

Разработка режимов шифрования для AES на python

Оглавление

Разработка режимов шифрования для AES на python.....	1
1.1. AES режимы шифрования.....	1
1.2. Реализация алгоритма.....	3
Алгоритм ECB	3
Алгоритм CBC	3
Алгоритм PCBC	4
Алгоритм CFB	4
Алгоритм OFB	5
Алгоритм CTR	6
Алгоритм GCM	7
Алгоритм EAX	8
1.3. Задание на работу.....	8
1.4. Тестирование NIST.....	9
1.5. Варианты заданий на лабораторную работу, определяются по номеру в группе.....	9
1.6. Загрузка результата выполнения лабораторной работы.....	10

1.1. AES режимы шифрования

AES (Advanced Encryption Standard) – симметричный алгоритм блочного шифрования, с размерностью блока 128 бит и ключом 128/192/256 бит.

Режим шифрования – метод применения блочного блочного шифра, который позволяет привести последовательность открытых блоков данных к зашифрованному виду. Режимы шифрования применяются для внесения изменения в стандартный процесс, что позволяет достигать уникальности каждого шифруемого блока вне зависимости от данных, а также скрывая их внутреннюю структуру.

ECB (Electronic code book) – режим при котором незашифрованное сообщение разделяется на блоки фиксированной длины, после чего каждый блок шифруется независимо от остальных с использованием одного и того же ключа.

CBC (Cipher Block Chaining) – режим сцепления блоков симметричного блочного шифра с использованием механизма обратной связи, применяемый для AES. Где, все блоки открытого текста за исключением первого складываются побитово по модулю 2 с результатом предыдущего шифрования.

PCBC (Propagating Cipher Block Chaining) – режим распространяющегося сцепления блоков шифра. Отличается от предыдущего тем, что каждый блок открытого текста перед шифрованием XOR-руется не с предыдущим блоком, а с результатом XOR между предыдущим блоком открытого текста и предыдущим блоком шифротекста. В результате каждый последующий блок шифротекста зависит от предыдущего.

CFB (Cipher FeedBack) – режим обратной связи по шифротексту, при котором реализуется возможность шифрования отдельных байтов, а не только целых блоков. При вызове алгоритма происходит сложение по модулю 2 каждого блока открытого текста с блоком, зашифрованным на предыдущем шаге.

OFB (Output FeedBack) – режим обратной связи по выходу. Превращает блочный шифр в шифр синхронного потока, при котором генерируются ключевые блоки, получаемые путём сложения с блоками открытого текста.

CTR (Counter Mode) – режим счётчика. Создаётся специальный счётчик, значение которого предполагается возвращать на вход блочному алгоритму шифрования, накопившееся через некоторое время после запуска. Операция XOR применяется к последовательности блоков, что делает шифр потоковым. Блоки исходного текста и шифротекста идентичны по размеру.

GCM (Galois Counter Mode) – режим счётчика с аутентификацией Галуа и симметричным ключом. На вход алгоритма передаются ключ, открытый текст и некоторые связанные данные. Открытый текст шифруется с помощью ключа, вычисляется специальный Тег аутентификации, используя зашифрованное сообщение и связанные данные. С помощью ключа, тега и связанных данных может расшифровать сообщение и проверить его на изменение.

EAX/AEAD (Encrypt-then-Authenticate-then-Translate) – двухпроходный алгоритм с ассоциированными данными AEAD, где первый проход выполняется для достижения конфиденциальности, а второй для достижения аутентификации каждого блока. Для шифрования используется режим CTR, а для аутентификации OMAC (One Key MAC) над каждым блоком применяется метода композиции EAX.

1.2. Реализация алгоритма

Необходимо написать программу, реализующую режим шифрования/дешифрования CBC для AES на python, позволяющую зашифровать и расшифровать передаваемое сообщение соответствующим образом.

PyCryptodome – библиотека для python, предоставляющая широкий спектр криптографических функций и алгоритмов, включая шифрование, дешифрование, хеширования и проверки подписей, в т.ч. AES.

Для режимов необходимо вручную применять процесс шифрования/дешифрования AES (SubBytes, ShiftRows, MixColumns, ...), для первого алгоритма необходимые шаги указаны, для последующих опущены для краткости.

Из данной библиотеки понадобятся AES

```
from Crypto.Cipher import AES
```

Алгоритм ЕВС

Шифрование:

- Необходимо сгенерировать раундовые ключи
- Добавить паддинг, дополняя данные до кратного 16 размера, например PKCS7
- Выполнить разделение на блоки по 16 байт
- Зашифровать каждый блок средствами AES (SubBytes, ShiftRows, MixColumns, AddRoundKey)

Дешифрование:

- Подготовка раундовых ключей, но в обратном порядке
- Разбить шифротекст на блоки по 16 байт
- Дешифрование каждого блока по AES (InvShiftRows, InvSubBytes, AddRoundKey, InvMixColumns)
- Удалить добавленные паддингом байты

Алгоритм CBC

Шифрование:

- Необходимо сгенерировать случайный вектор инициализации IV длиной 16 байт

- Разбить текст сообщения на блоки по 16 байт
- Зашифровать каждый блок используя XOR к текущему блоку
- Применить AES шифрование
- Соединить полученные шифротексты вместе, поместив вектор инициализации в начало

Дешифрование:

- Выбрать вектор инициализации из начала шифротекста
- Обрезать шифротекст на длину вектора инициализации и разбить на блоки соответствующего размера
- Расшифровать блоки с помощью XOR к предыдущему
- Соединить блоки в исходный текст

Функция шифрования/дешифрования `encrypt(text, key, IV)` реализуется вручную, без встроенной из библиотеки. В качестве сообщения для шифрования передается ФИО и email студента.

Алгоритм PCBC

Аналогичным образом для PCBC, с учётом отличия XOR-ирования.

Алгоритм CFB

Шифрование:

- Подготовить ключ соответствующей длины без расширения
- Сгенерировать вектор инициализации 16 байт
- Разбить исходный текст на блоки соответствующего размера
- Паддинг не требуется, т.к. шифр потоковый
- Зашифровать IV с помощью AES
- Выполнить XOR результата с первым блоком открытого текста
- Использовать предыдущий блок шифротекста как IV для следующего
- Выполнить XOR с текущим блоком открытого текста
- Объединить блоки шифротекста и передать IV для расшифровки

Дешифрование:

- Подготовить тот же ключ, что и при шифровании
- Извлечь IV – первые 16 байт шифротекста
- Разделить оставшийся шифротекст на блоки по 16 байт
- Для первого блока

- o Зашифровать IV с AES
 - o Выполнить XOR с первым блоком шифротекста
- Для последующих блоков
 - o Зашифровать предыдущий блок шифротекста
 - o Выполнить XOR с текущим блоком шифротекста
- Объединить все блоки открытого текста

Алгоритм OFB

Шифрование:

- Подготовка ключа AES
- Генерация случайного уникального IV длиной 16 байт
- Разделить данные на блоки по 16 байт, шифр потоковый
- Для первого блока
 - o Зашифровать IV с AES
- Для последующих блоков
 - o Использовать предыдущий ключевой поток как вход для шифрования AES
- Шифрование блоков
 - o Выполнить XOR каждого блока открытого текста с текущим блоком ключевого потока
- Объединить все блоки шифротекста и передать IV

Дешифрование:

- Подготовка ключа, тот же, что и при шифровании
- Извлечение IV из первых 16 байт шифротекста
- Разбиение на блоки по 16 байт
- Создание ключевого потока
- Первый блок
 - o Зашифровать IV с AES
- Последующие блоки
 - o Использовать для шифрования AES предыдущий ключевой поток
- Дешифрование блоков
 - o Выполнить XOR для каждого блока шифротекста с текущим блоком ключевого потока
- Объединить все блоки открытого текста

Алгоритм CTR

Шифрование:

- Подготовка соответствующего исходного ключа AES
- Генерация Nonce/IV
 - Создать уникальное значение счётчика nonce определённой длины до 16(8, 12, ...) байт
 - Оставшуюся часть использовать для инкремента
- Разделение данных на блоки
 - Разбить исходный текст на блоки по 16 байт, потоковый шифр
- Генерация ключевого потока
 - Создать счётчик в hex или bin
 - Зашифровать счётчик с AES
- Шифрование блоков
 - Выполнить XOR каждого блока открытого текста с соответствующим блоком ключевого потока
- Объединить все блоки шифротекста и передать nonce

Дешифрование:

- Подготовка ключа, тот же, что и при шифровании
- Извлечение nonce из начала шифротекста
- Разбить текст на блоки по 16 байт
- Генерация ключевого потока
 - Восстановить счётчик – nonce плюс номер блока
 - Зашифровать счётчик с AES
- Дешифрование блоков
 - Выполнить XOR каждого блока шифротекста с соответствующим блоком ключевого потока
- Объединить блоки исходного текста

Алгоритм GCM

Шифрование:

- Подготовить ключ AES без расширения
- Генерация случайного и уникального nonce длиной менее 16 байт (Nonce не может повторяться для ключа)

- Дополнительные аутентификационные данные (AAD)
 - Выбрать данные для аутентификации, но без шифровки, включаются в Тег
- Шифрование данных
 - Разбить исходный текст на блоки по 16 байт
 - Использовать модифицированный CTR режим. Счётчик – nonce || 0x00000001 с соответствующей длиной, выбранной вами заранее
 - Зашифровать каждый блок с AES и выполнить XOR с открытым текстом, аналогично CTR
- Вычисление аутентификационного тега
 - Создание GHASH – хеш в поле Галуа на основе данных (AAD, Шифротекст, Длины AAD и шифротекста в битах)
 - Зашифровать начальный счётчик nonce || 0x00000000
 - Выполнить XOR с результатом GHASH
- Объединить nonce, шифротекст и tag nonce(bytesize) || ciphertext (N) || tag (bytesize)

Дешифрование:

- Подготовка ключа, тот же, что и при шифровании
- Извлечение ключевых компонентов: nonce, tag, шифротекст
- Проверить тег путём его повторного вычисления на основе AEAD, шифротекста и nonce
- Дешифрование данных
 - Сгенерировать ключевой поток с помощью nonce аналогично CTR
 - Выполнить XOR ключевого потока с шифротекстом для получения открытого текста
- В случае верного тега возвращаются расшифрованные данные, иначе – ошибка аутентификации

Алгоритм EAX

Шифрование:

- Подготовить ключ AES, единый для шифрования и аутентификации
- Сгенерировать Nonce уникальный и случайный выбранной длины
- Определить дополнительные аутентификационные данные для аутентификации, которые не будут зашифрованы

- Шифрование данных
 - Сгенерировать ключевой поток в режиме CTR
 - Выполнить XOR ключевого потока с открытым текстом
 - Вычислить OMAC-тег для nonce, AEAD и шифротекста. Tag – OMAC(nonce || AAD || ciphertext, key)
- Объединить nonce, шифротекст и AEAD

Дешифрование:

- Подготовить ключ, тот же, что и при шифровании
- Извлечь nonce, шифротекст и тег из данных
- Проверка аутентификационного тега
 - Повторить вычисление OMAC тега на основе nonce, AEAD и шифротекста
 - При несовпадении тега возвращать ошибку
- Дешифрование данных
 - Восстановить ключевой поток через AES-CTR
 - Выполнить XOR ключевого потока с шифротекстом
- При верном теге – вернуть данные, иначе вывести ошибку аутентификации

1.3. Задание на работу

- Реализовать режим шифрования поверх библиотечной реализации AES шифрования соответственно варианту, методы из AES используются вручную
- Реализовать ручную соответствующие операции XOR для режимов шифрования
- Выполнить выравнивание данных (Padding) по стандарту PKCS7 при необходимости
- Написать функции encrypt и decrypt, принимающие ключ, вектор инициализации/связанные данные, исходный и зашифрованный текст соответственно выбранному режиму
- Реализовать проверку аутентификации (для соответствующих режимов) и вывод ошибки
- Чтение и запись из/в текстовый файл исходного/зашифрованного сообщения (и метаданных)

- Проверить корректность на тестовых векторах из стандарта NIST

1.4. Тестирование NIST

Провести тестирование разработанного приложения с помощью тестов, предоставленных NIST и проанализировать полученные результаты.

NIST vector AES-ECB/CBC/...-128/192/256 ... [Cryptographic Algorithm Validation Program | CSRC](#)

1.5. Варианты заданий на лабораторную работу, определяются по номеру в группе

1. AES-CBC 128 bit key
2. AES-CBC 192 bit key
3. AES-CBC 256 bit key
4. AES-CFB 128 bit key
5. AES-CFB 192 bit key
6. AES-CFB 256 bit key
7. AES-OFB 128 bit key
8. AES-OFB 192 bit key
9. AES-OFB 256 bit key
10. AES-CTR 128 bit key
11. AES-CTR 192 bit key
12. AES-CTR 256 bit key
13. AES-GCM 128 bit key
14. AES-GCM 192 bit key
15. AES-GCM 256 bit key
16. AES-EAX 128 bit key
17. AES-EAX 192 bit key
18. AES-EAX 256 bit key
19. AES-PCBC 128 bit key
20. AES-PCBC 192 bit key
21. AES-PCBC 256 bit key
22. AES-ECB 128 bit key
23. AES-ECB 192 bit key
24. AES-ECB 256 bit key

1.6. Загрузка результата выполнения лабораторной работы

В качестве результата выполнения лабораторной работы необходимо загрузить в Moodle следующие файлы:

- Код реализованной программы на python
- Текстовый файл с исходным и зашифрованным текстом, а также другими соответствующими варианту данными