Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Новгородский государственный университет имени Ярослава Мудрого»

Политехнический институт

Кафедра прикладной математики и информатики

**Методы защиты информации**

Лабораторная работа 3

**Традиционные методы шифрования**

по дисциплине «Методы защиты информации»

по направлению подготовки 09.03.01  
 «Информатика и вычислительная техника»

направленность (профиль) «Программное обеспечение   
вычислительной техники и автоматизированных систем»

Проверил

Доцент КПМИ, доц., к. ф.-м.н.

Жгун Т.В.

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

Выполнил

студент группы 3091

Ильин Данила Александрович

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2024 г.

**Оглавление**

[План работы 3](#_Toc185336322)

[Теоретическое положение. 4](#_Toc185336323)

[Алгоритм работы программы 6](#_Toc185336324)

[Приложения А 7](#_Toc185336325)

[Приложение Б 10](#_Toc185336326)

# План работы

В рамках работы планируется реализовать и исследовать четыре метода шифрования. Для каждого метода будет разработан соответствующий алгоритм шифрования и декодирования, а также выполнен частотный анализ и расчет энтропии зашифрованных текстов.

1. Реализовать алгоритм шифрования с использованием метода маршрутной перестановки (треугольник).
2. Реализовать алгоритм шифрования с использованием ключа (константа, поговорка, псевдослучайная последовательность).
3. Реализовать частотный анализ зашифрованных текстов с использованием библиотеки matplotlib.
4. Реализовать расчет энтропии зашифрованных текстов для каждого метода шифрования.
5. Разработать графический интерфейс с использованием tkinter для загрузки текста, выбора метода шифрования и отображения результатов.
6. Реализовать сохранение зашифрованных текстов и результатов анализа в файл.
7. Подготовить отчёт, включающий теоретические основы выбранных методов шифрования, описание алгоритмов и анализ частотных характеристик.

# Теоретическое положение

В работе рассматриваются четыре метода шифрования, каждый из которых имеет свои особенности и применения в криптографии. Особое внимание уделяется традиционным методам шифрования, парному шифру, шифру Гронфельда и использованию псевдослучайных последовательностей (ПСП) для повышения безопасности.

**Шифрование с помощью смещения на константу**  
Этот метод, известный как шифр Цезаря, предполагает сдвиг каждого символа исходного текста на фиксированное количество позиций в алфавите. В данной работе используется константа 4. Каждый символ смещается на заданное число позиций, а при переполнении алфавита происходит оборачивание. Шифр Цезаря прост в реализации, но уязвим к криптоанализу, особенно при использовании коротких ключей, что позволяет восстановить исходный текст с помощью частотного анализа.

**Парный шифр и шифр Гронфельда**  
В этих методах каждый символ открытого текста сдвигается на количество шагов, заданное соответствующим символом ключа. Для шифра Гронфельда используется таблица, где символ шифротекста находится на пересечении строки, соответствующей числу в ключе, и столбца, соответствующего символу открытого текста. Этот метод представляет собой более сложную форму шифрования и обеспечивает более высокую безопасность по сравнению с предыдущими.

**Псевдослучайные последовательности (ПСП)**  
Для шифрования используется поток псевдослучайных чисел, генерируемых на основе начального значения (seed). Текст преобразуется в числа, после чего к ним добавляются псевдослучайные числа. Линейный рекуррентный генератор (LFSR) используется для создания последовательности чисел. Этот метод шифрования обеспечивает высокий уровень безопасности, так как использование качественного генератора псевдослучайных чисел значительно усложняет восстановление исходного текста.

# Алгоритм работы программы

**Алгоритм работы приложения 1: Шифрование с методом маршрутной перестановки**

Приложение позволяет шифровать текст с использованием метода маршрутной перестановки, которая работает через треугольную транспозицию с сдвигами. После запуска приложения пользователь может ввести исходный текст в текстовое поле или загрузить его из файла. Затем он выбирает метод шифрования — маршрутную перестановку.

Текст делится на блоки и проходит транспозицию в виде сдвигов, образующих треугольник. После выполнения операции, приложение выводит зашифрованный текст в текстовом поле.

**Алгоритм работы приложения 2: Шифрование с ключом и анализ шифрования**

Второе приложение предлагает шифрование текста с использованием различных типов ключей (например, константа, поговорка или псевдослучайная последовательность) и анализа шифрования. После загрузки или ввода исходного текста пользователь выбирает метод шифрования, в том числе возможность ввести ключ в соответствующее текстовое поле.

Для шифрования с ключом используется алгоритм, который может применить XOR-операцию или другие методы в зависимости от типа ключа. После шифрования приложение выводит зашифрованный текст и сохраняет его в файл, если необходимо.

Приложение также проводит анализ шифрования, генерируя частотную гистограмму, которая показывает распределение символов в зашифрованном тексте. Кроме того, рассчитывается энтропия шифрованного текста, отображаемая в виде числового значения, которое указывает на степень случайности текста и сложности его расшифровки.

Пользователь может сохранить результат анализа, а также скачать зашифрованный текст. В приложении предусмотрены подсказки и подробные инструкции, которые помогут пользователю на каждом шаге работы с приложением.

Программа начинается с запроса исходных данных. Пользователь вводит текст для шифрования и выбирает метод шифрования, такой как маршрутная перестановка, шифрование с использованием ключа или метод LFSR. В случае выбора метода с ключом, пользователю необходимо ввести ключ. При использовании псевдослучайных чисел, например, сгенерированного через LCG или BBS, программа автоматически генерирует последовательность чисел.

После этого программа подготавливает текст, очищая его от лишних символов и приводя к нужному формату. Затем в зависимости от выбранного метода шифрования выполняется соответствующий алгоритм. Для маршрутной перестановки символы текста переставляются в определенном порядке, для шифрования с ключом символы изменяются с учетом введенного ключа, а метод LFSR использует генерируемые псевдослучайные числа для преобразования текста с помощью операций XOR.

Затем программа выводит зашифрованный текст на экран. Пользователь может сохранить его в файл или расшифровать. При расшифровке программа применяет обратные операции, используя тот же метод шифрования. После дешифровки выводится исходный текст.

Дополнительно программа может предложить провести частотный анализ зашифрованного текста для выявления возможных закономерностей и вычислить энтропию текста, чтобы оценить его случайность.

В конце работы программа предлагает пользователю завершить операцию или выполнить другие действия, например, сменить метод шифрования.

Для демонстрации в приложения A данного отчета предоставлены примеры работы программы. В приложение Б,В расположен исходный код приложений.

# Приложения А

## Исходый текст

Зато вернулись Алексей Толстой и, позже, Александр Куприн и никуда не уехали Брюсов, Пришвин, Вересаев, Сергеев-Ценский, Вячеслав Шишков (последнюю фамилию точно помню в учебнике: я тогда уже был юноша начитанный и знал, что Шишков выпустил первую книгу в 1916 году, то есть относить его к мастерам старой школы было не совсем честно – подгребли для массовости).

Потом, в постсоветское время, возобладала точка зрения противоположная: все мастера уехали, а оставшиеся, за исключением разве что циника Алешки Толстого, ушли во внутреннюю эмиграцию: тот же Пришвин, например, и Андрей Белый так и не стали правоверными советскими писателями. Про Шишкова вообще забыли – хотя, как ни странно, его часто переиздают; просто он каким-то чудом не проходит по ведомству советской литературы, а числится скорей в составе литературы этнографической, где-то меж Мельниковым-Печерским и Маминым-Сибиряком.

Здесь вроде бы должно написать: сегодня пришло время спокойно согласиться с тем, что литература в Гражданскую была разрублена ровно пополам и что по обе стороны советской границы жило истинное русское слово. Но я так не напишу.

Я, может быть, и без злорадства – но соглашусь с Газдановым.Русская литература осталась в Советской России, – именно там она выживала и выжила. К концу тридцатых годов, то есть спустя всего двадцать лет после первой волны эмиграции, никакой эмигрантской литературы почти не осталось. Старые мастера доживали свой век, новой литературы не появилось. В прозе можно назвать только два по-настоящему больших имени – Газданов и Набоков.

Пожалуй, еще Алданов, который писать начал до революции, но дебютировал как замечательный исторический романист уже в эмиграции. Может быть, еще Борис Поплавский. Но он умер в 1935-м. А Набоков с 1939 года пишет и публикует новые книги на английском языке.

## Метод маршрутной перестановки

Зтву елпепдБи ививутвг авмот еи и ыппарв мумсозРц и жзлел каоелксокра рСй л жаыуомсмавовсвуиплеетаывяр тждоупосндосапе ристозсинюн,В учнй е сечсянт иириутррдаи иас рстоав ву.нсеййжанес,ея(иен ввстчстке,нрнн –зруыое вчнентиле н дксбьА ен о рчпюбеиы таеоса ие,ес дор м тнтосдервниоикллТи,диувВгео н п1нрсвк мзкн ахахыэ-бсГоо виц аоа йиио, р е,еесстибзу9оотоозаааннстоюо,тПыпр ебыиаптйзНр йк А нх реллокынс1сйнсерс юатеттд не оап ым,теу ваер.улКиаПеваечелат6и от етиАюпаля;иаочдкжорт. ыррвабво еуклрс-вдн: ли тш ивнесл рля, т геоод уьР–хвыетоомНтуииаЦ но ю,лгьк–)риркеэиим п чррлйаос,у о кьклао шееШю ян о о .еяалшмм икрпиасжннбс сигйп, оюн нввниюп очпделп

м юкиеп.аоосфкносекисмо о твциоо,сш отштеугыоПяпучигрр кс лиио к о кедвчно.иснвккфмоаор,о до,рее р,аП твичм сусебаноотол

ит ыоанг в бгт охнТа врноете ноют еянвливьП, уемюднШуткыровтаиоциоои дссиаг осз о,н око ум ааиюо лемоилели в оояк плбрл л ынйожнжек чш моб,звимсюАеШснм оМиаыоозитт е аоерниккеа л оо, т:нрит ссйасслнвлтаоэ лдл ионсснивбп ро дншрктк,маиаыооем моивудввгитте лоаагтрыкааво итт .рр еистайе иье дпал зооемонкургнььрс ааосгте ,бэ1 рслодоов,тйивни еды:саоНдтнтрарп юм9ноясажсе аомсйем язвосуаьалаоети3а тлнт ужБс ,-о --с ре тр цот-щиг5 сааачшееов тввтСесутявавсисунерр-а явтл лвоеое оиг бс а ыпиьра оамн:шоиПыеог тс ботлкт ожу,.ысАвц.ги рйтбочсомидееоа–сис тлаи лцви сщ уксернмнйк твтнСнодлиАиошткечдотжяя,а нааяитеяа . йваи аойа к гнолл ка щнк Нскмзсм вМопчрре ааварпеоаМакиат леемртоа сс скыомвкобобони л.иовнноьиеоеяу, жом етль

ш ниаг гй в зек еинЗллоцплвво мибкатояртидои ыиа ы эалоом взеке тп шшСждмсолтеб ыосвеожууоивитсьочысьррпи.свлагеьшрат еаол

ьеадрр.иыть1тлоЯ т.цаа хйе,9а ,сс ан В л 3и кКттд ипье9мГо ьсопминщ айк кжресыег олоионай онейвзит Бдт ае ьиоалл – ср им нтипоГаосиачр шаиПечотп и

## Константа 3:

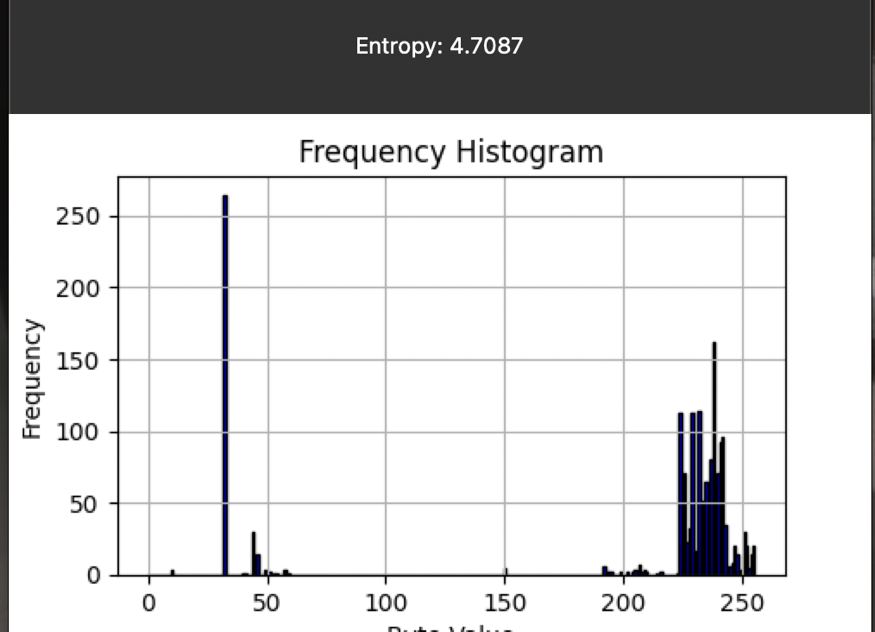
Кгхс#еиурцолфя#Гоинфим#Хсофхсм#л/#тскйи/#Гоинфгрзу#Нцтулр#л#рлнцзг#ри#цишгол#Дуфсе/#Тулыелр/#Еиуифгие/#Фиужиие0Щирфнлм/#Еъифоге#Ылынсе#+тсфоизр#чгплол#хсърс#тспр#е#цъидрлни=##хсжзг#цйи#дюо#рсыг#ргълхгррюм#л#крго/#ъхс#Ылынсе#еютцфхло#тиуец#нрлжц#е#4<49#жсзц/#хс#ифхя#схрсфлхя#ижс#н#пгфхиугп#фхгусм#ынсою#дюос#ри#фсефип#ъифхрс#™#тсзжуидол#зо#пгффсесфхл,1

Тсхсп/#е#тсфхфсеихфнси#еуип/#есксдогзгог#хсънг#куирл#тусхлестсосйрг=#ефи#пгфхиуг#цишгол/#г#сфхгеылиф/#кг#лфноъирлип#угкеи#ъхс#щлрлнг#Гоиынл#Хсофхсжс/#цыол#ес#ерцхуирр#плжугщл=#хсх#йи#Тулыелр/#ргтулпиу/#л#Грзуим#Диоюм#хгн#л#ри#фхгол#тугесеиурюпл#фсеихфнлпл#тлфгхиопл1#Тус#Ылынсег#ессдьи#кгдюол#™#шсх/#нгн#рл#фхугррс/#ижс#ъгфхс#тиуилкзгх>#тусфхс#ср#нгнлп0хс#ъцзсп#ри#тусшсзлх#тс#еизспфхец#фсеихфнсм#олхиугхцую/#г#ълфолхф#фнсуим#е#фсфхгеи#олхиугхцую#хрсжугчлъифнсм/#жзи0хс#пий#Пиоярлнсеюп0Тиъиуфнлп#л#Пгплрюп0Флдлунсп1

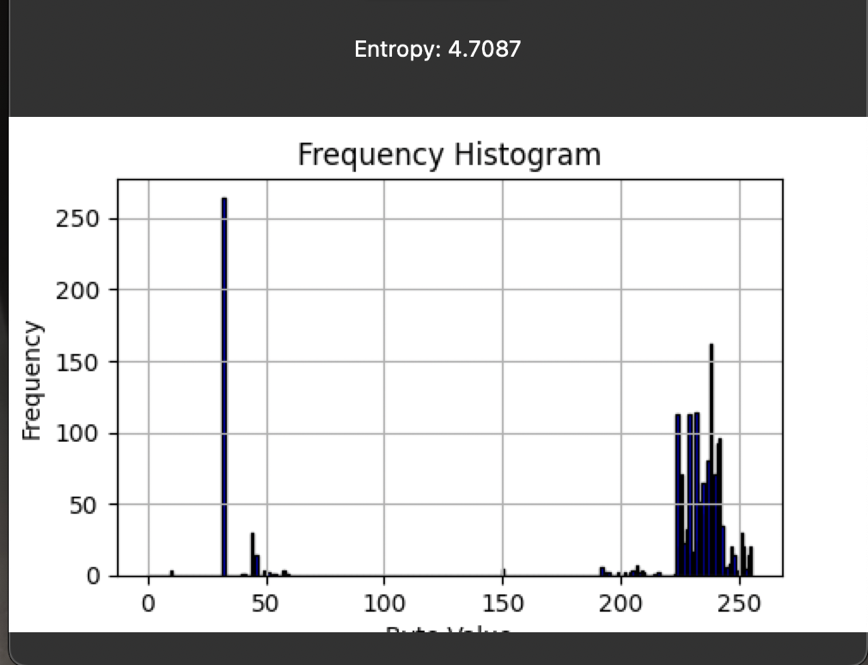
Кзифя#еусзи#дю#зсойрс#ргтлфгхя=#фижсзр#тулыос#еуип#фтснсмрс#фсжогфлхяф#ф#хип/#ъхс#олхиугхцуг#е#Жугйзгрфнц#дюог#угкуцдоирг#усерс#тстсогп#л#ъхс#тс#сди#фхсусрю#фсеихфнсм#жугрлщю#йлос#лфхлррси#уцффнси#фосес1#Рс##хгн#ри#ргтлыц1

в/#псйих#дюхя/#л#дик#косугзфхег#™#рс#фсжогыцфя#ф#Жгкзгрсеюп1Уцффнг#олхиугхцуг#сфхгогфя#е#Фсеихфнсм#Усффлл/#™#лпиррс#хгп#срг#еюйлегог#л#еюйлог1#Н#нсрщц#хулзщгхюш#жсзсе/#хс#ифхя#фтцфх#ефижс#зегзщгхя#оих#тсфои#тиуесм#есорю#плжугщлл/#рлнгнсм#плжугрхфнсм#олхиугхцую#тсъхл#ри#сфхгосфя1#Фхгуюи#пгфхиуг#зсйлегол#фесм#еин/#рсесм#олхиугхцую#ри#тселосфя1#Е#туски#псйрс#ргкегхя#хсоянс#зег#тс0ргфхсьипц#дсояылш#лпирл#™#Жгкзгрсе#л#Ргдснсе1

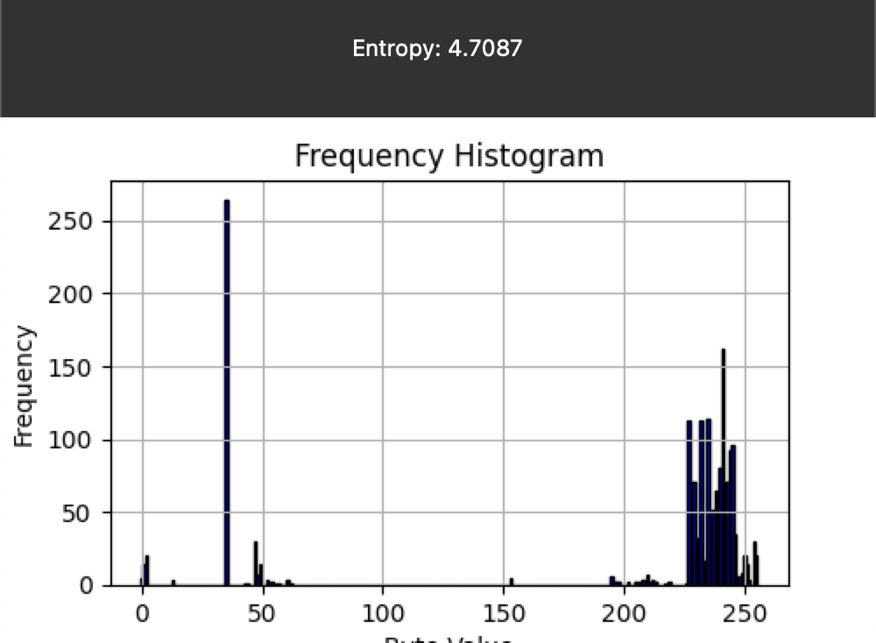
Тсйгоцм/#иьи#Гозгрсе/#нсхсуюм#тлфгхя#ргъго#зс#уиесощлл/#рс#зидхлусего#нгн#кгпиъгхиоярюм#лфхсулъифнлм#успгрлфх#цйи#е#плжугщлл1#Псйих#дюхя/#иьи#Дсулф#Тстогефнлм1#Рс#ср#цпиу#е#4<680п1#Г#Ргдснсе#ф#4<6<#жсзг#тлыих#л#тцдолнцих#рсеюи#нрлжл#рг#гржолмфнсп#кюни1



Изображение 1 оригинал



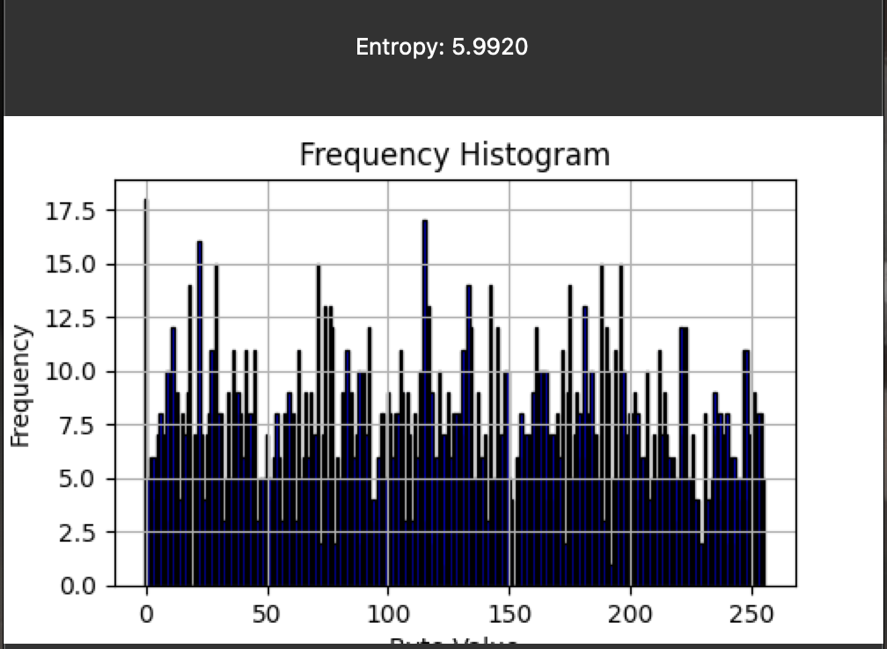
Изображение 2 маршрутная перестановка



Изображение 3 константа

## 

Изображение 4 поговорка



Изображение 5псп

# Приложение Б

import tkinter as tk

from tkinter import filedialog, messagebox

import numpy as np

import math

# Функция для шифрования методом Маршрутной перестановки(Треугольник с использованием случайных сдвигов и транспозицией)

def triangle\_encrypt(text):

num\_rows = int(np.ceil((-1 + math.sqrt(1 + 8 \* len(text))) / 2))

table = np.full((num\_rows, num\_rows), '')

index = 0

for i in range(num\_rows):

for j in range(i + 1):

if index < len(text):

table[i, j] = text[index]

index += 1

shift\_key = np.random.randint(1, 5, size=num\_rows)

# Шифруем

for i in range(num\_rows):

table[i][table[i] != ''] = np.roll(table[i][table[i] != ''], shift=shift\_key[i])

# Транспонируем треугольник (меняем строки и столбцы местами)

transposed\_table = table.T

# Соединяем строки транспонированной таблицы в зашифрованный текст

encrypted\_text = ''.join([''.join(row) for row in transposed\_table])

return encrypted\_text, shift\_key

# Функция для расшифровки методом Треугольника с использованием ключа сдвигов

def triangle\_decrypt(encrypted\_text, shift\_key):

num\_rows = int(np.ceil((-1 + math.sqrt(1 + 8 \* len(encrypted\_text))) / 2))

table = np.full((num\_rows, num\_rows), '')

elementLastBase = min(num\_rows, len(encrypted\_text) - (num\_rows - 1) \* num\_rows // 2)

# Заполняем таблицу зашифрованными строками

index = 0

for i in range(num\_rows):

if i < elementLastBase:

for j in range(num\_rows - i):

if index < len(encrypted\_text):

table[i, i + j] = encrypted\_text[index]

index += 1

else:

for j in range(num\_rows - i - 1):

if index < len(encrypted\_text):

table[i, i + j] = encrypted\_text[index]

index += 1

# Транспонируем матрицу

transposed\_table = table.T

# Шифруем

for i in range(num\_rows):

transposed\_table[i][transposed\_table[i] != ''] = np.roll(transposed\_table[i][transposed\_table[i] != ''], shift= (-1 \* shift\_key[i]))

# Соединяем строки транспонированной таблицы в зашифрованный текст

decrypted\_text = ''.join([''.join(row) for row in transposed\_table])

return decrypted\_text

class EncryptionApp:

def \_\_init\_\_(self, root):

self.root = root

self.root.title("Encryption Application")

self.filename = None

self.text = tk.Text(self.root, wrap=tk.WORD, height=20, width=60)

self.text.pack(pady=10)

btn\_frame = tk.Frame(self.root)

btn\_frame.pack()

tk.Button(btn\_frame, text="Open File", command=self.open\_file).grid(row=0, column=0, padx=5, pady=5)

tk.Button(btn\_frame, text="Encrypt", command=self.encrypt\_text).grid(row=0, column=1, padx=5, pady=5)

tk.Button(btn\_frame, text="Decrypt", command=self.decrypt\_text).grid(row=0, column=2, padx=5, pady=5)

def open\_file(self):

self.filename = filedialog.askopenfilename(filetypes=[("Text files", "\*.txt")])

if self.filename:

with open(self.filename, 'r', encoding='utf-8') as file:

content = file.read()

self.text.delete(1.0, tk.END)

self.text.insert(tk.END, content)

def encrypt\_text(self):

raw\_text = self.text.get(1.0, tk.END).strip()

if not raw\_text:

messagebox.showwarning("Warning", "No text to encrypt")

return

encrypted\_text, self.shift\_key = triangle\_encrypt(raw\_text)

self.text.delete(1.0, tk.END)

self.text.insert(tk.END, encrypted\_text)

with open("encrypted.txt", "w", encoding="utf-8") as f:

f.write(encrypted\_text)

messagebox.showinfo("Info", "Text encrypted and saved as 'encrypted.txt'")

def decrypt\_text(self):

encrypted\_text = self.text.get(1.0, tk.END).strip()

if not encrypted\_text:

messagebox.showwarning("Warning", "No text to decrypt")

return

if not hasattr(self, 'shift\_key'):

messagebox.showerror("Error", "No key available for decryption")

return

decrypted\_text = triangle\_decrypt(encrypted\_text, self.shift\_key)

self.text.delete(1.0, tk.END)

self.text.insert(tk.END, decrypted\_text)

if \_\_name\_\_ == "\_\_main\_\_":

root = tk.Tk()

app = EncryptionApp(root)

root.mainloop()

# Приложение В

import tkinter as tk

from tkinter import ttk, filedialog, messagebox

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

# Частотный анализ текста

def privateAnalysisText(text: str, encode="cp1251") -> np.ndarray:

privateAnalysis = np.zeros(shape=256, dtype=int)

for char in text:

privateAnalysis[char.encode(encode, errors="replace")[0]] += 1

return privateAnalysis

# Энтропия

def entropy(privateAnalysis):

total = sum(privateAnalysis)

probabilities = privateAnalysis / total

entropy = -np.nansum(probabilities \* np.log2(probabilities, where=privateAnalysis != 0))

return entropy

# Шифрование текста

def encrypt\_with\_constant\_cp1251(text, constant):

def char\_shift\_cp1251(char, constant):

byte\_value = char.encode('cp1251')[0]

shifted\_value = (byte\_value + constant) % 256

return bytes([shifted\_value]).decode('cp1251', errors='replace')

return ''.join(char\_shift\_cp1251(char, constant) for char in text)

def encrypt\_with\_proverb\_cp1251(text, key\_phrase):

def char\_to\_cp1251\_byte(char):

return char.encode('cp1251')[0]

def byte\_to\_cp1251\_char(byte):

return bytes([byte]).decode('cp1251', errors='replace')

alphabet = 'АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯабвгдежзийклмнопрстуфхцчшщъыьэюя'

key = [(alphabet.index(char.upper()) + 1) for char in key\_phrase if char.upper() in alphabet]

encrypted = []

for i, char in enumerate(text):

char\_byte = char\_to\_cp1251\_byte(char)

shifted\_byte = (char\_byte + key[i % len(key)]) % 256

encrypted\_char = byte\_to\_cp1251\_char(shifted\_byte)

encrypted.append(encrypted\_char)

return ''.join(encrypted)

# Реализация LFSR с использованием битовых операций и двоичного представления

def lfsr(seed, length):

# Параметры для LFSR

taps = [7, 4, 3, 2] # Полином x^8 + x^5 + x^4 + x^3 + 1

state = seed % 256 # Обрезаем если сид больше 8

sequence = []

for \_ in range(length):

new\_bit = 0

for tap in taps:

new\_bit ^= state >> (tap % 8)

new\_bit = new\_bit & 1

state = ((state >> 1) | (new\_bit << 7)) & 0xFF # Последние 8 бит результата

sequence.append(state)

return sequence

def encrypt\_with\_psp\_cp1251(text, psp):

def char\_to\_cp1251\_byte(char):

return char.encode('cp1251')[0]

def byte\_to\_cp1251\_char(byte):

return bytes([byte]).decode('cp1251', errors='replace')

encrypted = []

for i, char in enumerate(text):

char\_byte = char\_to\_cp1251\_byte(char)

shifted\_byte = (char\_byte + psp[i % len(psp)]) % 256

encrypted\_char = byte\_to\_cp1251\_char(shifted\_byte)

encrypted.append(encrypted\_char)

return ''.join(encrypted)

def plot\_histogram\_in\_tkinter(analysis, root):

# Create a new figure and axis for the plot

fig, ax = plt.subplots(figsize=(5, 3)) # Adjust the size as needed

x = list(range(256))

# Plot the histogram

ax.bar(x, analysis, width=1.0, color='blue', edgecolor='black')

ax.set\_title("Frequency Histogram")

ax.set\_xlabel('Byte Value')

ax.set\_ylabel('Frequency')

ax.grid(True)

# Remove any old plot from the canvas if it exists

if hasattr(root, 'canvas'):

root.canvas.get\_tk\_widget().destroy()

# Create the FigureCanvasTkAgg object and embed it in the Tkinter window

canvas = FigureCanvasTkAgg(fig, master=root)

canvas.draw()

# Pack the canvas widget into the Tkinter window

canvas.get\_tk\_widget().pack(pady=20)

# Store the canvas to update it later if needed

root.canvas = canvas

# Tkinter GUI Application

class EncryptionApp:

def \_\_init\_\_(self, root):

self.root = root

self.root.title("Text Encryption and Analysis")

self.text = ""

self.text\_box = tk.Text(root, height=10, width=50)

self.text\_box.pack(pady=10)

self.load\_button = tk.Button(root, text="Load Text", command=self.load\_text)

self.load\_button.pack(pady=5)

# Combobox for selecting encryption method

self.method\_var = tk.StringVar()

self.method\_menu = ttk.Combobox(root, textvariable=self.method\_var, values=["Const", "Proverb", "PSP"])

self.method\_menu.set("Const") # Default value

self.method\_menu.pack(pady=5)

# Entry for constant and proverb inputs

self.constant\_label = tk.Label(root, text="Enter constant key:")

self.constant\_label.pack(pady=5)

self.constant\_entry = tk.Entry(root)

self.constant\_entry.pack(pady=5)

self.proverb\_label = tk.Label(root, text="Enter proverb key:")

self.proverb\_label.pack(pady=5)

self.proverb\_entry = tk.Entry(root)

self.proverb\_entry.pack(pady=5)

self.encrypt\_button = tk.Button(root, text="Encrypt", command=self.encrypt\_text)

self.encrypt\_button.pack(pady=5)

self.histogram\_button = tk.Button(root, text="Generate Histogram", command=self.generate\_histogram)

self.histogram\_button.pack(pady=5)

self.entropy\_button = tk.Button(root, text="Calculate Entropy", command=self.calculate\_entropy)

self.entropy\_button.pack(pady=5)

self.save\_button = tk.Button(root, text="Save Text", command=self.save\_text)

self.save\_button.pack(pady=5)

self.result\_label = tk.Label(root, text="Entropy will be displayed here.", wraplength=400)

self.result\_label.pack(pady=10)

def load\_text(self):

file\_path = filedialog.askopenfilename(title="Select a Text File", filetypes=[("Text files", "\*.txt")])

if file\_path:

with open(file\_path, 'r') as f:

self.text = f.read()

self.text\_box.delete(1.0, tk.END)

self.text\_box.insert(tk.END, self.text)

def encrypt\_text(self):

if not self.text:

messagebox.showerror("Error", "Please load text first.")

return

method = self.method\_var.get()

if method == "Const":

try:

constant\_key = int(self.constant\_entry.get())

except ValueError:

messagebox.showerror("Error", "Please enter a valid integer for the constant key.")

return

encrypted\_text = encrypt\_with\_constant\_cp1251(self.text, constant\_key)

elif method == "Proverb":

proverb\_key = self.proverb\_entry.get()

if not proverb\_key:

messagebox.showerror("Error", "Please enter a proverb key.")

return

encrypted\_text = encrypt\_with\_proverb\_cp1251(self.text, proverb\_key)

elif method == "PSP":

try:

psp = lfsr(2134, len(self.text))

encrypted\_text = encrypt\_with\_psp\_cp1251(self.text, psp)

except Exception as e:

messagebox.showerror("Error", f"Error generating PSP: {e}")

return

else:

messagebox.showerror("Error", "Invalid encryption method selected.")

return

self.encrypted\_text = encrypted\_text

messagebox.showinfo("Encryption", f"Text successfully encrypted using {method} method.")

def generate\_histogram(self):

if not self.text:

messagebox.showerror("Error", "Please input text first.")

return

analysis = privateAnalysisText(self.text)

plot\_histogram\_in\_tkinter(analysis, self.root)

def calculate\_entropy(self):

if not self.text:

messagebox.showerror("Error", "Please input text first.")

return

analysis = privateAnalysisText(self.text)

result = entropy(analysis)

self.result\_label.config(text=f"Entropy: {result:.4f}")

def save\_text(self):

if not self.text:

messagebox.showerror("Error", "Please load text first.")

return

file\_path = filedialog.asksaveasfilename(defaultextension=".txt", filetypes=[("Text files", "\*.txt")])

if file\_path:

with open(file\_path, 'w') as f:

f.write(self.text)

messagebox.showinfo("Save", "Text saved successfully!")

if \_\_name\_\_ == "\_\_main\_\_":

root = tk.Tk()

app = EncryptionApp(root)

root.mainloop()