

Информация

Докладчик

- Надиа Эззакат
- студент группы НПМбд-02-20
- Факультет физико-математических и естественных наук
- Российский университет дружбы народов

Цель лабораторной работы

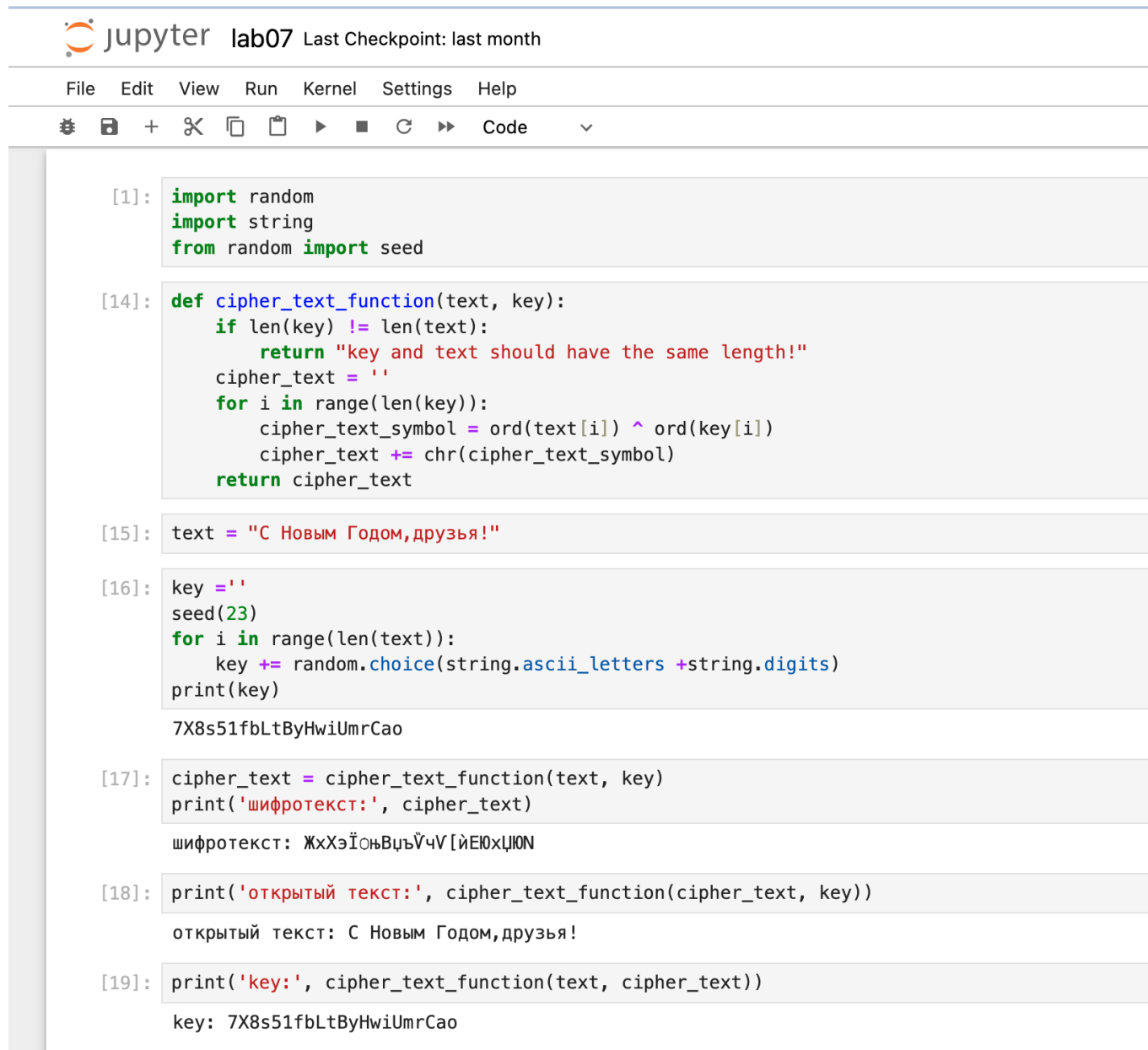
Освоить на практике применение режима однократного гаммирования

Создание программы

Для реализации процесса шифрования, я выполнила несколько важных шагов. В начале, на шаге In[1], была проведена импорт необходимых библиотек, обеспечивая доступ к необходимым функциям и возможностям. Далее, на шаге In[14], была разработана функция, выполняющая операцию сложения по модулю два для двух строк, что играет ключевую роль в процессе шифрования. После этого, на шаге In[15], был подготовлен открытый или исходный текст, который представляет собой исходное сообщение, подлежащее защите.

Для обеспечения безопасности, на шаге In[16], был создан ключ той же длины, что и открытый текст, что является неотъемлемой частью процесса шифрования. Затем, на шаге In[17], был получен шифротекст с использованием заранее разработанной функции, предоставляя, что известны открытый текст и ключ. Этот этап представляет собой конвертацию исходной информации в зашифрованный вид.

На шаге In[18], можно произвести обратную операцию и получить открытый текст с использованием функции, при условии, что известны шифротекст и ключ. Наконец, на шаге In[19], можно получить ключ с использованием функции, при условии, что известны открытый текст и шифротекст, что может быть важным для восстановления ключа в случае необходимости. Эти шаги важны для обеспечения процесса шифрования и расшифрования данных.



The image shows a JupyterLab interface with a code editor. The top bar includes the Jupyter logo, the text 'lab07', and 'Last Checkpoint: last month'. Below this is a menu bar with 'File', 'Edit', 'View', 'Run', 'Kernel', 'Settings', and 'Help'. A toolbar contains icons for file operations and execution. The code editor displays the following Python code:

```
[1]: import random
import string
from random import seed

[14]: def cipher_text_function(text, key):
    if len(key) != len(text):
        return "key and text should have the same length!"
    cipher_text = ''
    for i in range(len(key)):
        cipher_text_symbol = ord(text[i]) ^ ord(key[i])
        cipher_text += chr(cipher_text_symbol)
    return cipher_text

[15]: text = "С Новым Годом, друзья!"

[16]: key = ''
seed(23)
for i in range(len(text)):
    key += random.choice(string.ascii_letters + string.digits)
print(key)

7X8s51fbLtByHwiUmrCao

[17]: cipher_text = cipher_text_function(text, key)
print('шифротекст:', cipher_text)

шифротекст: ЖхХэЇоньВцъŸчV[йЕЮхЦЮN

[18]: print('открытый текст:', cipher_text_function(cipher_text, key))

открытый текст: С Новым Годом, друзья!

[19]: print('key:', cipher_text_function(text, cipher_text))

key: 7X8s51fbLtByHwiUmrCao
```

{#fig:001 width=50%}

Вывод

В ходе выполнения данной лабораторной работы я освоила на практике применение режима однократного гаммирования.