МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский технологический университет»

МИРЭА

Институт кибернетики

Кафедра информационной безопасности

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

по дисциплине

«Криптографические протоколы»

На тему:

«Реализация блочного шифра. Шифр AES»

Подготовил

студент группы ККСО–01–14 А.С. Першин

Руководитель работы

А.П. Никитин

Москва, 2019

Оглавление

1. Описание 3
2. Терминология....3
3. Шифрование....4
3.1 Преобразование SubBytes....5
3.2 Преобразование ShiftRows....6
3.3 Преобразование MixColomns....6
3.4 Преобразование AddRoundKey....7
3.5 Процедура KeyExpansion....7
4. Дещифрование 8
4.1 Преобразование InvShiftRows....9
4.2 Преобразование InvSubBytes 10
4.3 Преобразование InvMixColomns 10
5. Режимы шифрования 10
5.1 Режим ECB (Electronic Code Book) 10
5.2 Режим OFB (Output Feed Back)....11
5.3 Режим CTR (Counter)....12
6. Дополнение некратных блоков (Padding) 13
7. Результаты реализации алгоритма AES 128/192/256....13
Литература....17

1. Описание

AES представляет собой алгоритм шифрования 128-битных блоков данных ключами по 128, 192 и 256 бит. AES является упрощенной версией алгоритма Rijndael. Оригинальный алгоритм Rijndael отличается тем, что поддерживает более широкий набор длин блоков.

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.

В ходе выполнения работы был изучен документ FIPSPUB 197 «ADVANCED ENCRYPTION STANDARD (AES)» Национального института стандартов и технологий США(NIST).

2. Терминология

Байт — последовательность из 8 битов. В контексте данного алгоритма байт рассматривается как элемент поля Галуа. Операции над байтами производятся как над элементами поля Галуа $GF(2^8)$, то есть байту $\{b_7b_6b_5b_4b_3b_2b_1b_0\}$ соответствует многочлен $\sum_{i=0}^{7} b_i \cdot x^i$ в поле $GF(2^8)$.

Блок — последовательность из 16 байтов, над которой оперирует алгоритм. Блок служит входным и выходным данными алгоритма. Байты в блоке нумеруются с нуля.

Ключ — последовательность из 16, 24 или 32 байтов, используемая в качестве ключа шифрования. Байты в ключе нумеруются с нуля. Ключ, наряду с блоком, является входным данным алгоритма.

Форма (*State*) — двумерный массив байтов, состоящий из четырех строк. Байты в форме располагаются в порядке, изображенном в Табл. 1. В алгоритме AES форма используется для представления блока.

Табл. 1. Порядок байтов в форме

0	4	8	12
1	5	9	13
2	6	10	14
3	7	11	15

Раунд — итерация цикла преобразований над формой. В зависимости от длины ключа раундов может быть от 10 до 14, как показано в Табл. 2.

Ключ раунда (*round key*) — ключ, применяемый в раунде. Вычисляется для каждого раунда.

Таблица подстановок (*S-box*) — таблица, задающая биективное отображение байта в байт. Таблица подстановок представлена в Табл. 3.

Обратная таблица подстановок (*InvS-box*) — таблица, задающая отображение, обратное задаваемому таблицей подстановок. Обратная таблица подстановок представлена в Табл. 4.

Nb — количество слов (word) в блоке.

Nk — количество слов в ключе.

Nk может принимать значения 4, 6, 8.

Nr — количество раундов. Параметр *Nr* зависит от значений *Nk*. Соответствующие значения данных параметров приведены в Табл. 2.

Табл. 2. Зависимость *Nr* от *Nk*.

Nk	Nr
4	10
6	12
8	14

3. Шифрование

Для шифрования в алгоритме AES применяются следующие процедуры преобразования данных:

1. KeyExpansion — Вычисление раундовых ключей для всех раундов.
2. SubBytes — Подстановка байтов с помощью таблицы подстановок;
3. ShiftRows — Циклический сдвиг строк в форме на различные величины;
4. MixColumns — Смешивание данных внутри каждого столбца формы;
5. AddRoundKey — Сложение ключа раунда с формой.

Порядок выполнения процедур 2 и 3 можно поменять местами в силу линейности этих операций.

Процедуры 4 и 5 тоже можно выполнять в разном порядке, но при этом изменяется количество их вызовов, поскольку MixColumns(AddRoundKey(A, B)) = AddRoundKey(MixColumns(A), MixColumns(B)).

Шифрование производится по алгоритму, приведенному на Рис. 1.

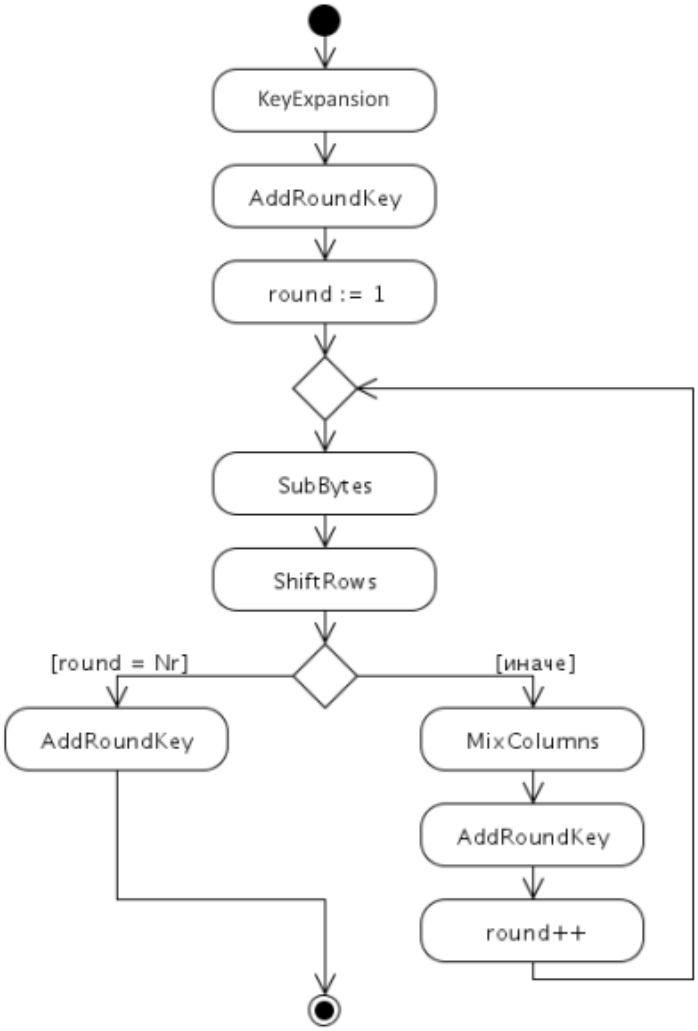


Рис. 1. Алгоритм шифрования

3.1 Преобразование SubBytes

Преобразование SubBytes заключается в замене каждого байта {xy} формы (где x и y обозначают шестнадцатиричные цифры) на другой в соответствии с Табл. 3.

Табл. 3. Таблица подстановок

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Например, байт {fe} заменится на {bb}.

3.2 Преобразование ShiftRows

Преобразование ShiftRows заключается в циклическом сдвиге влево строк формы. Преобразование схематично представлено на Рис. 2. Первая строка остается неизменной. Во второй производится сдвиг на 1 байт, то есть первый байт переносится в конец. В третьей — сдвиг на 2 байта, в четвертой — на 3.

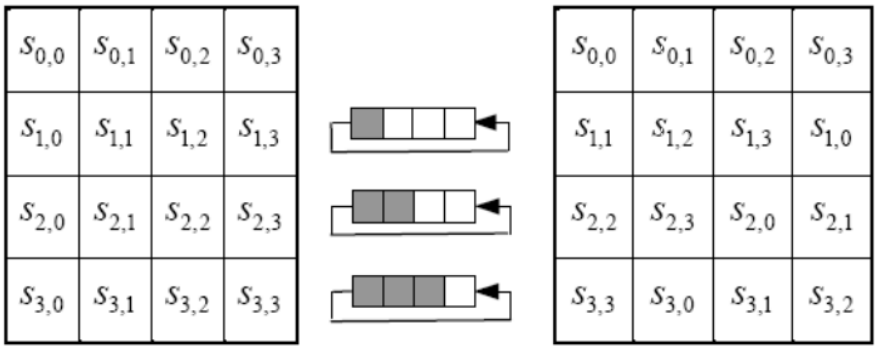


Рис. 2. Преобразование ShiftRows

3.3 Преобразование MixColomns

Преобразование MixColumns заключается в умножении квадратной матрицы 4-го порядка на каждый столбец формы:

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

Умножение производится в поле Галуа $GF(2^8)$.

Над каждым столбцом операция производится отдельно, как показано на Рис. 3.

Для реализации быстрого произведения в поле были использованы готовые таблицы, заменяющие умножения для всех элементов из $GF(2^8)$ на элементы (*0x02* и *0x03*) на подстановку [ресурс].

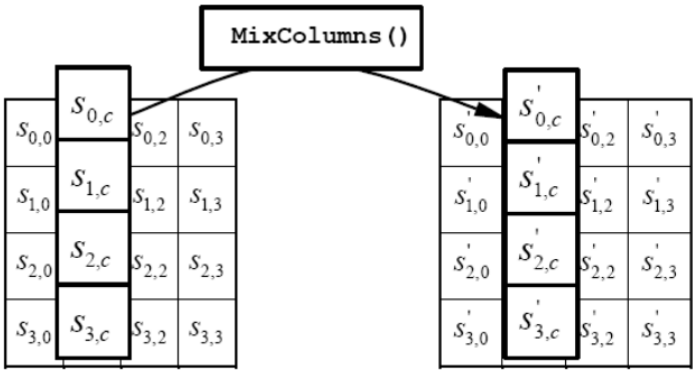


Рис. 3. Преобразование MixColumns

3.4 Преобразование AddRoundKey

В преобразовании AddRoundKey 32-битные слова раундового ключа прибавляются к столбцам формы с помощью побитовой операции XOR:

$$[s'_{0,c}, s'_{1,c}, s'_{2,c}, s'_{3,c}] = [s_{0,c}, s_{1,c}, s_{2,c}, s_{3,c}] \oplus [w_{round*Nb+c}]$$

Здесь w_i — это столбцы ключа. Над каждым столбцом операция производится отдельно, как показано на Рис. 4.

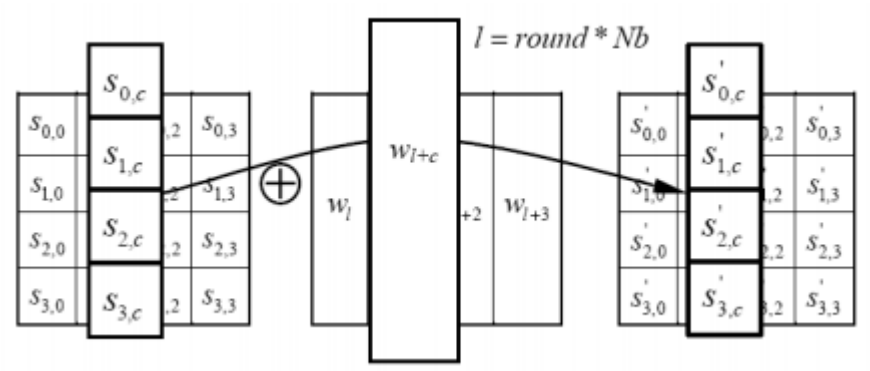


Рис. 4. Преобразование AddRoundKey

3.5 Процедура KeyExpansion

В алгоритме AES генерируются раундовые ключи на основе ключа шифрования с помощью процедуры KeyExpansion. Процедура KeyExpansion создает *Nb* * (*Nr* + 1) слов: алгоритму требуется начальный ключ размером *Nb*, плюс каждый из *Nr* раундов требует ключ из *Nb* слов. Ниже приведен псевдокод процедуры KeyExpansion:

```
// Процедура вычисляет ключи раундов.
// key — ключ
// out — результат
// Nk — количество слов в ключе
ExpandKey(byte key[4*Nk], word out[Nb*(Nr+1)], int Nk)
begin
     i = 0
     while (i < Nk)
          out[i] = word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
          i = i + 1
     end while
     i = Nk
     while (i < Nb * (Nr+1))
          word temp = out[i-1]
          if (i mod Nk = 0)
               temp = SubWord(RotWord(temp)) xor Rcon(i/Nk)
          else if ((Nk > 6) and (i mod Nk == 4))
               temp = SubWord(temp)
          end if
          out[i] = out[i-Nk] xor temp
          i = i + 1
```

```
    end while
end
```

Здесь использованы следующие функции:

SubWord осуществляет замену каждого байта в слове в соответствии с таблицей подстановок, представленной в Табл. 3.

RotWord осуществляет циклический сдвиг байтов в слове влево, как показано на Рис. 5.

Рис. 5. Процедура RotWord

Rcon(i) формирует слово $\left[02^{i-1},00,00,00\right]$.

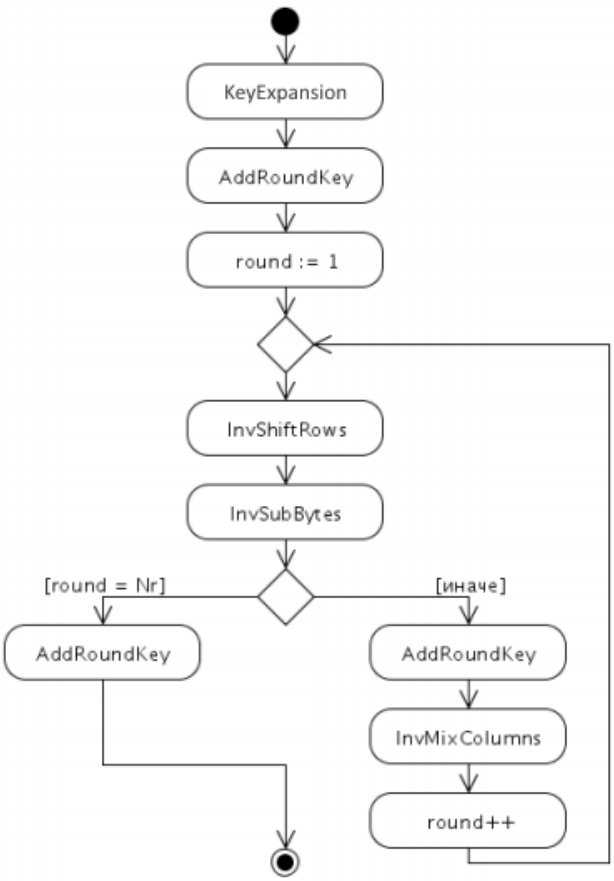
4. Дешифрование

При дешифровании все преобразования производятся в обратном порядке. Используются следующие обратные преобразования вместо соответствующих шифрующих:

InvSubBytes — Подстановка байтов с помощью обратной таблицы подстановок;

InvShiftRows — Циклический сдвиг строк в форме на различные величины;

InvMixColumns — Смешивание данных внутри каждого столбца формы;



Процедуры KeyExpansion и AddRoundKey остаются неизменными. Ключи раунда используются в обратном порядке. Алгоритм дешифрования представлен на Рис. 6.

4.1 Преобразование InvShiftRows

Это преобразование обратно преобразованию ShiftRows. Первая строка формы остается неизменной. Вторая строка циклически сдвигается вправо на 1 байт. Третья — на 2, четвертая — на 3. Схематично преобразование показано на Рис. 7.

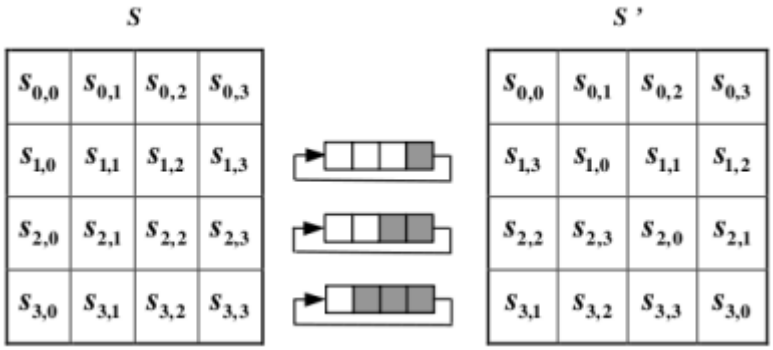


Рис. 7. Процедура InvShiftRows

4.2 Преобразование InvSubBytes

Это преобразование обратно преобразованию SubBytes. Подстановка байтов происходит аналогично с помощью обратной таблицы подстановок, представленной в Табл. 4.

Табл. 4. Обратная таблица подстановок

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
	1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
	2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
	3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
	4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
	5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
	6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
	7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
	8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
	9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
	a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
	b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
	c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
	d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
	e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
	f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

4.3 Преобразование InvMixColomns

Это преобразование обратно преобразованию MixColumns. InvMixColumns преобразует в форме каждый столбец отдельно. Преобразование происходит по следующей формуле:

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

Здесь умножение также производится в поле Галуа $GF(2^8)$.

Для реализации быстрого произведения в поле были использованы готовые таблицы, заменяющие умножения (для всех элементов из $GF(2^8)$ на элементы *0x09 0x0b* и *0x0d*) на подстановку [ресурс].

5. Режимы шифрования

Далее описаны 3 режима работы алгоритма, которые были реализованы в данной работе на практике.

5.1 Режим ECB (Electronic Code Book)

В режиме ECB каждый блок шифруется независимо от других, как показано на Рис. 8. Таким образом, одинаковые блоки открытого текста преобразуются в одинаковые блоки зашифрованного текста.

Шифрование

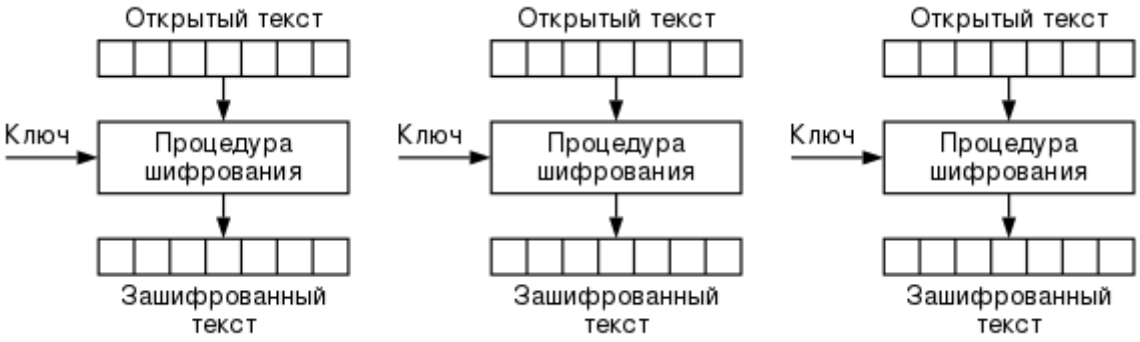


Рис. 8. Режим ECB

Дешифрование происходит по аналогичной схеме.

В режиме ECB можно производить шифрование и дешифрование нескольких блоков параллельно.

5.2 Режим OFB (Output Feed Back)

В режиме OFB входным блоком служит результат применения шифрования к предыдущему входному блоку. Первым входным блоком служит Initialization Vector.

Шифрование и дешифрование в режиме OFB представлены на Рис. 9 и Рис. 10.

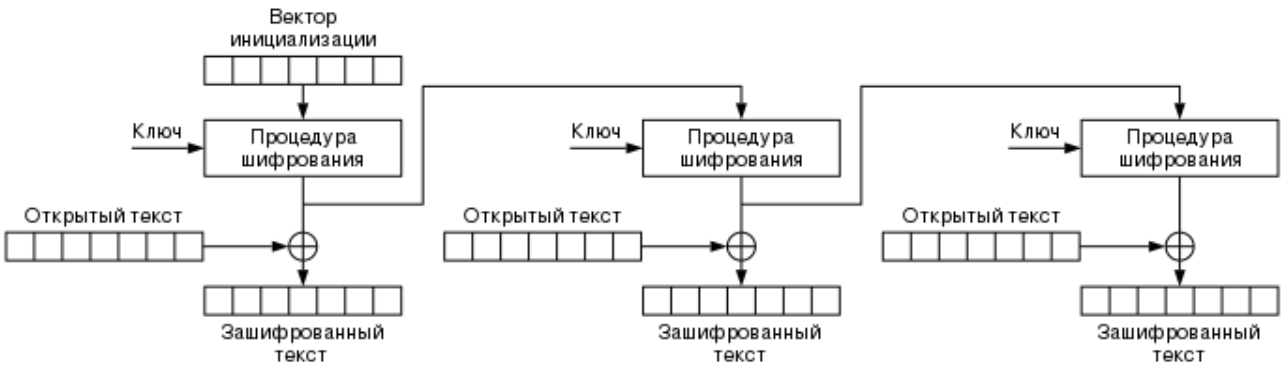


Рис. 9. Шифрование в режиме OFB

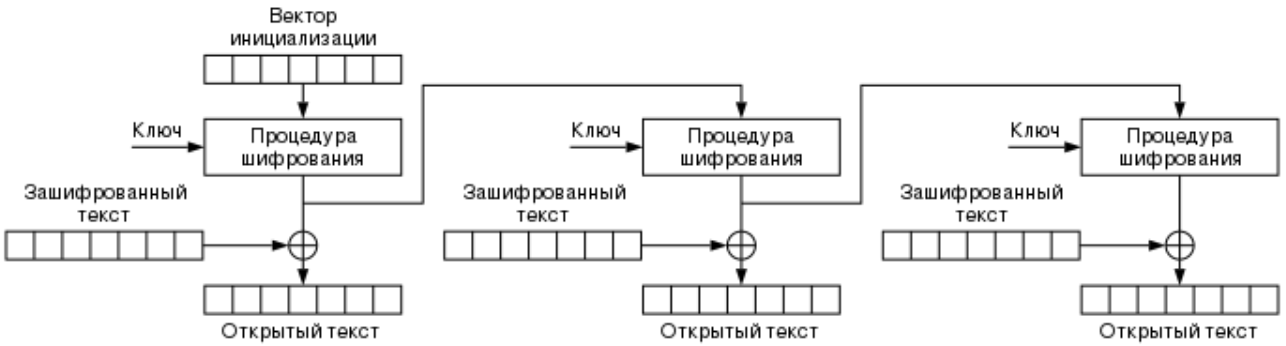


Рис. 10. Дешифрование в режиме OFB

В данном режиме работы шифра шифрование и дешифрование нескольких блоков одновременно произвести не получится.

5.3 Режим CTR (Counter)

В режиме CTR входными блоками являются значения некоторой функции *T(i)*, называемой счетчиком, где *i* — номер блока. Шифрование и дешифрование в режиме CTR представлены на Рис. 11 и Рис. 12.

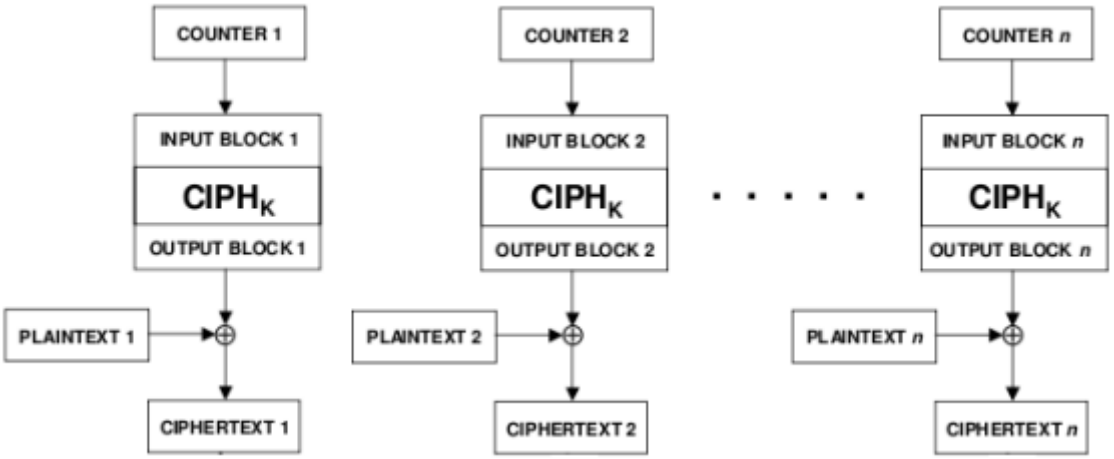


Рис. 11. Шифрование в режиме CTR

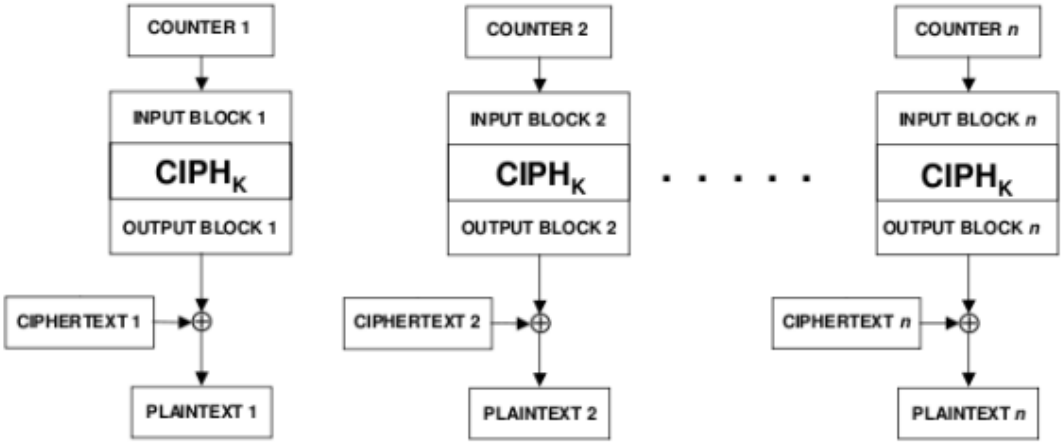


Рис. 12. Дешифрование в режиме CTR

Режим CTR допускает параллельное шифрование (и дешифрование) нескольких блоков.

6. Дополнение некратных блоков (Padding)

На практике происходят случаи, когда количество исходных данных в блочном шифрование не кратно числу элементов самого блока, для решения этой ситуации есть специальные методы дополнения данных до нужной кратности.

Алгоритм дополнения некратных блоков взят из ГОСТ 34.13-2015 «Режимы работы блочных шифров» из раздела 4.1 «Дополнение сообщения» процедуру №2. Пусть $|P| \equiv r \bmod L$. Положим $P^* = P||1||0^{L-r-1}$. Где |P| - мощность исходного открытого текста (ОТ), L = 16 – размер входного блока, r - остаток от деления мощности ОТ на размер входного блока. Если r=0, то мы ничего не дополняем, иначе, мы в конце ОТ дополняем его сначала 1(единицей), а затем числом 0(нулей), равным L-r-1. В конечном итоге получаем текст P* кратный нашему размеру входного блока L.

7. Результаты реализации алгоритма AES 128/192/256

Разработка производилась в IDE Microsoft Visual Studio 15 Pro. Для реализации задания лабораторной работы было создано общее решение с именем CryptoProtocols. Реализация алгоритма AES входит в проект AES_BlocksCipher решения CryptoProtocols.

Для тестирования корректности разрабатываемых проектов в решении CryptoProtocols был создан отдельный проект GoogleTestingSolutionProject модульного тестирования gtest (для unit testing) и gmock (для проверки корректности вызовов методов). Данные пакеты устанавливались через менеджер пакетов NuGet для Visual Studio.

Результат выполнения тест кейсов для проверки корректности работы функций шифрования/дешифрования одного блока AES 128/192/256 и фиксации времени выполнения для подсчета производительности работы (т.к. gtest замеряет работу вызовов кейсов в микросекундах, то для повышения точности была использована библиотека <chrono> c++11 с точностью до микросекунд) приведены на Рис. 13 и Рис. 14.

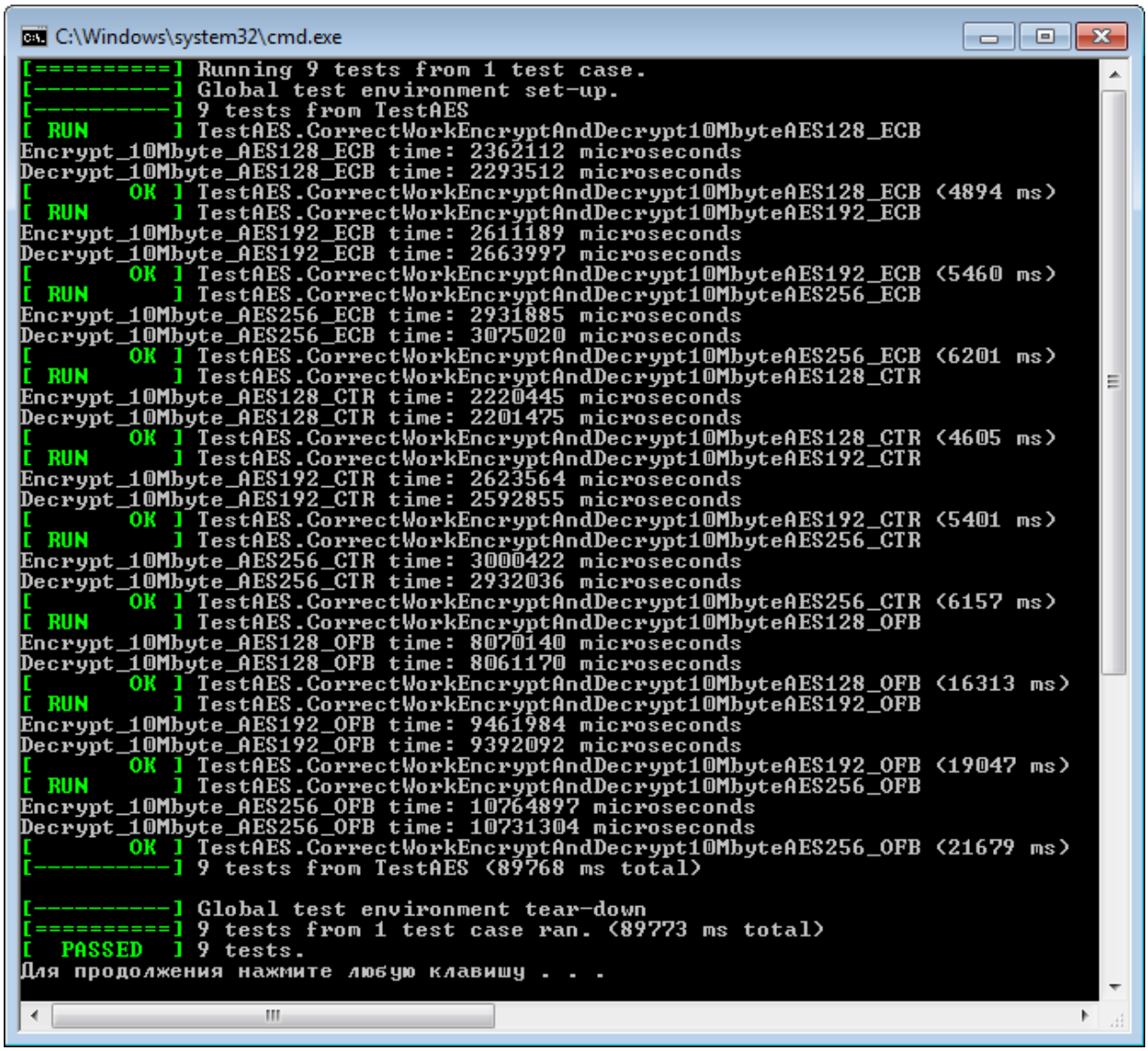


Рис. 13. Результат тестирования реализованного алгоритма AES (10 Мбайт данных)

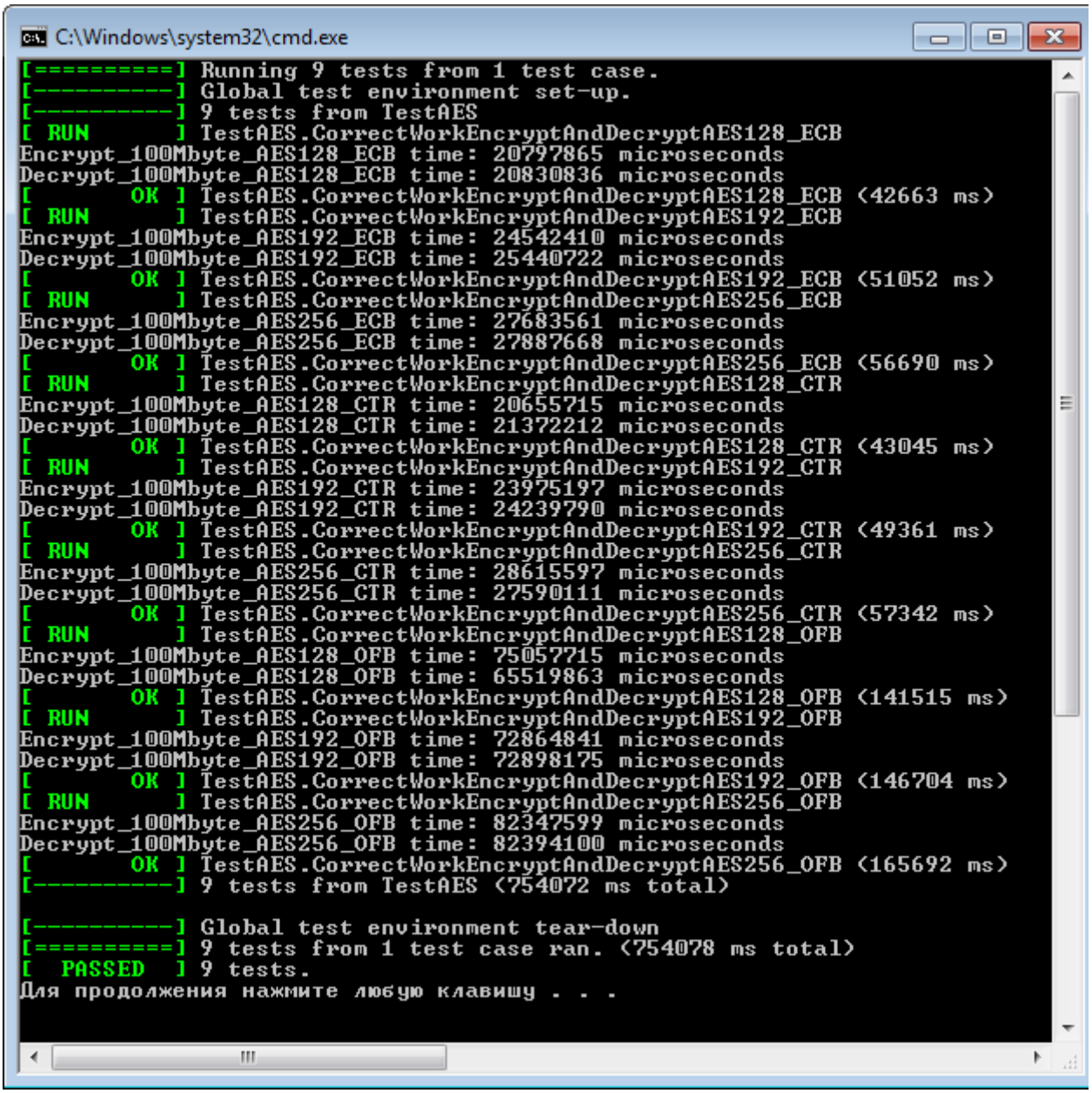


Рис. 14. Результат тестирования реализованного алгоритма AES (100 Мбайт данных)

Запускался тест на ЦП AMD A6-3410MX (4 ядра, 4 потока) на Рис.15. По полученным данным посчитаем скорость шифрования/дешифрования для данного ЦП. Данные приведены в Табл. 3.

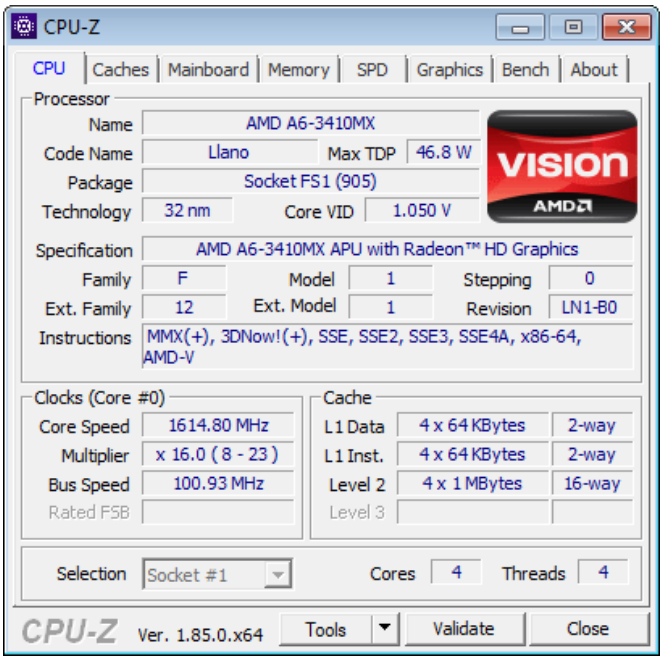


Рис. 15. ЦП AMD A6-3410MX (4 ядра, 4 потока)

Табл. 3. Скорость выполнения шифрования/дешифрования алгоритма AES

Алгоритм и размер ключа	Размер данных [Мбайт]	Шифрование/Дешифрование	Скорость [Мбайт/с]
В РЕЖИМЕ ECB (Реализован Multi Thread подход)			
Данные размером 10 Мбайт			
AES-128	10	Шифрование	4,2334995123
AES-128	10	Дешифрование	4,36012543209
AES-192	10	Шифрование	3,82967299571
AES-192	10	Дешифрование	3,75375798096
AES-256	10	Шифрование	3,41077497924
AES-256	10	Дешифрование	3,25201136903
Данные размером 100 Мбайт			
AES-128	100	Шифрование	4,80818584023
AES-128	100	Дешифрование	4,80057545458
AES-192	100	Шифрование	4,07457947284
AES-192	100	Дешифрование	3,93070605465
AES-256	100	Шифрование	3,61225205096
AES-256	100	Дешифрование	3,58581434633
В РЕЖИМЕ CTR (Реализован Multi Thread подход)			
Данные размером 10 Мбайт			
AES-128	10	Шифрование	4,5036017555
AES-128	10	Дешифрование	4,54240906665
AES-192	10	Шифрование	3,81160894112
AES-192	10	Дешифрование	3,85675249869

AES-256	10	Шифрование	3,33286451039
AES-256	10	Дешифрование	3,41059932416
Данные размером 100 Мбайт			
AES-128	100	Шифрование	4,84127516283
AES-128	100	Дешифрование	4,67897286439
AES-192	100	Шифрование	4,17097719781
AES-192	100	Дешифрование	4,12544828152
AES-256	100	Шифрование	3,49459771886
AES-256	100	Дешифрование	3,624487049
В РЕЖИМЕ OFB (Multi Thread подход не предусмотрен структурой)			
Данные размером 10 Мбайт			
AES-128	10	Шифрование	1,23913587621
AES-128	10	Дешифрование	1,24051471437
AES-192	10	Шифрование	1,05686080213
AES-192	10	Дешифрование	1,06472551589
AES-256	10	Шифрование	0,92894525604
AES-256	10	Дешифрование	0,93185320255
Данные размером 100 Мбайт			
AES-128	100	Шифрование	1,33230807786
AES-128	100	Дешифрование	1,5262547176
AES-192	100	Шифрование	1,37240401032
AES-192	100	Дешифрование	1,37177645394
AES-256	100	Шифрование	1,21436448924
AES-256	100	Дешифрование	1,21367913479

Литература

1. «Rijndael MixColomns» [Интернет ресурс], ссылка: https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/Rijndael_mix_columns.html
2. FIPS PUB 197 «ADVANCED ENCRYPTION STANDARD (AES)» [Интернет ресурс], ссылка https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.197.pdf