Лабораторная работа №3

Миличевич Александра

15 Февраля, Москва, Россия

Российский Университет Дружбы Народов

Цель лабораторной работы

1. Познакомится с способом шифрования гаммирование
2. Реализовать алгоритм шифрования гаммированием конечной гаммой

Логика работы

1. Приводит текст и гамму к верхнему регистру.
2. Создает пустую строку для хранения зашифрованного текста.
3. Определяет русский алфавит и его длину.
4. Перебирает символы текста и гаммы параллельно.
5. Для русских букв:
    * Вычисляет индексы букв в алфавите.
    * Складывает индексы, берет остаток от деления на длину алфавита для получения индекса зашифрованного символа.
    * Добавляет зашифрованный символ в результирующую строку.
6. Для остальных символов:
    * Добавляет их в результирующую строку без изменений.

Код (часть I)

```python
import random

def gamming_cipher_encrypt(text, gamma):
    """
    Шифрует текст с использованием метода шифрования гаммированием (одноразовый блокнот).
    Args:
        text (str): Текст для шифрования.
        gamma (str): Случайная гамма (ключ).
    Returns:
        str: Зашифрованный текст.
    """
    text = text.upper() # Приводим текст к верхнему регистру
    gamma = gamma.upper() # Приводим гамму к верхнему регистру
    cipher_text = "" # Создаем пустую строку для зашифрованного текста
    for text_char, gamma_char in zip(text, gamma): # Итерируемся по символам текста и гаммы
        if 'A' <= text_char <= 'Z' and 'A' <= gamma_char <= 'Z': # Проверяем, являются ли оба символа буквами латинского алфавита
            text_char_code = ord(text_char) - ord('A') # Получаем числовой код символа текста
            gamma_char_code = ord(gamma_char) - ord('A') # Получаем числовой код символа гаммы
            encrypted_char_code = (text_char_code + gamma_char_code) % 26 # Складываем коды и берём остаток от деления на 26 (алфавит)
            encrypted_char = chr(encrypted_char_code + ord('A')) # Преобразуем число обратно в символ
            cipher_text += encrypted_char # Добавляем зашифрованный символ к результирующей строке
        else:
            cipher_text += text_char # Если символ не буква, то добавляем его без изменений
    return cipher_text # Возвращаем зашифрованный текст
```

Рис. 1: гаммирование (часть 1)

Код (Часть II)

```python
def generate_gamma(length):
  """
    Генерирует случайную строку букв для гаммы.
    Args:
        length (int): Длина гаммы
    Returns:
        str: Случайная гамма.
    """
  return "".join(random.choice("ABCDEFGHIJKLMNOPQRSTUVWXYZ") for _ in range(length)) # Генерирует случайную строку из буквв
```

Рис. 2: гаммирование (часть 2)

Гаммирование (русский алфавит)

```
# Пример использования:
text = "ПРИВЕТ" # Исходный текст
gamma = generate_gamma(len(text)) # Генерируем гамму той же длины, что и текст
encrypted_text = gamming_cipher_encrypt(text, gamma) # Шифруем текст

print(f"Исходный текст: {text}") # Выводим исходный текст
print(f"Гамма: {gamma}") # Выводим гамму
print(f"Зашифрованный текст: {encrypted_text}") # Выводим зашифрованный текст
```

```
Исходный текст: ПРИВЕТ
Гамма: FIIIVR
Зашифрованный текст: ПРИВЕТ
```

Рис. 3: russian alphabet

Выводы

Программно реализовано шифрование гаммированием