Федеральное агентство по образованию
Государственное образовательное учреждение высшего профессионального образования
"ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ"
Механико-математический факультет
Кафедра системного программирования

Рецензент
канд. техн. наук
_____ П.Л. Цытович
“____”_____ 2009 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Зав. кафедрой СП
_____ Л.Б. Соколинский
“____”_____ 2009 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
на соискание академической степени бакалавра информационных технологий по направлению 010400.62 "Информационные технологии"

Адаптация алгоритма блочного симметричного шифрования AES для вычислительных систем на базе процессоров Cell

Ученый секретарь
_____ М.Л. Цымблер
“____”_____ 2009 г.

Научный руководитель
кандидат физ.-мат. наук, доцент
_____ М.Л. Цымблер

Автор работы
студент группы ММ-496
_____ К.С. Пан

Челябинск, 2009

Оглавление

Введение

АКТУАЛЬНОСТЬ ПРОБЛЕМЫ

В настоящее время шифрование применяется для решения множества задач. К ним относятся проверка подлинностицелостностии авторствасообщений, обеспечение сокрытия информации для передачи по ненадежным каналам, хранение секретных данных.

AES [1] является одним из наиболее распространенных стандартов шифрования.В качестве стандарта AES с 2001 г. используется алгоритм Rijndael [2], который является победителем 5-летнего конкурса, проведенного Национальным институтом стандартов и технологий США (NIST) среди 15 различных алгоритмов шифрования [3]. В 2003 г. Комитет США по национальным системам безопасности (The Committee on National Security Systems) утвердил AES как алгоритм, криптостойкость которого достаточна для защиты секретных данных уровня TOP SECRET [4].

На сегодня использование многоядерных вычислительных систем является одним из основных направлений развития суперкомпьютеров [5, 6]. Подавляющее большинство суперкомпьютеров списка Top500 базируются на многоядерной архитектуре [7]. На первом месте в списке Top500 стоит разработанный компанией IBM суперкомпьютер Roadrunner [8]. Он содержит более 20 000 многоядерных процессоров, из которых более 12 000 процессоров имеют архитектуру Cell Broadband Engine.

Cell Broadband Engine [9] представляет собой многоядерный процессор, содержащий 9 ядер, одно из которых является управляющим, а остальные — вычислительными. Управляющее ядро распределяет задачи по вычислительным ядрам. Все ядра поддерживают специальный набор SIMD-инструкций для обработки больших объемов данных. За счет этого Cell BE обладает огромным потенциалом как с точки зрения вычислительной мощности, так и с точки зрения энергопотребления [10].

Таким образом, актуальной является задача адаптации алгоритма блочного симметричного шифрования AES для вычислительных систем на базе процессора Cell.

СТРУКТУРА И ОБЪЕМ РАБОТЫ

Работа состоит из введения, пяти основных разделов, заключения и библиографии. Объем работы составляет 46 страницу, объем библиографии — 19 наименований.

ОБЗОР ЛИТЕРАТУРЫ

В настоящее время разработка новых и адаптация существующих методов и алгоритмов решения различных научных и прикладных задач на многопроцессорных вычислительных системах на базе многоядерных процессоров является динамично развивающейся областью системного программирования.

В [9] рассматривается архитектура Cell BE, модели и инструменты программирования.

В [11] описывается алгоритм сортировки CellSort для Cell с использованием SIMD-инструкций и двойной буферизации.

В работе [12] рассматриваются несколько задач, методы их решения для Cell и стратегии оптимизации. Показано влияние времени запуска SPE-нитей на эффективность и способы его минимизации, использование SIMD-инструкций в языках высокого уровня, прирост производительности при использовании PPE в вычислениях совместно с SPE и другие способы оптимизации.

В [13] представлен способ минимизации простоя SPE-нитей из-за задержек в доставке данных.

В [14] описан подход к реализации быстрого преобразования Фурье для вычислительных систем на базе процессоров Cell.

В [15] описан способ оптимизации алгоритма асимметричного шифрования RSA с использованием DMA и библиотеки IBM MPM (Multi-Precision Math library), оптимизированной для векторных вычислений.

Работа [16] описывает способы оптимизации алгоритмов с большим потоком входных и выходных данных за счет предварительной загрузки данных с помощью PPE и использования асинхронных DMA-передач.

СОДЕРЖАНИЕ РАБОТЫ

В первом разделе, «Архитектурные особенности вычислительных систем на базе процессора Cell», представлен обзор архитектуры многопроцессорных вычислительных систем на базе процессора IBM Cell.

Второй раздел, «Алгоритм блочного симметричного шифрования AES», содержит описание последовательного алгоритма Advanced Encryption Standard (AES).

В третьем разделе, «Параллельная версия алгоритма AES», предлагается версия алгоритма шифрования по стандарту AES в режиме счетчика (CTR), адаптированная для вычислительных систем на базе процессора Cell.

Четвертый раздел, «Реализация алгоритма PAES-CTR», описывает реализацию параллельной версии алгоритма AES на базе инструментальных средств компании IBM.

Пятый раздел, «Вычислительные эксперименты», содержит результаты вычислительных экспериментов, исследующих показатели эффективности предложенного алгоритма.

В заключении суммируются основные результаты работы и рассматриваются направления дальнейших исследований в данной области.

В приложение вынесены примеры шифрования и расшифрования модельных данных с промежуточными результатами на разных этапах.

1. Архитектурные особенности вычислительных систем на базе процессора Cell

В настоящее время использование многопроцессорных многоядерных архитектур является одним из основных направлений в развитии суперкомпьютеров. Cell Broadband Engine представляет собой перспективный в этой сфере многоядерный процессор. Далее представлена архитектура Cell BE и дана краткая техническая характеристика.

На Рис. 1 показаны элементы вычислительной системы на базе Cell. Это девять процессоров (PPE, SPE), контроллер памяти (MIC), контроллер ввода-вывода (BEI) и шина (EIB), которая соединяет между собой все элементы.

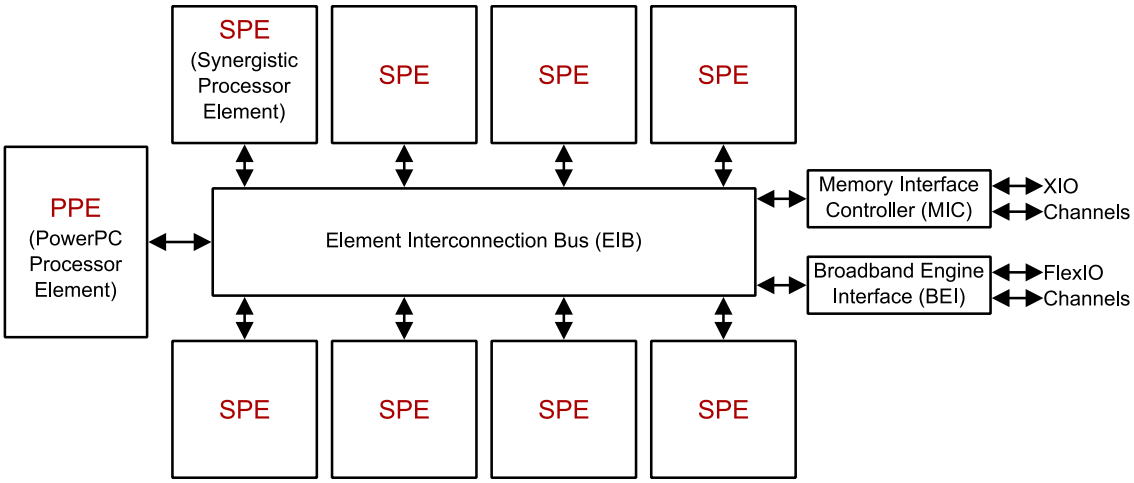

Рис. 1. Архитектура Cell BE

PPE (PowerPC Processor Element) — это процессор общего назначения на основе архитектуры PowerPC. Кроме набора инструкций PowerPC он поддерживает SIMD-инструкции из расширения Vector/SIMD Multimedia Extension. PPE является главным среди остальных процессоров в системе. На PPE выполняется операционная система, он управляет ресурсами и предназначен для управления SPE-нитями.

SPE (Synergistic Processor Element) — это RISC-процессоры, имеющие 128 регистров размером 128 бит каждый и локальную память размером 256 КБ. SPE поддерживают специальный набор SIMD-инструкций,

а также асинхронную передачу данных между своей локальной памятью и основной памятью системы. Процессоры SPE предназначены для выполнения нитей, которые запускает PPE.

2. Алгоритм блочного симметричного шифрования AES

В данном разделе описан алгоритм блочного симметричного шифрования Advanced Encryption Standard (AES) и режимы его применения. AES [1] представляет собой алгоритм шифрования 128-битных блоков данных ключами по 128, 192 и 256 бит. AES является упрощенной версией алгоритма Rijndael [2]. Оригинальный алгоритм Rijndael отличается тем, что поддерживает более широкий набор длин блоков.

Далее в изложении алгоритма нами будут использоваться следующие термины.

Байт — последовательность из 8 битов. В контексте данного алгоритма байт рассматривается как элемент поля Галуа $\mathrm{GF}(2^8)$ [17], то есть байту $b_7b_6b_5b_4b_3b_2b_1b_0$ соответствует многочлен $\sum_{i=0}^{n} b_i x^i$ в поле $\mathrm{GF}(2^8)$.

Слово (word) — последовательность из 4 байтов.

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. 1.

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 — 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

2.1. Шифрование

Шифрование производится по алгоритму, приведенному на Рис. 2. В данном алгоритме применяются следующие процедуры преобразования данных:

1. ExpandKey — Вычисление раундных 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.

В приложении приведен пример работы алгоритма шифрования. Далее преобразования и их применение при шифровании рассмотрены по-

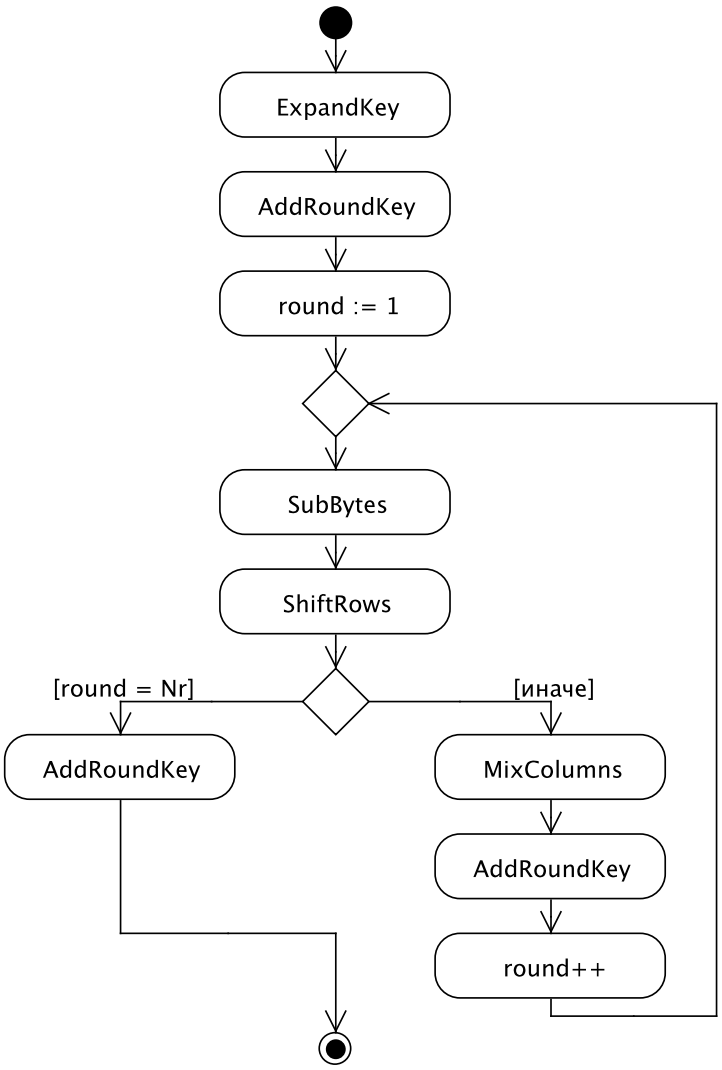

Рис. 2. Алгоритм шифрования

дробнее.

2.1.1. Преобразование SubBytes

Преобразование SubBytes заключается в замене каждого байта `{xy}` формы (где x и y обозначают шестнадцатеричные цифры) на другой в соответствии с Табл. 3. Например, байт `{fe}` заменится на `{bb}`.

2.1.2. Преобразование ShiftRows

Преобразование ShiftRows заключается в циклическом сдвиге влево строк формы. Преобразование схематично представлено на Рис. 3.

Первая строка остается неизменной. Во второй производится сдвиг на 1 байт, то есть первый байт переносится в конец. В третьей — сдвиг

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

на 2 байта, в четвертой — на 3.

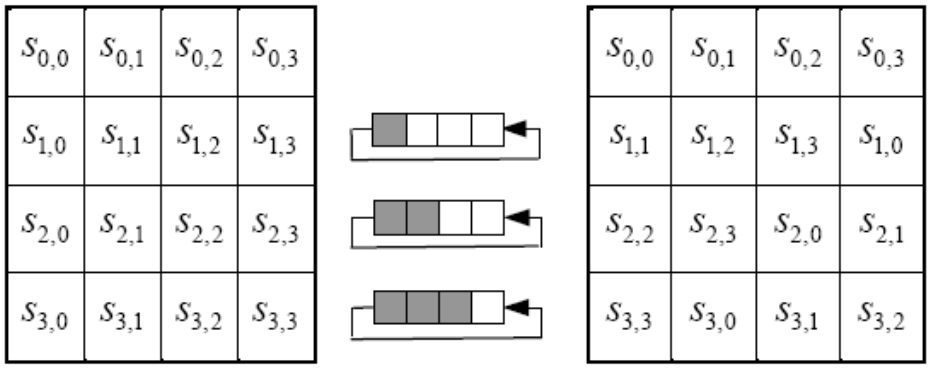

Рис. 3. Преобразование ShiftRows

2.1.3. Преобразование MixColumns

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

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

Умножение производится в поле Галуа $\mathrm{GF}(2^8)$.

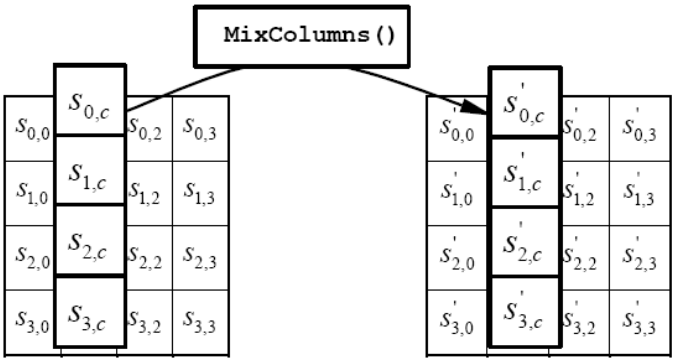

Рис. 4. Преобразование MixColumns

Над каждым столбцом операция производится отдельно, как показано на Рис. 4

2.1.4. Преобразование AddRoundKey

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

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

Здесь w_i — это столбцы ключа.

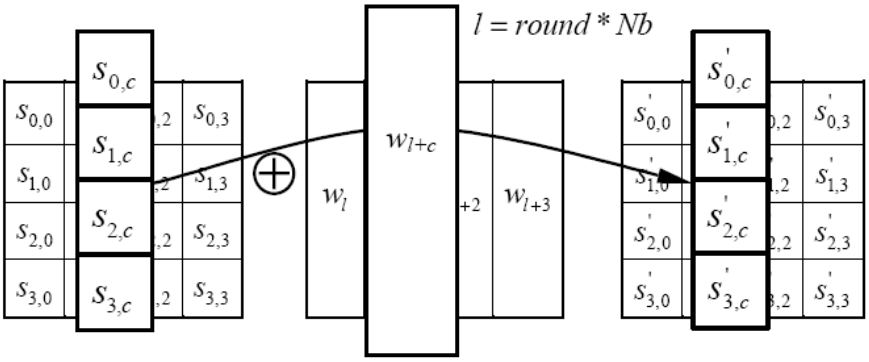

Рис. 5. Преобразование AddRoundKey

Над каждым столбцом операция производится отдельно, как показано на Рис. 5.

2.1.5. Процедура ExpandKey

В алгоритме AES генерируются раундные ключи на основе ключа шифрования с помощью процедуры ExpandKey. Процедура ExpandKey

создает $Nb*(Nr+1)$ слов: алгоритму требуется начальный ключ размером Nb, плюс каждый из Nr раундов требует ключ из Nb слов. На Рис. 6 приведен псевдокод процедуры ExpandKey.

```
// 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
```

Рис. 6. Псевдокод процедуры ExpandKey

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

Функция *RotWord* осуществляет циклический сдвиг байтов в слове влево, как показано на Рис. 7.

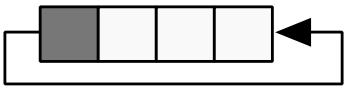

Рис. 7. Процедура RotWord

Функция *Rcon(i)* формирует слово $[02^{i-1}, 00, 00, 00]$.

2.2. Расшифрование

При расшифровании 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 *ExpandKey* и *AddRoundKey* остаются неизменными. Ключи раунда используются в обратном порядке.

Алгоритм расшифрования представлен на Рис. 8. В приложении приведен пример работы алгоритма расшифрования.

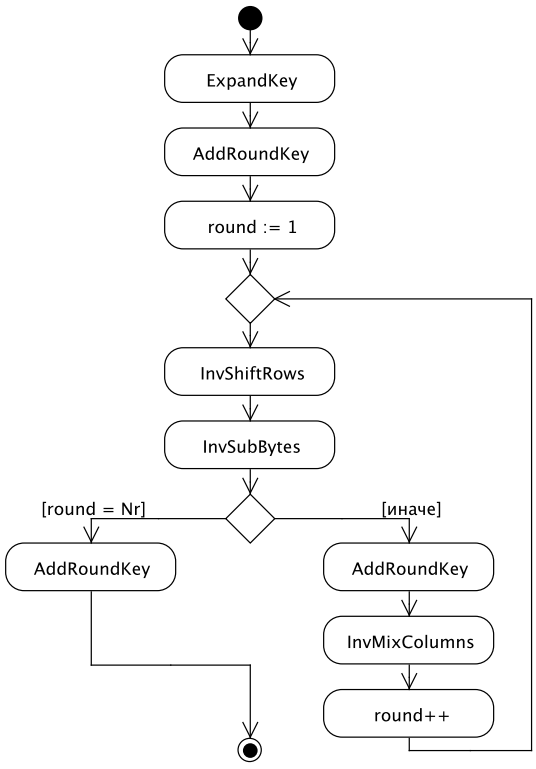

Рис. 8. Алгоритм расшифрования

2.2.1. Преобразование InvShiftRows

Данное преобразование обратно преобразованию ShiftRows. Схематично преобразование показано на Рис. 9

Первая строка формы остается неизменной. Вторая строка циклически сдвигается вправо на 1 байт. Третья — на 2, четвертая — на 3.

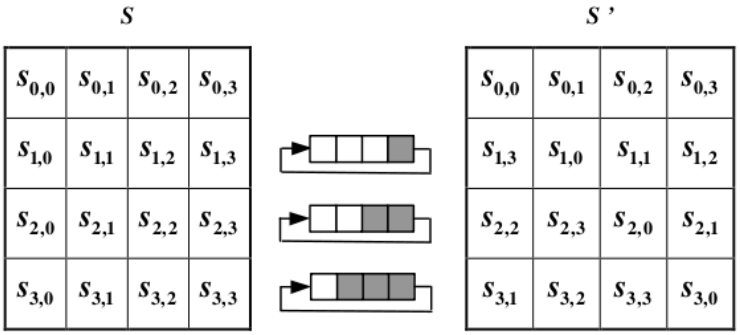

Рис. 9. Преобразование InvShiftRows

2.2.2. Преобразование InvSubBytes

Данное 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

2.2.3. Преобразование InvMixColumns

Данное преобразование обратно преобразованию MixColumns. InvMixColumns преобразует в форме каждый столбец отдельно по следу-

ющией 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).

2.3. Режимы применения

Далее описаны 5 режимов работы алгоритма, которые предусмотрены для алгоритмов блочного симметричного шифрования [18].

2.3.1. Режим ECB (Electronic Codebook)

В режиме ECB каждый блок шифруется независимо от других, как показано на Рис. 10. Таким образом, одинаковые блоки открытого текста преобразуются в одинаковые блоки зашифрованного текста.

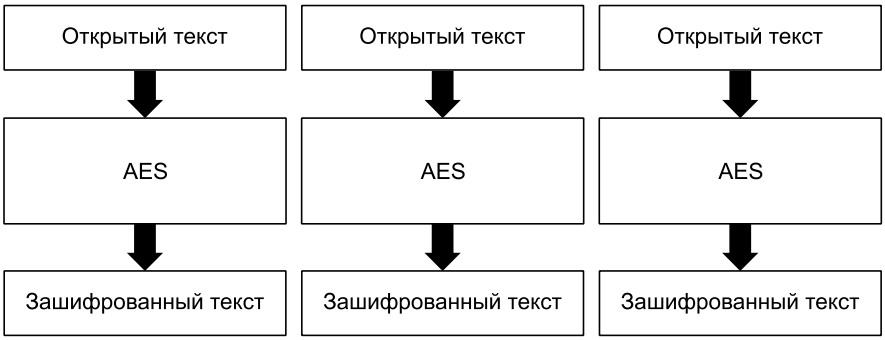

Рис. 10. Шифрование в режиме ECB

Расшифрование происходит по аналогичной схеме.

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

2.3.2. Режим CBC (Cipher Block Chaining)

В режиме CBC к каждому блоку открытого текста перед шифрованием прибавляется результат шифрования предыдущего блока с помощью побитовой операции XOR.

К первому блоку открытого текста прибавляется Initialization Vector, который генерируется случайным образом и обычно передается вместе с шифрованными данными, чтобы их можно было расшифровать.

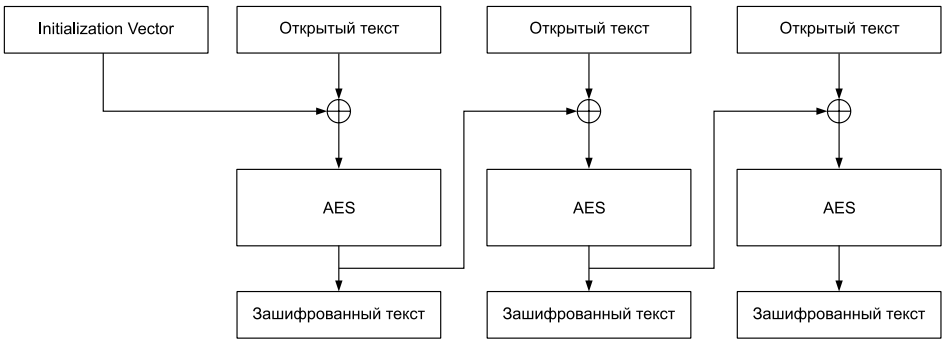

Рис. 11. Шифрование в режиме CBC

Шифрование и расшифрование в режиме CBC показаны на Рис. 11 и Рис. 12 соответственно.

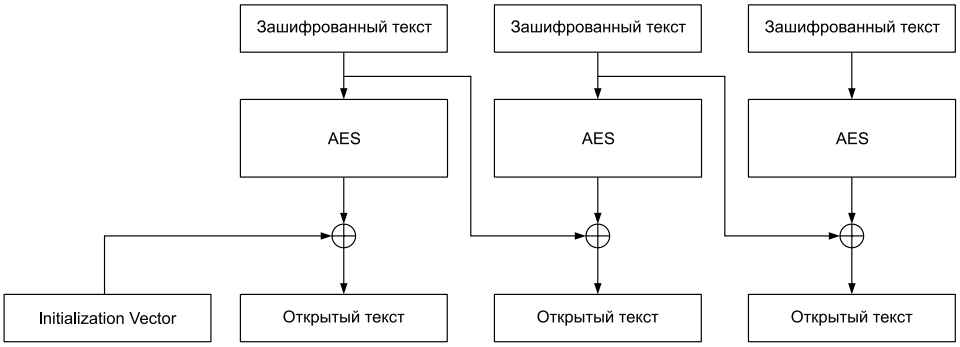

Рис. 12. Расшифрование в режиме CBC

В режиме CBC для шифрования каждого следующего блока нужно иметь результат шифрования предыдущего блока, поэтому шифровать несколько блоков одновременно нельзя. Но можно производить расшифрование нескольких блоков параллельно, поскольку для расшифрования каждого блока нужно иметь только этот блок и предыдущий.

2.3.3. Режим CFB (Cipher Feedback)

В режиме CFB при шифровании каждого блока используется только первые s битов результата. К ним прибавляются s битов открытого

(или зашифрованного, при расшифровании) текста с помощью побитовой операции XOR.

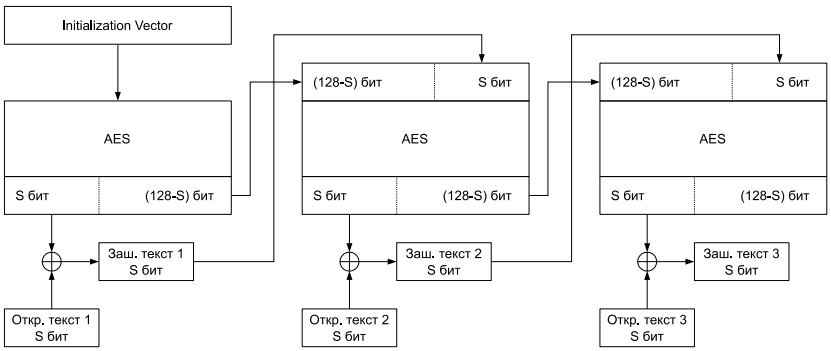

Рис. 13. Шифрование в режиме CFB

Шифрование и расшифрование в режиме CFB показаны на Рис. 13 и Рис. 14 соответственно.

В качестве первого входного блока используется Initialization Vector. Каждый следующий входной блок получается из предыдущего путем его побитового сдвига влево на s битов и прибавления первых s битов результата шифрования предыдущего блока.

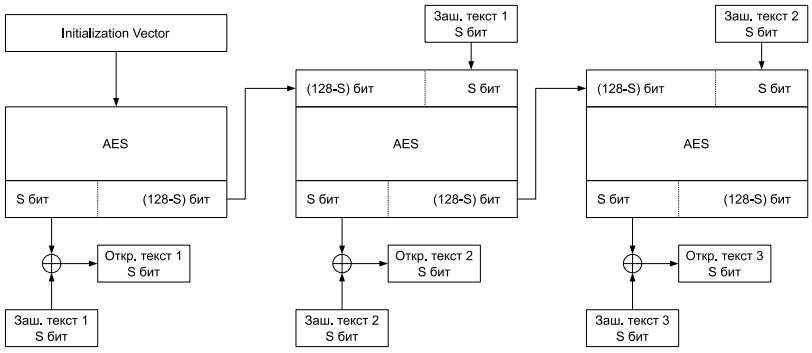

Рис. 14. Расшифрование в режиме CFB

При шифровании или расшифровании в режиме CFB для шифрования к каждого блока требуются результаты шифрования предыдущего блока. Поэтому шифрование или расшифрование нескольких блоков одновременно в данном режиме невозможно.

2.3.4. Режим OFB (Output Feedback)

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

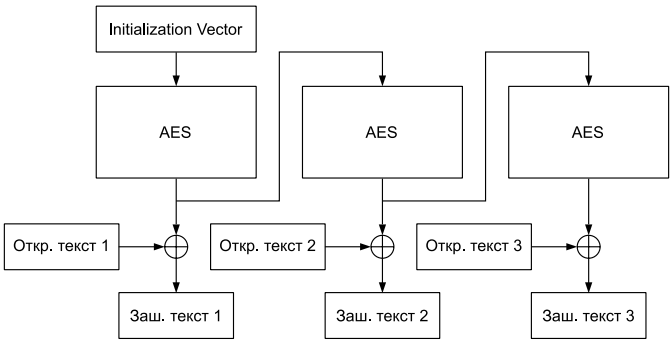

Рис. 15. Шифрование в режиме OFB

Шифрование и расшифрование в режиме OFB показаны на Рис. 15 и Рис. 16 соответственно.

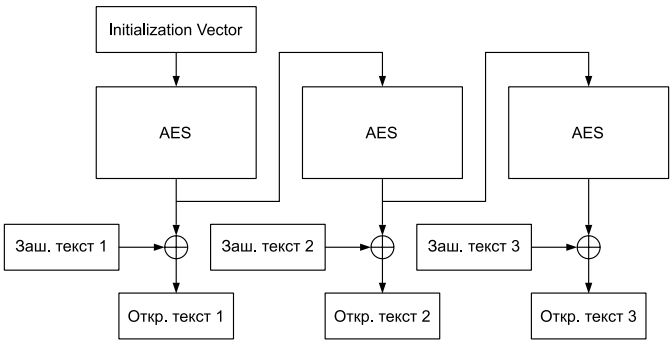

Рис. 16. Расшифрование в режиме OFB

Одновременное шифрование и расшифрование нескольких блоков невозможно, поскольку для применения шифрования к какому-либо блоку нужно зашифровать также и все предыдущие блоки.

2.3.5. Режим CTR (Counter)

В режиме CTR входными блоками являются значения некоторой функции $T(i)$, называемой счетчиком, где i — номер блока.

Шифрование и расшифрование в режиме CTR показаны на Рис. 17 и Рис. 18 соответственно.

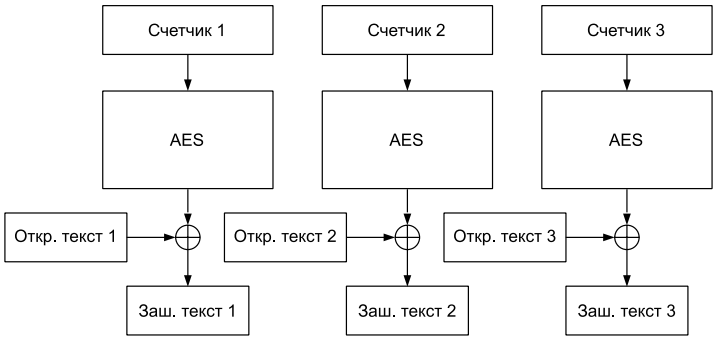

Рис. 17. Шифрование в режиме CTR

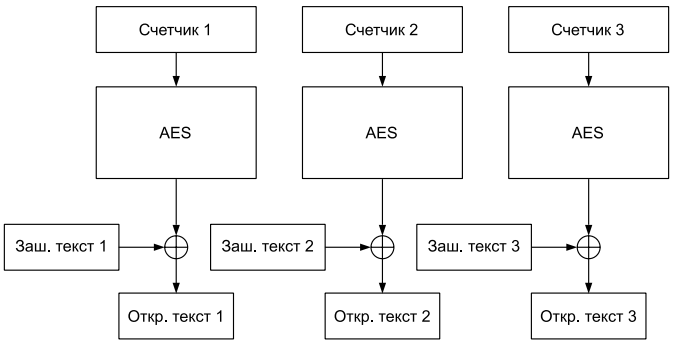

Рис. 18. Расшифрование в режиме CTR

В соответствии в определением, режим CTR допускает параллельное шифрование (и расшифрование) нескольких блоков.

3. Адаптированная версия алгоритма AES

В данном разделе предложена версия алгоритма AES, адаптированная для вычислительных систем на базе процессоров Cell.

Алгоритм блочного симметричного шифрования AES [1] может применяться в одном из следующих пяти режимов: ECB, CBC, CFB, OFB, CTR.

Режим ECB (Electronic Codebook) предполагает, что каждый блок открытого текста заменяется на блок зашифрованного текста, причем одинаковые блоки открытого текста дают одинаковые блоки зашифрованного текста.

Режим CBC (Cipher Block Chaining) предусматривает сложение блока открытого текста с предыдущим блоком зашифрованного текста с помощью побитовой операции XOR.

В режиме CFB (Cipher Feedback) каждый получаемый блок зашифрованного текста поступает на вход при шифровании следующего блока.

Режим OFB (Output Feedback) предполагает наличие так называемого инициализирующего блока, который поступает на вход в процедуру шифрования, а полученный зашифрованный блок опять проходит процедуру шифрования, и т. д. Получаемые в ходе этого блоки складываются с блоками открытого текста с помощью побитовой операции XOR.

Режим CTR (Counter) предполагает наличие счетчика. Счетчик — это 128-битное число, которое перед шифрованием инициализируется случайным значением и для каждого следующего блока открытого текста принимает новое значение, отличное от всех предыдущих. Значения счетчика шифруются и складываются с блоками открытого текста с помощью побитовой операции XOR. Начальное значение счетчика передается вместе с зашифрованными данными и используется при расшифровании.

Для выбора режима применения алгоритма AES, наиболее подходящего для параллельной реализации, нами был разработан следующий набор критериев.

Возможность распараллеливания процедуры шифрования (расшиф-

рования). Данный критерий означает возможность шифрования (расшифрования) двух и более блоков одновременно.

Простота реализации, означающая, что при шифровании и расшифровании используется только шифрующее преобразование AES, а расшифрующее не используется.

Возможность предварительных вычислений. Данный критерий означает, что можно начать вычисления до поступления открытого текста.

В Табл. 5 приведены оценки режимов по данному набору критериев. В этой таблице «+» означает, что режим удовлетворяет критерию, «−» — нет.

Табл. 5. Оценка режимов применения алгоритма AES

Критерий	**ECB**	**CBC**	**CFB**	**OFB**	**CTR**
Возможность распараллеливания процедуры шифрования	+	−	−	−	+
Возможность распараллеливания процедуры расшифрования	+	+	−	−	+
Простота реализации	−	−	+	+	+
Возможность предварительных вычислений	−	−	−	+	+

Таким образом, в соответствии с проведенным анализом для разработки параллельной версии алгоритма нами был выбран режим CTR. Параллельная версия алгоритма названа **PAES-CTR**.

3.1. Алгоритм PAES-CTR

В этом разделе предлагается параллельная версия алгоритма шифрования по стандарту AES в режиме CTR.

Далее описаны классы, реализующие параллельное шифрование в режиме CTR, и параметры алгоритма.

В изложении алгоритма, в дополнение к терминам, введенным в предыдущем разделе, будут использоваться следующие термины:

Блок — последовательность из 16 байтов.

Задание (Task) — набор последовательно идущих блоков. Количество блоков в задании является параметром алгоритма. Блоки объединяются в задания с целью снизить трафик при обменах данными между ядрами процессора Cell.

Буфер (Buffer) — область памяти, в которой хранятся блоки. Количество блоков, помещающихся в буфер, является параметром алгоритма.

3.1.1. Параметры алгоритма

В Табл. 6 перечислены параметры алгоритма, их семантика и значение по умолчанию.

Табл. 6. Параметры алгоритма

Параметр	**Семантика**	**Значение по умолчанию**
SLAVES_NUM	Количество нитей шифрования	8
BLOCKS_IN_TASK	Количество блоков в задании	8
TASKS_IN_BUFFER	Количество заданий в буфере	32
KEY_FNAME	Имя файла, содержащего ключ	key.txt
INP_FNAME	Имя файла со входными данными	STDIN
OUT_FNAME	Имя файла для выходных данных	STDOUT

3.1.2. Классы

На Рис. 19 представлена диаграмма классов UML, реализующих параллельное шифрование.

Далее описана семантика этих классов.

Класс Master

Класс Master выполняет следующие основные функции.

- чтение и запись блоков;
- формирование и отправку заданий;
- управление экземплярами Slave.

Класс обладает следующими методами.

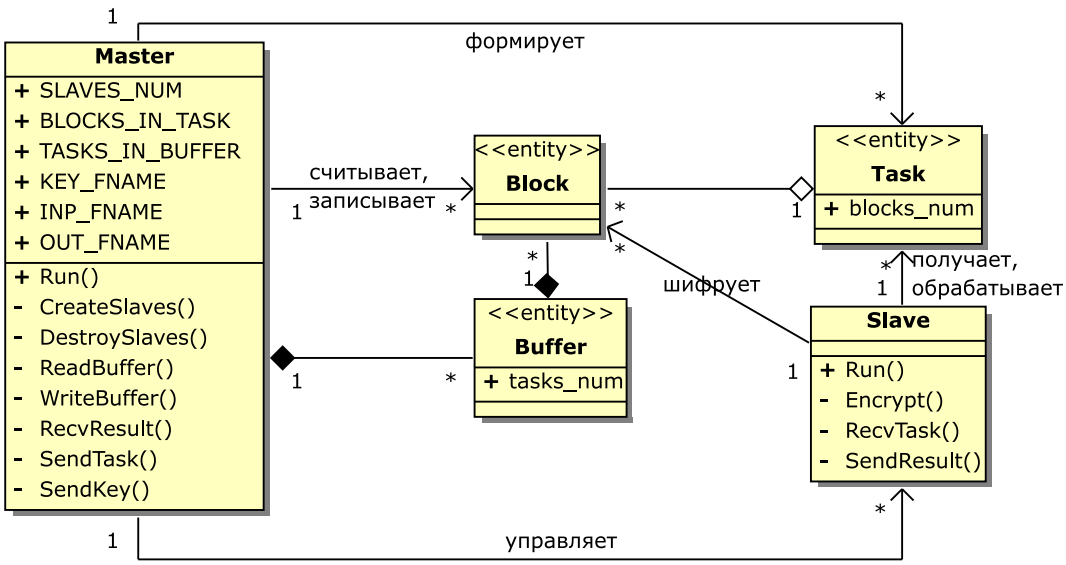

Рис. 19. Диаграмма классов, реализующих шифрование

1. *Run()* — рабочий цикл шифрования. Параметрами данного метода являются параметры алгоритма, перечисленные в Табл. 6. Данный метод описан диаграммой деятельности на Рис. 20.

2. *CreateSlaves()* — создание экземпляров класса Slave в количестве SLAVES_NUM.

3. *SendKey()* — рассылка ключа шифрования всем экземплярам класса Slave.

4. *DestroySlaves()* — уничтожение экземпляров класса Slave. При этом им рассылается пустое задание, которое служит сигналом к завершению.

5. *ReadBuffer()* — формирование буфера размером BLOCKS_IN_TASK * TASKS_IN_BUFFER блоков, считывание данных из файла в буфер и формирование заданий для блоков из этого буфера.

6. *WriteBuffer()* — вывод блоков зашифрованного текста из буфера *buffer* в выходной файл и уничтожение связанных с буфером заданий.

7. *RecvResult()* — получение результата выполнения задания от любого экземпляра класса Slave.

8. *SendTask(task, slave)* — передача задания *task* объекту *slave* для обработки.

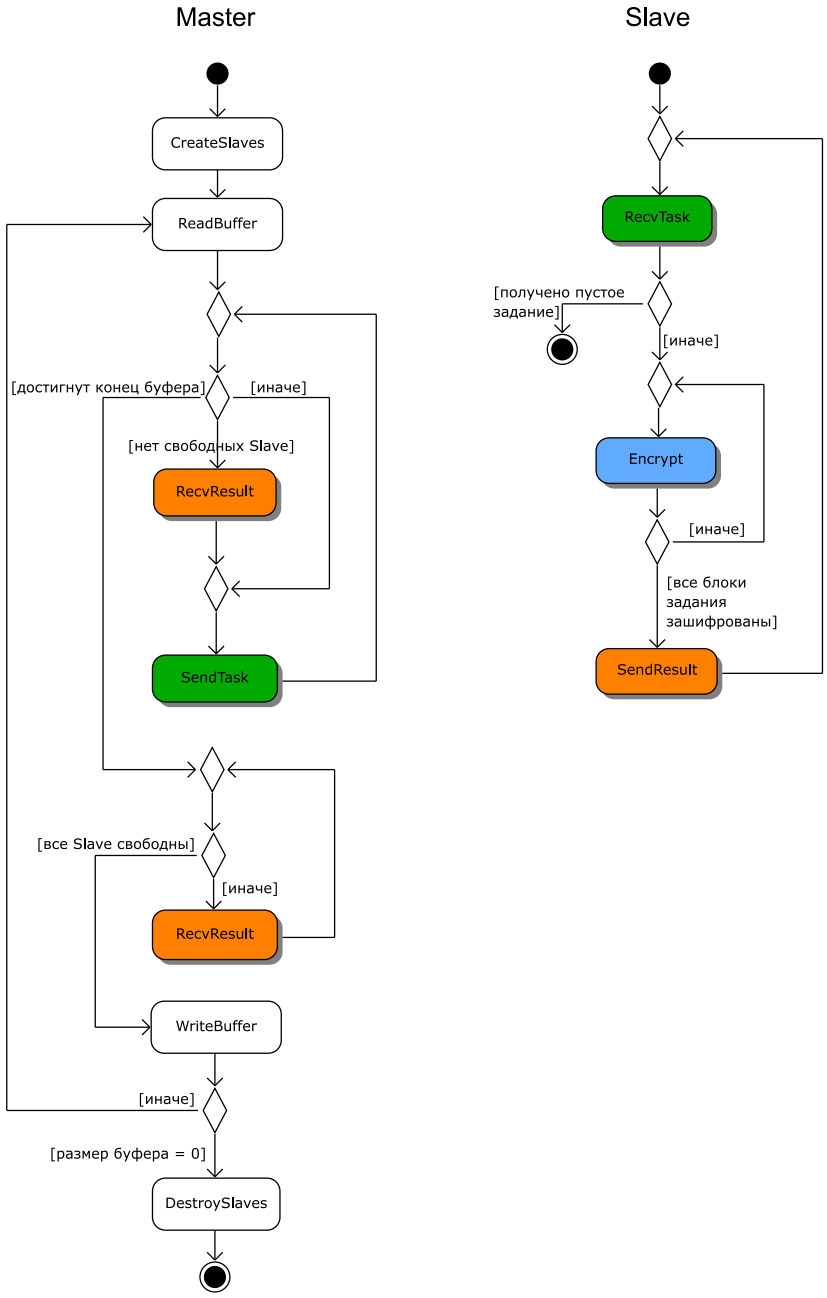

Рис. 20. Диаграммы деятельности классов Master и Slave

Класс Slave

Объекты класса Slave получают блоки открытого текста от Master, шифруют их и передают обратно результат. Объекты класса Slave могут находиться в одном из двух состояний, показанных на Рис. 21 с помощью

диаграммы состояний UML. Состояние «свободен» означает, что данному экземпляру класса Slave нужно отправить задание, а «занят» — что ему уже отправлено задание, но еще не получен результат.

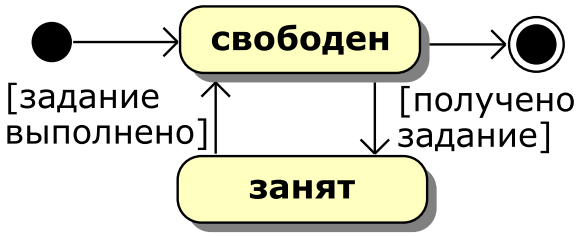

Рис. 21. Диаграмма состояний класса Slave

Класс обладает следующими методами.

1. *Run()* — запуск нити. Метод описывается диаграммой деятельности на Рис. 20.
2. *Encrypt(block)* — зашифровать блок *block* с помощью алгоритма AES.
3. *RecvTask()* — получение задания от объекта Master.
4. *SendResult(task)* — отправка результатов выполнения задания (task) объекту Master.

4. Реализация алгоритма PAES-CTR

В соответствии с разработанными алгоритмами, описанными в разделе 3, была реализована утилита PAES-CTR, выполняющая шифрование по стандарту AES в режиме CTR.

В данном разделе описана файловая структура текста программы и технические детали реализации алгоритмов.

4.1. Технические детали реализации

В реализации алгоритма нами для повышения эффективности его работы были использованы следующие возможности процессора Cell BE.

Прямая передача данных из основной памяти в локальную

Процессорв SPE может обрабатывать только те данные, которые находятся у него в локальной памяти размером 256 Кбайт. Для помещения данных в локальную память используется прямая асинхронная передача из основной памяти. Это позволяет свести время ожидания данных к нулю за счет использования двойной буферизации.

Обмен сообщениями с помощью «почтовых ящиков»

Каждый процессор SPE обладает так называемыми «почтовыми ящиками» (mailbox), представляющие собой очереди отправляемых и принимаемых сообщений, в которых каждое сообщение имеет размер 32 бита. Данная возможность используется для передачи процессорам SPE адресов в основной памяти, с помощью которых SPE инициируют прямую передачу данных. Один адрес передается с помозью двух сообщений, поскольку имеет размер 64 бита.

Векторные операции

PPE и SPE поддерживают SIMD-инструкции для векторов длиной 16 байтов. В алгоритме AES размер блока как раз составляет 16 байтов,

что позволяет свести такие операции как побитовое сложение двух блоков к вызову одной инструкции. Данное обстоятельство использовано в реализации метода Encrypt класса Slave.

4.2. Модульная структура текста программы

В данном разделе описываются модули, реализующие разработанный алгоритм. На Рис. 22 изображена модульная структура в виде диаграммы компонентов.

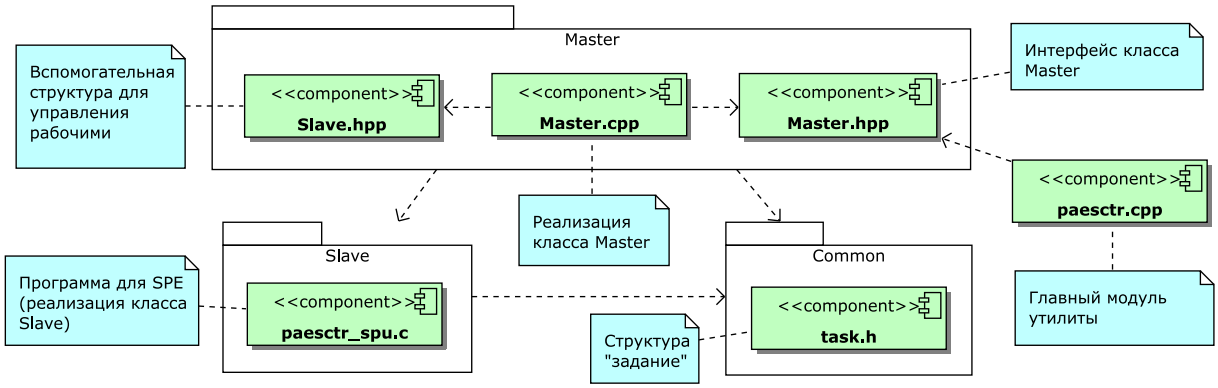

Рис. 22. Модульная структура текста программы

Исходные тексты программы размещены в сети Internet по адресу http://pcs.susu.ru/projects/3/3.PAES-CTR.shtml.

Модули разбиты на три пакета: **Master**, **Slave** и **Common**. В пакет **Master** включены модули, которые выполняются на управляющем ядре, в **Slave** — на вычислительных ядрах, а в **Common** включен один общий компонент — структура данных «Задание», передаваемая от мастера рабочим.

Далее изложена семантика компонентов программы.

paesctr.cpp

Главный модуль, содержит функцию *main()*. В этом модуле реализован разбор параметров запуска программы и создание экземпляра класса Master.

Master.hpp

Содержит описание интерфейса класса Master.

Master.cpp

Содержит реализацию класса Master.

Slave.hpp

Содержит описание вспомогательной структуры, с помощью которой Master управляет экземплярами Slave.

paesctr_spu.c

Программа для вычислительного ядра. Является реализацией класса Slave.

task.h

Структура «Задание», которая формируется мастером и передается рабочим.

5. Вычислительные эксперименты

В данном разделе описаны проведенные вычислительные эксперименты, исследующие эффективность предложенного алгоритма и оптимальные значения его параметров.

5.1. Цели экспериментов

Предлагаемый далее план экспериментов составлен исходя из следующих целей.

1. Вычислить, какое ускорение дает разработанный алгоритм, какова его эффективность и стоимость вычислений.
2. Выяснить расширяемость алгоритма.
3. Сравнить его с реализациями для других архитектур.
4. Определить оптимальные значения параметров алгоритма.

Ускорение

Ускорением называется величина

$$S_p(n) = \frac{T_1(n)}{T_p(n)},$$

где n — размер задачи, T_1 — время решения на одном процессоре, T_p — время решения на p процессорах [19].

Таким образом, чтобы вычислить ускорение для $p = \overline{1,8}$, нужно сделать 8 замеров времени при одинаковом размере данных n и разных p и, зная время $T_1(n)$, найти все $S_p(n)$. В разработанном алгоритме в качестве p выступает параметр SLAVES_NUM.

Эффективность

Под *эффективностью* понимается доля времени работы алгоритма, в течение которого процессоры реально используются для решения задачи [19], то есть эффективность — это величина

$$E_p(n) = \frac{T_1(n)}{p \cdot T_p(n)} = \frac{S_p(n)}{p}.$$

Таким образом, зная ускорение, можно посчитать эффективность.

Стоимость

Стоимость — это величина

$$C_p = p \cdot T_p.$$

Если C_p пропорционально времени выполнения наилучшего последовательного алгоритма, то параллельный алгоритм называется *стоимостно-оптимальным* [19]. Таким образом, имея результаты экспериментов, можно узнать, является ли разработанный алгоритм стоимостно-оптимальным.

Расширяемость

Расширяемость — зависимость времени выполнения алгоритма от размера задачи. Алгоритм считается хорошо расширяемым, если время выполнения зависит от размера задачи линейно.

5.2. Целевая система

Вычислительной системой для проведения экспериментов является Cellus в ЮУрГУ, имеющий конфигурацию, приведенную в Табл. 7.

Табл. 7. Конфигурация вычислительной системы Cellus

Процессор	PowerXCell8i
Количество процессоров	2
Тактовая частота процессора	3.2 ГГц
Пропускная способность шины EIB	200 ГБ/с

5.3. Результаты экспериментов

Для исследования разработанного алгоритма нами были проведены следующие вычислительные экспериментоы. Длина ключа во всех экспериментах — 16 байт.

5.3.1. Исследование ускорения, эффективности и стоимости

Данное исследование включает серию экспериментов с параметрами, перечисленными в Табл. 8. По результатам экспериментов вычислены все три показателя.

Табл. 8. Параметры экспериментов для исследования показателей эффективности

Вычислительная система	Cellus
Размер задачи	1 ГБ
SLAVES_NUM	меняеся от 1 до 8
BLOCKS_IN_TASK	1024
TASKS_IN_BUFFER	1024

Полученные результаты изображены в виде графиков на Рис. 23, 24 и 25.

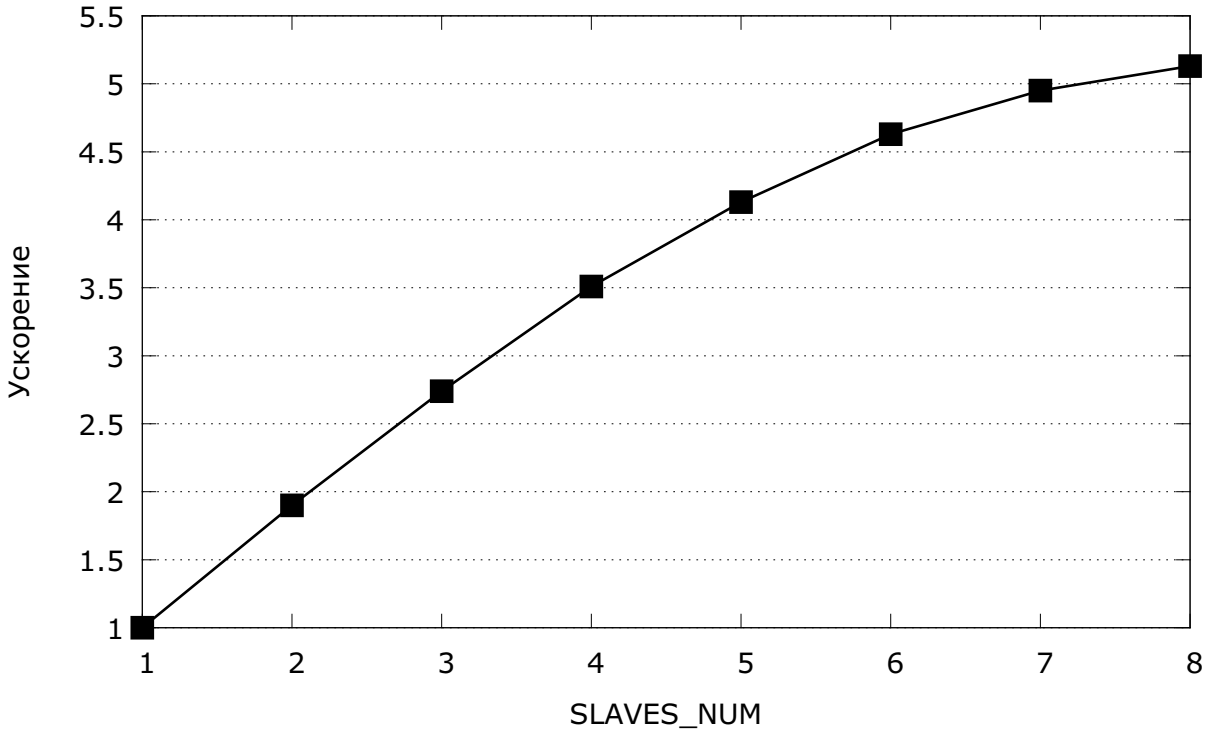

Рис. 23. Ускорение

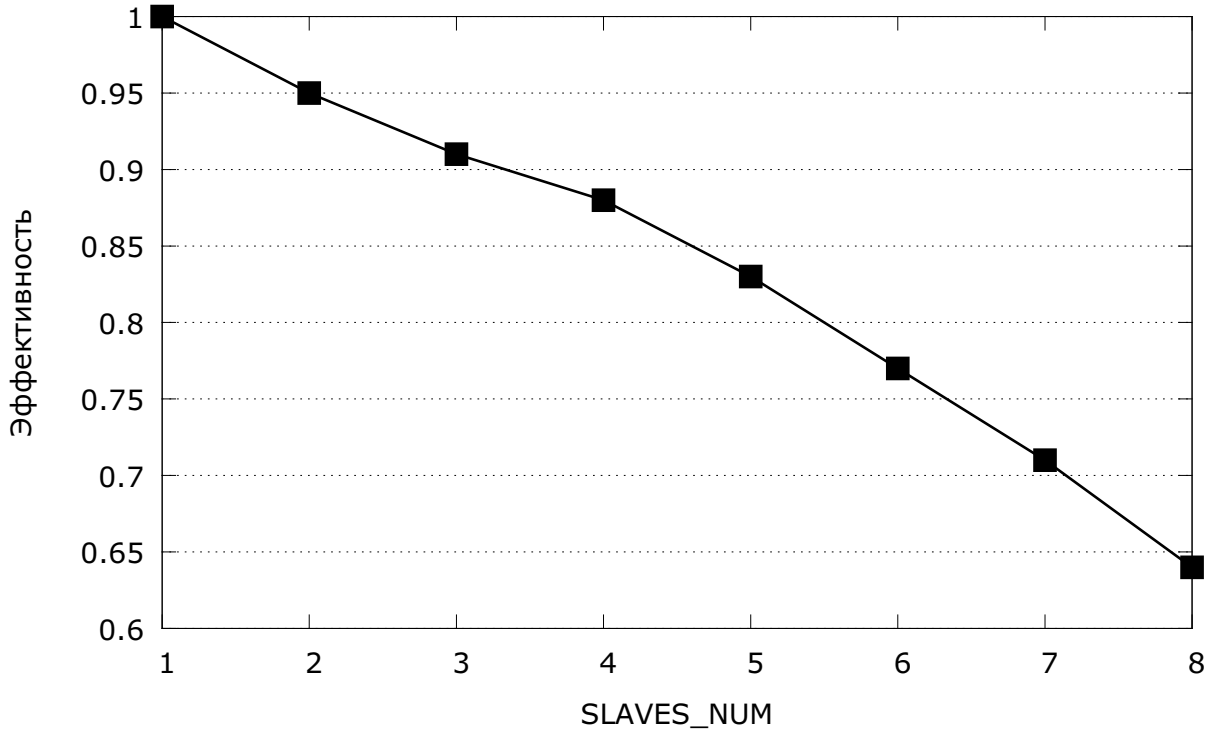

Рис. 24. Эффективность

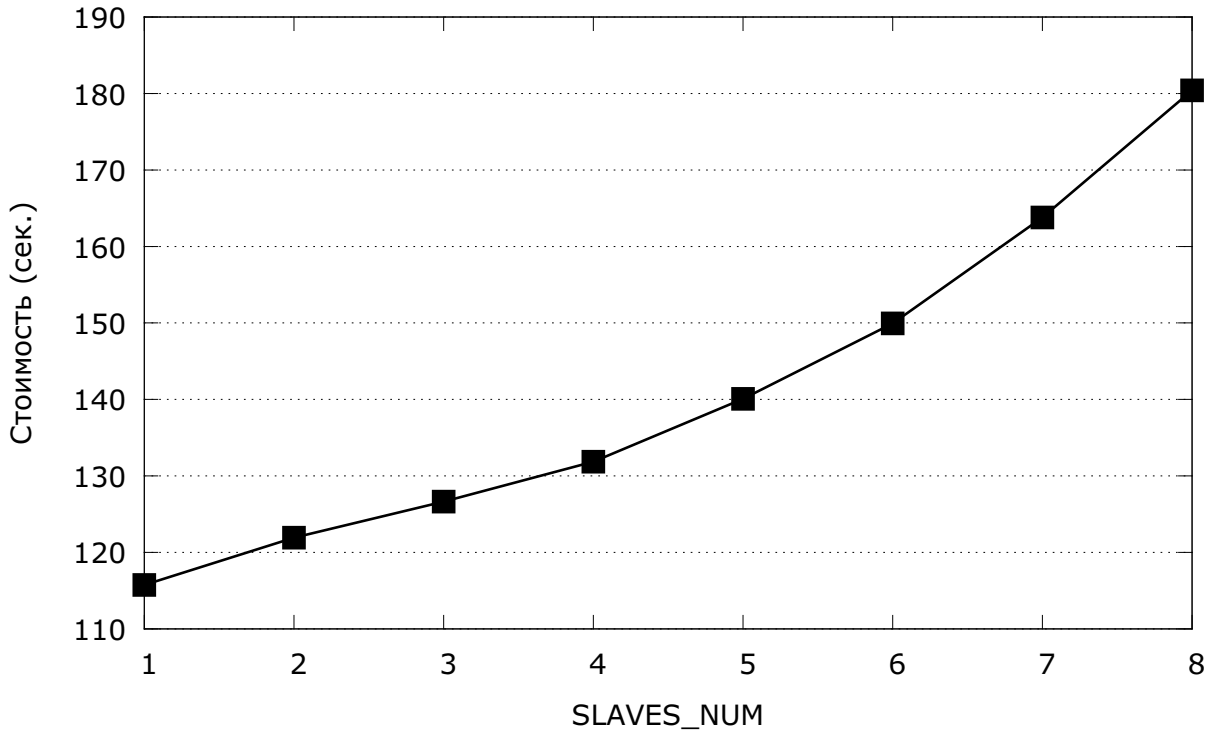

Рис. 25. Стоимость

5.3.2. Исследование расширяемости

Исследование расширяемости подразумевает эксперимент с приведенными в Табл. 9 параметрами.

Табл. 9. Параметры экспериментов для исследования расширяемости

Вычислительная система	Cellus
Размер задачи	0.25 ГБ, 0.5 ГБ, 1.0 ГБ
SLAVES_NUM	8
BLOCKS_IN_TASK	1024
TASKS_IN_BUFFER	1024

В соответствии с результатами проведенного эксперимента, изображенными в виде графика на Рис. 26, зависимость между временем и размером задачи является линейной.

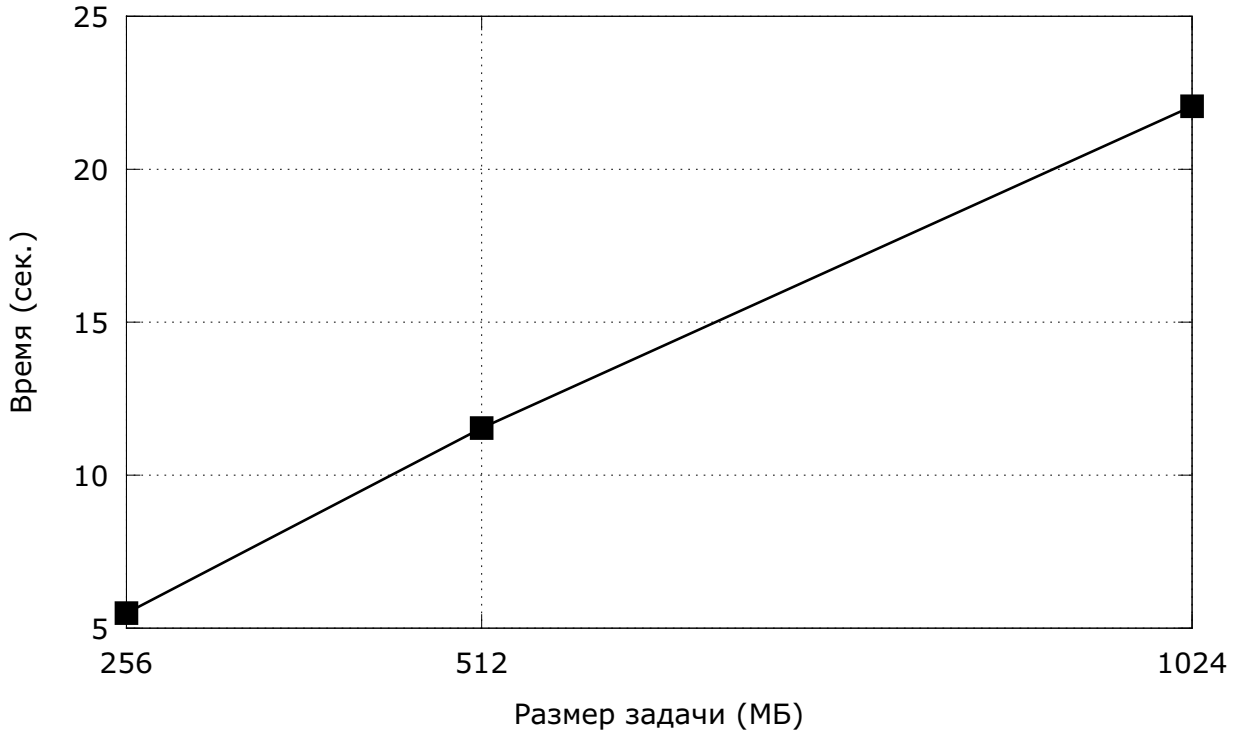

Рис. 26. Зависимость времени шифрования от размера задачи

5.3.3. Сравнение с реализациями для других архитектур

В данном эксперименте было произведено сравнение PAES-CTR с реализациями OpenSSL и GnuPG. OpenSSL и GnuPG шифруют только последовательно.

Табл. 10. Параметры эксперимента, сравнивающего алгоритм с реализациями для других архитектур

Вычислительная система	Cellus
Размер задачи	1.0 ГБ
Реализации	PAES-CTR, OpenSSL 0.9.8g, GnuPG 2.0.9

Результаты эксперимента изображены на Рис. 27.

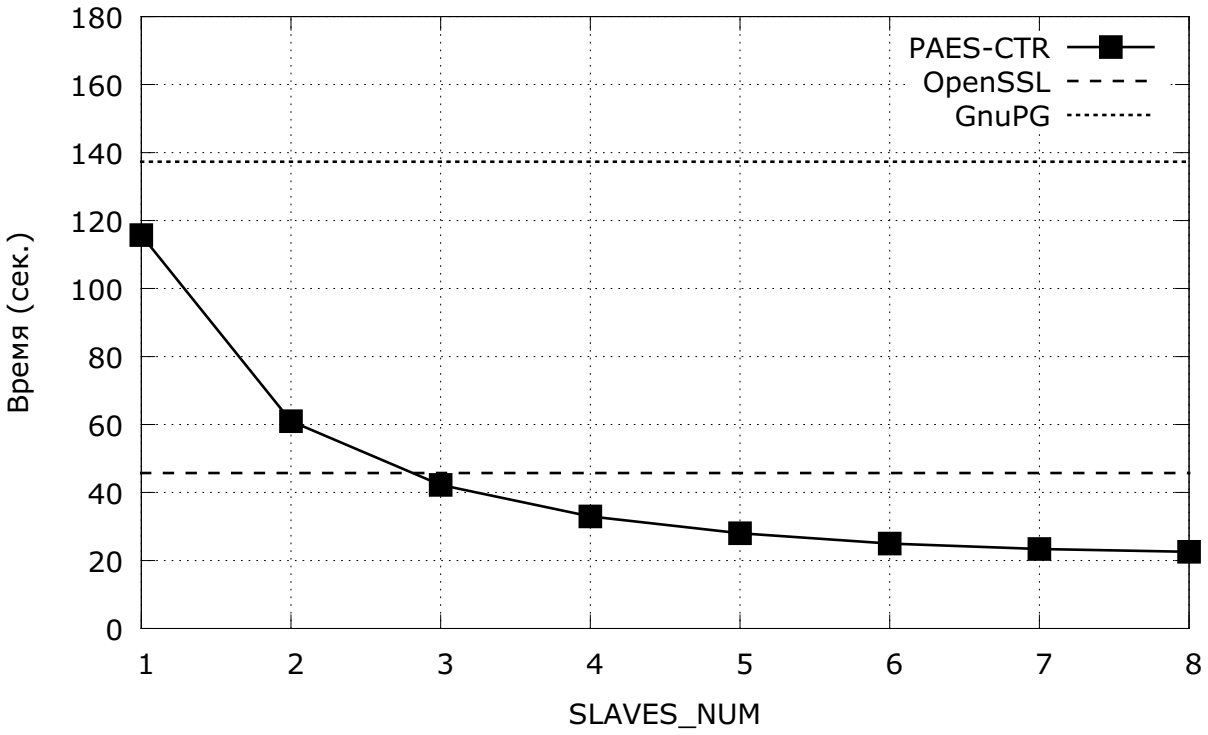

Рис. 27. Сравнение PAES-CTR с другими реализациями AES

5.3.4. Поведение алгоритма при изменении параметров

Данная серия экспериментов направлена на исследование влияния параметров алгоритма на время работы. Параметры данного эксперимента указаны в Табл. 11.

Табл. 11. Эксперименты для выявления оптимальных значений параметров

Вычислительная система	Cellus
Размер задачи	2 ГБ
SLAVES_NUM	8
BLOCKS_IN_TASK	меняется от 32 до 1024, удваиваясь
TASKS_IN_BUFFER	меняется от 32 до 1024, удваиваясь

На Рис. 28 показаны результаты эксперимента. На основе результатов данных экспериментов были выявлены оптимальные значения параметров TASKS_IN_BUFFER и BLOCKS_IN_TASK. Минимальное время достигается, когда оба параметра равны 1024.

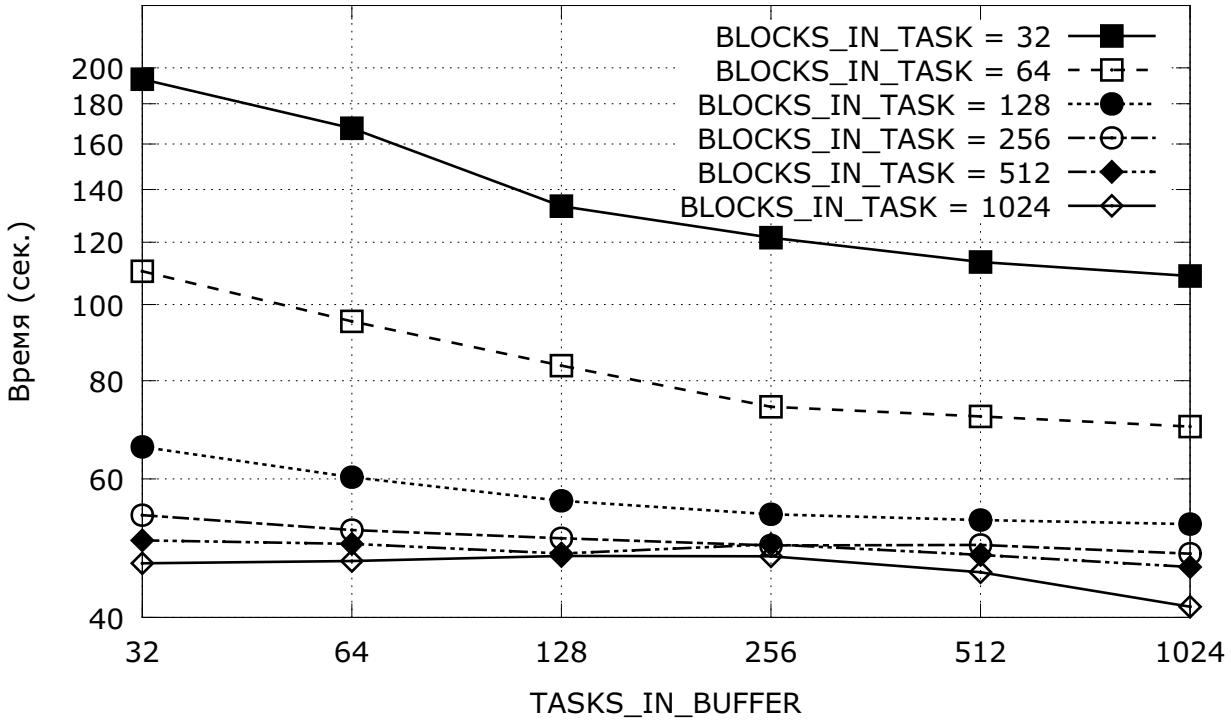

Рис. 28. Влияние параметров TASKS_IN_BUFFER и BLOCKS_IN_TASK

Также был проведен эксперимент, исследующий влияние опережающей подкачки с диска на время работы. Параметры эксперимента приведены в Табл. 12. Результаты эксперимента изображены на Рис. 29.

Табл. 12. Параметры эксперимента, исследующего влияние опережающей подкачки с диска

Вычислительная система	Cellus
Размер задачи	1.0 ГБ
SLAVES_NUM	меняется от 1 до 8
BLOCKS_IN_TASK	1024
TASKS_IN_BUFFER	1024
Опережающая подкачка	включена, выключена

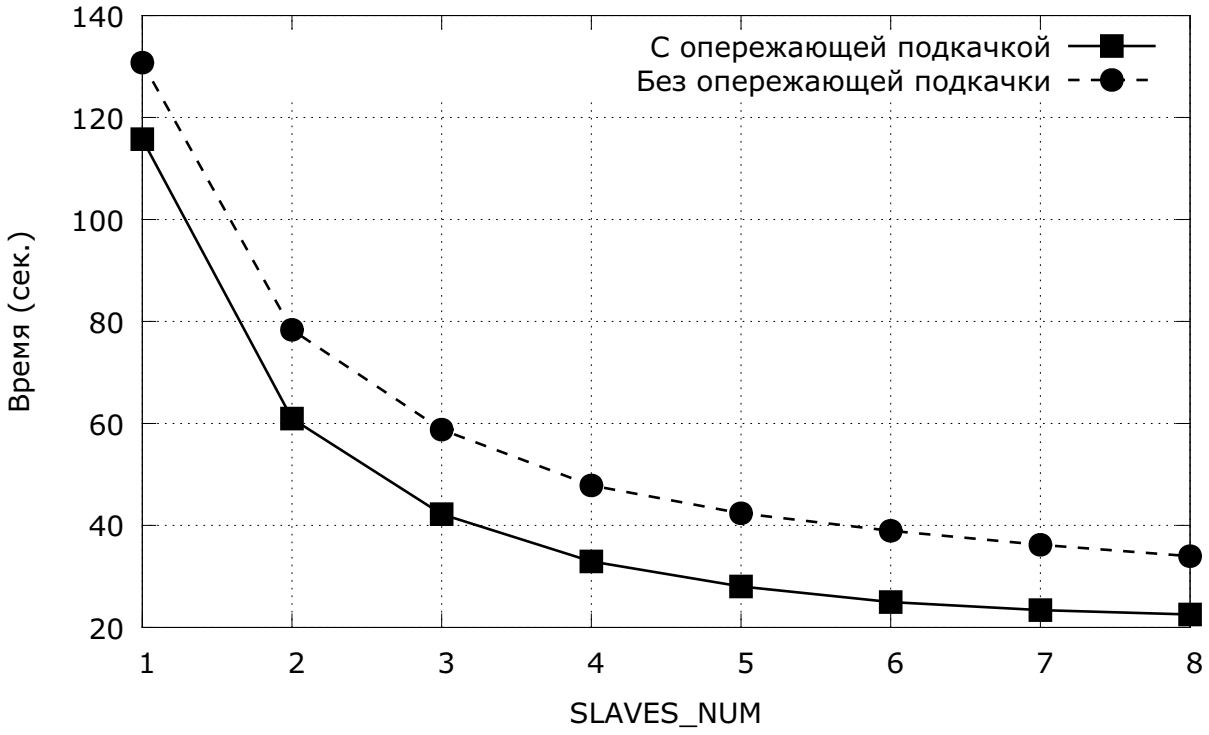

Рис. 29. Влияние опережающей подкачки с диска

Заключение

Данная работа посвящена адаптации алгоритма блочного симметичного шифрования AES для вычислительных систем на базе процессоров Cell.

В ходе работы были рассмотрены вопросы, связанные с задачей параллельного шифрования по стандарту AES на вычислительных системах на базе процессоров Cell.

Изучена архитектура процессора Cell BE, модель и системные средства разработки приложений для данного процессора. Изучен алгоритм блочного симметричного шифрования AES.

На базе модели параллелизма по данным и модели «мастер-рабочие» разработана параллельная версия алгоритма AES, адаптированная для архитектуры Cell BE.

Для исследования эффективности и нахождения оптимальных параметров разработанного алгоритма проведены вычислительные эксперименты на вычислительной системе, включающей в себя два процессора Cell BE. Получено пятикратное ускорение при использовании 8 ядер SPE.

АПРОБАЦИЯ РАБОТЫ

Результаты работы докладывались автором на конференции «Разработки РФ по приоритетным направлениям развития науки, технологии и техники» (11–13 мая 2009 г., Челябинск).

ОСНОВНЫЕ РЕЗУЛЬТАТЫ РАБОТЫ

1. Разработан и реализован алгоритм шифрования по стандарту AES в режиме CTR, оптимизированный для вычислительных систем на базе процессора Cell.

2. Проведены вычислительные эксперименты, подтверждающие эффективность разработанного алгоритма. На основе данных экспериментов определены оптимальные значения параметров алгоритма.

НАПРАВЛЕНИЯ ДАЛЬНЕЙШИХ ИССЛЕДОВАНИЙ

Дальнейшие исследования могут быть направлены на опережающую загрузку данных в локальную память SPE, а также на адаптацию других режимов шифрования алгоритма AES.

Литература

1. Specification for the Advanced Encryption Standard (AES). — Federal Information Processing Standards Publication 197. — 2001. URL: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf (дата обращения: 14.10.2008).

2. *Daemen, J.* The design of Rijndael: AES — the Advanced Encryption Standard / J. Daemen, V. Rijmen. — Springer-Verlag, 2002.

3. Report on the Development of the Advanced Encryption Standard (AES). — National Institute of Standards and Technology. URL: http://csrc.nist.gov/archive/aes/round2/r2report.pdf (дата обращения: 02.02.2008).

4. CNSS Policy No. 15, Fact Sheet No. 1, National Policy on the Use of the Advanced Encryption Standard (AES) to Protect National Security Systems and National Security Information. — National Security Agency. URL: http://www.cnss.gov/Assets/pdf/cnssp_15_fs.pdf (дата обращения: 17.10.2008).

5. *Dongarra J.* An Overview of High Performance Computing and Challenges for the Future // High Performance Computing for Computational Science - VECPAR 2008. — Springer Berlin, 2008. URL: http://www.springerlink.com/content/53q3h11t107552n7 (дата обращения: 12.05.2009).

6. *Gietl H., Meuer H.* The Top Trends in High Performance Computing. URL: http://top500.org/blog/2009/05/20/-top_trends_high_performance_computing (дата обращения: 21.05.2009).

7. Processor Generation share for 11/2008. URL: http://www.top500.org/charts/list/32/procgen (дата обращения: 02.05.2009).

8. Roadrunner in Top500. URL: http://www.top500.org/system/9707 (дата обращения: 02.05.2009).

9. Programming Tutorial // Software Development Kit for Multicore Acceleration Version 3.0. — IBM, 2007.

10. *Williams S., Shalf J., Oliker L.* The Potential of the Cell Processor for Scientific Computing // CF '06: Proceedings of the 3rd conference on Computing frontiers. — Italy.
URL: http://bebop.cs.berkeley.edu/pubs/williams2006-cell-scicomp.pdf (дата обращения: 02.04.2009).

11. *Gedik B., Bordawekar R.R., Yu P.S.* CellSort: High Performance Sorting on the Cell Processor // VLDB'07. — Vienna, 2007.

12. *Alam S.R., Meredith J.S., Vetter J.S.* Balancing Productivity and Performance on the Cell Broadband Engine. // 2007 IEEE International Conference on Cluster Computing. c. 149-158

13. *Bader D.A., Agarwal V., Madduri K.* On the Design and Analysis of Irregular Algorithmson the Cell Processor: A Case Study of List Ranking. URL: http://www.cc.gatech.edu/~bader/papers/ListrankCell-IPDPS2007.pdf (дата обращения: 24.03.2009).

14. *Chow A.C., Fossum G.C., Brokenshire D.A.* A Programming Example: Large FFT on the Cell Broadband Engine. URL: http://www.t-platforms.ru/pdf/GSPx_FFT_paper_legal_0115.pdf (дата обращения: 25.03.2009).

15. *Costigan N., Scott M.* Accelerating SSL using the Vector processors in IBM's Cell Broadband Engine for Sony's Playstation 3.
URL: http://eprint.iacr.org/2007/061.pdf (дата обращения: 20.03.2009).

16. Dma-based prefetching for i/o-intensive workloads on the cell architecture. URL: http://doi.acm.org/10.1145/1366230.1366236

17. *Лидл Р., Нидеррайтер Г.* Конечные поля. В 2-х томах. — М.:Мир, 1988

18. *Dworkin M.* NIST Special Publication 800-38A, 2001 Edition. Recommendation for Block Cipher Modes of Operation. Methods and Techniques.
URL: http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf (дата обращения: 14.10.2008).

19. *Гергель В.П.* Теория и практика параллельных вычислений. — ИНТУИТ, Бином. Лаборатория знаний, 2007. 424 с.

Приложение. Примеры шифрования и расшифрования модельных данных

В Табл. 1 приведен пример шифрования 128-битным ключом. Входной блок {00 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff} шифруется ключом {00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f}.

Табл. 1. Пример шифрования

Раунд	**Процедура**	**Результат**
	входной блок	00112233445566778899aabbccddeeff
	ключ	000102030405060708090a0b0c0d0e0f
	AddRoundKey	00102030405060708090a0b0c0d0e0f0
1	SubBytes	63cab7040953d051cd60e0e7ba70e18c
	ShiftRows	6353e08c0960e104cd70b751bacad0e7
	MixColumns	5f72641557f5bc92f7be3b291db9f91a
	ключ раунда	d6aa74fdd2af72fadaa678f1d6ab76fe
	AddRoundKey	89d810e8855ace682d1843d8cb128fe4
2	SubBytes	a761ca9b97be8b45d8ad1a611fc97369
	ShiftRows	a7be1a6997ad739bd8c9ca451f618b61
	MixColumns	ff87968431d86a51645151fa773ad009
	ключ раунда	b692cf0b643dbdf1be9bc5006830b3fe
	AddRoundKey	4915598f55e5d7a0daca94fa1f0a63f7
3	SubBytes	3b59cb73fcd90ee05774222dc067fb68
	ShiftRows	3bd92268fc74fb735767cbe0c0590e2d
	MixColumns	4c9c1e66f771f0762c3f868e534df256
	ключ раунда	b6ff744ed2c2c9bf6c590cbf0469bf41
	AddRoundKey	fa636a2825b339c940668a3157244d17
4	SubBytes	2dfb02343f6d12dd09337ec75b36e3f0
	ShiftRows	2d6d7ef03f33e334093602dd5bfb12c7
	MixColumns	6385b79ffc538df997be478e7547d691
	ключ раунда	47f7f7bc95353e03f96c32bcfd058dfd
	AddRoundKey	247240236966b3fa6ed2753288425b6c

Раунд	Процедура	Результат
5	SubBytes	36400926f9336d2d9fb59d23c42c3950
	ShiftRows	36339d50f9b539269f2c092dc4406d23
	MixColumns	f4bcd45432e554d075f1d6c51dd03b3c
	ключ раунда	3caaa3e8a99f9deb50f3af57adf622aa
	AddRoundKey	c81677bc9b7ac93b25027992b0261996
6	SubBytes	e847f56514dadde23f77b64fe7f7d490
	ShiftRows	e8dab6901477d4653ff7f5e2e747dd4f
	MixColumns	9816ee7400f87f556b2c049c8e5ad036
	ключ раунда	5e390f7df7a69296a7553dc10aa31f6b
	AddRoundKey	c62fe109f75eedc3cc79395d84f9cf5d
7	SubBytes	b415f8016858552e4bb6124c5f998a4c
	ShiftRows	b458124c68b68a014b99f82e5f15554c
	MixColumns	c57e1c159a9bd286f05f4be098c63439
	ключ раунда	14f9701ae35fe28c440adf4d4ea9c026
	AddRoundKey	d1876c0f79c4300ab45594add66ff41f
8	SubBytes	3e175076b61c04678dfc2295f6a8bfc0
	ShiftRows	3e1c22c0b6fcbf768da85067f6170495
	MixColumns	baa03de7a1f9b56ed5512cba5f414d23
	ключ раунда	47438735a41c65b9e016baf4aebf7ad2
	AddRoundKey	fde3bad205e5d0d73547964ef1fe37f1
9	SubBytes	5411f4b56bd9700e96a0902fa1bb9aa1
	ShiftRows	54d990a16ba09ab596bbf40ea111702f
	MixColumns	e9f74eec023020f61bf2ccf2353c21c7
	ключ раунда	549932d1f08557681093ed9cbe2c974e
	AddRoundKey	bd6e7c3df2b5779e0b61216e8b10b689
10	SubBytes	7a9f102789d5f50b2beffd9f3dca4ea7
	ShiftRows	7ad5fda789ef4e272bca100b3d9ff59f
	ключ раунда	13111d7fe3944a17f307a78b4d2b30c5
	AddRoundKey	69c4e0d86a7b0430d8cdb78070b4c55a
	выходной блок	69c4e0d86a7b0430d8cdb78070b4c55a

В Табл. 2 приведен пример расшифрования 128-битным ключом. Входной блок {69 c4 e0 d8 6a 7b 04 30 d8 cd b7 80 70 b4 c5 5a} расшифруется ключом {00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f}.

Табл. 2. Пример расшифрования

Раунд	**Процедура**	**Результат**
	входной блок	69c4e0d86a7b0430d8cdb78070b4c55a
	ключ раунда	13111d7fe3944a17f307a78b4d2b30c5
	AddRoundKey	7ad5fda789ef4e272bca100b3d9ff59f
1	InvShiftRows	7a9f102789d5f50b2beffd9f3dca4ea7
	InvSubBytes	bd6e7c3df2b5779e0b61216e8b10b689
	ключ раунда	549932d1f08557681093ed9cbe2c974e
	AddRoundKey	e9f74eec023020f61bf2ccf2353c21c7
2	InvMixColumns	54d990a16ba09ab596bbf40ea111702f
	InvShiftRows	5411f4b56bd9700e96a0902fa1bb9aa1
	InvSubBytes	fde3bad205e5d0d73547964ef1fe37f1
	ключ раунда	47438735a41c65b9e016baf4aebf7ad2
	AddRoundKey	baa03de7a1f9b56ed5512cba5f414d23
3	InvMixColumns	3e1c22c0b6fcbf768da85067f6170495
	InvShiftRows	3e175076b61c04678dfc2295f6a8bfc0
	InvSubBytes	d1876c0f79c4300ab45594add66ff41f
	ключ раунда	14f9701ae35fe28c440adf4d4ea9c026
	AddRoundKey	c57e1c159a9bd286f05f4be098c63439
4	InvMixColumns	b458124c68b68a014b99f82e5f15554c
	InvShiftRows	b415f8016858552e4bb6124c5f998a4c
	InvSubBytes	c62fe109f75eedc3cc79395d84f9cf5d
	ключ раунда	5e390f7df7a69296a7553dc10aa31f6b
	AddRoundKey	9816ee7400f87f556b2c049c8e5ad036

Раунд	Процедура	Результат
5	InvMixColumns	e8dab6901477d4653ff7f5e2e747dd4f
	InvShiftRows	e847f56514dadde23f77b64fe7f7d490
	InvSubBytes	c81677bc9b7ac93b25027992b0261996
	ключ раунда	3caaa3e8a99f9deb50f3af57adf622aa
	AddRoundKey	f4bcd45432e554d075f1d6c51dd03b3c
6	InvMixColumns	36339d50f9b539269f2c092dc4406d23
	InvShiftRows	36400926f9336d2d9fb59d23c42c3950
	InvSubBytes	247240236966b3fa6ed2753288425b6c
	ключ раунда	47f7f7bc95353e03f96c32bcfd058dfd
	AddRoundKey	6385b79ffc538df997be478e7547d691
7	InvMixColumns	2d6d7ef03f33e334093602dd5bfb12c7
	InvShiftRows	2dfb02343f6d12dd09337ec75b36e3f0
	InvSubBytes	fa636a2825b339c940668a3157244d17
	ключ раунда	b6ff744ed2c2c9bf6c590cbf0469bf41
	AddRoundKey	4c9c1e66f771f0762c3f868e534df256
8	InvMixColumns	3bd92268fc74fb735767cbe0c0590e2d
	InvShiftRows	3b59cb73fcd90ee05774222dc067fb68
	InvSubBytes	4915598f55e5d7a0daca94fa1f0a63f7
	ключ раунда	b692cf0b643dbdf1be9bc5006830b3fe
	AddRoundKey	ff87968431d86a51645151fa773ad009
9	InvMixColumns	a7be1a6997ad739bd8c9ca451f618b61
	InvShiftRows	a761ca9b97be8b45d8ad1a611fc97369
	InvSubBytes	89d810e8855ace682d1843d8cb128fe4
	ключ раунда	d6aa74fdd2af72fadaa678f1d6ab76fe
	AddRoundKey	5f72641557f5bc92f7be3b291db9f91a
10	InvMixColumns	6353e08c0960e104cd70b751bacad0e7
	InvShiftRows	63cab7040953d051cd60e0e7ba70e18c
	InvSubBytes	00102030405060708090a0b0c0d0e0f0
	ключ раунда	000102030405060708090a0b0c0d0e0f
	AddRoundKey	00112233445566778899aabbccddeeff
	выходной блок	00112233445566778899aabbccddeeff