НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
июль–август 2018 Том 18 № 4 ISSN 2226-1494 http://ntv.ifmo.ru/
SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS
July–August 2018 Vol. 18 No 4 ISSN 2226-1494 http://ntv.ifmo.ru/en

УДК 004.056.55

ИССЛЕДОВАНИЕ СПОСОБОВ СКОРОСТНОЙ РЕАЛИЗАЦИИ ЭЛЕМЕНТОВ СИММЕТРИЧНЫХ АЛГОРИТМОВ ШИФРОВАНИЯ ПРИ ПРОВЕДЕНИИ ВЫЧИСЛЕНИЙ НА ГРАФИЧЕСКОМ ПРОЦЕССОРЕ

В.А. Удальцов[a], Н.С. Кармановский[a]

[a] Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация
Адрес для переписки: udalv1@rambler.ru

Информация о статье
Поступила в редакцию 12.03.18, принята к печати 25.05.18
doi: 10.17586/2226-1494-2018-18-4-646-653
Язык статьи – русский

Ссылка для цитирования: Удальцов В.А., Кармановский Н.С. Исследование способов скоростной реализации элементов симметричных алгоритмов шифрования при проведении вычислений на графическом процессоре // Научно-технический вестник информационных технологий, механики и оптики. 2018. Т. 18. № 4. С. 646–653. doi: 10.17586/2226-1494-2018-18-4-646-653

Аннотация

Предмет исследования. Проведено исследование преобразований, используемых в современных симметричных алгоритмах, с целью определения наиболее скоростных способов их реализации на графическом процессоре с использованием технологий CUDA и OpenCL. **Метод.** Для достижения поставленной цели рассмотрены LSX и ARX структуры блочных алгоритмов на примере шифров AES, «Кузнечик», LEA, Rectangle, Simon и Speck. Определены наиболее часто применяемые преобразования, в число которых входят умножение в полях Галуа, применение таблиц подстановок, побитовые операции, сложение длинных чисел. Как неотъемлемая часть вычислений на графических устройствах рассмотрен обмен данных с глобальной памятью. Рассмотрены варианты реализации данных вычислений, разработаны и апробированы синтетические тесты, предназначенные для определения времени их выполнения. **Основные результаты.** Определены наилучшие способы реализации перечисленных преобразований. При выполнении умножения в полях Галуа в случае, если один из множителей является константным, наилучшее время показал метод с использованием таблицы предвычислений. Выявлено, что наиболее эффективным с точки зрения скорости является хранение таблиц замены в разделяемой памяти. Осуществление побитовых операций и сложение длинных чисел наиболее эффективно в случае разбиения входных данных на 8-битные элементы. В качестве апробации результатов осуществлена реализация алгоритма CLEFIA. Время шифрования 1 ГБ данных составило 1542 мс, что в 16 раз меньше времени шифрования на центральном процессоре. Применение вариантов реализации исследуемых преобразований, показавших худшие временные результаты при проведении синтетических тестов на графических процессорах, дает четырехкратное увеличение скорости в сравнении с центральным процессором. **Практическая значимость.** Результаты проведенного исследования применимы при использовании графических процессоров для быстрой и эффективной реализации существующих алгоритмов шифрования. Результаты исследования могут служить основой разработки новых алгоритмов шифрования с применением графических процессоров.

Ключевые слова
CUDA, OpenCL, криптографические преобразования, симметричные алгоритмы, ускорение шифрования

STUDY OF HIGH-SPEED REALIZATION TECHNICS FOR ELEMENTS OF SYMMETRIC ENCRYPTION ALGORITHMS DURING CALCULATIONS ON GRAPHICS PROCESSOR

V.A. Udaltsov[a], N.S. Karmanovskiy[a]

[a] ITMO University, Saint Petersburg, 197101, Russian Federation
Corresponding author: udalv1@rambler.ru

Article info
Received 12.03.18, accepted 25.05.18
doi: 10.17586/2226-1494-2018-18-4-646-653
Article in Russian

For citation: Udaltsov V.A., Karmanovskiy N.S. Study of high-speed realization technics for elements of symmetric encryption algorithms during calculations on graphics processor. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2018, vol. 18, no. 4, pp. 646–653 (in Russian). doi: 10.17586/2226-1494-2018-18-4-646-653

Abstract

Subject of Research. The paper deals with the research of transformations used in up-to-date symmetric algorithms aimed at definition of the most high-speed ways of their realization on the graphics processor with the use of CUDA and OpenCL technologies. **Method.** To achieve this goal, we considered LSX and ARX structures of block algorithms on the example of the following ciphers: AES, «Kuznyechik», LEA, Rectangle, Simon and Speck. The main types of transformations were detected, which include: multiplication in Galois fields, the use of lookup tables, bitwise operations, long number addition and data exchange with the global memory as an integral part of the calculations on graphics devices. The variants of the implementation of these calculations were considered and synthetic tests were carried out to determine their execution time. **Main Results**. The best ways for implementation of these transformations were determined. When performing multiplication in Galois fields, if one of the multipliers is constant, the best time was shown by the method using the pre-calculation table. It was also found that the most effective in terms of speed is the storage of replacement tables in shared memory and the implementation of bitwise operations with the division of input data into 8-bit elements, as in the case of long numbers addition. The result approbation was carried out by CLEFIA algorithm realization.The encryption time of 1 GB of data was 1542 mc. This result is 16 times less than the encryption time on the general-purpose processor. The application of realization variants for studied transformations that show the worst time results during synthetic tests on graphics processors gives fourfold speed increase compared with the central processor. **Practical Relevance.** The study results are applicable for the speedy and efficient use of graphics processors in the implementation of existing encryption algorithms. The results can become the basis for the development of new encryption algorithms with the use of graphics processors.

Keywords

CUDA, OpenCL, cryptographic transformations, symmetric algorithms, encryption acceleration

Введение

На данный момент существует большое количество работ, посвященных применению графических процессоров для криптографических вычислений. Для алгоритмов AES (Advanced Encryption Standard) [1], Blowfish [2], RSA (Rivest, Shamir, Adleman) [3], а также отечественных «Магма» [4] и «Кузнечик» [5] продемонстрирован существенный прирост скорости шифрования при реализации на графическом процессоре по сравнению с реализациями на центральном процессоре. Оцениваются возможности применения графического оборудования в качестве широко доступной альтернативы дорогостоящим криптоплатам или для частичного перенесения нагрузки с центрального процессора. Недостатком перечисленных работ является отсутствие подробного описания технических особенностей используемых реализаций.

В статьях [6, 7] отсутствует этот недостаток. Однако они направлены на оптимизацию конкретных алгоритмов, конкретного шифра, а не отдельных криптографических преобразований. Это ограничивает применение упомянутых работ при анализе других алгоритмов, к примеру, при криптоанализе разрабатываемых или недавно опубликованных шифров.

Несмотря за значительные вычислительные мощности современных видеокарт, их возможности ограничены применением только массивно распараллеленных алгоритмов. Для криптографии это может вызвать падение криптостойкости из-за ограниченности в выборе режима шифрования. Но, с учетом последних тенденций развития легковесной криптографии, описанных в [8–10], а также разработки специализированных алгоритмов для RFID (Radio Frequency IDentification) меток [11] и Интернета вещей [12], скоростные качества которых достигаются за счет уменьшения криптостойкости, можно ожидать появления алгоритмов, предназначенных для применения на графических процессорах.

Таким образом, быстрое и эффективное использование графических процессоров для новых алгоритмов шифрования может быть востребовано не только для увеличения скорости шифрования, но и для криптоанализа. Исходя из этого, целью настоящей работы является определение ключевых частей симметричных алгоритмов шифрования и наиболее скоростного способа их реализации.

Анализ алгоритмов шифрования

Структуры алгоритмов. Ключевым параметром криптографических алгоритмов является их структура. Ниже приводятся наиболее распространенные на данный момент структуры блочных шифров:

- LSX-структура (Linear transformation, S-box, XOR (eXception OR)) включает в себя линейное преобразование, которое обеспечивает его свойства диффузии и обычно является наиболее медленной частью шифра [13], нелинейное преобразование, выполняющее отображение n-битного входного блока на выход длиной m бит [14] и используемое для противодействия дифференциальному криптоанализу, а также операцию XOR, используемую для смешения с раундовым ключом;
- ARX-структура (Addition, Rotation, XOR) состоит из сложения по модулю $2n$, которое в данном случае обеспечивает нелинейность алгоритма [15], битовых смещений, выполняющих роль линейного преобразования, и операции XOR.

Для исследования были выбраны алгоритмы, которые, согласно [16], показали наилучшую производительность при программной реализации. К ним был добавлен шифр «Кузнечик», являющийся одним

из самых молодых алгоритмов, который, следовательно, позволяет оценить тенденции развития симметричной криптографии. Далее приведено краткое описание наиболее значимых для данной работы алгоритмов.

– AES[1] является наиболее популярным алгоритмом симметричного шифрования, имеет длину ключа 128, 192 и 256 бит и размер блока 128 бит. Он имеет LSX-структуру, при этом для S-преобразования используется таблица из 256 однобайтовых элементов, преобразование L состоит из циклического сдвига строк матрицы шифруемого блока и смешения столбцов, в рамках которого используется умножение и сложение в поле Галуа.
– «Кузнечик»[2] также имеет LSX-структуру, длину блока 128 бит и длину ключа 256 бит, для S-преобразования используется таблица подстановок, содержащая 256 элементов по одному байту, L-преобразование осуществляется посредством перемножения элемента блока с заданными константами и последующего сложения произведений в поле Галуа.
– Simon и Speck [17, 18] – ARX-шифры, основанные на сети Фейстеля, имеющие длину ключа от 64 до 256 бит и длину блока от 32 до 128 бит. Кроме операций циклического сдвига и сложения по модулю, в алгоритме Simon добавляется побитовое И.
– LEA [19] – ARX-шифр, имеющий длину ключа 128, 192 и 256 бит и размер блока 128 бит. Отличительной особенностью является его структура – по сути, сеть Фейстеля, состоящая из четырех меняющихся местами блоков.
– Rectangle [20] – LSX-шифр с длиной ключа 80 или 128 бит и длиной блока 64 бита. Он имеет похожую на AES-форму представления шифруемого блока в виде матрицы, но минимальной единицей данных является бит, а не байт; это отражается на L-преобразовании, состоящим из циклического битового сдвига строки, и специфическом S-преобразовании, выполняемом побитого относительно столбцов и использующем таблицу подстановок из 16 4-битных элементов.

Режимы работы. Кроме структуры данных, важную роль играет режим работы алгоритма, определяемый ГОСТ 34.11[3]. Приведенный в нем перечень можно разделить на две группы:

– режимы, имеющие связи между блоками шифртекста, – гаммирование с обратной связью по выходу, гаммирование с обратной связью по шифртексту, режим простой замены с зацеплением и выработка имитовставки;
– режимы, не имеющие связи между блоками шифртекста, – гаммирование и режим простой замены.

Так как режимы первой группы требуют последовательного вычисления блоков шифртекста, далее рассматриваются только режимы второй группы. Режим простой замены состоит только из преобразований, включенных в сам алгоритм, и не требует дополнительных вычислений. Режим гаммирования дополнительно требует вычисления начального значения, при этом длина начального значения составляет 128 бит, следовательно, требует реализации сложения длинных чисел.

Исследование способа реализации криптографических преобразований

Особенности реализации. Последующие реализации рассматриваемых криптографических преобразований, предназначенные для выполнения на графическом процессоре, были осуществлены с помощью наиболее популярных на данный момент технологий – CUDA и OpenCL. Выбор технологии CUDA обусловлен ее превосходством в сравнении с аналогами в возможностях языка и компилятора, OpenCL обладает схожими моделями памяти и вычислений, позволяющими осуществлять эффективное использование графических процессоров, обеспечивая при этом кроссплатформенность. DirectCompute, DirectX и Compute Shader OpenGL не использовались в работе, так как применяемые в них шейдерные языки предназначены для описания графических вычислений и не обеспечивают необходимую гибкость для эффективной реализации криптографических преобразований. OpenAAC обеспечивает значительно большую гибкость и удобство использования, но при этом обладает значительно более низкой производительностью, как показано в [21]. С целью сравнения результатов аналогичные реализации криптографических преобразований были осуществлены также для центрального процессора.

В ходе реализации и последующих замеров времени выполнения рассмотренных криптографических преобразований использовались центральный процессор Intel Core i7-3537-U, компилятор MSVC 2015, видеокарта GeForce GT 750M, CUDA 9.1, OpenCl 1.2.

В предыдущей работе [22] показано, что преимущество в скорости шифрования на графическом процессоре в сравнении с центральным процессором становится существенным при обработке минимум

[1] Specification for the Advanced Encryption Standard (AES). http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.
[2] ГОСТ Р 34.12-2015. Информационные технологии. Криптографическая защита информации. Блочные шрифты https://tc26.ru/standarts/natsionalnye-standarty/gost-r-34-12-2015-informatsionnaya-tekhnologiya-kriptograficheskaya-zashchita-informatsii-blochnye-shifry.html.
[3] ГОСТ Р 34.13-2015. Информационные технологии. Криптографическая защита информации. Режимы работы блочных шрифтов. https://tc26.ru/standarts/natsionalnye-standarty/gost-r-34-13-2015-informatsionnaya-tekhnologiya-kriptograficheskaya-zashchita-informatsii-rezhimy-raboty-blochnykh-shifrov.html.

1 ГБ данных. В связи с этим в описанных ниже синтетических тестах количество запускаемых потоков было подобрано так, чтобы за один запуск ядра осуществлялась обработка или генерация 1 ГБ данных.

Количество раундов криптографических преобразований в рамках тестов соответствовало среднему числу подобных преобразований, используемых для шифрования одного блока данных в описанных выше алгоритмах.

Умножение в поле Галуа. Исходя из рассмотренных выше шифров, можно сделать вывод, что наибольшую сложность при реализации LSX-структуры будет составлять достаточно быстрое умножение в полях Галуа. Его эффективной реализации посвящены работы [23, 24]. В настоящей работе используется поле Галуа порядка 2 как наиболее часто встречающееся в современной симметричной криптографии, его реализация представлена ниже в виде кода на языке C:

```
unsigned char result = 0;
unsigned char tmp;
for (int counter = 0; counter < 8; counter++)
{
        if (b & 1)
                result ^= a;
        tmp = (a & 0x80);
        a <<= 1;
        if (tmp)
                a ^= polynomial;
        b >>= 1;
}
```

Альтернативным методом является таблица предвычислений, повышающая скорость вычислений на центральном процессоре. Однако, учитывая вычислительные мощности и сложную структуру памяти графического процессора, данный подход может не быть столь эффективным.

Для исследования скорости работы данных методов была реализована программа, производящая перемножение двух массивов по 16 элементов каждый с 50-кратным повтором за один запуск ядра. Изначально в качестве таблицы предвычислений использовался массив, содержащий все возможные значения, получаемые при перемножении двух 8-битных значений.

В ходе эксперимента вычисления первым способом было показано среднее время для CUDA 5716 мс и 5827 мс для OpenCL, а при использовании таблицы предвычислений был получены результаты в 14779 мс и 15365 мс.

Однако, исходя из анализа описанных алгоритмов, было выявлено, что таблицы предвычислений требуются только для нескольких констант, используемых в конкретном алгоритме – максимум 8 в случае с алгоритмом «Кузнечик». После сокращения объема используемого массива измерения показали 3392 мс и 3828 мс для CUDA и OpenCL соответственно. Кроме того, как показано в предыдущей работе [22], целесообразно организовывать многократный доступ к одной и той же таблице максимально близко, а также воспользоваться свойством мультипликативности. Это позволит ускорить алгоритм в среднем на 10–15%.

При измерении времени выполнения данных вычислений на центральном процессоре было получено: для первого способа – 2738438 мс, для альтернативного способа – 95294 мс при использовании таблицы, содержащей все возможные произведения 8-битных значений, и 59210 мс при использовании таблиц для определенных констант.

Таблицы подстановки. Важной составляющей LSX-структуры является нелинейное преобразование, для которого в большинстве шифров используется таблица подстановки, которая значительно уменьшает скорость вычислений на графическом процессоре, так как предполагает в основном операции чтения из памяти. Анализ замены таблицы на соответствующие ей вычисления, например, представленные в [25], или алгоритмы, полученные в ходе криптоанализа [26], выходит за рамки данной работы, так как сложность и скорость подобных вычислений зависят от конкретного шифра.

Для оценки времени выполнения преобразования таблица подстановок из 256 однобайтовых значений была расположена в константной и разделяемой памяти и проведены 10 раундов S-преобразований (использовалось нелинейное преобразования шифра «Кузнечик»). При этом использовалось три типа входных данных: случайные значения (сгенерированы функцией qrand), группы по 100 одинаковых случайных значений и массив, полностью состоящий из одинаковых элементов. В результате замеров времени вычислений с применением константной и разделяемой памяти были получены значения 1222 мс и 238 мс в случае использования CUDA, а в случае использования OpenCL – 1253 мс и 253 мс, при этом дублирующиеся данные не ухудшили время выполнения.

После этого таблица подстановок была заменена на таблицу из 16 полубайтовых элементов, и эксперимент был проведен в тех же условиях, в результате чего были получены значения 484 мс в случае применения константной памяти и 253 мс в случае применения разделяемой памяти для CUDA, для OpenCL полученные результаты равны 493 мс и 257 мс соответственно.

Так как в случае использования центрального процессора проблема выбора памяти отсутствует, то были проведены только два типа экспериментов – с использованием 8-битных и 4-битных таблиц подстановок, в результате которых было получено время, равное 22217 мс и 24631 мс соответственно.

Побитовые операции. Так как операция сложения по модулю $2n$ в современных симметричных алгоритмах чаще всего используется при $n = 1$, для данного эксперимента была использована соответствующая данному случаю операция XOR. Кроме того, данная операция применяется при смешении с раундовым ключом в рассмотренных выше алгоритмах. Для эксперимента была реализована программа, которая загружает два массива по 16 Б каждый и производит 100 раундов операции XOR между их элементами с записью в выходной массив. При этом элементы массива представляются в виде 8, 16, 32 и 64-битных переменных.

В результате эксперимента время вычисления при использовании CUDA составило 1694 мс, 1856 мс, 1847 мс и 1737 мс, для OpenCL – 1681 мс, 1856 мс, 1851 мс и 1738 мс, для центрального процессора – 207941 мс, 107281 мс, 65418 мс и 58785 мс соответственно. Данные результаты также актуальны для операций побитового И и побитового ИЛИ.

Сложение длинных чисел. Как было сказано ранее, режим гаммирования требует применения операции сложения с длинными числами, так как длина синхропосылки должна совпадать с длиной блока, которая может превышать максимальный размер данных в CUDA, равный 64 битам. Само значение счетчика вычисляется путем сложения синхропосылки и глобального номера потока. В рамках настоящей работы длина блока составляет 128 бит, при этом размерность второго слагаемого зависит от количества запускаемых потоков. Исходя из того, что каждое ядро требует минимум 16 Б глобальной памяти, даже в случае наличия 8 ГБ памяти количество возможных потоков не превысит одного миллиарда, а следовательно, для хранения номера потока достаточно беззнаковой переменной длиной 4 Б.

Исходя из вышесказанного, слагаемые можно разделить на 8, 16 и 32-битные составляющие или 64-битные составляющие с увеличением размера второго слагаемого. Так как CUDA поддерживает только little-endian порядок байт, то необходимо осуществлять предварительный переворот данных для первого слагаемого, кроме случая разбиения на 8-битные составляющие. Использованные для замеров времени алгоритмы приведены ниже в виде кода на языке C.

```
T* thread_ptr = (T*)&thread;
const int k = 16 / sizeof(T);
const int n = sizeof(thread) / sizeof(T);
long long result = 0;
for(int i = 0; i < n; i++)
{
        result += thread_ ptr [i];
        result += counter[k - i - 1];
        counter[k - i - 1] = result;
        result >>= sizeof(T) * 8;
}
for(int i = k - n - 1; i >= 0; i--)
{
        result += counter[i];
        counter[i] = result;
        result >>= sizeof(T) * 8;
}
```

В случае использования 64 составляющих алгоритм имеет следующий вид:

```
long long result = counter[0];
counter[0] += thread;
if(counter[0] < result || counter[0] < thread)
        counter[1]++;
```

Хотя для шифрования одного блока данных достаточно осуществить сложение синхропосылки и глобального номера потока один раз, из-за незначительной разницы в полученных результатах было принято решение увеличить количество раундов данного действия до 10 за один запуск ядра. Среднее время вычислений с разбиением слагаемых на 8, 16, 32 и 64-битные составляющие в случае использования CUDA равно 154 мс, 167 мс, 176 мс и 230 мс, в случае использования OpenCL равно 157 мс, 171 мс, 176 мс и 229 мс, в случае использования центрального процессора равно 27411 мс, 58386 мс, 51615 мс и 45050 мс соответственно.

Обмен данными с глобальной памятью. Загрузка данных из глобальной памяти в локальные переменные не является частью криптографических преобразований, но крайне важна при вычислениях на графических процессорах. Для сравнения использовались два способа: первый заключается в передаче данных из входного буфера в локальный массив размером 16 Б и выгрузке их в выходной буфер с исполь-

зованием функции memcpy для CUDA и оператора присвоения невыровненной структуры для OpenCL, второй способ заключается в использовании структуры с полем в виде массива на 16 Б, выровненной с помощью макроса __align__ или конструкции __attribute__((aligned())), и оператора присвоения по умолчанию. В результате замеров было получено время для CUDA 340 мс и 101 мс, для OpenCL 300 мс и 96 мс.

Для наглядности результаты проведенных экспериментов представлены в таблице.

Тип преобразования		Время выполнения, мс		
		CUDA	OpenCL	Центральный процессор
Умножение в поле Галуа	путем вычисления	5716	5827	2738438
	с использованием полной таблицы предвычислений	14779	15365	95294
	с использованием таблиц предвычислений для известных констант	3392	3828	59210
Расположение 8-битной таблицы подстановки	в константной памяти	1222	1253	22217
	в разделяемой памяти	238	253	
Расположение 4-битной таблицы подстановки	в константной памяти	484	253	24631
	в разделяемой памяти	493	257	
Разбиение при побитовых операциях	8-битные элементы	1694	1681	207941
	16-битные элементы	1856	1856	107281
	32-битные элементы	1847	1851	65418
	64-битные элементы	1737	1738	58785
Разбиение при сложении длинных чисел	8-битные элементы	154	157	27411
	16-битные элементы	167	171	58386
	32-битные элементы	176	176	51615
	64-битные элементы	230	229	45050
Обмен данными с глобальной памятью	выровненная структура	101	96	–
	не выровненная структура	340	300	

Таблица. Время выполнения преобразований

Реализация алгоритма CLEFIA. С целью оценки возможности практического применения результатов исследования на графическом процессоре был реализован случайно выбранный алгоритм, не рассмотренный в рамках данной работы, – шифр CLEFIA.

CLEFIA – LSX-алгоритм с длиной блока 128 бит, основанный на сети Фейстеля. Он имеет схожую с AES-структуру линейного преобразования, но более сложное S-преобразование с использованием двух таблиц замены. Существуют три вариации данного алгоритма с длиной блока 128, 192 и 256 бит, из которых для последующей реализации был выбран вариант с длиной блока 128 бит.

При реализации для графического процессора применялись варианты технического исполнения криптографических преобразований, показавшие наилучшие результаты в описанных выше синтетических тестах. Так, для умножения в поле Галуа использовались таблицы предвычислений, подгружаемые в разделяемую память вместе с таблицами замены, побитовые операции производились с разбиением операндов на элементы размером 1 Б, а загрузка данных из глобальной памяти осуществлялась с использованием выровненной по 16 Б структуры. Для измерения скорости осуществлялось шифрование блока данных размером 1 ГБ. Время, полученное в результате эксперимента, составляет 1542 мс для CUDA и 1563 мс для OpenCL с учетом времени загрузки и выгрузки данных на устройство. Время реализации алгоритма на центральном процессоре, для которой также учитывались результаты тестов, составило 25239 мс. В то же время наихудшее время реализации с применением вариантов исполнения криптографических преобразований составило 6172 мс для CUDA и 6241 мс для OpenCL.

Таким образом, применение результатов проведенного исследования позволило избежать более чем четырехкратной потери скорости.

Заключение

В данной работе проведен анализ реализации преобразований, применяемых при использовании симметричных алгоритмов шифрования, на графическом процессоре с применением технологий CUDA и OpenCL. В рамках анализа был определен перечень наиболее значимых вычислений: умножение в полях Галуа, замена элементов блока с использованием таблицы подстановки, побитовые операции, сложение длинных чисел и загрузка данных из глобальной памяти.

В ходе проведенных экспериментов было выявлено, что при умножении в полях Галуа, в случае если один из множителей является заведомо известной константой, эффективнее всего использовать таблицы предвычислений; если же ни один из множителей не известен на этапе компиляции, наибольшую скорость можно получить путем прямого вычисления произведения. Также было выявлено, что наилучшим с точки зрения скорости является хранение таблиц подстановки в разделяемой памяти устройства, это же относится и к таблицам предвычислений, но применимо только при наличии необходимого объема разделяемой памяти. При проведении побитовых операций наиболее скоростным методом является представление блока данных в виде массива беззнаковых 8-битных элементов, что также характерно и для сложения длинных чисел. Также были представлены практические рекомендации по осуществлению обмена данными с глобальной памятью.

С использованием полученных результатов была осуществлена реализация блочного алгоритма CLEFIA, позволяющая производить шифрование 1 ГБ данных за время, равное 1542 мс, с учетом обмена данными с устройством. Путем сравнения наиболее и наименее эффективных реализаций было показано, что применение результатов исследования позволяет избежать более чем четырехкратной потери скорости шифрования.

Литература

1. Tomoiaga D., Stratulat M. AES algorithm adapted on GPU using CUDA for small data and large data volume encryption // International Journal of Applied Mathematics and Informatics. 2011. V. 5. P. 71–81.
2. Earanky K., Elmiligi H., Rahman M. GPU-acceleration of blowfish cryptographic algorithm // Proc. IEEE Pacific RIM Conference on Communications, Computers and Signal Processing. Victoria, Canada, 2015. P. 507–512. doi: 10.1109/PACRIM.2015.7334889
3. Fan W., Chen X., Li X. Parallelization of RSA algorithm based on compute unified device architecture // Proc. 9th IEEE Int. Conf. on Grid and Cooperative Computing. Nanjing, China, 2010. P. 174–178 doi: 10.1109/GCC.2010.44
4. Ищукова Е.А., Богданов К.И. Реализация алгоритма шифрования магма с использованием технологии nvidia CUDA // Международный журнал прикладных и фундаментальных исследований. 2015. № 12. С. 789–793.
5. Сизоненко А.Б., Ткаченко Д.А. Способ высокопроизводительной реализации алгоритма криптографического преобразования ГОСТ Р 34.12-2015 «Кузнечик» на массивно-параллельных сопроцессорах // Сборник материалов XXXVI Международной научно-практической конференции «Перспективы развития информационных технологий». Новосибирск, 2017. С. 174–178.
6. Keisuke I., Naoki N., Takakazu K. Acceleration of AES encryption on CUDA GPU // International Journal of Networking and Computing. 2012. V. 2. N 1. P. 131–145. doi: 10.15803/ijnc.2.1_131
7. Гибадуллин Р.Ф., Яковлев А.С., Новиков А.А., Перухин М.Ю. Ускорение AES шифрования на аппаратно-программной платформе NVIDIA CUDA // Вестник Казанского технологического университета. 2017. Т. 20. № 12. С. 97–103.
8. Жуков А.Е. Легковесная криптография. Часть 1 // Вопросы кибербезопасности. 2015. № 1(9). С. 26–43.
9. Жуков А.Е. Легковесная криптография. Часть 2 // Вопросы кибербезопасности. 2015. № 2(10). С. 2–10.
10. Knezevic M., Nikov V., Rombouts P. Low-latency encryption – is “Lightweight = Light + Wait”? // Lecture Notes in Computer Science. 2012. V. 7428. P. 426–446. doi: 10.1007/978-3-642-33027-8_25
11. Bogdanov A., Knezevic M., Leander G., Toz D., Varici K., Verbauwhede I. Spongent: a lightweight hash function // Lecture Notes in Computer Science. 2011. V. 6917. P. 312–325. doi: 10.1007/978-3-642-23951-9_21

References

1. Tomoiaga D., Stratulat M. AES algorithm adapted on GPU using CUDA for small data and large data volume encryption. *International Journal of Applied Mathematics and Informatics*, 2011, vol. 5, pp. 71–81.
2. Earanky K., Elmiligi H., Rahman M. GPU-acceleration of blowfish cryptographic algorithm. *Proc. IEEE Pacific RIM Conference on Communications, Computers and Signal Processing*. Victoria, Canada, 2015, pp. 507–512. doi: 10.1109/PACRIM.2015.7334889
3. Fan W., Chen X., Li X. Parallelization of RSA algorithm based on compute unified device architecture. *Proc. 9th IEEE Int. Conf. on Grid and Cooperative Computing*. Nanjing, China, 2010, pp. 174–178 doi: 10.1109/GCC.2010.44
4. Ishchukova E.A., Bogdanov K.I. Implementation of the encryption algorithm MAGMA using NVIDIA CUDA technology. *Mezhdunarodnyi Zhurnal Prikladnykh i Fundamental'nykh Issledovanii*, 2015, no. 12, pp. 789–793. (in Russian)
5. Sizonenko A.B., Tkachenko D.A. Method of high-performance implementation of the cryptographic transformation algorithm GOST R 34.12-2015 "Kuznyechik" on massively parallel coprocessors. *Proc. 26th Int. Conf. on Prospects for the Development of Information Technology*. Novosibirsk, 2017, pp. 174–178. (in Russian)
6. Keisuke I., Naoki N., Takakazu K. Acceleration of AES encryption on CUDA GPU. *International Journal of Networking and Computing*, 2012, vol. 2, no. 1, pp. 131–145. doi: 10.15803/ijnc.2.1_131
7. Gibadullin R.F., Yakovlev A.S., Novikov A.A., Perukhin M.Yu. Acceleration of AES encryption on the NVIDIA CUDA hardware-software platform. *Vestnik Kazanskogo Tekhnologicheskogo Universiteta*, 2017, no. 12, pp. 97–103. (in Russian)
8. Zhukov A.E. Lightweight cryptography. Part 1. *Voprosy Kiberbezopasnosti*, 2015, no. 1, pp. 26–43. (in Russian)
9. Zhukov A.E. Lightweight cryptography. Part 2. *Voprosy Kiberbezopasnosti*, 2015, no. 2, pp. 2–10. (in Russian)
10. Knezevic M., Nikov V., Rombouts P. Low-latency encryption – is “Lightweight = Light + Wait”? *Lecture Notes in Computer Science*, 2012, vol. 7428, pp. 426–446. doi: 10.1007/978-3-642-33027-8_25
11. Bogdanov A., Knezevic M., Leander G., Toz D., Varici K., Verbauwhede I. Spongent: a lightweight hash function. *Lecture Notes in Computer Science*, 2011, vol. 6917, pp. 312–325. doi: 10.1007/978-3-642-23951-9_21
12. Usman M., Ahmed I., Aslam M.I., Khan S., Shah U.A. SIT: a

12. Usman M., Ahmed I., Aslam M.I., Khan S., Shah U.A. SIT: a lightweight encryption algorithm for secure internet of things // International Journal of Advanced Computer Science and Applications. 2017. V. 8. N 1. P. 402–411. doi: 10.14569/IJACSA.2017.080151
13. Borisenko N., Nguyen L. On implementation method of large size linear transformation // Proc. 9th Workshop on Current Trends in Cryptology. Kazan, Russia, 2015. P. 183–195.
14. Казимиров О.В., Казимирова В.Н., Олейников Р.В. Метод генерации сильно нелинейных S-блоков на основе градиентного спуска // Математические основы криптографии. 2014. Т. 5. № 2. С. 71–78.
15. Biryukov A., Roy A., Velichkov V. Differential analysis of block ciphers SIMON and SPECK // Lecture Notes in Computer Science. 2015. V. 8540. P. 546–570. doi: 10.1007/978-3-662-46706-0_28
16. Biryukov A., Perrin L. State of the Art in Lightweight Symmetric Cryptography [Электронный ресурс]. Режим доступа: http://orbilu.uni.lu/handle/10993/31319, свободный. Яз. англ. (дата обращения 24.05.2018).
17. Beaulieu R., Shors R., Smith D., Treatman-Clark J., Weeks S., Wingers B. The SIMON and SPECK families of lightweight block ciphers // Proc. 52nd ACM/EDAC/IEEE design Automation Conference. San Francisco, USA, 2015. P. 1–10. doi: 10.1145/2744769.2747946
18. Beaulieu R., Shors D., Smith J., Treatman-Clark S., Weeks B., Wingers L. SIMON and SPECK: block ciphers for the internet of things // Proc. NIST Lightweight Cryptography Workshop. Gaithersburg, USA, 2015. P. 1–15.
19. Hong D., Lee J.K., Kim D.C., Kwon D., Ryu K.H., Lee D.G. LEA: a 128-bit block cipher for fast encryption on common processors // Lecture Notes in Computer Science. 2014. V. 8267. P. 3–27. doi: 10.1007/978-3-319-05149-9_1
20. Zhang WT., Bao ZZ., Lin DD., Rijmen V., Yang BH., Verbauwhede I. RECTANGLE: a bit-slice lightweight block cipher suitable for multiple platforms // Science China Information Sciences. 2015. V. 58. P. 1–15. doi: 10.1007/s11432-015-5459-7
21. Hoshino T., Maruyama N., Matsuoka S., Takaki R. CUDA vs OpenACC: performance case studies with kernel benchmarks and a memory-bound CFD application // Proc. 13th IEEE/ACM Int. Symposium on Cluster, Cloud and Grid Computing. Delft, Netherlands, 2013. P. 136–143. doi: 10.1109/CCGrid.2013.12
22. Удальцов В.А., Павлов В.Э. Увеличение скорости работы алгоритма шифрования «Кузнечик» с использованием технологии CUDA // Теория. Практика. Инновации. 2017. № 4(16). С. 5–11.
23. Jullien G.A., Bajard J., Imbert L. Parallel Montgomery multiplication in GF (2^k) using trinomial residue arithmetic // Proc. 17th IEEE Symposium on Computer Arithmetic. Massachusetts, USA, 2005. P. 164–171. doi:10.1109/ARITH.2005.34
24. Рахман П.А. Эффективные вычислительные схемы для арифметики поля Галуа GF(2^8) в технологии помехоустойчивого кодирования // Международный журнал прикладных и фундаментальных исследований. 2016. № 7. С. 360–365.
25. Фомичев В.М., Лолич Д.М., Юзбашев А.В. Алгоритмическая реализация s-боксов на основе модифицированных аддитивных генераторов // Прикладная дискретная математика. Приложение. 2017. № 10. С. 102–104. doi: 10.17223/2226308X/10/41
26. Biryukov A., Perrin L., Udovenko A. Reverse-engineering the S-box of Streebog, Kuznyechik and STRIBOBr1 // Lecture Notes in Computer Science. 2016. V. 9665. P. 372–402. doi: 10.1007/978-3-662-49890-3_15

lightweight encryption algorithm for secure internet of things. *International Journal of Advanced Computer Science and Applications*, 2017, vol. 8, no. 1, pp. 402–411. doi: 10.14569/IJACSA.2017.080151
13. Borisenko N., Nguyen L. On implementation method of large size linear transformation. *Proc. 9th Workshop on Current Trends in Cryptology*. Kazan, Russia, 2015, pp. 183–195.
14. Kazymyrov O.V., Kazymyrova V.N., Oliynykov R.V. A method for generation of high-nonlinear S-boxes based on gradient descent. *Mathematical Aspects of Cryptography*, 2014, vol. 5, pp. 71–78. doi: 10.4213/mvk118
15. Biryukov A., Roy A., Velichkov V. Differential analysis of block ciphers SIMON and SPECK. *Lecture Notes in Computer Science*, 2015, vol. 8540, pp. 546–570. doi: 10.1007/978-3-662-46706-0_28
16. Biryukov A., Perrin L. *State of the Art in Lightweight Symmetric Cryptography*. Available at: http://orbilu.uni.lu/handle/10993/31319 (accessed 24.05.2018).
17. Beaulieu R., Shors R., Smith D., Treatman-Clark J., Weeks S., Wingers B. The SIMON and SPECK families of lightweight block ciphers. *Proc. 52nd ACM/EDAC/IEEE design Automation Conference*. San Francisco, USA, 2015, pp. 1–10. doi: 10.1145/2744769.2747946
18. Beaulieu R., Shors D., Smith J., Treatman-Clark S., Weeks B., Wingers L. SIMON and SPECK: block ciphers for the internet of things. *Proc. NIST Lightweight Cryptography Workshop*. Gaithersburg, USA, 2015, pp. 1–15.
19. Hong D., Lee J.K., Kim D.C., Kwon D., Ryu K.H., Lee D.G. LEA: a 128-bit block cipher for fast encryption on common processors. *Lecture Notes in Computer Science*, 2014, vol. 8267, pp. 3–27. doi: 10.1007/978-3-319-05149-9_1
20. Zhang WT., Bao ZZ., Lin DD., Rijmen V., Yang BH., Verbauwhede I. RECTANGLE: a bit-slice lightweight block cipher suitable for multiple platforms. *Science China Information Sciences*, 2015, vol. 58, pp. 1–15. doi: 10.1007/s11432-015-5459-7
21. Hoshino T., Maruyama N., Matsuoka S., Takaki R. CUDA vs OpenACC: performance case studies with kernel benchmarks and a memory-bound CFD application. *Proc. 13th IEEE/ACM Int. Symposium on Cluster, Cloud and Grid Computing*. Delft, Netherlands, 2013, pp. 136–143. doi: 10.1109/CCGrid.2013.12
22. Udaltsov V.A., Pavlov V.E. Operation speed increase of the cryptoalgorithm “Kuznyechik” with the use of CUDA technology. *Teoriya, Praktika, Innovatsii*, 2017, no. 4, pp. 5–11. (in Russian)
23. Jullien G.A., Bajard J., Imbert L. Parallel Montgomery multiplication in GF (2^k) using trinomial residue arithmetic. *Proc. 17th IEEE Symposium on Computer Arithmetic*. Massachusetts, USA, 2005, pp. 164–171. doi:10.1109/ARITH.2005.34
24. Rahman P.A. Effective computational schemes for the arithmetic of Galois field GF(2^8) in the error-correcting coding technology. *Mezhdunarodnyi Zhurnal Prikladnykh i Fundamental'nykh Issledovanii*, 2016, no. 7, pp. 360–365. (in Russian)
25. Fomichev V.M., Lolich D.M., Yuzbashev A.V. S-boxes algorithmic realization based on modified additive generators. *Prikladnaya diskretnaya matematika. Prilozhenie*, 2017, no. 10, pp. 102–104. (in Russian) doi: 10.17223/2226308X/10/41
26. Biryukov A., Perrin L., Udovenko A. Reverse-engineering the S-box of Streebog, Kuznyechik and STRIBOBr1. *Lecture Notes in Computer Science*, 2016, vol. 9665, pp. 372–402. doi: 10.1007/978-3-662-49890-3_15

Авторы

Удальцов Валерий Анатольевич – аспирант, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, ORCID ID: 0000-0002-7726-1298, udalv1@rambler.ru

Кармановский Николай Сергеевич – кандидат технических наук, доцент, доцент, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, Scopus ID: 57192385103, ORCID ID: 0000-0002-0533-9893, Karmanov50@mail.ru

Authors

Valeriy A. Udaltsov – postgraduate, ITMO University, Saint Petersburg, 197101, Russian Federation, ORCID ID: 0000-0002-7726-1298, udalv1@rambler.ru

Nikolai S. Karmanovskiy – PhD, Associate Professor, Associate Professor, ITMO University, Saint Petersburg, 197101, Russian Federation, Scopus ID: 57192385103, ORCID ID: 0000-0002-0533-9893, Karmanov50@mail.ru