Отчет по выполненной работе

1. Добавлен (изменен) файл ak\_aes.c, содержащий реализацию алгоритма шифрования, которая включает в себя следующее:

Развертка ключа и выработка обратных раундовых ключей – void ak\_key\_expansion()

Шифрование/дешифрование 128-битного блока алгоритмом aes:

* void ak\_aes128\_enc(),
* void ak\_aes128\_dec()

Функции для взаимодействия с библиотекой (встраивания) –

* ak\_aes\_encrypt(), ak\_aes\_decrypt() – для шифрования/дешифрования
* ak\_aes\_delete\_keys() – для освобождения памяти от раундовых ключей
* ak\_skey\_set\_mask(), ak\_skey\_set\_unmask() – определения функций маскирования (ничего не делают)
* ak\_aes\_schedule\_keys() – развертка ключа (вызов ak\_key\_expansion)
* ak\_bckey\_create\_aes() – инициализация значений объекта класса: функции-обработчики, значения ключа

1. Добавлен файл test-aes.c, в котором тестируется правильность работы программы на примере значений, указанных в стандарте алгоритма AES-128 (страница 27, Appendix A, A.1). Приведены значения раундовых ключей в 16-ричном виде, результаты шифрования 128-битного блока и дешифрования полученного шифра
2. Добавлены описания функций, не относящихся к встраиванию в библиотеку (неэкспортируемые функции), в файл libakrypt-internal.h, а также описание ak\_bckey\_create\_aes() в libakrypt.h

В ходе работы были использованы интринзиковые функции, специально разработанные для ускорения работы алгоритма AES:

AESENC – Выполнить один раунд шифрования. Выполняет ряд последовательных раундовых преобразований над 128-битным блоком с использованием ключа, который подается на вход. Последовательно выполняются ShiftRows(), SubBytes(), MixColumns(), xor

AESDEC - Выполнить один раунд дешифрования (используется алгоритм Equivalent Inverse Cipher). Выполняет ряд последовательных раундовых преобразований над 128-битным блоком с использованием ключа, который подается на вход. Последовательно выполняются InvShiftRows(), InvSubBytes(), InvMixColumns(), xor.

AESENCLAST – Все аналогично AESENC за исключением того, что здесь отсутствует преобразование MixColumns. Данная функция предназначена для последнего раунда шифрования.

AESDECLAST – Все аналогично AESDEC за исключением того, что здесь отсутствует преобразование InvMixColumns. Данная функция предназначена для последнего раунда дешифрования.

AESIMC – Сгенерировать очередной обратный раундовый ключ. AESDEC и AESDECLAST дешифруют блок бит алгоритмом эквивалентного инверсного шифрования (Equivalent Inverse Cipher), для этого им нужно обратный ключ (то есть тот, над которым провели преобразование InvMixColumns). Функция AESIMC выполняет преобразование InvMixColumns над очередным раундовым ключом шифрования, полученный блок битов является обратным ключом (для дешифрования)

AESKEYGENASSIST – Сгенерировать очередной раундовый ключ (для шифрования). Функция принимает на вход вектор матрицы Rcon (содержит значения степеней многочленов поля Галуа), предыдущий раундовый ключ и выполняет SubWord(RotWord(tmp)) xor RCON[i/Nk], где SubWord и RotWord – функции подстановки и перестановки соответственно, tmp – предыдущий ключ, Nk – длина ключа. В случае, если идет работа с 256-битным блоком добавляется преобразование SubWord(tmp) после применения предыдущих преобразований.

Функции AESIMC и AESKEYGENASSIST используются в паре для генерации раундовых ключей шифрования и дешифрования, которые в дальнейшем будут использоваться с командами AECENC, AESDEC, AESENCLAST, AESDECLAST. В данной работе развертка ключа осуществлялась без использования интринзиков (за исключением выработки обратных ключей), то есть команда AESKEYGENASSIST не использовалась.

Данные команды могут использоваться для 128, 192 и 256-битных блоков, для каждого блока они имеют разные названия. Для 128 бит:

\_mm\_aesenc\_si128

\_\_mm\_aesenclast\_si128

\_mm\_aesdec\_si128

\_mm\_aesdeclast\_si128

\_\_mm\_aesimc\_si128

Помимо команд, созданных специально для алгоритма AES, в работе использовались и другие интринзики:

\_mm\_loadu\_si128 – загружает в буфер 128 бит данных из памяти (integer), возвращает 128 бит (\_\_m128i)

\_mm\_storeu\_si128 – загружает 128 бит данных (integer) из памяти по заданному адресу (указателю)

Принципиальной разницы между предыдущими двумя командами нет, можно было использовать только одну из них

\_mm\_xor\_si128 – выполнить операцию XOR над двумя наборами по 128 бит.

Использованные материалы:

<https://csrc.nist.gov/csrc/media/publications/fips/197/final/documents/fips-197.pdf> - стандарт AES;

<https://www.intel.com/content/dam/doc/white-paper/advanced-encryption-standard-new-instructions-set-paper.pdf> - статья по интринзиковым AES инструкциям

<https://software.intel.com/sites/landingpage/IntrinsicsGuide/#expand=0,233,3416,3416,5653,6170&text=_mm_xor_si128> – справочник по интринзикам

«Криптографические методы защиты информации» - А.Б. Лось, А.Ю. Нестеренко, М.И. Рожков