**Учреждение образования**

**«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»**

**Кафедра интеллектуальных информационных технологий**

**Лабораторная работа №2 по курсу «СиМЗИИС» на тему:**

**«Простейшие криптографические преобразования»**

**Вариант 3**

Выполнил студент: Исамиддинов Ботир

группы 121731

Проверил(а): Сальников Д.А

**МИНСК**

2023

Вариант 3

**3) Шифр Скитала.**

**Задания**

1) Реализовать в виде программы шифр (зашифрование и расшифрование) в соответствии с вариантом. Язык исходного текста русский или английский по выбору исполнителя.

2) Реализовать в виде программы атаку полным перебором ключа, используя для оценки правильности выбора ключа визуальный метод или исходный текст для автоматического сравнения результата дешифрования.

3) Оценить криптографическую стойкость реализованного шифра.

4) Предложить варианты усложнения шифра. Предложенные варианты оформить в виде алгоритма.

1. Скрип шифровки методом Скитала.
2. def encrypt(text, key):  
    result = ''  
    for char in text:  
    if char.isalpha():  
    shift = ord('a') if char.islower() else ord('A')  
    result += chr((ord(char) - shift + key) % 26 + shift)  
    else:  
    result += char  
    return result  
     
   # def decrypt(text, key):  
   # return encrypt(text, -key)  
     
   # Пользовательский ввод на английском  
   plaintext = input("Enter the text to encrypt: ")  
   key = int(input("Enter the key (integer): "))  
     
   encrypted\_text = encrypt(plaintext, key)  
   # decrypted\_text = decrypt(encrypted\_text, key)  
     
   print(f"Original text: {plaintext}")  
   print(f"Encrypted text: {encrypted\_text}")  
   # print(f"Decrypted text: {decrypted\_text}")
3. **encrypt(text, key)** - это функция, которая принимает два аргумента: **text** (текст для шифрования) и **key** (целое число, на которое будет сдвигаться каждая буква). Она возвращает зашифрованный текст.
4. **result = ''** - создается пустая строка **result**, в которой будет накапливаться зашифрованный текст.
5. **for char in text:** - начинается цикл, в котором перебираются все символы в исходном тексте.
6. **if char.isalpha():** - проверяется, является ли текущий символ буквой.
7. **shift = ord('a') if char.islower() else ord('A')** - определяется смещение (**shift**) в алфавите. Если текущая буква строчная, то смещение будет соответствовать коду буквы 'a' (97 в ASCII), если заглавная - то коду 'A' (65 в ASCII).
8. **result += chr((ord(char) - shift + key) % 26 + shift)** - символ шифруется следующим образом:
   * **ord(char)** - получаем числовой код текущего символа.
   * **(ord(char) - shift + key) % 26** - производится арифметическое действие, которое возвращает новый код символа с учетом сдвига. **% 26** обеспечивает зацикливание в пределах алфавита.
   * **chr(...)** - новый код преобразуется обратно в символ.
9. **else:** - если символ не является буквой, он добавляется к **result** без изменений.
10. В конце функция возвращает зашифрованный текст в виде строки.
11. Ввод пользовательского текста и ключа осуществляется с помощью **input**.
12. **encrypted\_text = encrypt(plaintext, key)** - вызывается функция **encrypt** с пользовательским вводом и результат сохраняется в переменной **encrypted\_text**.
13. **print(f"Original text: {plaintext}")** и **print(f"Encrypted text: {encrypted\_text}")** - выводят оригинальный и зашифрованный тексты соответственно.

Комментированный код **decrypt** в настоящий момент закомментирован, но он предполагает, что можно расшифровать текст, применяя обратное действие к функции **encrypt**. Например, если зашифровать текст ключом 3, то его можно расшифровать, применив тот же ключ -3 (так как -3 % 26 = 23, что равно 23 в модульной арифметике по модулю 26).

--------------------------------------------------------------------------------------------------

**2.** Реализовать в виде программы атаку полным перебором ключа, используя для оценки правильности выбора ключа визуальный метод или исходный текст для автоматического сравнения результата дешифрования.

def decript(text, key, alphabet):  
 result = ''  
 for char in text:  
 if char.isalpha():  
 shift = ord('a') if char.islower() else ord('A')  
 result += chr((ord(char) - shift + key) % 26 + shift)  
 else:  
 result += char  
 return result  
  
def brute\_force\_attack(ciphertext, alphabet, num\_keys):  
 for key in range(1, num\_keys + 1): # Полный перебор ключей  
 decrypted\_text = decript(ciphertext, -key, alphabet)  
 print(f"При ключе {key}: {decrypted\_text}")  
  
# Пользовательский ввод  
ciphertext = input("Enter the text to decrypt: ")  
num\_keys = int(input("Enter the key (integer): "))  
  
# Пример использования с английским алфавитом  
english\_alphabet = ('a', 'A')  
brute\_force\_attack(ciphertext, english\_alphabet, num\_keys)

Этот код дополняет предыдущий и включает две новые функции: **decrypt** и **brute\_force\_attack**, а также основной блок кода.

1. **def decrypt(text, key, alphabet)** - это функция расшифровки текста. Она принимает три аргумента: **text** (текст для расшифровки), **key** (целое число, на которое был зашифрован текст) и **alphabet** (кортеж, содержащий два символа - начальные символы нижнего и верхнего регистров алфавита). Функция возвращает расшифрованный текст.
2. **result = ''** - создается пустая строка **result**, в которой будет накапливаться расшифрованный текст.
3. Остальная часть функции аналогична функции **encrypt**, только с обратными действиями: буквы сдвигаются назад в алфавите.
4. **return result** - функция возвращает расшифрованный текст в виде строки.
5. **def brute\_force\_attack(ciphertext, alphabet, num\_keys)** - это функция для "грубой силы", которая пытается все возможные ключи для расшифровки текста.
6. **for key in range(1, num\_keys + 1):** - начинается цикл, в котором перебираются все возможные ключи для расшифровки текста. **num\_keys** - это число, указанное пользователем и предположительно является максимальным значением ключа.
7. **decrypted\_text = decrypt(ciphertext, -key, alphabet)** - вызывается функция **decrypt** с текущим ключом. Обратите внимание, что ключ передается со знаком минус, так как мы хотим осуществить обратное действие по отношению к зашифрованному тексту.
8. **print(f"При ключе {key}: {decrypted\_text}")** - выводится расшифрованный текст с указанием использованного ключа.
9. В блоке пользовательского ввода вводится зашифрованный текст (**ciphertext**) и предполагаемое максимальное значение ключа (**num\_keys**).
10. **english\_alphabet = ('a', 'A')** - определяется английский алфавит, где первый элемент кортежа (**'a'**) соответствует начальной букве строчного алфавита, а второй элемент (**'A'**) - начальной букве заглавного алфавита.
11. **brute\_force\_attack(ciphertext, english\_alphabet, num\_keys)** - вызывается функция **brute\_force\_attack** с введенным пользователем зашифрованным текстом, английским алфавитом и предполагаемым максимальным значением ключа.

Этот код позволяет вам попробовать разные ключи для расшифровки текста и найти правильный ключ, если он был утерян.

--------------------------------------------------------------------------------------------------

**3.** Реализованный шифр Скитала (и его усложненные варианты) не обладает высокой криптографической стойкостью с точки зрения современных стандартов. Он поддается атакам перебора ключа, статистического анализа и другим методам.

Вот несколько причин, почему шифр Скитала не является криптографически стойким:

1. Однократное применение ключа: В шифре Скитала используется один ключ для всех символов. Это делает его уязвимым к атакам перебора.
2. Малое пространство ключей: Шифр Скитала имеет ограниченное количество возможных ключей. Например, в простейшем варианте (сдвиг на одну позицию) всего 26 ключей для английского алфавита.
3. Сохранение статистических характеристик: Шифр Скитала не изменяет частоту встречаемости букв в тексте. Это делает его поддающимся атакам на основе частотного анализа.
4. Отсутствие дополнительных механизмов защиты: В усложненных вариантах шифра Скитала могут быть применены более сложные алгоритмы, но они также остаются уязвимыми к криптографическим атакам.
5. Отсутствие ключевого обмена и аутентификации: В реальных криптографических протоколах требуется обеспечить безопасность ключей и процессы аутентификации.

В общем, шифр Скитала - это простой учебный пример шифра и не предназначен для защиты важной конфиденциальной информации. В реальных приложениях для обеспечения криптографической стойкости применяются более сложные алгоритмы, такие как AES, RSA, и другие.

--------------------------------------------------------------------------------------------------

4. Усложнения могут включать в себя различные комбинации алгоритмов, дополнительные шаги и сложные математические операции. При реализации новых методов следует также учесть возможные уязвимости и поддержать механизмы обратного преобразования (расшифрования).

1. Множественные сдвиги (перестановки): Вместо одного фиксированного сдвига для всех символов, можно использовать список сдвигов. Каждый символ будет сдвигаться на свое собственное значение.
2. Смешанный шифр (комбинирование с другими алгоритмами): Этот метод предполагает применение нескольких различных шифров. Например, можно сначала применить замену символов, а затем сдвиг.
3. Использование блоков и раундов: Здесь текст разбивается на блоки, и к каждому блоку применяется свой собственный ключ. Это усложняет анализ и расшифрование текста.

Все эти методы усложнения шифра Скитала вносят дополнительные элементы и шаги в процесс шифрования, что делает его более сложным для анализа и взлома. Однако, даже с такими улучшениями, важно помнить, что шифр Скитала не предназначен для реальной криптографической защиты и используется скорее в учебных или развлекательных целях.

**Вывод**

Усложнение базового шифра Скитала с помощью вышеперечисленных методов позволяет повысить уровень безопасности, однако даже после усложнения, этот шифр не обладает высокой криптографической стойкостью с точки зрения современных стандартов. Его использование следует ограничивать учебными и развлекательными целями.

1. Необходимость Контроля Ключей: При усложнении шифра Скитала необходимо внимательно управлять ключами. Варианты, такие как множественные сдвиги и комбинированный шифр, требуют точной организации ключей для правильной расшифровки.
2. Уровень Усложнения и Поддержка: При выборе метода усложнения важно учесть, насколько сложно будет поддерживать и администрировать такую систему. Усложненные алгоритмы могут быть более трудными для отладки и модификации.
3. Обучение и Понимание: Важно, чтобы все пользователи и администраторы системы понимали, как работает усложненный шифр. Обучение и понимание ключевых аспектов криптографических методов является ключевым элементом обеспечения безопасности.
4. Регулярное Тестирование и Анализ: После внедрения усложненного шифра, рекомендуется проводить регулярные тесты на безопасность для проверки его криптографической стойкости.
5. Разработка и Тестирование Криптографических Методов: При разработке собственных криптографических методов следует учитывать множество факторов, включая проверку на уязвимости, обеспечение защиты от атак и соответствие современным стандартам безопасности.

Добавление этих аспектов поможет в создании более полного понимания о том, как усложнение шифра Скитала может быть применено в практике и какие аспекты безопасности следует учитывать при его использовании.