ОТЧЁТ О ЛЕТНЕЙ ПРАКТИКЕ

Cipher Mode Picker (HKCERT CTF)

Нецветайлов А. А. Флягин А. И.

«БАЛТИЙСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ ИМЕНИ ИММАНУИЛА КАНТА»

Образовательно-научный кластер «Институт высоких технологий»

Компьютерная безопасность математические методы защитый информации
3 курс

8 июля 2023

Предисловие

Мы выбрали задачу **Cipher Mode Picker (HKCERT CTF)**. В этой задаче используется криптосистема AES-256. В университете мы её не изучали, нам стало интересно разобраться самим и мы посчитали, что необходимо в рассказать об этом алгоритме в презентации-отчёте; поэтому слайды со 2 до 28 включительно это описание криптосистемы. Решение задачи начинается на 29 слайде.

Определения

Byte – целое число от 0 до 255 включительно.

Word – слово, состоящее из 4 Byte.

Nk – число 32-битных слов, составляющих шифроключ. Для AES Nk = 4, 6, или 8.

Nr – число раундов, которое является функцией Nk и Nb. Для AES Nr = 10, 12, 14.

Rcon[] – массив, который состоит из битов 32-разрядного слова и является постоянным для данного раунда.

State – промежуточный результат шифрования, который может быть представлен как прямоугольный массив байтов, имеющий 4 строки и Nb колонок.

Определения

Block – последовательность бит, из которых состоит input, output, State и Round Key. Также под Block можно понимать последовательность байтов.

Cipher Key – секретный криптографический ключ, который используется Key Expansion процедурой, чтобы произвести набор ключей для раундов (Round Keys); может быть представлен как прямоугольный массив байтов, имеющий четыре строки и Nk колонок.

Round Key – Round Keys получаются из Cipher Key использованием процедуры Key Expansion. Они применяются к State при шифровании и расшифровании.

Ciphertext – выходные данные алгоритма шифрования.

Key Expansion – процедура генерации Round Keys из Cipher Key.

Вспомогательные процедуры

AddRoundKey() – трансформация при шифровании и обратном шифровании, при которой Round Key XOR'ится с State. Длина RoundKey равна размеру State (то есть если Nb = 4, то длина RoundKey равна 128 бит или 16 байт).

InvMixColumns() – трансформация при расшифровании, которая является обратной по отношению к MixColumns().

InvShiftRows() – трансформация при расшифровании, которая является обратной по отношению к ShiftRows().

InvSubBytes() – трансформация при расшифровании, которая является обратной по отношению к SubBytes().

MixColumns() – трансформация при шифровании, которая берёт все столбцы State и смешивает их данные (независимо друг от друга), чтобы получить новые столбцы.

Вспомогательные процедуры

RotWord() – функция, использующаяся в процедуре Key Expansion, которая берёт 4-байтовое слово и производит над ним циклическую перестановку.

ShiftRows() – трансформации при шифровании, которые обрабатывают State, циклически смещая последние три строки State на разные величины.

SubBytes() – трансформации при шифровании, которые обрабатывают State, используя нелинейную таблицу замещения байтов (S-box), применяя её независимо к каждому байту State.

SubWord() – функция, используемая в процедуре Key Expansion, которая берёт на входе четырёхбайтовое слово и, применяя S-box к каждому из четырёх байтов, выдаёт выходное слово.

Описание криптосистемы

Определяем размеры Nk и Nr.

$Nk = type/32$ $Nr = Nk + 6$

Генерируем $Nr + 1$ раундовый ключ. Каждый раундовый ключ имеет размер 128 бит.

$RoundKey = KeyExpansion(key)$

Смешиваем открытый текст с первым ключом, чтобы лишить возможности аналитика подавать на вход специальный открытый текст

$State = PlainText \oplus RoundKey_1$

Описание криптосистемы

Для $i = 1,...,Nr\text{-}1$ - выполняем

$State = Raund(State, RaundKey_{i+1})$

Последний раунд отличается от остальных

$State = EndRaund(State, RaundKey_{Nr+1})$

Вывод результата

$CipherText = State$

Описание Round

Input

$State = (s_1, ..., s_16), s_i - Byte$
$RoundKey = (k_1, ..., k_16), k_i - Byte$

Output

$State = (s_1, ..., s_16), s_i - Byte$

1. $State = SubBytes(State)$
2. $State = ShiftRows(State)$
3. $State = MixColumns(State)$
4. State $= State \oplus RoundKey$

Описание EndRound

Input

$State = (s_1, ..., s_16), s_i - Byte$
$RoundKey = (k_1, ..., k_16), k_i - Byte$

Output

$State = (s_1, ..., s_16), s_i - Byte$

1. $State = SubBytes(State)$
2. $State = ShiftRows(State)$
3. $State = State \oplus RoundKey$

Описание SubBytes()

Процедура SubBytes() обрабатывает каждый байт состояния, независимо производя нелинейную замену байтов, используя таблицу замен (S-box). Такая операция обеспечивает нелинейность алгоритма шифрования.

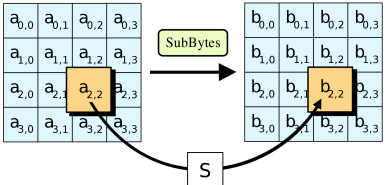
Построение S-box

1. Взятие обратного числа в $GF(2^8)$
2. К каждому байту, принадлежащему S-box, применяется операция:

$$b'_i = b_i \oplus b_{i+4 mod 8} \oplus b_{i+5 mod 8} \oplus b_{i+7 mod 8} \oplus c_i$$

где $0 \leq i < 8$, и где b_i есть i-ый бит константы

$c = 6316 = 9910 = 01100011_2$



Немного математики

Операция сложения в $GF(2^8)$

$$GF(2^8) = \{s_7, s_6, s_5, s_4, s_3, s_2, s_1, s_0\},\ s_i \in \mathbb{Z}_2$$

Элемент

$$(s_7, s_6, ..., s_0) = s_7 2^7 + s_6 2^6 + ... + s_1 2 + s_0$$

назовём **байтом**

Отождествим каждый байт с многочленом

$$(s_7, ..., s_1, s_0) \leftrightarrow s_7 x^7 + ... + s_1 x + s_0$$

Из вышесказанного, сумму байт можно отождествить с побитовой операцией XOR.

Немного математики

Операция умножения в $GF(2^8)$

Определим операцию умножения при помощи многочлена

$$x^8 + x^4 + x^3 + x + 1$$

$\exists s(x)$ и $t(x)$ - многочлены 7 степени в $GF(2^8)$

Разделим их произведение на $f(x)$:

$$s(x)t(x) = g(x)f(x) + r(x),\ deg(f(x)) < 8$$

тогда

$$s(x) * t(x) = r(x)$$

Немного математики

Операция умножения в $GF(2^8)$

По расширенному алгоритму Евклида:

$$u(x)s(x) + v(x)f(x) = \gcd(s(x), f(x))$$

Если

$$\gcd(s(x), f(x)) = 1$$

то

$$(u(x))^{-1} = s(x)$$

иначе $s(x)$ необратим

Если gcd равен 1 $\Rightarrow$ $f(x)$ раскладывается на множители над $\mathbb{Z}_2$(. Но $f(x)$ неприводим по условию. Из всего вышеперечисленного следует, что $GF(2^8)$ - поле.

Матрица S-box

Байт y будет заменять байт x по правилу:

$$y = Ax^{-1} + b$$

где
x^{-1} - обратный в $GF(2^8)$

Шифр перестановки ShiftRows

$$State = \begin{bmatrix} s_1 & s^5 & s^9 & s^{13} \\ s^2 & s^6 & s^{10} & s^{14} \\ s^3 & s^7 & s^{11} & s^{15} \\ s^4 & s^8 & s^{12} & s^{16} \end{bmatrix}$$

Второй, третий и четвёртый ряд сдвигаются влево на 1, 2, 3 соответственно:

$$State = \begin{bmatrix} s_1 & s^5 & s^9 & s^{13} \\ s^2 & s^6 & s^{10} & s^{14} \\ s^3 & s^7 & s^{11} & s^{15} \\ s^4 & s^8 & s^{12} & s^{16} \end{bmatrix} \rightarrow \begin{bmatrix} s_1 & s^5 & s^9 & s^{13} \\ s^6 & s^{10} & s^{14} & s^2 \\ s^{11} & s^{15} & s^3 & s^7 \\ s^{16} & s^4 & s^8 & s^{12} \end{bmatrix}$$

Ещё немного математики

Расширим $GF(2^8)$, присоединив многочлен $x^4 + 1$ Зададим множество s_1, s_2, s_3, s_4, $s_i \in GF(2^8)$

Сложение в $GF(2^8)_{x^4+1}$

$$(s_1, s_2, s_3, s_4) + (t_1, t_2, t_3, t_4) = (s_1 + t_1, s_2 + t_2, s_3 + t_3, s_4 + t_4)$$

$$\forall s_i + t_i \in GF(2^8)$$

Умножение в $GF(2^8)_{x^4+1}$

$$s(x) = s_1 + s_2 x + s_3 x^2 + s_4 x^3$$

$$s(x) \times t(x) = s(x)t(x)(mod x^4 + 1) = k(x)$$

$$s(x), t(x), k(x) \in GF(2^8)$$

Ещё немного математики

Делители нуля в $GF(2^8)_{x^4+1}$
$\lhd x^4 + 1$

$$x^4 + 1 = (x-1)^4 \Rightarrow (x^2+1)(x^2+1) = 0 \ (mod x^4+1)$$

Таким образом, в $GF(2^8)_{x^4+1}$ существуют делители нуля и $GF(2^8)_{x^4+1}$ – кольцо.

Ещё немного математики

Более подробно об умножение в $GF(2^8)_{x^4+1}$

При делении с остатком на $x^4 + 1$ можно считать, что $x^4 + 1 = 0$ или $x^4 = -1 = 1$, тогда:

$a_1 + a_2x + a_3x^2 + a_4x^3)(s_1 + s_2x + s_3x^2 + s_4x^3) = a_1s_1 + (a_2s_1 + a_1s_2)x + (a_3s_1 + a_2s_2 + a_1s_3)x^2 + (+a_4s_1 + a_3s_2 + a_2s_3 + a - 1s_4)x^3 + (a_4s_2 + a_3s_3 + a_2s_4)x^4 + (a_4s_3 + a_3 + s_4)x^5 + a_4s_4x^6 = a_1s_1 + a_4s_2 + a_3s_3 + a_2s_4 + (s_1 + s_2x + s_3x^2 + s_4x^3)x + (a_3s_1 + a_2s_2 + a_1s_3 + a_4s_4)x^2 + (+a_4s_1 + a_3s_2 + a_2s_3 + a_1s_4)x^3$

$$as = \begin{bmatrix} a_1 & a^4 & a^3 & a^2 \\ a^2 & a^1 & a^4 & a^3 \\ s^3 & a^2 & a^1 & a^4 \\ s^4 & a^3 & a^2 & a^1 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix}$$

MixColumns

Функция перемножает State и многочлен $a(x)$ в $GF(2^8)_{x^4+1}$

$$State = \begin{bmatrix} s_1 & s^5 & s^9 & s^{13} \\ s^2 & s^6 & s^{10} & s^{14} \\ s^3 & s^7 & s^{11} & s^{15} \\ s^4 & s^8 & s^{12} & s^{16} \end{bmatrix}$$

$$a(x) = 2 + x + x^2 + 3x^3$$

Cipher Key

Input

Nk – количество слов в ключе *key*. Nr – количество раундов в алгоритме.

$$key = (k_1, k_2, ..., k_{4Nk})$$

k_i – байт

Output

Расширенный ключ

$$w = (w_0, w_1, ..., w_{4Nr+1} - 1)$$

$$RoundKey_i = (w_{4i-4}, w_{ri-3}, w_{4i-2}, w_{4i-1})$$

Cipher Key

1. Для $i = 0, .., Nk - 1$:
   $w_i = (k_{4i+1}, k_{4i+2}, k_{4i+3}, k_{4i+4})$
2. Для $i = Nk, ..., 4(Nk + 1) - 1$:
   $temp = w_{i-1}$
   Если $i \equiv 0 (mod\ Nk)$
   то: $temp = SubWord(RotWord(temp)) \oplus Rcon(i/Nk)$
   иначе:
   Если $Nk > 6 and i \equiv 4 (mod\ Nk)$
   то: $temp = SubWord(temp)$
   $w_i = temp \oplus w_{i-Nk}$
3. Output (Nr+1 сеансовых ключа):
   $(w_0, w_1, w_2, w_3), ..., (w_{4Nk}, w_{4Nk+1}, w_{4Nk+2}, w_{4Nk+3})$

Вспомогательные функции

SubWord()

$SubWord(w) = Subword((w_1, w_2, w_3, w_4)) =$
$(SubBytes(w_1), SubBytes(w_2), SubBytes(w_3), SubBytes(w_4))$
w_i - байт.

RotWord()

$RotWord(w) = RotateLeft(w, 1)$

Rcon()

$Rcon(i) = (2^{i-1}, 0, 0, 0)$
2^{i-1} – возведение в степень в $GF(2^8)$

Заполнение сообщения

Необходимые требования

- Сообщение должно быть определённой длины
- Чтобы шифровать сообщение произвольной длины, необходим алгоритм дополнения входных данных до нужной длины
- У этого алгоритма должен быть обратный алгоритм, который может восстановить сообщение по дополненной части.

Заполнение сообщения

Input

$m = (m_1, m_2, ..., m_k,\ m_i)$ – байт

Output

$M = (M_1, M_2, ..., M_L),\ M_i$ – блок
$M_i = (M_{i,1}, ..., M_{i,16}),\ M_{i,j}$ – байт

1. Вычисляем количество недостающих байт

$$n = 16 - Mod(k, 16)$$

2. К m справа добавляем n байт, равных n.
3. Разбиваем сообщение m на блоки по 16 байт.

Режим шифрования

Режим шифрования – метод применения блочного шифра (алгоритма), позволяющий преобразовать последовательность блоков открытых данных в последовательность блоков зашифрованных данных. При этом для шифрования одного блока могут использоваться данные другого блока.

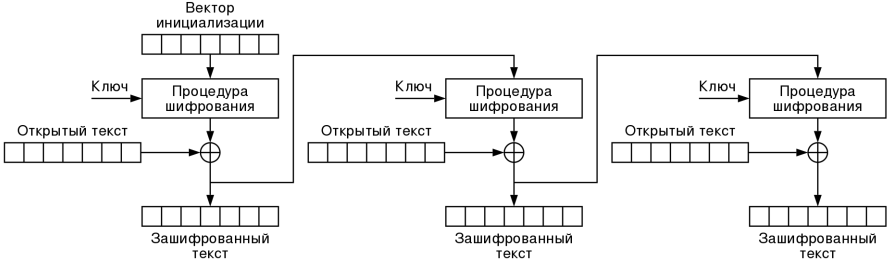
Обычно режимы шифрования используются для изменения процесса шифрования так, чтобы результат шифрования каждого блока был уникальным вне зависимости от шифруемых данных и не позволял сделать какие-либо выводы об их структуре. Это обусловлено, прежде всего, тем, что блочные шифры шифруют данные блоками фиксированного размера, и поэтому существует потенциальная возможность утечки информации о повторяющихся частях данных, шифруемых на одном и том же ключе.

Для решения задачи используются CFB и OFB, поэтому мы рассмотрим только их.

Cipher Feedback

Во время шифрования каждый блок открытого текста складывается по модулю 2 с блоком, зашифрованным на предыдущем шаге.

$C_0 = IV$
$C_i = E_k(C_{i-1}, k \oplus P_i)$
$P_i = E_k(C_{i-1}, k \oplus C_i)$



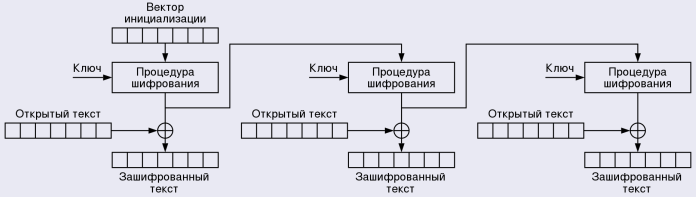
Output Feedback

Режим (OFB) обратной связи вывода превращает блочный шифр в синхронный шифр потока: он генерирует ключевые блоки, которые являются результатом сложения с блоками открытого текста, чтобы получить зашифрованный текст. Так же, как с другими шифрами потока, зеркальное отражение в зашифрованном тексте производит зеркально отражённый бит в открытом тексте в том же самом местоположении. Это свойство позволяет многим кодам с исправлением ошибок функционировать как обычно, даже когда исправление ошибок применено перед кодированием.
Из-за симметрии операции сложения, шифрование и расшифрование похожи:

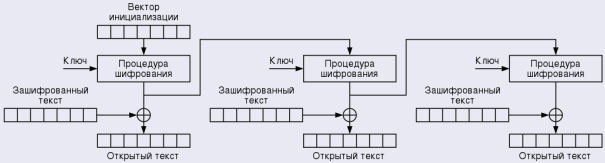
$C_i = P_i \oplus O_i)$
$P_i = C_i \oplus O_i)$
$O_i = E_k(O_{i-1})$
$C_0 = IV$

Output Feedback

Шифрование в режиме OFB



Расшифрование в режиме OFB



Решение

Из приложенного файла можно сделать следующие выводы

1. Длина флага 80 символов.
2. *key* и *IV* используются для каждого соединения.
3. Реализовано пять режимов шифрования: 'ECB', 'CBC', 'CFB', 'OFB', 'CTR'

Мы можем дать серверу данные для шифрования, позволить серверу шифровать флаг в течении 5 минут за одно подключение. Но можем использовать только один режим шифрования за одно подключение. Выше мы рассмотрели функции CFB и OFB. Они практически идентичны, поэтому мы их и будем использовать

Решение

Введём на вход нули и применим CFB, он вернёт шифрование каждого блока. Затем запросим зашифрованный флаг с режимом OFB. Объедем их вместе и получим флаг.

Код для CFB

$C_0 = IV$
$C_i = E_k(C_{i-1}) \oplus P_i$
$P = 0 \Rightarrow C_i = E_k(C_{i-1})$

Код для OFB

$O_0 = IV$
$O_i = E_k(O_{i-1})$
$C_i = E_k(O_{i-1}) \oplus P_i$

Код программы

```
from pwn import *
from Crypto.Util.number import long_to_bytes

def xor(s1, s2):
    return ''.join([str(int(a) ^ int(b)) for a,b in zip(s1,s2)])

r = remote("chalp.hkcert21.pwnable.hk", 28102)
r.recvuntil(b'> ')
r.sendline(b'cfb data '+ b'0'*160)
s1 = r.recvline()[:-1]
r.recvuntil(b'> ')
r.sendline(b'ofb flag')
s2 = r.recvline()[:-1]
r.close()
t = xor(bin(int(s1,16))[2:],bin(int(s2,16))[2:])
print(long_to_bytes(int(t,2)))
```

Ответ

Флаг

```
In [7]: runfile('/home/andrew/Documents/Summer 2023/solution.py', wdir='/home/
andrew/Documents/Summer 2023')
w3_sh0u1d_n0t_g1v3_much_fr3ed0m_t0_us3r5_wh3n_1t_c0m3s_t0_cryp70gr4phy
```

Ответ

hkcert21{w3_sh0u1d_n0t_g1v3_much_fr3ed0m_t0_us3r5_wh3n_1t_c0m3s_t0_cryp70gr4phy}