УДК 004.056.55

ШИФРОВАНИЕ/РАСШИФРОВАНИЕ СОДЕРЖИМОГО ФАЙЛОВ С ИСПОЛЬЗОВАНИЕМ ГЕНЕРАТОРА ПСЕВДОСЛУЧАЙНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ БИТ НА ОСНОВЕ AES-128

Киясов Мурат Русланович, студент,
Георгиева Марьяна Альбековна, старший преподаватель кафедры КТИБ,
Закаунов Эльдар Замирович, студент
ФГБОУ ВО «КБГУ», г. Нальчик, Россия

ENCRYPTION/DECRYPTION FILE CONTENTS USING AES-128 BASED PSEUDO-RANDOM BIT SEQUENCE GENERATOR

Kiyasov Murat Ruslanovich, student, myratik0545@gmail.com
Georgieva Maryana Albekovna, Senior Lecturer, maryana.g@list.ru
Zakaunov Eldar Zamirovich, student, zakaunov99@bk.ru
Kabardino-Balkarian State University, Nalchik, Russia

В рамках исследования была разработана и реализована программная система для шифрования и расшифрования содержимого файлов с использованием генератора псевдослучайных последовательностей бит на основе алгоритма AES-128. Эта система была оформлена в виде удобно-

го настольного приложения, написанного на языке Python. Пользователь имеет возможность выбрать файл для шифрования и, при необходимости, выполнить его расшифровку. Результатом работы программы является создание нового файла, содержащего зашифрованные или расшифрованные данные.

Ключевые слова: *шифрование, расшифрование, алгоритм AES-128, генератор псевдослучайных последовательностей бит, блочное шифрование, криптографическая стойкость, безопасность данных.*

As part of the research, a software system was developed and created for encryption and decryption of file contents using a pseudo-random bit sequence generator based on the AES-128 algorithm. This system was designed as a convenient desktop application written in Python. The user has the ability to select a file for encryption and, if necessary, decrypt it. The result of the program is the creation of a new file containing encrypted or decrypted data.

Keywords: *encryption, decryption, AES-128 algorithm, pseudo-random bit sequence generator, block encryption, cryptographic strength, data security.*

Современные методы информационной безопасности требуют эффективного шифрования данных. Использование генератора псевдослучайных последовательностей на основе AES-128 обеспечивает защиту информации при передаче по незащищенным каналам. AES-128 — блочный шифр, обрабатывающий данные фиксированными блоками, что гарантирует высокую криптостойкость [4].

Цель работы — разработка программной системы шифрования/расшифрования файлов с использованием генератора на базе AES-128 для обеспечения конфиденциальности данных при хранении и передаче.

Особенность AES — процедура расширения ключа, генерирующая набор ключей для каждого раунда шифрования, что повышает безопасность процесса [3].

Разработано настольное приложение на Python, реализующее функции шифрования данных. В ходе работы изучены стандарты применения AES-128 для выбора требований к системе [1].

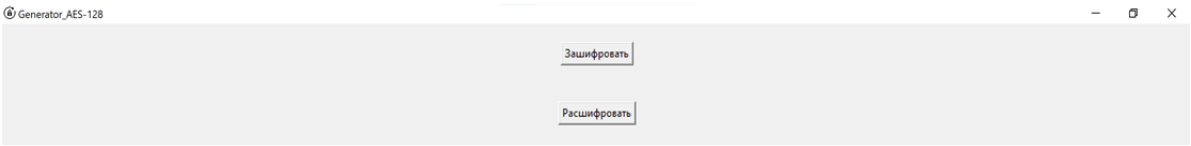

Рисунок 1. – Пользовательский интерфейс

На представленном рисунке изображен пользовательский интерфейс настольного приложения. Пользователю предлагается выбрать одну из двух основных функций: «Зашифровать» и «Расшифровать».

Для демонстрации процесса возьмем тестовый файл с именем «test.txt».

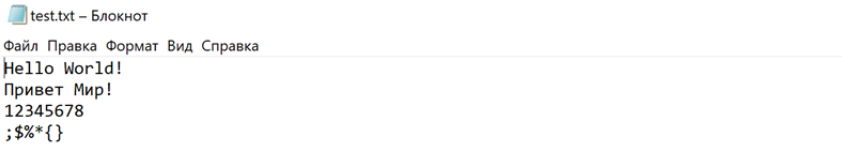

Рисунок 2. – Содержимое файла «test.txt»

На данном изображении представлено содержимое файла «test.txt».

После выбора файла для шифрования, программа обрабатывает данные и предлагает пользователю указать место сохранения, название и формат файла, который будет содержать зашифрованные данные.

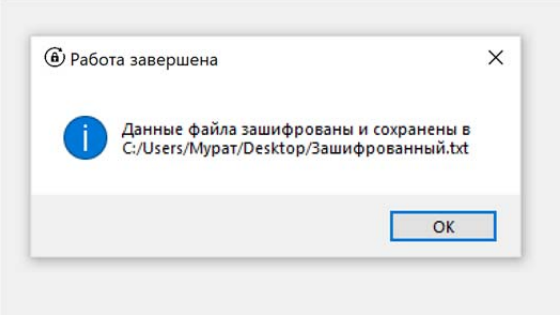

Рисунок 3. – Завершение работы шифрования

На следующем изображении представлено уведомление о завершении шифрования.

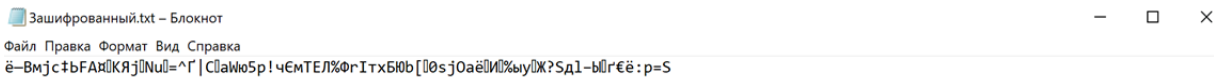

Рисунок 4. – Содержимое файла «Зашифрованный.txt»

Здесь показано содержимое файла, содержащего зашифрованные данные из «test.txt».

После шифрования файл можно безопасно передать по каналам связи. Получатель сможет расшифровать данные, используя тот же программный продукт, который был применен на этапе шифрования.

Для расшифрования пользователь выбирает файл, содержащий зашифрованные данные. Программа обрабатывает эти данные и предлагает указать место сохранения, название и формат файла для расшифрованных данных.

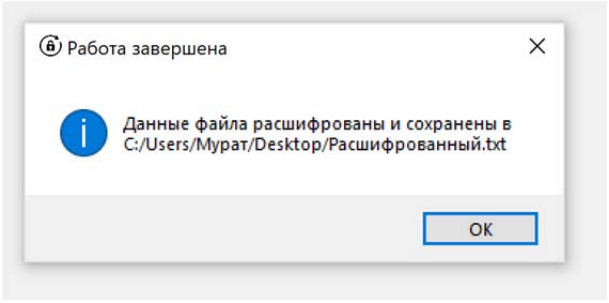

Рисунок 5. – Завершение работы расшифрования

На данном изображении показано сообщение о завершении процесса расшифрования.

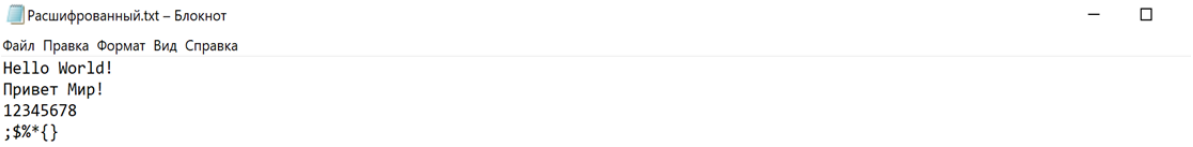

Рисунок 6. – Содержимое файла «Расшифрованный.txt»

На последнем изображении показано содержимое файла «Расшифрованный.txt», полученного после расшифровки файла «Зашифрованный.txt».

Листинг 2 – Шифрование файла

```
def encrypt_file():
file_path = filedialog.askopenfilename(title="Выбор файла")
if file_path:
original_extension = os.path.splitext(file_path)[1]
with open(file_path, "rb") as f:
input_data = f.read()
key = generate_key()
```

```
nonce = secrets.token_bytes(8)
ctr = Counter.new(64, prefix=nonce, initial_value=0)
cipher = AES.new(key, AES.MODE_CTR, counter=ctr)
encrypted_data = cipher.encrypt(input_data)
output_encrypted_file_name = filedialog.asksaveasfilename(
defaultextension=original_extension,
filetypes=[(f"Файл", f"*{original_extension}"), ("All Files", "*.*")],
title="Сохранение файла, содержащего зашифрованные данные"
      )
if output_encrypted_file_name:
with open(output_encrypted_file_name, "wb") as f:
f.write(key + nonce + encrypted_data)
messagebox.showinfo("Работа завершена", f"Данные сохранены в {output_encrypted_file_name}")
```

Листинг 3 – Расшифрование файла

```
def decrypt_file():
file_path = filedialog.askopenfilename(title="Выбор зашифрованного файла")
if file_path:
original_extension = os.path.splitext(file_path)[1]
with open(file_path, "rb") as f:
key = f.read(16)
nonce = f.read(8)
ciphertext = f.read()
ctr = Counter.new(64, prefix=nonce, initial_value=0)
cipher = AES.new(key, AES.MODE_CTR, counter=ctr)
decrypted_data = cipher.decrypt(ciphertext)
output_decrypted_file_name = filedialog.asksaveasfilename(
defaultextension=original_extension,
filetypes=[(f"Файл", f"*{original_extension}"), ("All Files", "*.*")],
title="Сохранение файла, содержащего расшифрованные данные"
      )
if output_decrypted_file_name:
with open(output_decrypted_file_name, "wb") as f:
f.write(decrypted_data)
messagebox.showinfo("Работа завершена", f"Данные сохранены в {output_decrypted_file_name}")
```

Процесс шифрования начинается с выбора файла через диалоговое окно. После этого генерируется 16-байтный криптографический ключ и 8-байтный nonce, который обеспечивает уникальность шифрования для каждого сеанса. Счетчик CTR инициализируется с использованием nonce, что гарантирует неповторимость обработки блоков данных [5]. Далее данные шифруются алгоритмом AES-128 в режиме CTR, где применяется процедура расширения ключа, повышающая криптостойкость за счет генерации раундовых ключей [2]. Результат (ключ, nonce и зашифрованные данные) сохраняется в файл, выбранный пользователем через диалоговое окно.

При расшифровании из файла последовательно извлекаются ключ (первые 16 байт) и nonce (следующие 8 байт). На основе nonce восстанавливается счетчик CTR, после чего данные декодируются с использованием того же ключа. Это обеспечивает точное восстановление исходной информации.

Интерфейс приложения адаптируется под разрешение экрана пользователя, а кнопки «Зашифровать» и «Расшифровать» размещены с отступами (pady=20), что улучшает навигацию. Для идентификации программы используется иконка «Generaton AES-128.ico».

Реализованная система демонстрирует, что AES-128 в режиме CTR обеспечивает безопасность за счет уникальности шифрования блоков (благодаря счетчику и nonce [5]), процедуры расширения ключа [2], а также устойчивости к атакам на основе анализа шифртекста [4]. Соответствие стандартам криптографической защиты [1, 3] подтверждает надежность решения для хранения и передачи конфиденциальных данных.

Список использованных источников

1. Иванов М.А., Чугунков И.В. Теория, применение и оценка качества генераторов псевдослучайных последовательностей. – Москва: КУДИЦ-ОБРАЗ, 2003. – С. 5–55 с.

2. Габидулин Э.М., Кшевецкий А.С., Колыбельников А.И. Защита информации // Учебное пособие. – Москва, 2011. – С. 46–72.

3. Панасенко С. Алгоритмы шифрования // Специальный справочник. – Санкт-Петербург, 2009. – С. 84–97.

4. Рацеев, С.М. Математические методы защиты информации. – Ульяновск, 2018. – С. 396–404.

5. Режимы работы блочного шифра // Хабр, 2024. URL: https://habr.com/ru/companies/otus/articles/812181/ (дата обращения: 17.07.2025).