ИССЛЕДОВАНИЕ И РЕАЛИЗАЦИЯ АЛГОРИТМА RIJNDAEL

***Мурат Бердимуратов*[1],**
***Дилара Баллиева*[2]**
Мурат Бердимуратов -кандидат физико-математических наук, доцент кафедры Информационной системы и технологии;
Дилара Баллиева -магистрант кафедры Информационной системы и технологии, Каракалпакский государственный университет имени Бердаха;

Аннотация. *В статье рассматриваются теоретические основы и практическая реализация алгоритма Rijndael, который лежит в основе международного стандарта шифрования AES. Подробно описаны ключевые этапы работы алгоритма: генерация раундовых ключей, операции SubBytes, ShiftRows, MixColumns и AddRoundKey, а также обратные преобразования для дешифрования. Представлен пример программной реализации на языке Python с использованием библиотеки Crypto.Cipher, демонстрирующий процесс шифрования и расшифрования текстовых сообщений. Сделан вывод о значимости Rijndael в современных системах защиты информации и необходимости дальнейших исследований в области повышения устойчивости к новым видам атак.*

Ключевое слово. *Rijndael, AES (Advanced Encryption Standard), симметричное шифрование, блочный шифр, безопасность данных, шифрование и дешифрование, генерация ключей, Python-реализация*

Введение. Алгоритм Rijndael, разработанный бельгийскими криптографами Винсентом Рэйменом и Йоаном Дайменом, является основой стандарта шифрования AES (Advanced Encryption Standard), принятого правительством США в 2001 году. Благодаря высокой скорости, эффективности и уровню безопасности, алгоритм широко применяется в современных системах защиты данных [1]-[3].

Основные принципы алгоритма Rijndael. Алгоритм Rijndael является симметричным блочным шифром, который использует ключи длиной 128, 192 или 256 бит и обрабатывает данные блоками по 128 бит. Процесс шифрования включает несколько раундов (10, 12 или 14), состоящих из четырёх основных операций [4]:

1. **SubBytes (замена байтов)** – нелинейное преобразование каждого байта блока по специальной S-блок таблице.
2. **ShiftRows (сдвиг строк)** – циклическое смещение строк блока.

3. **MixColumns (перемешивание столбцов)** – линейная операция перемешивания байтов внутри каждого столбца блока.
4. **AddRoundKey (сложение с раундовым ключом)** – побитовая операция XOR блока данных с ключом раунда.

Реализация алгоритма Rijndael. Реализация алгоритма Rijndael требует четкого понимания его математической структуры и этапов обработки данных. Основные шаги реализации:

1. Генерация ключей

Создание набора раундовых ключей из исходного ключа осуществляется через процедуру KeyExpansion, использующую операции SubBytes, циклические сдвиги и константы Rcon.

2. Шифрование

Алгоритм начинается с операции AddRoundKey, затем следует выполнение заданного количества раундов. В последнем раунде пропускается этап MixColumns.

3. Дешифрование

Процесс дешифрования использует обратные операции (InvSubBytes, InvShiftRows, InvMixColumns), но в обратном порядке.

Практическое применение

Алгоритм Rijndael используется в:

- Защите конфиденциальных данных.
- Протоколах передачи данных (например, TLS).
- Программном обеспечении для шифрования файлов и дисков.

Конкретный пример реализации на Python

Предположим, у нас есть текстовое сообщение, которое необходимо надежно передать:

```
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes
# Генерация случайного ключа
def generate_key():
    return get_random_bytes(16)  # 128 бит
# Шифрование текста
def encrypt_message(message, key):
    cipher = AES.new(key, AES.MODE_EAX)
    ciphertext, tag = cipher.encrypt_and_digest(message.encode('utf-8'))
    return cipher.nonce, ciphertext, tag
# Дешифрование текста
def decrypt_message(nonce, ciphertext, tag, key):
    cipher = AES.new(key, AES.MODE_EAX, nonce)
```

```
    decrypted_message = cipher.decrypt_and_verify(ciphertext, tag)
    return decrypted_message.decode('utf-8')
# Пример задачи
message = 'Пароль: SuperSecure123'
key = generate_key()
nonce, ciphertext, tag = encrypt_message(message, key)
print(f'Зашифрованное сообщение: {ciphertext}')
# Получение исходного сообщения
original_message = decrypt_message(nonce, ciphertext, tag, key)
print(f'Расшифрованное сообщение: {original_message}')
```

Преимущества и ограничения

Преимущества:

- Высокая скорость обработки.
- Гибкость выбора размера ключа.
- Надёжность и устойчивость к криптографическим атакам.

Ограничения:

- Симметричность (требует надёжного распространения ключей).
- Потенциальная уязвимость к атакам по сторонним каналам (side-channel attacks).

Заключение. Алгоритм Rijndael благодаря своей эффективности и доказанной стойкости остаётся важнейшим компонентом криптографических решений, обеспечивая защиту информации во множестве приложений и систем по всему миру. Исследования алгоритма и постоянное совершенствование реализации остаются актуальными задачами для специалистов в области информационной безопасности.

Список литературы

1. Daemen J., Rijmen V. The Design of Rijndael: AES – The Advanced Encryption Standard. Springer, 2002.
2. Schneier B. Applied Cryptography: Protocols, Algorithms, and Source Code in C. Wiley, 1996.
3. Stallings W. Cryptography and Network Security: Principles and Practice. Pearson Education, 2017.
4. Menezes A., van Oorschot P., Vanstone S. Handbook of Applied Cryptography. CRC Press, 1996.