Научно-исследовательская работа

Информатика

СОЗДАНИЕ ПРОГРАММЫ ДЛЯ ШИФРОВАНИЯ И ДЕШИФРОВАНИЯ ТЕКСТОВОЙ ИНФОРМАЦИИ

Выполнил:

Алтунин Александр Романович

учащийся 11Б класса

МАОУ Лицей № 3, Россия, г. Красноярск

Руководитель:

Малеев Олег Николаевич

учитель информатики

МАОУ Лицей № 3, Россия, г. Красноярск

Введение

Бурное развитие средств вычислительной техники привело к созданию большого числа разного рода автоматизированных информационных и управляющих систем, к возникновению принципиально новых, так называемых, информационных технологий. Информационные технологии приобрели глобальный трансграничный характер и стали неотъемлемой частью всех сфер деятельности личности, общества и государства. Их эффективное применение является фактором ускорения экономического развития государства и формирования информационного общества.

Однако любое фундаментальное техническое или технологическое новшество, открывая широкие перспективы для развития личности и общества, всегда становится источником новых потенциальных опасностей. По данным экспертно-аналитического центра InfoWatch, в первой половине 2022 года по сравнению с первым полугодием 2021 года число утечек информации ограниченного доступа выросло в два раза в мире и в полтора раза – в России.

Обеспечение информационной безопасности в настоящее время превратилось в одну из основных задач государства. Реализация национальных интересов в информационной сфере направлена на формирование безопасной и устойчивой к различным видам воздействия информационной инфраструктуры в целях обеспечения конституционных прав и свобод человека и гражданина, стабильного социально-экономического развития страны, а также национальной безопасности Российской Федерации.

Растущий уровень угроз приводит к необходимости искать собственные методы защиты. Для защиты информации сейчас существует множество различных программ с различными функциями. Я решил внести свой вклад в защиту информации и создать для этих целей собственную программу.

Цель работы: создание программы для кодирования и декодирования текстовой информации.

Задачи:

1. Разработать алгоритмы шифрования и дешифрования;

2. Проанализировать криптостойкость алгоритма;

3. Реализовать разработанный алгоритм на языке программирования Python, написать и отладить программу;

4. Создать графический интерфейс программы для удобства работы пользователя.

Алгоритм шифрования

Шифрование — процесс обратимого преобразования информации, который защищает её от посторонних лиц, оставляя доступной только для авторизованных пользователей. Главная его цель заключается в соблюдении конфиденциальности. Любой алгоритм шифрования основывается на применении ключа, поэтому авторизованным считается пользователь, которому он принадлежит. Также этот процесс состоит из двух компонентов: шифрования и дешифрования.

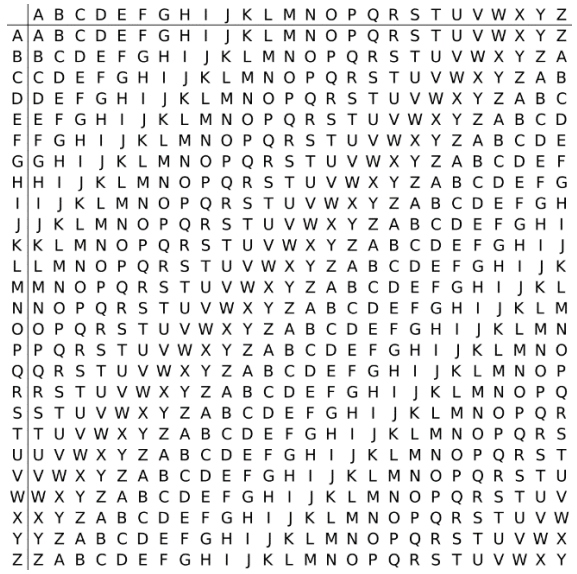
Основным показателем надёжности шифра является его криптографическая стойкость — способность противостоять дешифрованию. Ключ, используемый алгоритмом, содержит определенную секретную информацию. Её объем измеряется в битах.

Существует два вида шифрования: симметричное и ассиметричное.

Симметричный метод предполагает, что для шифровки и информации используется один и тот же криптографический ключ. Благодаря этому алгоритм шифрования упрощается, но возникает задача безопасно передать ключ, ведь тот, кто его перехватит, сможет без проблем прочесть передаваемую информацию.

Ассиметричное же шифрование использует пару ключей. Открытый ключ используется только для кодирования информации и находится в свободном доступе. Секретный ключ используется только для декодирования и доступен только получателю. Алгоритм ассиметричного шифрования сложнее, зато позволяет безопасно передавать данные: после кодирования их не сможет прочесть никто, кроме получателя.

Мы остановили свой выбор на симметричном методе шифрования. Наш алгоритм напоминает шифр Виженера. Суть этого шифра заключается в последовательном применении шифра Цезаря к каждому символу исходного текста. При этом параметр сдвига для этого символа определяется соответствующим символом ключевого слова (например, «a» – это сдвиг на 0, «b» – сдвиг на +1). В нашем алгоритме ключевого слова, как такового, нет: параметр сдвига для каждого символа определяется с помощью генератора псевдослучайных чисел. Кроме того, наш алгоритм способен при желании пользователя «произвольно» размещать случайные символы в шифротексте. Эта модификация призвана усложнить подбор пароля (об этом далее).



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Рис.1 – «Квадрат Виженера» для кодирования на английском языке

Шифр Виженера является полиалфавитным, то есть пригоден для кодирования любого алфавита. В качестве алфавита для нашего алгоритма мы решили использовать символы с 0 по 55291 по стандарту UTF-8. В этот промежуток входят все символы, необходимые для работы с текстами на различных языках (русский, английский, китайский и т.д.), спецсимволы, математические знаки и т.п.

Таблица 1

Некоторые символы и их коды по стандарту кодирования UTF-8

Код символа	Символ	Код символа	Символ
33	!	1041	Б
44	,	1081	й
53	5	6058	ᗪ
78	N	10363	⠻
113	q	55291	ﾛｴ

В процессе шифрования происходит перебор всех символов текста. Каждый символ сдвигается по алфавиту на псевдослучайное число вперёд или назад. При достижении края алфавита происходит цикличное перемещение в начало или в конец, соответственно. Далее при необходимости генерируются несколько (10% от длины исходного текста) псевдослучайных чисел – позиций дополнительных символов в шифротексте, на эти позиции ставятся случайные символы, сдвигая позиции последующих.

Расшифровка производится зеркально шифрованию. При необходимости из шифротекста удаляются лишние символы, далее текст разбивается на символы, и они сдвигаются назад или вперёд на нужное значение.

Итак, алгоритм способен работать с любым языком и любыми символами. Шифрование производится путём сдвига каждого отдельного символа сообщения в алфавите.

Анализ криптографической стойкости алгоритма

Криптографическая стойкость (или криптостойкость) – это способность криптографического алгоритма противостоять криптоанализу. Устойчивым называется тот алгоритмический код, успешное криптоаналитическое вскрытие которого предписывает заинтересованному лицу иметь в своём распоряжении настолько колоссальные вычислительно-алгоритмические или временные

ресурсы на дешифровку перехваченных сообщений, что задачу решить либо практически невозможно, либо к тому времени, когда она будет решена, зашифрованная информация утратит свою ценность для злоумышленников.

Исследования по теоретической составляющей стойкости алгоритмов впервые были проведены К. Шенноном. В своих публикациях он, применяя вероятностную модель алгоритма, впервые сформулировал определение совершенно стойкого алгоритма и доказал, что такой алгоритм действительно существует.

Криптостойкость алгоритма можно определять с помощью специальных единиц измерения стойкости, таких как:

- затраты времени на взлом ключа, включая разработку соответствующей вычислительной модели;
- необходимый объём памяти для взлома ключа;
- стоимостные затраты на взлом ключа;
- количество необходимой энергии, затрачиваемой на взлом ключа;
- временная сложность наилучшего из известных алгоритмов, нарушающих безопасность;
- физический объём вычислительной модели для взлома ключа.

Так как наш алгоритм является модификацией шифра Виженера, стоит начать с методов взлома этого шифра. Созданный в XVII веке, он считался неподдающимся взлому на протяжении почти 3 веков. В 1863 году Фридрих Касики стал первым, кто опубликовал успешный алгоритм атаки на шифр Виженера. Его метод опирается на то, что некоторые слова могут быть зашифрованы одинаковыми символами, что приводит к повторению групп символов в зашифрованном тексте. В 1920 году Вильямом Фридманом был изобретён тест Фридмана, использующий индекс совпадения, который измеряет частоты повторения символов. Эти методы позволяют определить длину кодового слова, а дальше взлом не представляет особой сложности. Однако в нашем алгоритме длина «кодового слова» равна длине исходного текста, поэтому тесты Касики и Фридмана неприменимы.

Существует вариант шифра Виженера с бегущим ключом, где в качестве кодового слова используется блок текста, равный по длине исходному тексту. Методы Фридмана и Касики здесь неприменимы, так как ключ не повторяется. Недостатком данного метода является то, что криптоаналитик обладает статистической информацией о ключе (если он написан на известном языке), и эта информация будет отражаться в шифре.

Но в нашем алгоритме ключ генерируется практически случайно. Сказать «абсолютно случайно» мы не можем, поскольку все сгенерированные генератором псевдослучайных чисел значения являются порождениями некоторого исходного значения (сида), поэтому они связаны друг с другом. Чтобы усложнить поиск закономерностей при перед каждой генерацией сид обновляется. Если при использовании программы для каждого сообщения будет использоваться новый пароль, то алгоритм можно будет назвать (с некоторой оговоркой) шифром Вернама-Виженера, для которого доказана абсолютная криптостойкость «по Шеннону».

Если предположить, что взломщику известна часть исходного сообщения длины l (для простоты возьмём первые l символов), то он легко сможет вычислить первые l сгенерированных псевдослучайных чисел. Это позволит ему получить некоторое количество паролей, при использовании которых первые l сгенерированных чисел совпадут с найденными. В таком случае существует вероятность (впрочем, стремящаяся к нулю), что, расшифровав сообщение каждым из найденных паролей, злоумышленник поймёт, какое сообщение на самом деле было зашифровано. Чтобы не допустить этого, в нашем алгоритме реализована псевдослучайность знака параметра сдвига («+» или «-»). Например, комбинацию «ГАВ» из исходного текста «ГДЕ» в алфавите «АБВГДЕ» можно получить 8 способами («+0+2+3», «+0+2-3», «+0-4+3», «+0-4-3», «-0+2+3», «-0+2-3», «-0-4-3»). Таким образом, некоторые l символов исходного текста могут быть зашифрованы в одну и ту же последовательность 2^l способами, то есть количество потенциально верных паролей увеличивается. Пароль может содержать 55292 символа UTF-8, а его длина может составлять

от 0 до максимум 2^{63}-1 символов. Таким образом, максимальное количество возможных паролей равно $1+55292+55292^2+...+55292^{2\wedge 63-1}$. С учётом того, что при желании пользователя в шифротекст могут быть включены дополнительные символы, игнорирование которых приведёт к неверной расшифровке, это, и без того огромное, число удваивается.

Таким образом, наш алгоритм является криптостойким, даже если злоумышленник обладает частью исходного сообщения.

Реализация алгоритма на Python

Python – высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ. Язык очень прост в освоении и использовании, а огромное множество всевозможных библиотек делают область его применения практически необозримой. Именно его мы выбрали для реализации поставленной цели.

Прежде всего была написана функция (приложение 1), которая осуществляет кодирование переданного ей текста с помощью переданного пароля, а также зеркальная ей функция декодирования. Для реализации алгоритма кодирования (декодирования) была импортирована библиотека random, входящая в стандартный устанавливаемый пакет. Пароль пользователя в виде строки передаётся в качестве параметра для random.seed(). Это позволяет успешно декодировать информацию, так как при одинаковых ключах генераторы псевдослучайностей будут выдавать одинаковые значения. Если бы ключ не был явно указан, в его качестве был бы взят некоторый параметр системы, в таком случае декодирование произвести было бы невозможно, так как генерируемое «кодовое слово» отличалось бы от исходного.

Далее были написаны функции, осуществляющие работу с файлами:

1. Функция, которая получает текст из файлов и передаёт его на шифрование (дешифрование);
2. Функции, сохраняющие текст (как полученный из файла, так и введённый с клавиатуры);
3. Функции, осуществляющие отправку текста из файлов на кодирование (декодирование) в режиме реального времени;
4. Функция для генерации пароля.

Осуществление работы с файлами было реализовано с помощью встроенной библиотеки os, а также её метода os.path. Для реализации шифрования (дешифрования) в режиме реального времени была использована встроенная библиотека threading.

После написания программы она была успешно отлажена.

Создание графического интерфейса

Для создания графического интерфейса была использована встроенная библиотека tkinter – кросс-платформенная событийно-ориентированная графическая библиотека на основе средств Tk.

Интерфейс состоит из главного окна (Рис.2), в котором отображается одна из восьми возможных страниц (рамок). Главное меню представляет собой семь кнопок, нажатие на которые ведёт на соответствующую страницу – шифрование текста (Рис.3), шифрование файлов, шифрование в режиме реального времени, дешифрование текста, дешифрование файлов, дешифрование в режиме реального времени, генератор паролей. Нажатие на кнопку «Назад» на этих страницах выводит на экран главное меню.

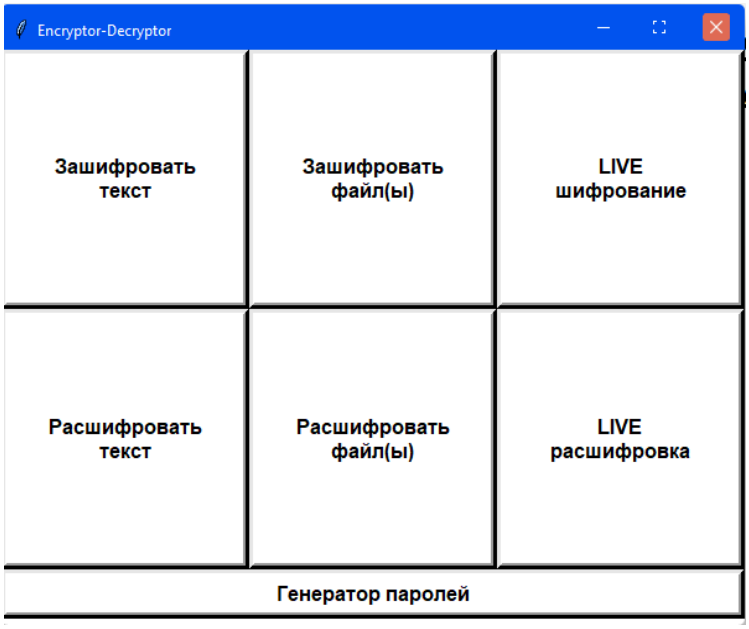


Рис.2 – Главное меню программы

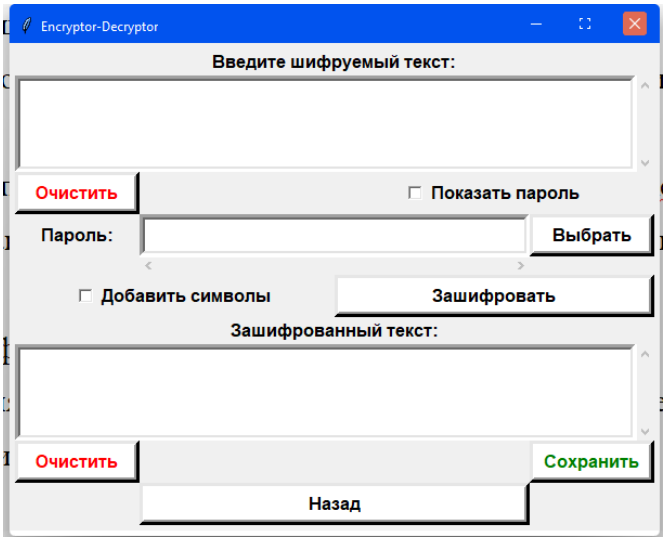


Рис.3 – Страница для шифрования текста

Так как в интерфейсе присутствует большое количество кнопок и ярлыков, было решено создать соответствующие классы (наследующие признаки классов tkinter) для хранения некоторых нужных, но не заданных по умолчанию, параметров (ширина, шрифт, толщина границы и т. п.).

Также были использованы модули tkinter.filedialog и tkinter.messagebox для создания диалоговых окон сохранения/выбора файлов и показа предупреждающего окна соответственно.

Была написана функция для показа/скрытия введённого пароля.

После написания и отладки всех функций программа была успешно скомпилирована в исполняемый файл при помощи pyinstaller.

Возможности использования программы

Созданная нами программа распространяется в виде одного исполняемого файла (.exe), вес которого составляет всего 9,37 Мб. Она не требует установки и готова к работе сразу после скачивания.

Для начала работы с программой нужно скачать исполняемый файл на компьютер с ОС Windows либо подключить к нему внешний носитель с файлом (программа способна работать даже без копирования на компьютер). Далее двойным щелчком левой кнопки мыши необходимо запустить файл.

В открывшемся окне необходимо ввести требуемые данные на нажать кнопку «Зашифровать». При работе с текстом, введённым пользователем прямо в программе, есть возможность сохранить зашифрованный текст, для этого нужно нажать на соответствующую кнопку. При работе с файлами программам автоматически сохраняет результат (под именем исходного файла), удаляя исходный файл.

Пока программа способна работать только с простыми текстовыми файлами и текстом, введённым пользователем. Планируется расширить диапазон допустимых типов данный и файлов.

Программа предназначена, в первую очередь, для защиты личной информации людей (паролей, дневников и т.п.), однако может применяться и для безопасного хранения информации ограниченного доступа различных компаний. Если найти способ безопасно передать ключ, программа сможет быть использована для безопасного обмена данными.

Преимуществами нашей программы перед аналогами являются:

1. Отсутствие необходимости в установке. Программа распространяется в виде одного исполняемого файла (.exe) и готова к работе сразу после скачивания;

2. Программу можно запускать прямо со съёмного носителя, без копирования на компьютер;

3. Интуитивно понятный интерфейс;

4. Программа способна работать не только с введённым с клавиатуры текстом, но и с файлами.

Перспективы развития проекта:

1. Увеличение количества поддерживаемых типов файлов;

2. Разработка способа безопасной передачи ключа по открытому каналу.

3. Создание web-интерфейса или Telegram-бота для программы.

Заключение

В ходе проведения работы выполнены следующие действия:

1. разработан алгоритм, обеспечивающий надёжное шифрование личной информации;

2. проанализирована криптостойкость алгоритма;

3. алгоритм успешно реализован на языке программирования Python, программа была написана со всеми желаемыми функциями;

4. создан графический интерфейс программы;

Таким образом, основные задачи работы выполнены.

Цель работы по созданию программы, позволяющей осуществлять кодирование и декодирование информации, достигнута.

Список использованной литературы

1. Доктрина информационной безопасности Российской Федерации, утверждённая Указом Президента РФ от 5 декабря 2016 г. № 646 [Электронный ресурс]. Доступ из справочно-правовой системы «КонсультантПлюс».

2. Лутц М. Изучаем Python. – М.: Диалектика, 2019. – 832 с.

3. Яковлев А.В., Безбогов А.А., Родин В.В., Шамкин В.Н. Криптографическая защита информации: Учебное пособие. - Тамбов: Издательство ТГТУ, 2006. - 140 с.

4. Антонов Ю.С. Занимательная криптография [Электронный ресурс] // КиберЛенинка: [сайт]. URL: https://cyberleninka.ru/article/n/zanimatelnaya-kriptografiya/viewer.

5. Горьков А.В. Шифр Вижинера и его разгадка [Электронный ресурс] // Хабр: [сайт]. URL: https://habr.com/ru/post/103055/.

6. Филяк П.Ю. Актуальность обеспечения информационной и экономической безопасности в условиях информационного общества [Электронный ресурс] // КиберЛенинка: [сайт]. URL: https://cyberleninka.ru/article/n/aktualnost-obespecheniya-informatsionnoy-i-ekonomicheskoy-bezopasnosti-v-usloviyah-informatsionnogo-obschestva.

7. Шурховецкий Г.Н. Криптостойкость алгоритмов шифрования [Электронный ресурс] // Электронный научный журнал «Молодая наука Сибири». URL: https://mnv.irgups.ru/sites/default/files/articles_pdf_files/shurhoveckiygnkriptostoykost_algoritmov_shifrovaniya.pdf.

8. Отчёт об исследовании утечек информации ограниченного доступа в I половине 2022 года [Электронный ресурс] // Экспертно-аналитический центр InfoWatch: [сайт]. URL: https://www.infowatch.ru/sites/default/files/analytics/files/otchyot-ob-utechkakh-dannykh-za-1-polugodie-2022-goda_1.pdf.

9. Python 3.10.5 documentation [Электронный ресурс] // Официальный сайт python. URL: https://docs.python.org/release/3.10.5/.

10. tkinter – Python interface to Tcl/Tk [Электронный ресурс] //Официальный сайт python. URL: https://docs.python.org/3/library/tkinter.html.

11. tkinter.filedialog — File selection dialogs [Электронный ресурс] // Официальный сайт python. URL:

https://docs.python.org/3/library/dialog.html?highlight=filedialog#module-tkinter.filedialog.

12. tkinter.messagebox — Tkinter message prompts [Электронный ресурс] // Официальный сайт python. URL: https://docs.python.org/3/library/tkinter.messagebox.html?highlight=message#module-tkinter.messagebox.

13. random – Generate pseudo-random numbers [Электронный ресурс] // Официальный сайт python. URL: https://docs.python.org/3/library/random.html.

14. os — Miscellaneous operating system interfaces [Электронный ресурс] // Официальный сайт python. URL: https://docs.python.org/3/library/os.html.

15. os.path — Common pathname manipulations [Электронный ресурс] // Официальный сайт python. URL: https://docs.python.org/3/library/os.path.html?highlight=path#module-os.path.

16. threading — Thread-based parallelism [Электронный ресурс] // Официальный сайт python. URL: https://docs.python.org/3/library/threading.html?highlight=threading%20thread#threading.Thread.

Приложение 1. Функция, осуществляющая кодирование

```
def encryption(text: str, password: str, add_symbols: bool, use_csum: bool):
    #Задаёт сид для генератора псевдослучайных чисел
    random.seed(password + str(len(text)))

    # Перебирает каждый символ исходного текста
    for i in range(len(text)):
        s = text[i]
        sc = ord(s)

        # Перезадаёт сид и генерирует случайное число
        a = random.randint(0, len(password))
        b = random.randint(0, len(password))
        random.seed( password[min(a,b) : max(a,b)] + str(random.random()))
        del a
        del b
        modification = random.randint(0, 55291)

        if random.choice([True, False]):
            if sc + modification > 55291: #Если при сложении превысили 55291, прибавляем лишнее к 0
                text = text.replace(s, chr(sc + modification - 55292), 1) #Меняет символ на новый
            else:
                text = text.replace(s, chr(sc + modification), 1)

        else:
            if sc - modification < 0: #Если при вычитании опустились ниже 0, отнимаем лишнее от 55291
```

```
            text = text.replace(s, chr(sc - modification + 55292), 1)
        else:
            text = text.replace(s, chr(sc - modification))

    #Если нужно добавить символы в шифротекст (повышает стойкость)
    if add_symbols:
        random.seed(password[::-1]) #Задаёт сид для генератора псевдослучайных чисел
        len_enc = len(text) #Получаем длину шифротекста (будут добавлены 10% символов от исходного кол-ва)
        poses = []
        for i in range(len(text) // 10): #Получаем случайные позиции для вставки доп. символов (10% от исходного кол-ва символов)
            poses.append(random.randint(0, len_enc + 1))
            len_enc += 1
        poses.sort() #Сортируем полученные позиции по возрастанию
        for i in poses:
            text = text[:i] + chr(random.randint(0, 55291)) + text[i:] #Вставляем на позиции случайные символы

    # Контрольная сумма
    if use_csum:
        random.seed(text)
        csum = hex(random.randint(0, 4294967296))[2:]
        while len(csum)<8:
            csum = '0' + csum
        text += csum

    return text
```