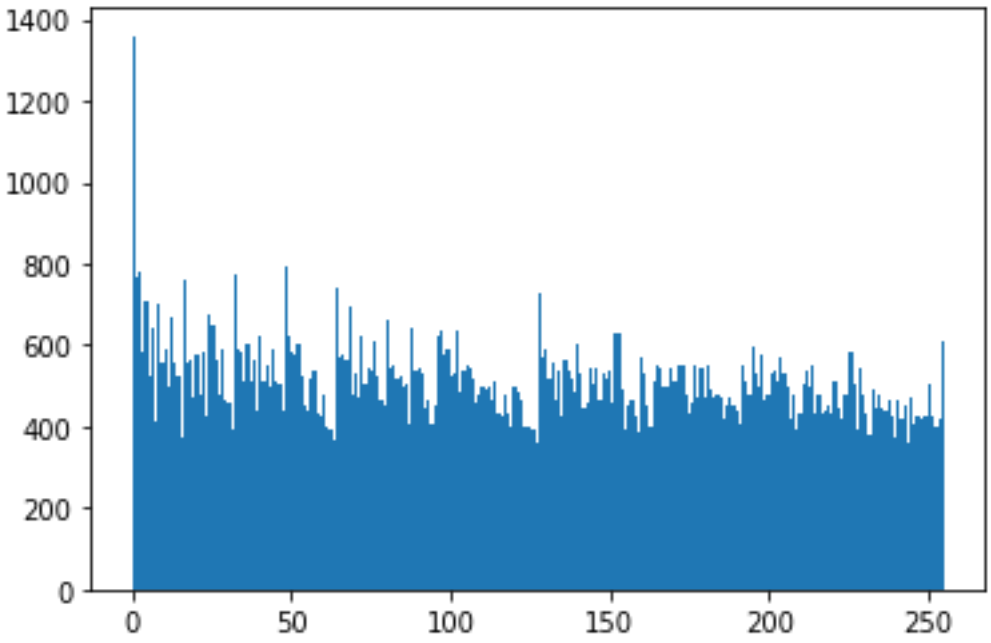

Рисунок 3.5 – Гістограма нерівномірно розподіленої дискретної величини

Хі-квадрат критерій дорівнює 4737.4414, що свідчить про непідтвердження гіпотези рівномірності даного розподілу.

Наступним кроком буде перетворення масиву десяткових чисел від 0 до 255 довжиною 131072 у двійкову послідовність довжиною 1048576. Для цього просто виведемо його у файл у двійковому форматі. Подальше тестування NIST буде проходити з цим файлом.

3.3 Тести NIST

Першим тестом є частотний. Для його реалізаціє просто сумуємо двійкову послідовність, але замість нуля додаємо -1. Таким чином отримуємо суму послідовності S_n. Далі рахуємо статистику:

$$S_{obs} = \frac{|S|}{\sqrt{n}}$$

Наступним кроком буде обрахування P-значення через додаткову функцію помилок:

$$P_{value} = erfc(\frac{S_{obs}}{\sqrt{2}})$$

Додаткова функція помилок визначається:

$$erfc(x) = \frac{2}{\sqrt{\pi}} \int_x^{\infty} e^{-t^2} dt$$

Наступним тестом йду частотний блочний. Він рахується як і попередній, але спочатку розіб'ємо послідовність на блоки. Потім порахуємо відношення одиниць до загальної довжини кожного блоку π_i. Далі вираховуємо статистику за методом хі-квадрат з n (кількість блоків) ступенями свободи[15]:

$$\chi^2_{obs} = 4 * M * \sum_{i=1}^{N} (\pi_i - 1/2)^2$$

Вираховуємо P-значення через певну функцію Q:

$$P_{value} = Q(\frac{N}{2}, \frac{\chi^2_{obs}}{2})$$

Q – це неповна верхня гама-функція, яка визначається наступним чином:

$$Q(a, x) = \frac{1}{\Gamma(a)} \int_x^{\infty} e^{-t} t^{a-1} dt$$

Г – стандартна гама-функція

$$\Gamma(z) = \int_0^{\infty} t^{z-1} e^{-t} dt$$

3.4 Кодування та декодування

Сам процес кодування доволі простий. Він полягає у посимвольному додаванні по модулю 2, тобто операція XOR, ключа та вхідного тексту. Що стосується декодування, то цей процес є зворотнім. Проводимо операцію XOR над закодованим текстом та ключем.

```
text = input("Enter a text:\n")
key=string
keyLength = len(key)
output=""

for i in range(0, len(text)):
    j = i % keyLength

    xor = ord(text[i]) ^ ord(key[j])
    output = output + chr(xor)

print(output)
```

Рисунок 3.6 – Частина коду програми

РОЗДІЛ 4. ТЕСТУВАННЯ СИСТЕМИ

Проведемо дослідження mp3-файлів з різними жанрами музики. Наприклад, візьмемо по три композиції наступних жанрів: класична музика, популярна музика, рок-музика.

Гістограми та хі-квадрат критерій:

Рок-музика:

Композиція №1

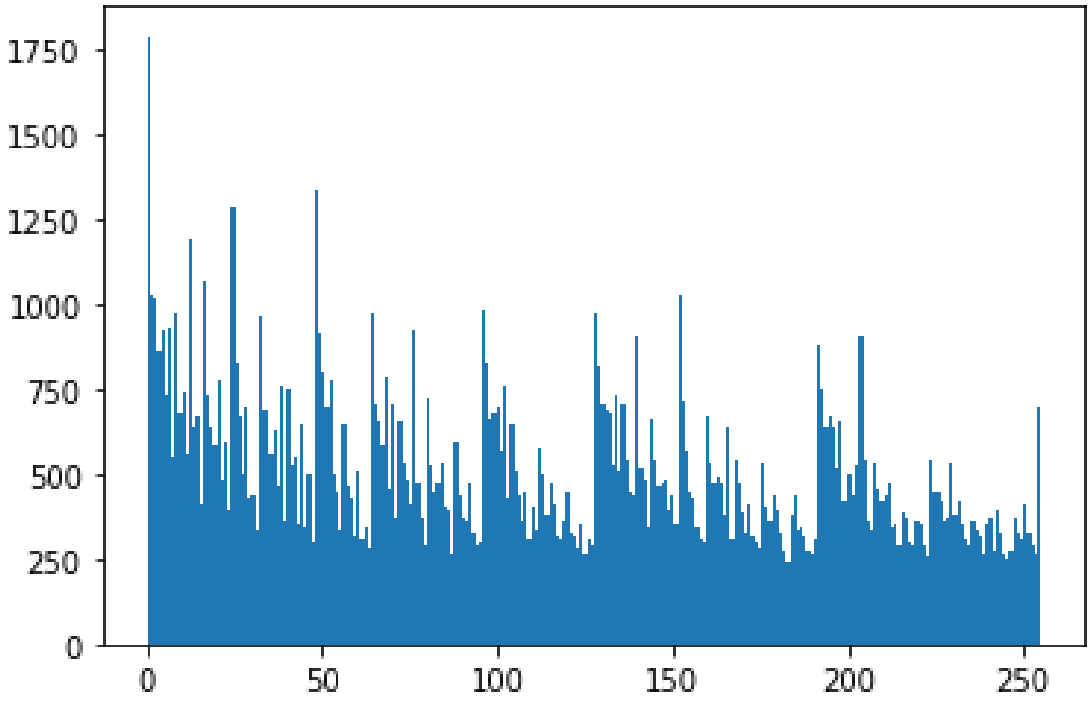

Рисунок 4.1 – Гістограма рок-композиції №1

χ^2 = 24373.6992

```
SUMMARY
-------
monobit_test                            0.0                 FAIL
frequency_within_block_test             0.0                 FAIL
runs_test                               0.0                 FAIL
longest_run_ones_in_a_block_test        4.24637394806657e-82 FAIL
binary_matrix_rank_test                 0.643389439875055   PASS
dft_test                                3.3481450504906335e-31 FAIL
non_overlapping_template_matching_test  0.9799879419297476 PASS
overlapping_template_matching_test      4.535279100650627e-134 FAIL
maurers_universal_test                  0.0                 FAIL
linear_complexity_test                  0.9708941035532213 PASS
serial_test                             0.0                 FAIL
approximate_entropy_test                0.0                 FAIL
cumulative_sums_test                    0.0                 FAIL
random_excursion_test                   0.0016575451314226326 FAIL
random_excursion_variant_test           0.3545394797735013 PASS
```

Рисунок 4.2 – Результати статистичних тестів NIST рок-композиції №1

Композиція №2

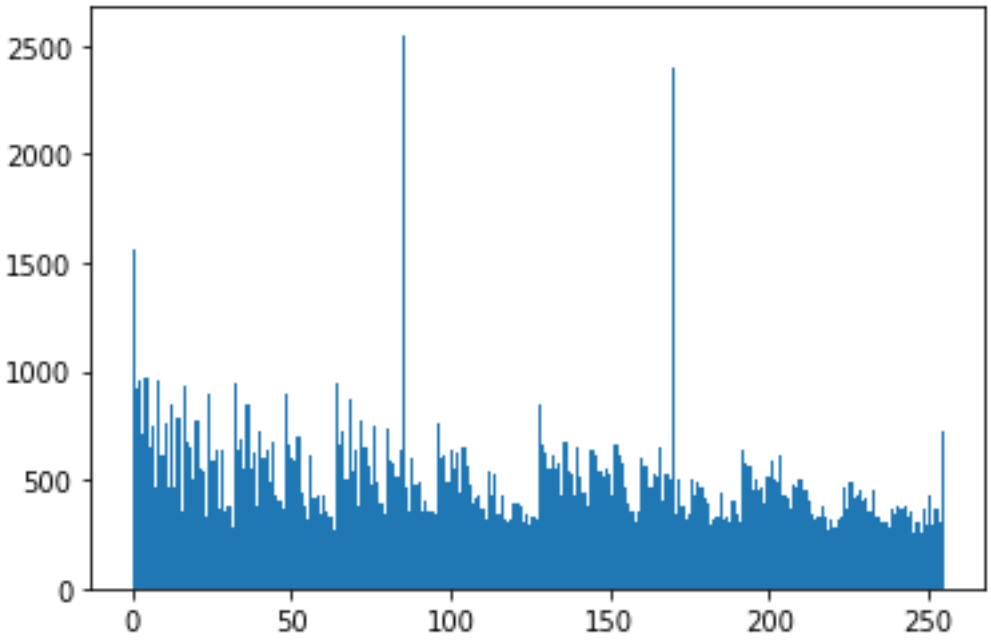

Рисунок 4.3 – Гістограма рок-композиції №2

χ^2 = 29968.75

```
SUMMARY
-------
monobit_test                            0.0                 FAIL
frequency_within_block_test             0.0                 FAIL
runs_test                               0.0                 FAIL
longest_run_ones_in_a_block_test        1.0775685912041501e-41 FAIL
binary_matrix_rank_test                 1.84502794581348 57e-23 FAIL
dft_test                                5.554740684073529e-67 FAIL
non_overlapping_template_matching_test  0.9999177317132636 PASS
overlapping_template_matching_test      4.7776095432896943e-94 FAIL
maurers_universal_test                  0.0                 FAIL
linear_complexity_test                  0.000676833304497325 FAIL
serial_test                             0.0                 FAIL
approximate_entropy_test                0.0                 FAIL
cumulative_sums_test                    0.0                 FAIL
random_excursion_test                   0.0046921060959793665 FAIL
random_excursion_variant_test           0.201242620957724   PASS
```

Рисунок 4.4 – Результати статистичних тестів NIST рок-композиції №2

Композиція №3

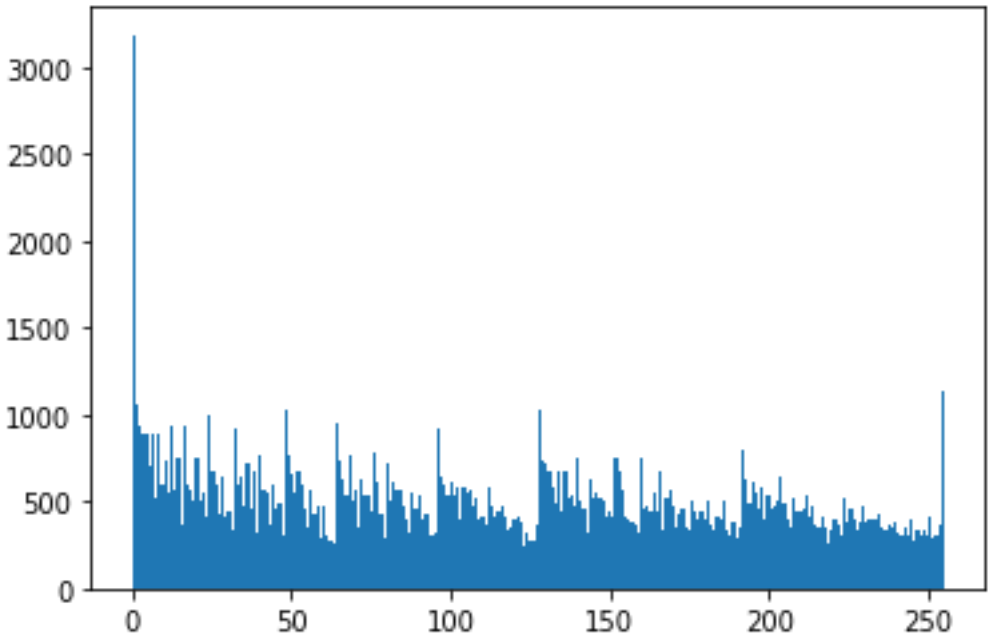

Рисунок 4.5 – Гістограма рок-композиції №3

χ^2 = 28283.90625

```
SUMMARY
-------
monobit_test                            0.0                 FAIL
frequency_within_block_test             0.0                 FAIL
runs_test                               0.0                 FAIL
longest_run_ones_in_a_block_test        3.8083899547103 88e-40 FAIL
binary_matrix_rank_test                 1.169558520470945e-15 FAIL
dft_test                                6.571325326116638e-76 FAIL
non_overlapping_template_matching_test  0.993234567140453   PASS
overlapping_template_matching_test      1.3562428032116618e-80 FAIL
maurers_universal_test                  0.0                 FAIL
linear_complexity_test                  0.3058455929580982  PASS
serial_test                             0.0                 FAIL
approximate_entropy_test                0.0                 FAIL
cumulative_sums_test                    0.0                 FAIL
random_excursion_test                   0.012228140467104357 PASS
random_excursion_variant_test           0.4142161782425251  PASS
```

Рисунок 4.6 – Результати статистичних тестів NIST рок-композиції №3

Популярна музика:

Композиція №1

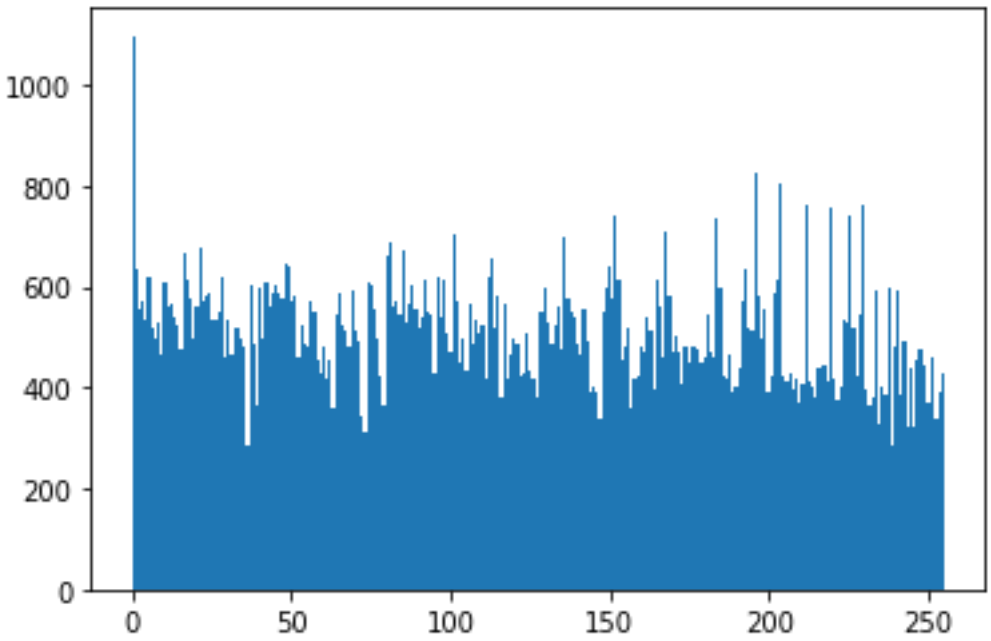

Рисунок 4.7 – Гістограма поп-композиції №1

χ^2 = 5341.0078125

```
SUMMARY
-------
monobit_test                          4.870634697295882e-249 FAIL
frequency_within_block_test           0.0                    FAIL
runs_test                             0.0                    FAIL
longest_run_ones_in_a_block_test      1.0294057304684916e-25 FAIL
binary_matrix_rank_test               0.28221110649593023    PASS
dft_test                              0.47563567114249855    PASS
non_overlapping_template_matching_test 0.9999997376399054    PASS
overlapping_template_matching_test    2.2176524546047636e-26 FAIL
maurers_universal_test                1.4508202488351982e-46 FAIL
linear_complexity_test                0.9860923863862948     PASS
serial_test                           0.0                    FAIL
approximate_entropy_test              0.0                    FAIL
cumulative_sums_test                  0.0                    FAIL
random_excursion_test                 0.20262495612657722    PASS
random_excursion_variant_test         0.29738237660847733    PASS
```

Рисунок 4.8 – Результати статистичних тестів NIST поп-композиції №1

Композиція №2

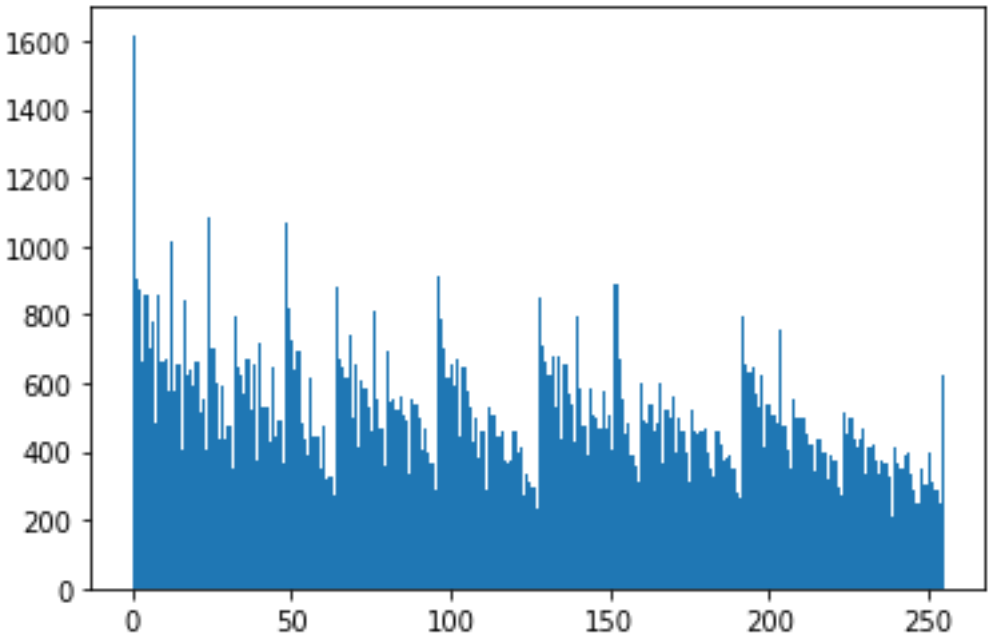

Рисунок 4.9 – Гістограма поп-композиції №2

χ^2 = 14769.984375

```
SUMMARY
-------
monobit_test                            0.0                 FAIL
frequency_within_block_test             0.0                 FAIL
runs_test                               0.0                 FAIL
longest_run_ones_in_a_block_test        1.2763928751649935e-24 FAIL
binary_matrix_rank_test                 0.5884685698761754 PASS
dft_test                                6.88056309733434e-13 FAIL
non_overlapping_template_matching_test  0.991522068582404  PASS
overlapping_template_matching_test      1.0294608287968798e-51 FAIL
maurers_universal_test                  0.0                 FAIL
linear_complexity_test                  0.18782147456026183 PASS
serial_test                             0.0                 FAIL
approximate_entropy_test                0.0                 FAIL
cumulative_sums_test                    0.0                 FAIL
random_excursion_test                   0.19696688943675392 PASS
random_excursion_variant_test           0.326109452020489   PASS
```

Рисунок 4.10 – Результати тестів NIST поп-композиції №2

Композиція №3

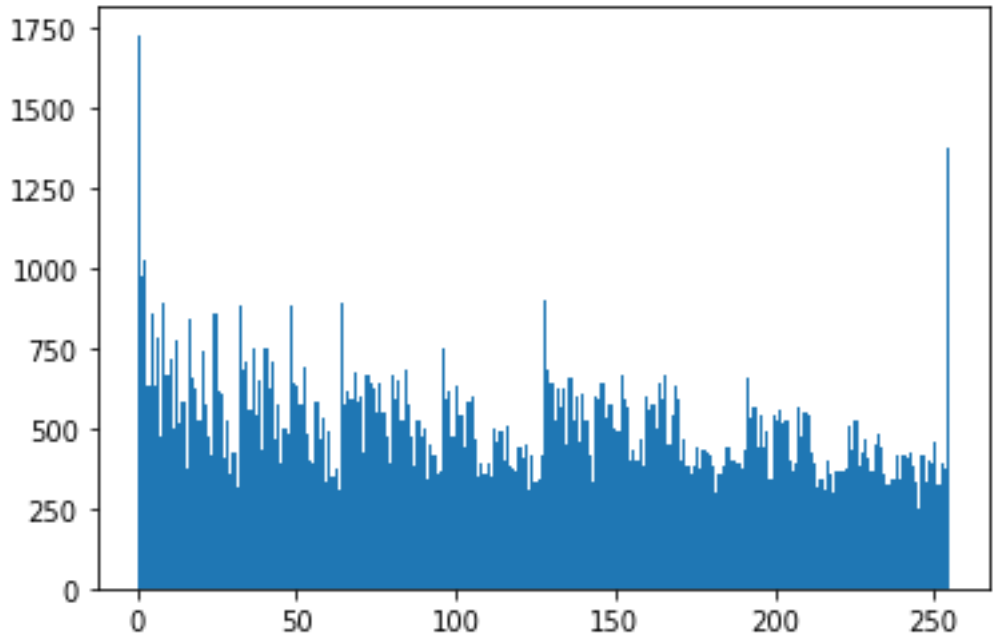

Рисунок 4.11 – Гістограма поп-композиції №3

χ^2 = 13886.984375

```
SUMMARY
-------
monobit_test                            0.0                 FAIL
frequency_within_block_test             0.0                 FAIL
runs_test                               0.0                 FAIL
longest_run_ones_in_a_block_test        6.064928576081367e-149 FAIL
binary_matrix_rank_test                 0.31223437829282613 PASS
dft_test                                2.444624238958949e-10 FAIL
non_overlapping_template_matching_test  0.9999615214651288 PASS
overlapping_template_matching_test      2.1303147737284667e-233 FAIL
maurers_universal_test                  0.0                 FAIL
linear_complexity_test                  0.49411308594730347 PASS
serial_test                             0.0                 FAIL
approximate_entropy_test                0.0                 FAIL
cumulative_sums_test                    0.0                 FAIL
random_excursion_test                   0.039392184415519275 PASS
random_excursion_variant_test           0.07688120490870527 PASS
```

Рисунок 4.12 – Результати тестів NIST поп-композиції №3

Класична музика:

Композиція №1

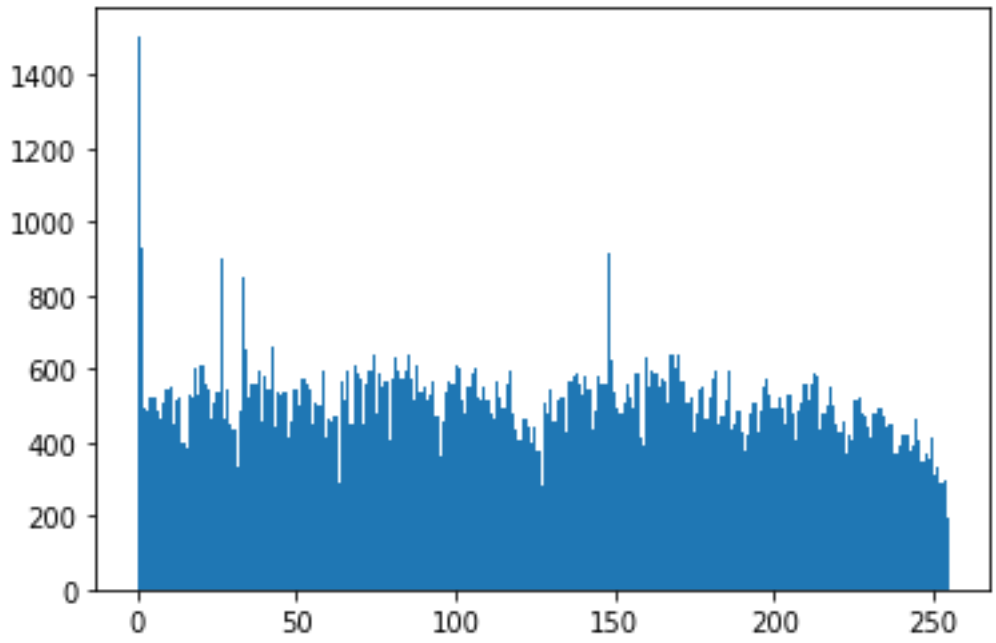

Рисунок 4.13 – Гістограма класичної композиції №1

χ^2 = 6027.75

```
SUMMARY
-------
monobit_test                            0.0                 FAIL
frequency_within_block_test             0.0                 FAIL
runs_test                               0.0                 FAIL
longest_run_ones_in_a_block_test        1.880510228513234e-18 FAIL
binary_matrix_rank_test                 0.10477586098169248 PASS
dft_test                                0.0002612860160203519 FAIL
non_overlapping_template_matching_test  0.9999952852412725 PASS
overlapping_template_matching_test      8.737914075992677e-50 FAIL
maurers_universal_test                  5.944256178451929e-70 FAIL
linear_complexity_test                  0.7284455319631657 PASS
serial_test                             0.0                 FAIL
approximate_entropy_test                0.0                 FAIL
cumulative_sums_test                    0.0                 FAIL
random_excursion_test                   1.336913265879106e-07 FAIL
random_excursion_variant_test           0.09725442843900325 PASS
```

Рисунок 4.14 – Результати тестів NIST класичної композиції №1

Композиція №2

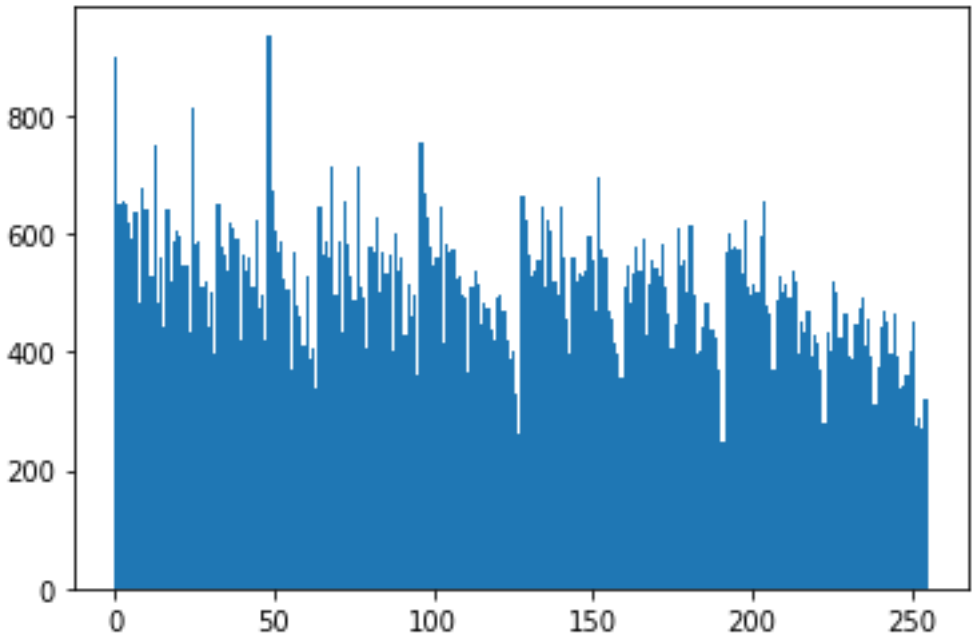

Рисунок 4.15 – Гістограма класичної композиції №2

χ^2 = 5203.91796875

```
SUMMARY
-------
monobit_test                            0.0                 FAIL
frequency_within_block_test             0.0                 FAIL
runs_test                               0.0                 FAIL
longest_run_ones_in_a_block_test        2.2945582703789632e-20 FAIL
binary_matrix_rank_test                 0.6831935340454058 PASS
dft_test                                0.0001534864651310746 FAIL
non_overlapping_template_matching_test  0.9999951801062147 PASS
overlapping_template_matching_test      3.390431079546186e-32 FAIL
maurers_universal_test                  0.0                 FAIL
linear_complexity_test                  0.6579257263907651 PASS
serial_test                             0.0                 FAIL
approximate_entropy_test                0.0                 FAIL
cumulative_sums_test                    0.0                 FAIL
random_excursion_test                   0.147330120397369   PASS
random_excursion_variant_test           0.017244708366320843 PASS
```

Рисунок 4.16 – Результати тестів NIST класичної композиції №2

Композиція №3

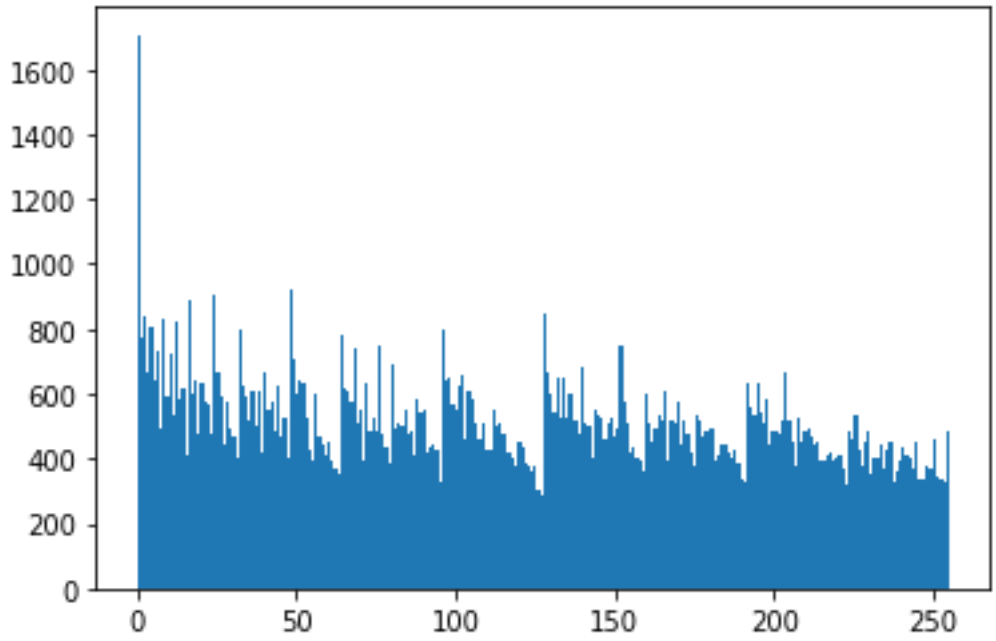

Рисунок 4.17 – Гістограма класичної композиції №3

χ^2 = 9930.53125

```
SUMMARY
-------
monobit_test                          0.0                 FAIL
frequency_within_block_test           0.0                 FAIL
runs_test                             0.0                 FAIL
longest_run_ones_in_a_block_test      4.515121208825155e-24 FAIL
binary_matrix_rank_test               0.7491702942388597  PASS
dft_test                              1.3678163547756096e-15 FAIL
non_overlapping_template_matching_test 0.9999964313044761 PASS
overlapping_template_matching_test    4.57412651786024e-46 FAIL
maurers_universal_test                8.877717141730203e-299 FAIL
linear_complexity_test                0.704049981305906   PASS
serial_test                           0.0                 FAIL
approximate_entropy_test              0.0                 FAIL
cumulative_sums_test                  0.0                 FAIL
random_excursion_test                 1.524678880022459e-06 FAIL
random_excursion_variant_test         0.010515245935858918 PASS
```

Рисунок 4.18 – Результати тестів NIST класичної композиції №3

Тепер перевіримо різні шуми:

Шум №1

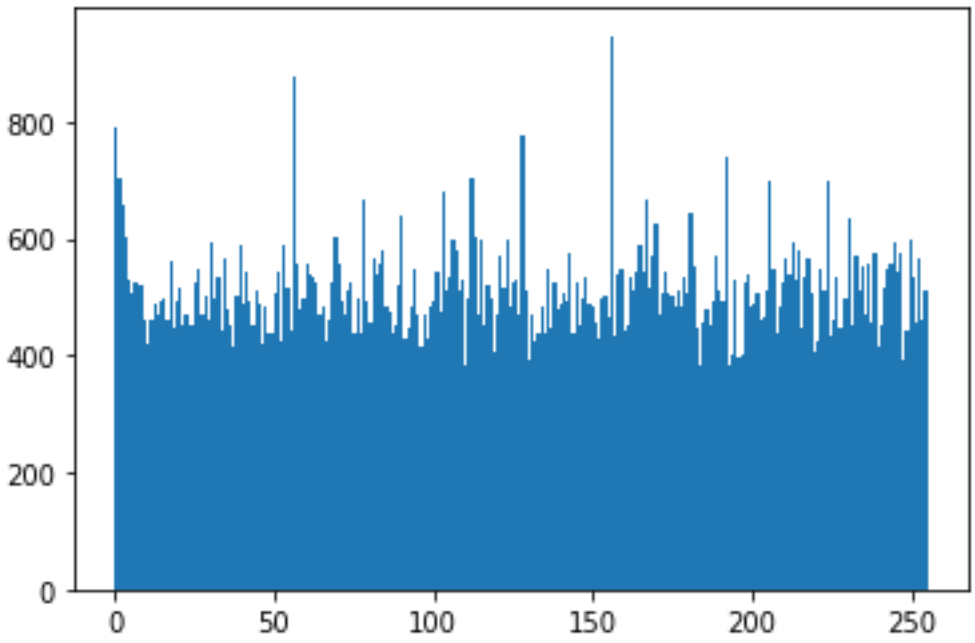

Рисунок 4.13 – Гістограма шуму №1

χ^2 = 2245.49675

```
SUMMARY
-------
monobit_test                          0.772533549822382  PASS
frequency_within_block_test           2.1716601125489003e-23 FAIL
runs_test                             6.937056135587e-05 FAIL
longest_run_ones_in_a_block_test      2.9351262549855206e-09 FAIL
binary_matrix_rank_test               0.672420631026681  PASS
dft_test                              0.04805085526745377 PASS
non_overlapping_template_matching_test 0.9999999973237754 PASS
overlapping_template_matching_test    3.5370642205885323e-56 FAIL
maurers_universal_test                1.421641780931107e-28 FAIL
linear_complexity_test                0.04872466523833006 PASS
serial_test                           5.806163616405766e-65 FAIL
approximate_entropy_test              9.40178187483772e-65 FAIL
cumulative_sums_test                  0.17708871022106654 PASS
random_excursion_test                 0.07327945171195625 PASS
random_excursion_variant_test         0.1905146977096144 PASS
```

Рисунок 4.14 – Результати тестів NIST шуму №1

Шум №2

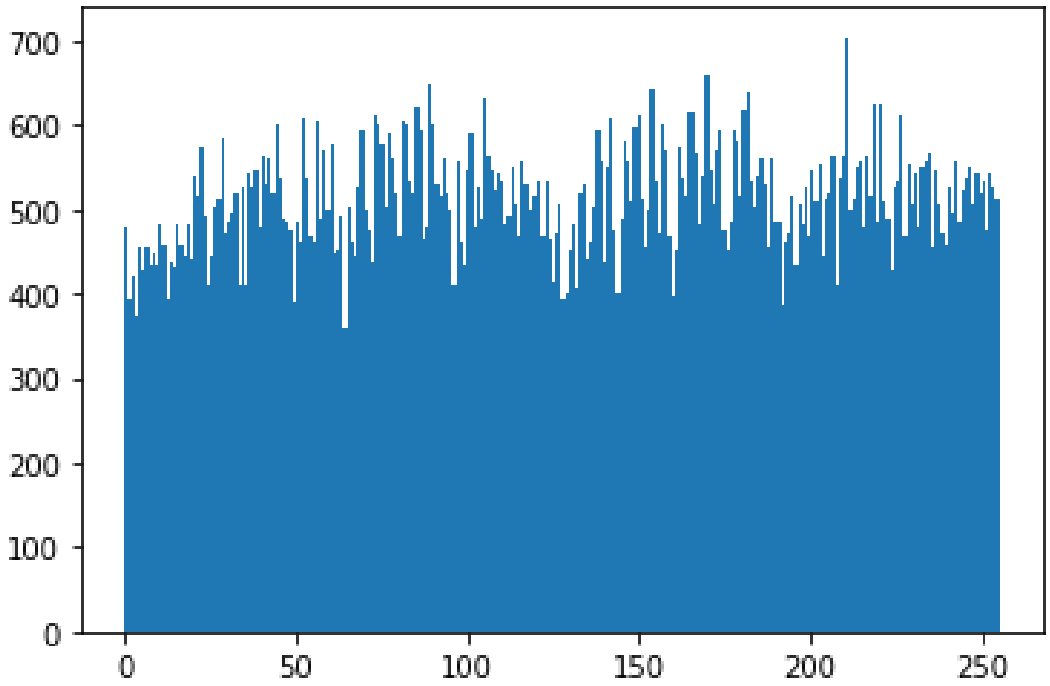

Рисунок 4.15 – Гістограма шуму №2

χ^2 = 1685.484215

```
SUMMARY
-------
monobit_test                            2.4206614716423483e-46 FAIL
frequency_within_block_test             6.374335839339333e-47 FAIL
runs_test                               0.0                    FAIL
longest_run_ones_in_a_block_test        6.176621781377435e-22 FAIL
binary_matrix_rank_test                 0.37683934075908976 PASS
dft_test                                0.008072201862028576 FAIL
non_overlapping_template_matching_test  0.9999999971857099 PASS
overlapping_template_matching_test      2.400500123664917e-118 FAIL
maurers_universal_test                  3.582018363099952e-62 FAIL
linear_complexity_test                  0.6899111389364707 PASS
serial_test                             6.813588061464858e-141 FAIL
approximate_entropy_test                4.534088089669699e-143 FAIL
cumulative_sums_test                    0.0                    FAIL
random_excursion_test                   0.43894262502072323 PASS
random_excursion_variant_test           0.1038987701628236 PASS
```

Рисунок 4.16 – Результати тестів NIST шуму №2

Телевізійний шум

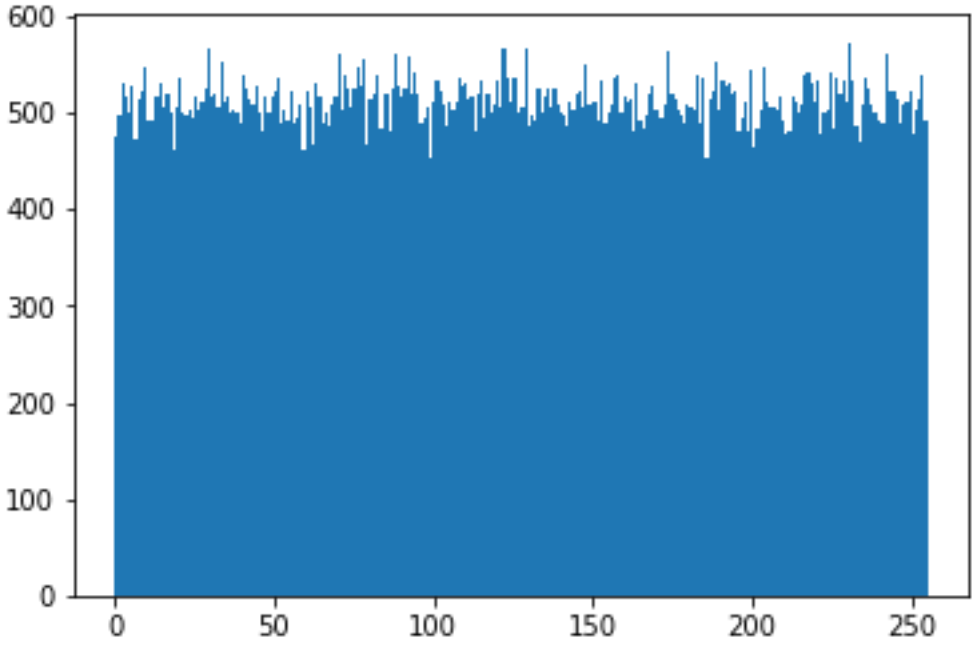

Рисунок 4.17 – Гістограма шуму №3

χ^2 = 232.69140625

```
SUMMARY
-------
monobit_test                            0.4795470580000091 PASS
frequency_within_block_test             0.5180655749103592 PASS
runs_test                               0.9225928852596187 PASS
longest_run_ones_in_a_block_test        0.06899884793832442 PASS
binary_matrix_rank_test                 0.09064057937587627 PASS
dft_test                                0.3313330763855039 PASS
non_overlapping_template_matching_test  1.0000000238255444 PASS
overlapping_template_matching_test      0.34658545061406437 PASS
maurers_universal_test                  0.8657583127843645 PASS
linear_complexity_test                  0.5017281999386923 PASS
serial_test                             0.40042417731573376 PASS
approximate_entropy_test                0.6515827732949205 PASS
cumulative_sums_test                    0.4800694853080729 PASS
random_excursion_test                   0.06467005478328913 PASS
random_excursion_variant_test           0.34911181014305115 PASS
```

Рисунок 4.18 – Результати тестів NIST шуму №3

Тепер перевіримо стандартний генератор псевдовипадкових чисел мови Python – randint:

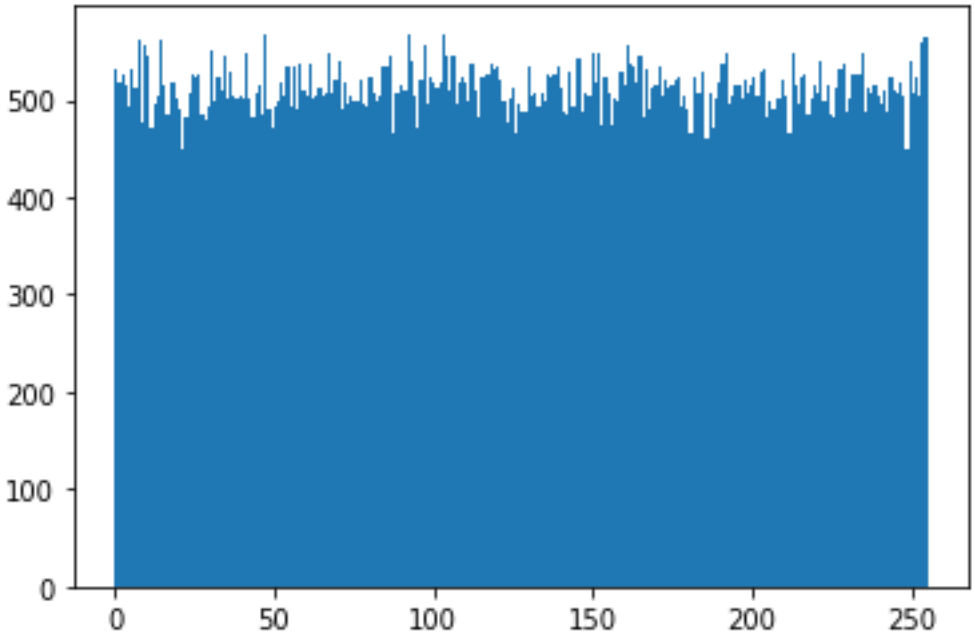

Рисунок 4.19 – Гістограма випадкового генератора Python

χ^2 = 240.21484375

```
SUMMARY
-------
monobit_test                            0.5513735701990394 PASS
frequency_within_block_test             0.2553180229408204 PASS
runs_test                               0.7692810878441497 PASS
longest_run_ones_in_a_block_test        0.7191176223465182 PASS
binary_matrix_rank_test                 0.5541799892301433 PASS
dft_test                                0.38764633036326174 PASS
non_overlapping_template_matching_test  0.9999999744085265 PASS
overlapping_template_matching_test      0.7754179638893669 PASS
maurers_universal_test                  0.5156270751571078 PASS
linear_complexity_test                  0.3219289196536404 PASS
serial_test                             0.8417964474941203 PASS
approximate_entropy_test                0.9463178791399165 PASS
cumulative_sums_test                    0.24377983388489421 PASS
random_excursion_test                   0.2534558464850147 PASS
random_excursion_variant_test           0.15078556502797127 PASS
```

Рисунок 4.20 – Результати тестів NIST випадкового генератора Python

Як бачимо музичні файли поводять себе приблизно однаково, вони не дуже «випадкові». Краще проявляють себе різні шуми в цій ситуації. Білий телевізійний шум навіть проходить тест на рівномірний розподіл та статистичні тести NIST. І, очевидно, найкраще в даному випадку є псевдовипадковий генератор randint.

Проведемо шифрування та дешифрування за допомогою останнього в якості ключа:

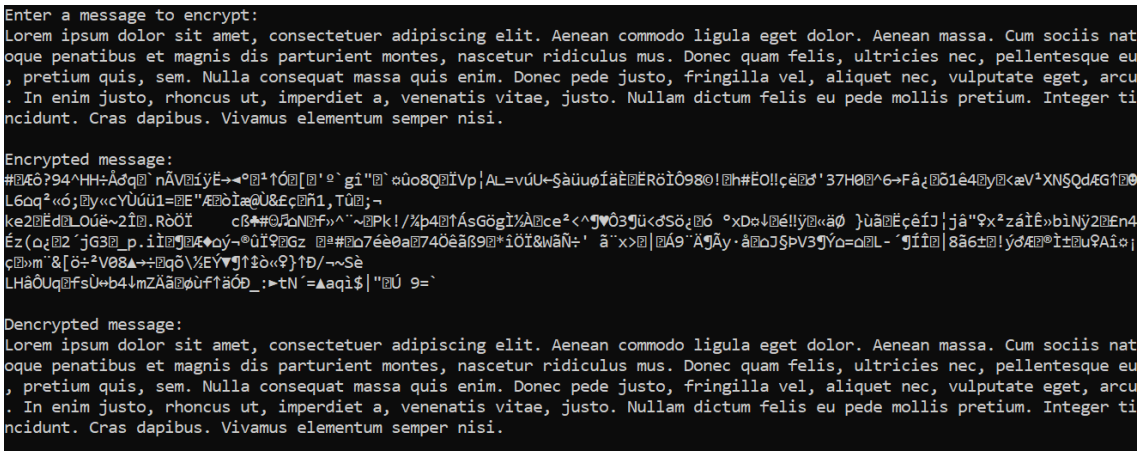

Рисунок 4.21 – Результати шифрування та дешифрування

ВИСНОВКИ

В ході виконання роботи було проведено дослідження предметної області, визначено головні вимоги до системи та бізнес-процеси. Метою дипломної роботи була розробка програмного забезпечення, яке могло б шифрувати та дешифрувати текст, а також перевіряти випадковість ключа шифрування.

У процесі виконання були виконані наступні завдання:

- розглянуто та опановано мову програмування Python
- здійснено перевірку на рівномірний розподіл 15 випадкових дискретних послідовностей
- здійснено перевірку статистичними тестами NIST 15 двійкових послідовностей
- реалізовано алгоритм XOR-шифрування
- здійснено шифрування та дешифрування тестів

У системі реалізовано шифрування та дешифрування за допомогою XOR-алгоритму. Отже, базовий функціонал повністю реалізований.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Криптографія [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Криптографія.
2. Глинчук Л. Я. Криптологія: навч.-метод. посіб. / Людмила Ярославівна Глинчук. – Луцьк: Вежа-Друк, 2014. – 164 с.
3. Принцип Керкгоффса [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Принцип_Керкгоффса.
4. Принцип Керкгоффза [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Принцип_Керкгоффза.
5. Брюс Шнайер. Прикладная криптография. Протоколы, алгоритмы и исходные тексты на языке С (неопр.). — 2-е издание. — Триумф, 1995. — ISBN 5-89392-055-4.
6. ШИФРЫ ГАММИРОВАНИЯ [Електронний ресурс] – Режим доступу до ресурсу: https://sites.google.com/site/anisimovkhv/learning/kripto/lecture/tema6
7. Гаммирование [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Гаммирование.
8. Фомичев В. М. ДИСКРЕТНАЯ МАТЕМАТИКА И КРИПТОЛОГИЯ. Курс лекций / Под общ. Ред. д-ра. физ.-мат. н. Н. Д. Подуфалова. – М.: ДИАЛОГ-МИФИ, 2003 – 400 с.
9. Бабаш А. В., Гольев Ю. И., Ларин Д. А., Шанкин Г. П. Криптографические идеи XIX века // Защита информации. Конфидент — СПб.: 2004. — вып. 3.
10. Шеннон, Клод [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Шеннон,_Клод.
11. Абсолютно стойкий шифр [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Абсолютно_стойкий_шифр.
12. Классические методы статистики: критерий хи-квадрат [Електронний ресурс] – Режим доступу до ресурсу: https://r-analytics.blogspot.com/2012/08/blog-post.html.

13. Критерий согласия Пирсона [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Критерий_согласия_Пирсона.
14. Статистические тесты NIST [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Статистические_тесты_NIST.
15. Статистическая проверка случайности двоичных последовательностей методами NIST [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/company/securitycode/blog/237695/.