1. Расшифровать файл dd1_saes_c_all.bmp, M = [[1, 4], [4, 1]]. Неприводимый многочлен x^4+x+1. Режим ECB. Key = 834.

```
matrix = list([['1', '4'], ['4', '1']])
mod = int('10011', 2)
key = 834

data = read_write_file.read_data_2byte('encrypt_data/dd1_saes_c_all.bmp')

saes_d = em.ecb_d(data, key, mod=mod, matrix=matrix, crypto_mode='saes')
print(saes_d[:4])
read_write_file.write_data_2byte('decrypt_data/dd1_saes_c_all_decrypt.bmp', saes_d)

data = read_write_file.read_data_2byte('decrypt_data/dd1_saes_c_all_decrypt.bmp')
saes_ecb50 = em.ecb_e(data[50:], key, mod=mod, matrix=matrix, crypto_mode='saes')
read_write_file.write_data_2byte('encrypt_data/dd1_saes_c_all_50.bmp', data[:50] + saes_ecb50)
```

Результат:


Оригинал

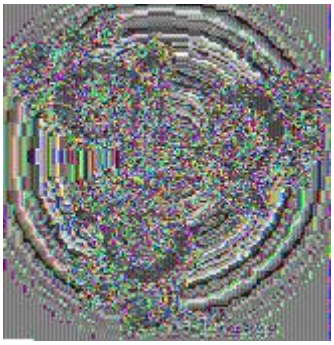

Режим ECB

2. Расшифровать файл im43_saes_c_all.bmp. M = [[b, 4], [e, d]], Неприводимый многочлен = x^4+x+1. Режим ECB. Key = 2318.

```
matrix = list([['b', '4'], ['e', 'd']])
mod = int('10011', 2)
key = 2318

data = read_write_file.read_data_2byte('encrypt_data/im43_saes_c_all.bmp')

saes_d = em.ecb_d(data, key, mod=mod, matrix=matrix, crypto_mode='saes')
print(saes_d[:4])
read_write_file.write_data_2byte('decrypt_data/im43_saes_c_all_decrypt.bmp', saes_d)

data = read_write_file.read_data_2byte('decrypt_data/im43_saes_c_all_decrypt.bmp')
saes_ecb50 = em.ecb_e(data[50:], key, mod=mod, matrix=matrix, crypto_mode='saes')
read_write_file.write_data_2byte('encrypt_data/im43_saes_c_all_50.bmp', data[:50] + saes_ecb50)
```

Результат:


Оригинал

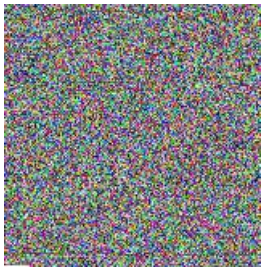

Режим ECB

3. Расшифровать файл dd5_saes_cbc_c_all.bmp. M = [[a, c], [8, 6]], Неприводимый многочлен = x^4+x^3+1. Режим CBC. Key = 1021, iv = 456.

```
matrix = list([['a', 'c'], ['8', '6']])
mod = int('11001', 2)
key = 1021
iv = 456

data = read_write_file.read_data_2byte('encrypt_data/dd5_saes_cbc_c_all.bmp')

saes_d = em.cbc_d(data, key, iv, mod=mod, matrix=matrix, crypto_mode='saes')
read_write_file.write_data_2byte('decrypt_data/dd5_saes_cbc_c_all_decrypt.bmp', saes_d)

data = read_write_file.read_data_2byte('decrypt_data/dd5_saes_cbc_c_all_decrypt.bmp')
saes_cbc50 = em.cbc_e(data[50:], key, iv, mod=mod, matrix=matrix, crypto_mode='saes')
read_write_file.write_data_2byte('encrypt_data/dd5_saes_cbc_c_all_50.bmp', data[:50] + saes_cbc50)
```

Результат:

Оригинал



Режим CBC

4. Расшифровать файл dd8_saes_ofb_c_all.bmp. M = [[5, 3], [2, c]], Неприводимый многочлен = x^4+x^3+1. Режим OFB. Key = 12345, iv = 5171.

```
matrix = list([['5', '3'], ['2', 'c']])
mod = int('11001', 2)
key = 12345
iv = 5171

data = read_write_file.read_data_2byte('encrypt_data/dd8_saes_ofb_c_all.bmp')

saes_d = em.ofb_d(data, key, iv, mod=mod, matrix=matrix, crypto_mode='saes')
read_write_file.write_data_2byte('decrypt_data/dd8_saes_ofb_c_all_decrypt.bmp', saes_d)

data = read_write_file.read_data_2byte('decrypt_data/dd8_saes_ofb_c_all_decrypt.bmp')
saes_ofb50 = em.ofb_e(data[50:], key, iv, mod=mod, matrix=matrix, crypto_mode='saes')
read_write_file.write_data_2byte('encrypt_data/dd8_saes_ofb_c_all_50.bmp', data[:50] + saes_ofb50)
```

Результат:

Оригинал

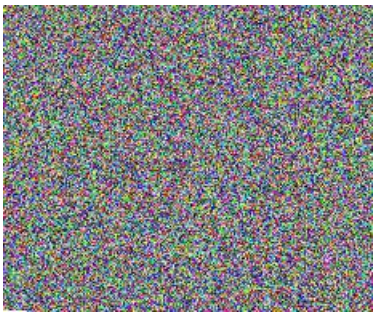
Режим OFB

5. Расшифровать файл dd10_saes_cfb_c_all.bmp. M = [[7, d], [4, 5]], Неприводимый многочлен = x^4+x^3+1. Режим CFB. Key = 24545, iv = 9165.

```
matrix = list([['7', 'd'], ['4', '5']])
mod = int('11001', 2)
key = 24545
iv = 9165

data = read_write_file.read_data_2byte('encrypt_data/dd10_saes_cfb_c_all.bmp')

saes_d = em.cfb_d(data, key, iv, mod=mod, matrix=matrix, crypto_mode='saes')
read_write_file.write_data_2byte('decrypt_data/dd10_saes_cfb_c_all_decrypt.bmp', saes_d)

data = read_write_file.read_data_2byte('decrypt_data/dd10_saes_cfb_c_all_decrypt.bmp')
saes_cfb50 = em.cfb_e(data[50:], key, iv, mod=mod, matrix=matrix, crypto_mode='saes')
read_write_file.write_data_2byte('encrypt_data/dd10_saes_cfb_c_all_50.bmp', data[:50] + saes_cfb50)
```

Результат:

Оригинал

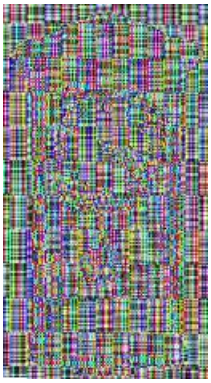


Режим CFB

6. Расшифровать файл dd12_saes_ctr_c_all.bmp. M = [[7, 3], [2, e]], Неприводимый многочлен = x^4+x+1. Режим CFB. Key = 2645, iv = 23184.

```
matrix = list([['7', '3'], ['2', 'e']])
mod = int('10011', 2)
key = 2645
iv = 23184

data = read_write_file.read_data_2byte('encrypt_data/dd12_saes_ctr_c_all.bmp')

saes_d = em.ctr_d(data, key, counter=iv, mod=mod, matrix=matrix, crypto_mode='saes')
read_write_file.write_data_2byte('decrypt_data/dd12_saes_ctr_c_all_decrypt.bmp', saes_d)

data = read_write_file.read_data_2byte('decrypt_data/dd12_saes_ctr_c_all_decrypt.bmp')
saes_ctr50 = em.ctr_e(data[50:], key, counter=iv, mod=mod, matrix=matrix, crypto_mode='saes')
read_write_file.write_data_2byte('encrypt_data/dd12_saes_ctr_c_all_50.bmp', data[:50] + saes_ctr50)
```

Оригинал



Режим CTR

AES

Advanced Encryption Standard, также известный как Rijndael — симметричный алгоритм блочного шифрования (размер блока 128 бит, ключ 128/192/256 бит), принятый в качестве стандарта шифрования правительством по результатам конкурса AES. Этот алгоритм хорошо проанализирован и сейчас широко используется, как это было с его предшественником DES. 26 мая 2002 года AES был объявлен стандартом шифрования. По состоянию на 2009 год AES является одним из самых распространённых алгоритмов симметричного шифрования. Поддержка AES (и только его) введена фирмой Intel в семейство процессоров x86 начиная с Intel Core i7-980X Extreme Edition, а затем на процессорах Sandy Bridge.

Формальное начало процессу разработки нового криптостандарта было положено 2 января 1997 года, когда НИСТ объявил о своем решении разрабатывать AES по причине моральной устаревшести DES и недостаточной эффективности заменившего его алгоритма "Triple DES" (по сути дела, тот же самый шифр, применяемый три раза). Затем, 12 сентября того же года был опубликован официальный призыв ко всем заинтересованным кругам о выдвижении своих алгоритмов в качестве возможных кандидатов. Тогда же были оговорены следующие первичные требования, которым должен удовлетворять AES: это должен быть незасекреченный, открыто опубликованный алгоритм шифрования, бесплатно доступный по всему миру. Спустя некоторое время было уточнено, что AES будет блочным шифром, реализующим криптографию с симметричным ключом, причем алгоритм (как минимум) должен поддерживать 128-битную длину шифруемого блока текста и длины ключей размером 128, 192 и 256 бит. Если же формулировать цель процесса AES совсем кратко, то ее можно свести к таким словам: "новый блочный шифр должен быть более стойким и более эффективным, чем Triple DES".

Критерии оценки алгоритмов-кандидатов были разработаны на основе первичных требований НИСТ и последовавшего затем публичного обсуждения на проведенном в Институте стандартов открытом семинаре 15 апреля 1997 года. Все оценочные критерии были разбиты на три основные категории: стойкость, стоимость, характеристики алгоритма.

Стойкость расценивается как самой важный фактор при оценке кандидатов. Под стойкостью понимаются такие свойства шифра как неподдаваемость алгоритма криптоаналитическому вскрытию, прочность его математического фундамента, равновероятность выхода алгоритма и, наконец, относительная стойкость алгоритма в сравнении с остальными кандидатами.

Стоимость - это вторая важная область оценки, куда включаются и вопросы (аннулирования необходимости) лицензирования, и вычислительная эффективность (скорость) на разнообразных платформах, и требования к памяти.

Характеристики алгоритма и его реализации, такие как гибкость, простота, программная и аппаратная "укладываемость" - это третья важная область оценки кандидатов. Под гибкостью понимается способность алгоритма работать с различными длинами ключей и блоков текста, возможность реализации алгоритма в качестве поточного шифра, алгоритма хеширования, генератора случайных чисел и т.д.

20 августа 1998 года прошла "Первая конференция кандидатов на AES" или AES1, где НИСТ объявил о приеме на конкурс 15 алгоритмов-кандидатов. Во второй раунд прошли только пятеро: MARS, RC6, Rijndael, Serpent, Twofish.

Второй раунд длился до 15 мая 2000 года, в этот период решался вопрос о принятии одного или нескольких алгоритмов как победителей. Однако 2 октября 2000 года было объявлено, что победителем стал алгоритм Rijndael и 26 ноября 2001 года алгоритм был стандартизирован как FIPS 197.

Озвученные на конференции CRYPTO 2011 результаты дополнительного криптоанализа алгоритма блочного шифрования AES (Advanced Encryption Standard) указывают на новый способ атаки, позволяющий в четыре раза сократить трудоёмкость выполнения операций по подбору секретного ключа. Иными словами на деле криптостойкость AES-128 сводится к AES-126, а AES-192 к AES-189, что само по себе остается достаточно внушительным показателем (для взлома AES-128 требуется выполнить 2 в 126 степени операций). Предложенный метод атаки работает со всеми вариантами AES. Возможность совершения атаки указанным методом признали создатели AES, Винсент Рэймен и Йоан Даймен.

На вход алгоритму подается последовательности из 128 бит, ключ шифрования принимает значения из 128, 192 и 256 бит.

В алгоритме AES все байты рассматриваются в качестве элементов конечного поля $GF(2^8)$ с операциями сложения и умножения.

Сложение представляется как сложение по модулю 2, т.е. a + b = a xor b. Со следующими свойствами:

1 + 1 = 0; 1 + 0 = 1; 0 + 0 = 0; a + a = 0; -a = 0 – a = a; a – b = a + (-b) = a + b;

Умножение производится по модулю неприводимого многочлена: $x^8 + x^4 + x^3 + x + 1$.

В начале процесса шифрования входные данные разбиваются на блоки размером 16 байт, которые обычно представляются в виде матрицы состояний 4x4.

Каждый блок шифруется в несколько этапов – раундов. Схема алгоритма шифрования:

1. Расширение ключа
2. Суммирование с основным ключом
3. Пока количество раундов меньше 10, 12 или 14:
   - 3.1. Заменить байты в таблице состояний по таблице замен
   - 3.2. Сделать циклический сдвиг строк матрицы состояний
   - 3.3. Переставить столбцы матрицы состояний
   - 3.4. Суммирование с раундовым ключом
4. На последнем раунде:
   - 4.1. Сделать замену байт матрицы состояний по таблице замен
   - 4.2. Сделать циклический сдвиг строк матрицы состояний
   - 4.3. Суммирование с раундовым ключом

Для процедуры расшифрования нужно проделать шаги в обратном порядке (использовать обратные преобразования для сдвига строк, замены, сдвига столбцов и суммирования раундового ключа)

Источники

[1] http://byrd.narod.ru/aes/aes2.htm
[2] https://xakep.ru/2011/08/18/56520/
[3] FIPS 197 AES
[4] https://habrahabr.ru/post/112733/