Муниципальное бюджетное общеобразовательное учреждение средняя общеобразовательная школа №1 им. Л.Б. Ермина с.Засечное

Проектная работа
«Разработка Python-приложения для шифрования данных»

Автор проекта: Клочков Максим Михайлович
Руководитель: Колгушкин Андрей Анатольевич

с. Засечное
2024 г.

Содержание

Введение

Современный мир – мир высоких технологий и изобретений. А самым важным и ценным ресурсом в настоящее время является информация. Поэтому она должна подлежать защите. А самая надежная защита информации осуществляется благодаря шифрованию.

Оно помогает защитить личные данные, финансовые транзакции, коммерческую информацию и государственные секреты от несанкционированного доступа. Без правильного шифрования данных информационная безопасность становится уязвимой. Хакеры и киберпреступники могут повредить частные жизни людей, нарушить работу предприятий и даже угрожать национальной безопасности. Шифрование данных способствует созданию защитного барьера и предотвращает распространение вредоносного программного обеспечения, фишинговых атак и других видов кибератак.

Актуальность: мы живём в информационном обществе, где с каждым годом информация увеличивается. Но информация становится уязвимее, появляются новые способы взлома данных. Поэтому данная тема является очень актуальной в современном мире.

Цель проекта: разработать Python-приложение для шифрования данных шифром Цезаря.

Задачи:

1. Изучить, что такое шифрование и как оно появилось.
2. Узнать какие бывают методы шифрования и выявить их преимущества и недостатки.
3. Создать программу, которая будет шифровать по вводимым данным.

Гипотеза: с помощью Python-приложения можно осуществлять шифрование данных.

Объект исследования: криптография.

Предмет исследования: способы шифрования и дешифрования данных.

Методы исследования: изучение теоретических основ по данной теме, создание программы для шифрования данных.

Глава 1. Теоретическая часть. История создания. Основные понятия. Методы шифрования.

1.1. Что такое шифрование

Шифрование — обратимое преобразование информации в целях сокрытия от неавторизованных лиц с предоставлением в это же время авторизованными пользователям доступа к ней. Главным образом, шифрование служит для соблюдения конфиденциальности передаваемой информации. Важной особенностью любого алгоритма шифрования является использование ключа, который утверждает выбор конкретного преобразования из совокупности возможных для данного алгоритма.

В целом шифрование состоит из двух составляющих: зашифрование и расшифрование.

С помощью шифрования обеспечиваются **три состояния безопасности информации:**

- **Конфиденциальность**

Шифрование используется для скрытия информации от неавторизованных пользователей при передаче или при хранении.

- **Целостность**

Шифрование используется для предотвращения изменения информации при передаче или хранении.

- **Идентифицируемость**

Шифрование используется для аутентификации источника информации и предотвращения отказа отправителя информации от того факта, что данные были отправлены именно им.

Шифрование применяется для хранения важной информации в ненадёжных источниках и передачи её по незащищённым каналам связи. Такая передача данных включает **два взаимно обратных процесса:**

- Перед отправлением данных по линии связи или перед помещением на хранение они подвергаются ***зашифровыванию***.
- Для восстановления исходных данных из зашифрованных к ним применяется процедура ***расшифровки (или дешифровки)***.

Шифрование изначально использовалось только для передачи конфиденциальной информации. Однако впоследствии шифровать информацию начали с целью её хранения в ненадёжных источниках. Шифрование информации с целью её хранения применяется и сейчас, это позволяет избежать необходимости в физически защищённом хранилище.

Шифром называется пара алгоритмов, реализующих каждое из указанных преобразований. Эти алгоритмы применяются к данным с использованием ключа. Ключи для шифрования и для расшифровки могут различаться, а могут быть одинаковыми. Секретность второго (расшифровывающего) из них делает данные недоступными для несанкционированного ознакомления, а секретность первого (шифрующего) делает невозможным внесение ложных данных. В первых методах шифрования использовались одинаковые ключи, однако в 1976 году были разработаны алгоритмы с применением разных ключей. Сохранение этих ключей в секретности и правильное их разделение между адресатами является очень важной задачей для сохранения конфиденциальности передаваемой информации. Эта задача исследуется в теории управления ключами (в некоторых источниках она упоминается как разделение секрета).

1.2. История шифрования данных и криптографии:

Криптография и шифрование тысячи лет используются людьми для защиты своих секретов. История шифрования делится на несколько периодов:

Первый период (приблизительно с 3-го тысячелетия до н. э.) характеризуется господством моноалфавитных шифров (основной

принцип — замена алфавита исходного текста другим алфавитом через замену букв другими буквами или символами):

- Древний Египет

Самый древний текст с элементами криптографии найден в гробнице древнеегипетского вельможи Хнумхотепа II (рисунок 1). Где-то около 1900 г до н.э. писарь Хнумхотепа описывал жизнь своего господина в его гробнице. Среди иероглифов он использовал несколько необычных символов, которые скрывают прямое значение текста. Такой метод шифрования фактически представляет собой шифр подстановки, когда элементы исходного текста заменяются другими элементами по определённым правилам.

По мере развития египетской культуры замены иероглифов встречались всё чаще. Есть разные версии, почему египтяне использовали такую систему шифрования. По одной версии, они хотели охранить свои религиозные ритуалы от обычных людей. По другой версии, таким образом писцы придавали тексту некий формальный вид, как в наше время юристы используют специфические выражения для замены обычных слов. Как и сейчас, египетская криптография тоже могла быть способом писца впечатлить других людей — показать, что он может изъясняться на более высоком уровне, чем они.

- Древняя Греция

Примерно в 500 г до н.э. спартанцы разработали устройство под названием скитала (рисунок 2), созданное для отправки и получения секретных сообщений. Оно представляло собой цилиндр, обёрнутый по спирали узкой полоской пергамента. Послание писалось вдоль скиталы, но если полоску развернуть, оно становилось нечитаемым. Для прочтения текста требовалась скитала такого же диаметра. Только в этом случае буквы становились в ряд, чтобы восстановить оригинальное сообщение.

Скитала является примером перестановочного шифра, в котором элементы исходного текста меняют местами, а не заменяют другими

символами. По современным стандартам скиталу было бы очень просто взломать, но 2500 лет назад очень мало людей умели читать и писать. Скитала обеспечила спартанцам защищённую связь.

- Древний Рим

Самый ранний известный способ военного применения криптографии принадлежит Юлию Цезарю. Около 2000 лет назад Цезарь, будучи полководцем римской армии, решил проблему безопасных коммуникаций со своими полками. Проблема была в том, что гонцы с секретными военными сообщениями часто перехватывались неприятелем. Цезарь разработал шифр подстановки, в котором заменял одни буквы другими. Только тот, кто знал таблицу подстановки, мог расшифровать секретное сообщение. Теперь, даже если гонец попадёт в руки врага, шифровки не будут рассекречены. Это дало римлянам огромное преимущество в войне.

Цезарь обычно просто сдвигал буквы на некое определённое число (рисунок 3). Это число было шифровальным ключом для его алгоритма. Случайный порядок замены символов обеспечивает гораздо лучшую безопасность благодаря большему количеству возможных таблиц замены.

Второй период (хронологические рамки — с IX века на Ближнем Востоке (Ал-Кинди) и с XV века в Европе (Леон Баттиста Альберти) — до начала XX века) ознаменовался введением в обиход полиалфавитных шифров. Леон Батиста Альберти изобрёл систему шифрования на основе шифровального диска. Это было механическое устройство со скользящими дисками, которые допускали много разных методов подстановки символов (рисунок 4).

Третий период (с начала и до середины XX века) характеризуется внедрением электромеханических устройств в работу шифровальщиков (рисунок 5). При этом продолжалось использование полиалфавитных шифров.

Четвёртый период — с середины до 70-х годов XX века — период перехода к математической криптографии. В работе Шеннона появляются строгие математические определения количества информации, передачи данных, энтропии, функций шифрования. Обязательным этапом создания шифра считается изучение его уязвимости для различных известных атак — линейного и дифференциального криптоанализа. Однако до 1975 года криптография оставалась «классической» или же, более корректно, криптографией с секретным ключом.

Современный период развития криптографии (с конца 1970-х годов по настоящее время) отличается зарождением и развитием нового направления — криптография с открытым ключом. Её появление знаменуется не только новыми техническими возможностями, но и сравнительно широким распространением криптографии для использования частными лицами. Правовое регулирование использования криптографии частными лицами в разных странах сильно различается — от разрешения до полного запрета.

1.3. Методы шифрования

В настоящий момент существует огромное количество методов шифрования. Главным образом эти методы подразделяются в зависимости от структуры используемых ключей: на симметричные методы и асимметричные методы. Кроме того, методы шифрования могут обладать различной криптостойкостью и по-разному обрабатывать входные данные — блочные шифры и поточные шифры. Всеми этими методами, их созданием и анализом занимается наука криптография.

- **Классическое шифрование** имеет ограниченное количество ключей.
- **Симметричное шифрование** использует один и тот же ключ (элемент "ключик") и для зашифровывания, и для расшифровывания.

- **Асимметричное шифрование** использует два разных ключа: один для зашифровывания (который также называется открытым), другой для расшифровывания (называется закрытым).
- **Хеширование** не изменяет количество ключей. Оно преобразует входные данные (например, строки или целые числа) в фиксированную длину хеш-кода, который служит в качестве уникальной идентификации входных данных.
- **Цифровая подпись** использует криптографическую ключевую пару, которая состоит из приватного и публичного ключей.

Самое распространённое – **симметричное и ассиметричное шифрование.**

Симметричное шифрование

В симметричных криптосистемах для шифрования и расшифровывания используется один и тот же ключ. Отсюда название — *симметричные*. Алгоритм и ключ выбирается заранее и известен обеим сторонам (рисунок 1). Сохранение ключа в секретности является важной задачей для установления и поддержки защищённого канала связи. В связи с этим, возникает проблема начальной передачи ключа (синхронизации ключей). Кроме того, существуют методы криптоатак, позволяющие так или иначе дешифровать информацию не имея ключа или же с помощью его перехвата на этапе согласования. В целом эти моменты являются проблемой криптостойкости конкретного алгоритма шифрования и являются аргументом при выборе конкретного алгоритма.

Преимущества:

— Очень быстро;

— Простота реализации;

— Не требует больших вычислительных ресурсов.

Недостатки:

— Ключ должен быть передан тайно, иначе любой, кто знает ключ, сможет прочитать зашифрованные данные;

— Не обеспечивает аутентификацию и целостность информации.

Ассиметричное шифрование

В системах с открытым ключом используются два ключа — открытый и закрытый, связанные определённым математическим образом друг с другом. Открытый ключ передаётся по открытому (то есть незащищённому, доступному для наблюдения) каналу и используется для шифрования сообщения и для проверки ЭЦП. Для расшифровки сообщения и для генерации ЭЦП используется секретный ключ (рисунок 2).

Данная схема решает проблему симметричных схем, связанную с начальной передачей ключа другой стороне. Если в симметричных схемах злоумышленник перехватит ключ, то он сможет как «слушать», так и вносить правки в передаваемую информацию. В асимметричных системах другой стороне передается открытый ключ, который позволяет шифровать, но не расшифровывать информацию. Таким образом решается проблема симметричных систем, связанная с синхронизацией ключей.

Преимущества:

— Использует два ключа: открытый и закрытый;

— Открытый ключ может быть передан открытым способом, что обеспечивает безопасность передачи данных;

— Позволяет достичь целостности, аутентификации и конфиденциальности данных.

Недостатки:

— Замедленное выполнение в сравнении с симметричным шифрованием;

— Более сложная реализация.

1.4. Управление ключами

Как было сказано ранее, при шифровании очень важно правильно содержать и распространять ключи между собеседниками, так как это является наиболее уязвимым местом любой криптосистемы. Если вы с собеседником обмениваетесь информацией посредством идеальной шифрующей системы, то

всегда существует возможность найти дефект не в используемой системе, а в тех, кто её использует. Можно выкрасть ключи у доверенного лица или подкупить его, и зачастую это оказывается гораздо дешевле, чем взламывание шифра. Поэтому процесс, содержанием которого является *составление* и *распределение* ключей между пользователями, играет важнейшую роль в криптографии как основа для обеспечения конфиденциальности обмена информацией.

Цели управления ключами

- Сохранение конфиденциальности закрытых ключей и передаваемой информации.
- Обеспечение надёжности сгенерированных ключей.
- Предотвращение несанкционированного использования закрытых или открытых ключей, например использование ключа, срок действия которого истек.

Вывод по Главе 1

В связи с увеличением информации, намного чаще стало использоваться шифрование. Шифрование данных - это процесс преобразования информации в такой вид, который не может быть прочитан или понятен без знания определенного ключа или алгоритма расшифровки. В данной главе мы изучили историю шифрования данных, существующие методы шифрования, выявили их преимущества и недостатки, узнали о необходимости управления ключами в шифровании.

Глава 2. Практическая часть.
Разработка программы шифрования.

2.1. Подготовка к разработке программы шифрования

Для разработки программы шифрования я выбираю шифр Цезаря и язык программирования Python, так как он обладает богатыми возможностями для работы с данными.

Программа работает следующим образом: пользователь вводит в строку «сдвиг» любое число, затем вводит какое-то сообщение в строку «введите текст» и нажимает кнопку «Зашифровать», после чего получает зашифрованный текст. Для дешифровки информации необходимо перейти во вкладку «Дешифрование», ввести тот же сдвиг, который был при шифровании информации и вставить зашифрованный текст. После нажатия кнопки «Дешифровать» пользователь получает первоначальное сообщение.

2.2. Разработка программы шифрования

В своём коде я использовал функции шифрования и дешифрования информации, эта часть кода выглядит следующим образом:

```
# функция для шифрования текста
def encrypt_text():
    shift = int(shift_entry.get())
    text = input_text.get("1.0", "end-1c")  # Получаем введенный текст
    encrypted_text = ""
    for char in text:
        if char.isalpha():
            if char.isupper():
                encrypted_text += chr((ord(char) - 1040 + shift) % 32 + 1040)  # для больших букв
            else:
                encrypted_text += chr((ord(char) - 1072 + shift) % 32 + 1072)  # для маленьких букв
        else:
```

```
            encrypted_text += char  # оставляем неизменными символы, которые не являются буквами
    output_text.delete("1.0", tk.END)
    output_text.insert(tk.END, encrypted_text)


# функция для дешифрования текста
def encrypt_text2():
    shift2 = -int(shift_entry2.get())
    text = input_text2.get("1.0", "end-1c")  # Получаем введенный текст
    encrypted_text2 = ""
    for char in text:
        if char.isalpha():
            if char.isupper():
                encrypted_text2 += chr((ord(char) - 1040 + shift2) % 32 + 1040)  # для больших букв
            else:
                encrypted_text2 += chr((ord(char) - 1072 + shift2) % 32 + 1072)  # для мал букв
        else:
            encrypted_text2 += char
    output_text2.delete("1.0", tk.END)
    output_text2.insert(tk.END, encrypted_text2)
```

2.3. Создание графического интерфейса приложения к программе

Кроме того, я создал графический интерфейс своего приложения и проработал функцию каждой кнопки:

```
# создаем графический интерфейс
window = root = tk.Tk()
root.title("Шифрование данных")
root.geometry("1000x600")
#добавление вкладок для нескольких шифрований
tab_control = ttk.Notebook(root)
tab1 = ttk.Frame(tab_control)
tab_control.add(tab1, text='Шифрование')
```

```
lbl1 = Label(tab1, text='Вкладка 1')
lbl1.pack(expand=0)
tab_control.pack(expand=0, fill='both')
#для дешифр
tab2 = ttk.Frame(tab_control)
tab_control.add(tab2, text='Дешифрование')
lbl2 = Label(tab2, text='Вкладка 2')
lbl2.pack(expand=0)
tab_control.pack(expand=0, fill='both')
# метка и поле для ввода сдвига
shift_label = tk.Label(tab1, text="Сдвиг:")
shift_label.pack()
shift_entry = tk.Entry(tab1)
shift_entry.pack()
# метка и поле для ввода текста
input_label = tk.Label(tab1, text="Введите текст на русском:")
input_label.pack()
input_text = tk.Text(tab1, height=10, width=40)
input_text.pack()
# кнопка для шифрования
encrypt_button = tk.Button(tab1, text="Зашифровать", command=encrypt_text)
encrypt_button.pack()
# метка и поле для вывода зашифрованного текста
output_label = tk.Label(tab1, text="Зашифрованный текст:")
output_label.pack()
output_text = tk.Text(tab1, height=10, width=40)
output_text.pack()
# метка и поле для ввода сдвига
shift_label2 = tk.Label(tab2, text="Сдвиг:")
shift_label2.pack()
shift_entry2 = tk.Entry(tab2)
```

```
shift_entry2.pack()
# метка и поле для ввода текста
input_label2 = tk.Label(tab2, text="Введите текст на русском:")
input_label2.pack()
input_text2 = tk.Text(tab2, height=10, width=40)
input_text2.pack()

# кнопка для дешифрования
encrypt_button2 = tk.Button(tab2, text="Дешифровать", command=encrypt_text2)
encrypt_button2.pack()
# метка и поле для вывода дешифрованного текста
output_label2 = tk.Label(tab2, text="Дешифрованный текст:")
output_label2.pack()
output_text2 = tk.Text(tab2, height=10, width=40)
output_text2.pack()
root.mainloop()
```

Вывод по Главе 2

В ходе работы над практической частью был разработан программный код, который позволяет шифровать и расшифровывать введённую информацию. Выполнение практической части индивидуального проекта позволило углубить знания в области криптографии, приобрести практические навыки в реализации алгоритмов шифрования и увидеть их применение для защиты данных.

Заключение

Выполнение проекта по теме "Шифрование данных" позволило углубить знания и понимание в области криптографии. Были изучены различные методы шифрования и их применение в современном мире, а также были приобретены навыки использования языка программирования для создания программ, осуществляющих шифрование и дешифрование данных.

В ходе проекта были проведены эксперименты, протестированы различные алгоритмы и проанализирована их эффективность и надежность, были изучены принципы работы шифров.

Данный проект позволил развить навыки программирования, аналитического мышления и самостоятельной работы.

В целом, выполнение данного проекта дает большой опыт работы с шифрованием данных и расширяет знания в области информационной безопасности. Мы убедились в важности шифрования для защиты конфиденциальной информации и готовы применять полученные навыки и знания в дальнейшем.

Список используемых источников

1. Книга Кольцов Д. М. – «Python. Полное руководство», 2023 г. -…стр.

2. Шифрование — Википедия (wikipedia.org)

3. Криптография и главные способы шифрования информации (proglib.io)

4. Криптоалгоритмы. Классификация с точки зрения количества ключей / Хабр (habr.com)

5. Какие преимущества и недостатки у различных видов шифрования? (qaa-engineer.ru)

Приложение

Рисунок 1. Древний текст с элементами криптографии из гробницы Хнумхотепа II

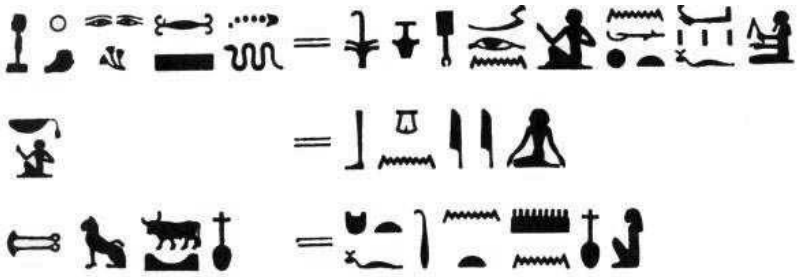

Рисунок 2. Скитала

Рисунок 3. Пример шифра Цезаря

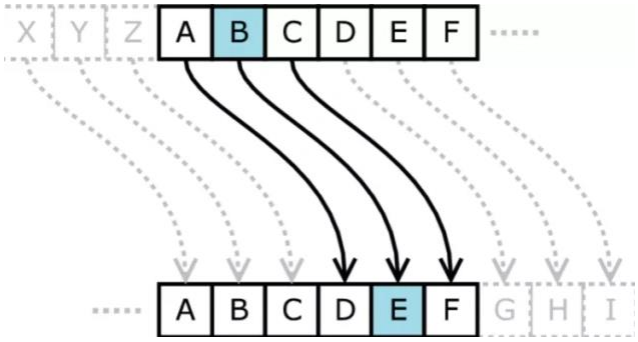

Рисунок 4. Шифровальный диск Леона Батиста Альберти

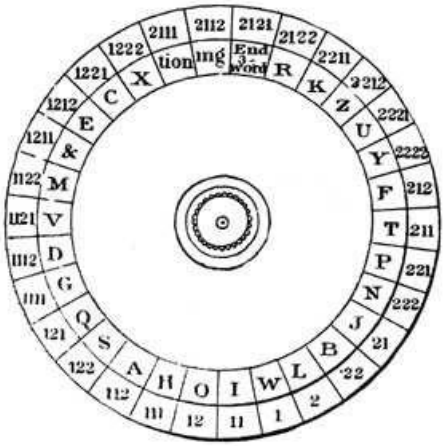

Рисунок 5. Шифровальная машинка

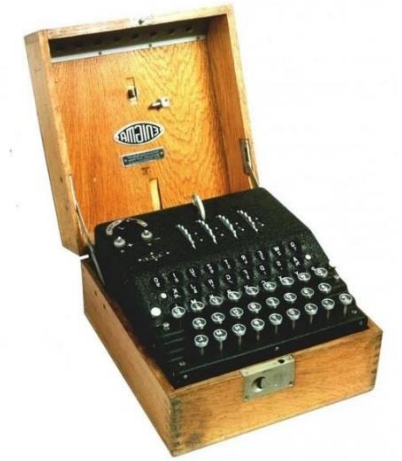

Рисунок 6. Алгоритм симметричного шифрования

Рисунок 2. Алгоритм асимметричного шифрования