V.A. Koryagin, N.P. Vasilyev

Parallel Block Encrypting Method Based On Standard Algorithms

Keywords: block cipher, parallel computations, dynamically specified block size.

Current publication proposes block cipher design that offers availability to specify block size in the initialization phase without maximum block size limit. Publication contains theoretical bases and common description of algorithms that are based on proposed design. Current publication describes how to implement part of block cipher job: diffusion; to implement the rest of the functionality – confusionand key depended operations – itis supposed to use standard algorithms. Algorithms, based on proposed design, are targeted on maximum parallelism and can be implemented for hybrid systems with graphical co-processor.

В.А. Корягин, Н.П. Васильев

МЕТОД ПАРАЛЛЕЛЬНОГО БЛОЧНОГО ШИФРОВАНИЯ НА ОСНОВЕ СТАНДАРТНЫХ АЛГОРИТМОВ

Введение

Блочный шифр – это один из способов обеспечения конфиденциальности информации, отличительной особенностью которого является обработка данных порциями фиксированной длины (блоками). Для зашифрования и расшифрования используется один и тот же секретный ключ.

Базовые принципы блочного шифрования

Для сокрытия избыточности информации применяются две базовые техники: рассеивание и перемешивание (diffusionandconfusion) [1, часть 11.1].

Перемешивание – это операция, направленная на усложнение выяснения взаимосвязей между открытым текстом, ключом и шифртекстом. Это срывает попытки исследования шифртекста при помощи поиска избыточности и характерных статистических особенностей. Самый простой способ реализовать такую операцию – выполнить замену в соответствии с таблицей замен (substitution). [1, часть 11.1]

Рассеивание предполагает распространение влияния одного знака открытого текста, а также одного знака ключа на значительное количество знаков шифртекста. Наличие у шифра этого свойства позволяет скрыть статистическую зависимость между знаками открытого текста, иначе говоря, перераспределить избыточность исходного языка посредством распространения ее на весь текст, и не позволяет восстанавливать неизвестный ключ по частям. [5, стр. 60]

Блочные шифры строятся по принципу многократного чередования этих двух операций для достижения так называемого «Лавинного эффекта» (Avalancheeffect)[1, часть 12.3, Number of Rounds], при котором каждый бит зашифрованных данных зависит от всех бит исходных данных, а также от всех бит ключа.

Основные понятия и определения

В настоящей статье описывается алгоритм, обрабатывающий данные, находящиеся в памяти ЭВМ. Минимальной единицей информации является бит. ЭВМ хранит

данные в памяти в виде одномерного массива бит, сгруппированного в ячейки. Адресация осуществляется к **ячейкам** памяти. Рассматривается общий случай, при котором ячейка содержит $byteSize$ бит. В описываемом методе все операции выполняются над ячейками памяти.

В основе описываемого преобразования лежит **базовая функция** f_b. Базовая функция выполняет рассеивание и перемешивание входной информации фиксированной длины $\boldsymbol{cubeSide}$ ячеек. Результатом выполнения базовой функции над блоком данных является блок обработанных данных размером $cubeSide$ ячеек.

Описываемый метод позволяет проектировать блочные шифры с задаваемым размером блока. Размер полученного блока будем обозначать как **size**. Однако нельзя задавать произвольный размер. В качестве размера блока можно использовать только число, соответствующее формуле

$$size = cubeSide * k_1 * k_2 * \ldots * k_n;\ \forall k_i: k_i \in (1;\ cubeSide] \quad (1)$$

Размер блока, соответствующий этому критерию, будем именовать **корректным размером**.

В связи с тем, что размер блока должен соответствовать определённым критериям, перед обработкой данных выполняется процедура дополнения блока до корректного размера. **Контейнером** будем называть структуру данных корректного размера. Контейнер может одновременно содержать как полезную информацию, так и псевдослучайные данные и служебную информацию криптосистемы. Размер контейнера (в ячейках) будем обозначать словом **size**.

Под **расстоянием** между ячейками будет пониматься разница между индексами этих ячеек, (например, соседние ячейки находятся на расстоянии 1 друг от друга).

Во время обработки данных последовательно выполняется определённое количество раундов. **Раундом** будем называть множество операций выполнения **базовой функции**, для которых выбираются ячейки на одинаковом расстоянии.

В каждом раунде базовая функция выполняется над $\frac{size}{cubeSide}$ множествами ячеек. Каждое множество содержит cubeSide ячеек, находящихся на одинаковом расстоянии друг от друга. Для каждого раунда это расстояние отличается. Для удобства введено понятие **очередь расстояний**. Очередь расстояний – это структура данных, содержащая целые положительные числа. Количество элементов в очереди равно количеству раундов, требуемому для обработки блока размером size. Каждый элемент очереди – это расстояние, на котором в соответствующем раунде нужно выбрать cubeSide ячеек для выполнения над ними базовой функции (порядковый номер раунда соответствует порядковому номеру значения из очереди).

Теорема о распространении рассеивания

Пусть даны две функции f_1 и f_2, обрабатывающие блоки данных размером bs_1 и bs_2 ячеек соответственно. Оба преобразования выполняют рассеивание информации. Дополнительным свойством является то, что для обеих функций любое изменение блока входных данных приводит к изменению каждой ячейки выходных данных.

Даны входные данные размером $size$ ячеек. Размер входных данных соответствует следующим условиям:

$$size \bmod bs_1 = 0,$$
$$size \bmod bs_2 = 0,$$
$$size \in (bs_2;\ bs_1 * bs_2].$$

Введём функцию F, последовательно применяющую функции f_1 и f_2 над данными размером $size$ следующим образом:

Шаг 1: Входные данные рассматриваются в качестве $\frac{size}{bs_1}$ множеств ячеек. Каждое множество представляет собой bs_1 ячеек, находящихся на расстоянии $\frac{size}{bs_1}$ ячеек друг от друга (соответствующие ячейки разных множеств располагаются последовательно друг за другом). Над каждым множеством ячеек выполняется функция f_1.

Шаг 2: Входные данные рассматриваются в качестве $\frac{size}{bs_2}$ множеств ячеек. Каждое множество представляет собой bs_2 ячеек, расположенных последовательно друг за другом (сами множества также располагаются последовательно друг за другом). Над каждым множеством ячеек выполняется функция f_2.

Тогда функция F обеспечивает влияние одного бита входных данных на все биты выходных данных.

Ограничения на размер статьи не позволяют опубликовать здесь доказательство теоремы. Полный текст доказательства содержится в [6].

Следствия из теоремы

Построение функции с задаваемым размером блока на основе одной функции раундового блочного преобразования с фиксированным размером блока. Описанная выше функция F обеспечивает влияние всех битов входных данных на каждый бит выходных данных (т.е. выполняет рассеивание). Следовательно, её можно рекурсивно применять в качестве функции f_1 илиf_2, так как она сама соответствует требованиям, предъявляемым к этим функциям.

При рекурсивном запуске функции F функция f_1 (f_2) будет являться базовой функцией, и первый (второй) шаг преобразования F остаётся неизменным. Тогда в качестве функции f_2 (f_1), выполним функцию F над каждым из множеств, размером bs_2 (bs_1) ячеек.

Множество операций выполнения базовой функции в одном конкретном рекурсивном вызове будем называть раундом.

Корректный размер. Здесь и далее подразумевается, что в качестве базовой функции выбрана функция $\boldsymbol{f_1}$.

По условию теоремы, для размеров входных данных существуют определённые ограничения:

$$size mod bs_1 = 0,$$
$$size mod bs_2 = 0,$$
$$size \in (bs_2;\ bs_1 * bs_2].$$

Последнее условие можно переформулировать, как два ограничения для bs_2:$bs_2 < size$и $bs_2 \geq \frac{size}{bs_1}$.

При рекурсивном запуске функции F следующее значение $size$ становится равным текущему значению bs_2, т.е. уменьшается в $k = \frac{size}{bs_2}$ раз, где $k \in [2; bs_1]$, поскольку $size \in (bs_2, bs_1 * bs_2]$.

Для того чтобы рекурсивно запускать функцию F описанным выше способом, необходимо иметь возможность каждый раз представить входные данные как k множеств размером bs_2 ячеек.

Как указано выше, $k \in [2; bs_1]$. Кроме того, в итоге при последнем рекурсивном вызове необходимо получить $bs_2 = bs_1$.

Для того чтобы все перечисленные условия выполнились, исходный размер входных данных должен являться произведением целых положительных коэффициентов, каждый из которых меньше либо равен $cubeSide$, а один из них равен $cubeSide$:

$$size = cubeSide * k_1 * k_2 * \dots * k_n;\ \forall k_i: k_i \in (1;\ cubeSide] \quad (1)$$

Размер входных данных, который соответствует данному критерию, будет называться **корректным размером**, а любые другие корректными быть не могут.

Очередь расстояний. Рассматривая рекурсивное выполнение функции $\boldsymbol{F}$ как множество операций выполнения базовой функции, есть возможность отказаться от рассмотрения двух различных функций блочного шифрования$\boldsymbol{f_1}$ и $\boldsymbol{f_2}$, и рассматривать только расстояния, на которых будут выбираться ячейки для выполнения над ними базовой функции. Перед выполнением раундов в таком случае необходимо составить очередь расстояний между ячейками так, чтобы для каждого раунда заранее было известно расстояние.

Если в качестве базовой функции выбрана f_1, то первым расстоянием в очереди будет значение $\frac{size}{cubeSide}$, и каждое следующее расстояние, как уже было сказано выше, будет уменьшаться в $k \in [2; cubeSide]$ раз. Последнее расстояние будет равно 1 (будут выбраны множества из соседних $cubeSide$ элементов).

Если базовой функцией является f_2, то очередь начинается со значения 1 и увеличивается аналогичным образом.

Количество раундов. В случае, когда каждый коэффициент $\boldsymbol{k_i = cubeSide}$, размер контейнера является степенью числа $\boldsymbol{cubeSide}$, а общее количество раундов равно $\boldsymbol{\log_{cubeSide} size}$. Любой размер входных данных можно дополнить до значения $\boldsymbol{cubeSide^n}$, где n – натуральное число. Следовательно, для любого размера водных данных $\boldsymbol{src\_size}$ возможно подобрать такой размер контейнера, при котором количество раундов не будет превышать

$$\lceil \log_{cubeSide} src\_size \rceil. \quad (2)$$

Перемешивание информации. Функция $\boldsymbol{F}$ выполняет рассевание информации согласно представленной в статье теореме. Если в качестве базовой взять функцию, выполняющую как рассеивание, так и перемешивание, то функция F, построенная на основе такой базовой функции, также будет выполнять перемешивание, так как после каждого раунда в каждую ячейку будет записан результат выполнения базовой функции над несколькими ячейками, включая рассматриваемую.

Практическое применение

Построение функции блочного шифрования на основе функции F. Как уже было сказано ранее, блочный шифр – это обратимое преобразование, зависящее от ключа шифрования $\boldsymbol{k}$ и выполняющее рассеивание и перемешивания информации.

Рассеивание информации выполняет сама функция F, подробно описанная в предыдущих разделах. Для обеспечения перемешивания необходимо правильно выбрать базовую функцию. Также нужно в каждом раунде дополнительно выполнять преобразование, зависимое от ключа.

Авторы рекомендуют выбрать базовую функцию на основе примитивов, использующихся в существующих блочных шифрах. Например, в стандарте ГОСТ 28147-89 совокупность таких примитивов обозначена, как «один цикл шифрования» [4, прил. 3, чертеж 5, стр. 21]. В стандарте AES такие примитивы называются «слоями» [2, стр. 8].

В алгоритме Serpent, предложенном в качестве кандидата на роль криптографического стандарта AES и занявшем в результате конкурса 2-е место, такие примитивы перечислены, как 3 операции, которые составляют раунд шифрования [3, стр. 5].

Представление данных. Авторы рекомендуют рассматривать промежуточные данные как статическую трёхмерную матрицу. Такое представление позволяет наиболее просто и эффективно реализовать алгоритм шифрования, так как при таком подходе есть возможность отказаться от рекурсивной обработки данных и выполнять операции итеративно.

В i-м раунде шифрования блок входных данных рассматривается как вектор двухмерных матриц. В каждой матрице $cubeSide$ строк и $subSize$ (i-е значение в очереди) столбцов. Для выполнения раунда, необходимо выполнить базовую функцию над каждым столбцом каждой матрицы.

Операции выполнения базовой функции над столбцами в отдельном раунде можно производить независимо друг от друга. Раунды необходимо выполнять последовательно.

Для расшифрования раунды необходимо выполнять в обратном порядке, и использовать функцию шифрования, обратную к базовой функции.

Вычислительная сложность

На основе описанного алгоритма можно оценить его вычислительную сложность. В каждом раунде над каждым столбцом матрицы выполняется базовая функция. При переходе от раунда к раунду во время зашифрования количество столбцов в матрицах уменьшается в $k \in (1; cubeSide]$ раз, а количество самих матриц обратно пропорционально возрастает. Таким образом, общее количество вызовов базовой функции остаётся неизменным в каждом раунде и равно $\frac{size}{cubeSide}$. Количество раундов описано в формуле (2) и логарифмически зависит от размера входных данных.

Исходя из этого, общая вычислительная сложность шифрования не будет превышать

$$O(\lceil \log_{cubeSide} src\_size \rceil * \frac{size}{cubeSide}). \quad (3)$$

При этом вычислительная сложность расшифрования эквивалентна вычислительной сложности зашифрования, так как при расшифровании выполняются те же действия, но в обратном порядке.

Отметим также, что производительность напрямую зависит от выбранных криптографических примитивов.

Параллелизм. Алгоритм предполагает выполнение базовой функции над отдельными столбцами отдельных матриц. Все эти операции можно производить независимо, из чего следует, что степень параллелизма при выполнении одного раунда составляет $\frac{size}{cubeSide}$ операций. При этом раунды необходимо выполнять последовательно.

Заключение

В статье предложен новый способ построения блочных шифров. Метод предоставляет гибкие возможности по формированию алгоритмов блочного шифрования. Задаваемый размер блока открывает новые возможности проектирования криптографических протоколов с использованием блочных шифров. Возможность параллельной обработки данных позволяет выполнить эффективную реализацию с использованием современных вычислительных технологий, таких как гибридные вычисления с графическим сопроцессором.

СПИСОК ЛИТЕРАТУРЫ:

1. Schneier B. Applied cryptography // John Wiley & Sons, 1996. 784 p.
2. Daemen J., Rijmen V. AES Proposal: Rijndael [Электронный ресурс] / National Institute of Standards and Technology's web site – *Электрон. текстовые дан*. – Режим доступа: http://csrc.nist.gov/archive/aes/rijndael/Rijndael-ammended.pdf
3. Anderson R., Biham E., Knudsen L. Serpent: A Proposal for the Advanced Encryption Standard [Электронныйресурс] / Serpent home page – *Электрон. текстовые дан*. – Режим доступа: http://www.cl.cam.ac.uk/~rja14/Papers/serpent.pdf
4. ГОСТ 28147-89 Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования – Электрон. текстовые дан. – Режим доступа: http://protect.gost.ru/v.aspx?control=7&id=139177
5. Иванов М.А., Чугунков И.В. Криптографические методы защиты информации в компьютерных системах и сетях : Учебное пособие / Под ред. М.А. Иванова. М.: НИЯУ МИФИ, 2012. 400 с.
6. Корягин В.А. Пояснительная записка к дипломному проекту (выпускной квалификационной работе) на тему: Параллельное блочное преобразование с задаваемым размером блока. М.: НИЯУ МИФИ. 2014.

REFERENCES:

1. Schneier B. Applied cryptography // John Wiley & Sons, 1996. 784 p.
2. Daemen J., Rijmen V. AES Proposal: Rijndael [Электронный ресурс] / National Instituteof Standards and Technology's web site – URL:http://csrc.nist.gov/archive/aes/rijndael/Rijndael-ammended.pdf
3. Anderson R., Biham E., Knudsen L. Serpent: A Proposal for the Advanced EncryptionStandard [Электронный ресурс] / Serpent home page – URL: http://www.cl.cam.ac.uk/~rja14/Papers/serpent.pdf
4. GOST 28147-89 Systemy obrabotki informatii. Zaschita cryptographicheskaya. Algorithm cryptographycheskogo preobrasovaniya – URL: http://protect.gost.ru/v.aspx?control=7&id=139177
5. Ivanov M.A., Chugunkov I.V. Cryptographycheskie methody zaschity informatii v computernyhsystemah i setyah: Uchebnoe posobie / Pod red. M.A. Ivanova M.: NRNU MEPhI, 2012. 400 p.
6. Koryagin V.A. Poyasnitelnaya zapiska k diplomnomu projectu (vypusknoj qualificationnoj rabote)na temu: Parallelnoe blochnoe preobrazovanie s zadavaemym razmerom blocka / NRNU MEPhI. 2014.