Секция 5

ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ В ДИСКРЕТНОЙ МАТЕМАТИКЕ

УДК 519.7

РЕШЕНИЕ ЗАДАЧ КРИПТОАНАЛИЗА В ГРИД-СИСТЕМАХ (НА ПРИМЕРЕ BOINC)[1]

О. С. Заикин

Логический криптоанализ (постановка задачи криптоанализа как задачи поиска выполняющих наборов пропозициональных выражений) показал свою перспективность в отношении некоторых систем поточного шифрования. Основы круноблочной параллельной технологии решения SAT-задач описаны в [1]. Реализация данной технологии в виде параллельного SAT-решателя PD-SAT с использованием библиотеки MPI представлена в [2]. С применением решателя PD-SAT были успешно решены на вычислительных кластерах задачи логического криптоанализа суммирующего генератора, порогового генератора, а также генератора Гиффорда (с длинами ключей 66, 80, 64 бит). Логический криптоанализ генератора A5/1 потребовал весьма значительных вычислительных ресурсов, получение которых в рамках одного (пусть даже очень мощного) вычислительного кластера, как правило, проблематично. В связи с этим было принято решение о переносе технологии распараллеливания SAT-задач в распределенные вычислительные среды. Первым удачным опытом такого рода стало успешное решение задач криптоанализа генератора A5/1 в грид-системе BNB-Grid [3].

В настоящее время большое число исследований посвящено решению трудных комбинаторных задач в так называемых Desktop-Grid — грид-системах, в которых в роли вычислительных узлов выступают обычные персональные компьютеры (ПК). Принцип организации таких грид-систем заключается в предоставлении пользователями ресурсов своих ПК на добровольной основе. Наиболее популярной платформой для организации Desktop-Grid является Berley Open Infrastructure for Network Computing (BOINC, [4]). Пользователь устанавливает клиент BOINC, с помощью которого подключается через Интернет к проекту, после чего ПК начинает получать и обрабатывать задания. Режим обработки заданий гибко настраивается (по умолчанию обработка заданий начинается, когда ПК некоторое время не используется). Существуют проекты на основе BOINC, объединяющие сотни тысяч ПК (например, проект поиска радиосигналов внеземных цивилизаций SETI@home).

Реализована новая грид-версия решателя PD-SAT, предназначенная для использования в проектах на основе платформы BOINC. Для этой цели использована библиотека DC-API [5]. Применение PD-SAT позволило успешно решить в рамках BOINC-проекта задачи криптоанализа ряда генераторов (суммирующего, порогового, генератора Гиффорда). В ближайшее время планируется организовать BOINC-проект, состоящий из нескольких тысяч ПК для решения более масштабных задач криптоанализа.

[1]Работа поддержана грантом «Лаврентьевский конкурс молодежных проектов СО РАН» 2010–2011 гг. и грантом РФФИ, проект № 11-07-00377-а.

ЛИТЕРАТУРА

1. *Заикин О. С., Семенов А. А.* Технология крупноблочного параллелизма в SAT-задачах // Проблемы управления. 2008. № 1. С. 43–50.
2. *Заикин О. С.* Реализация процедур прогнозирования трудоемкости параллельного решения SAT-задач // Вестник УГАТУ. 2010. Т. 14. № 4(39). С. 210–220.
3. *Посыпкин М. А., Заикин О. С., Беспалов Д. В., Семенов А. А.* Решение задач криптоанализа поточных шифров в распределенных вычислительных средах // Труды ИСА РАН. 2009. Т. 46. С. 119–137.
4. `http://boinc.berkeley.edu` — Berkeley Open Infrastructure for Network Computing.
5. `http://www.desktopgrid.hu` — SZTAKI desktop grig.

УДК 519.7

BOOLEAN FUNCTIONS — СИСТЕМА ДЛЯ РАБОТЫ С БУЛЕВЫМИ ФУНКЦИЯМИ[1]

Н. А. Коломеец, А. В. Павлов

Предлагается новая система для работы с булевыми функциями *Boolean Functions*. Эта система ориентирована на пользователей-программистов и представляет собой библиотеку классов и функций на языке C++; она может быть полезна для проведения компьютерных экспериментов и тестов, связанных с булевыми функциями. В отличие от других систем, таких, как MAGMA, система является узконаправленной на работу с булевыми функциями специального вида. Кроме того, *Boolean Functions* бесплатна и свободно распространяема.

Для решения многих математических задач требуется подготовка экспериментальной базы и проведение компьютерных тестов. Каждый раз для новой подобной задачи приходится заново создавать все вспомогательные функции, такие, как проверка принадлежности функции какому-либо классу или перевод функции в различные представления (АНФ, таблица истинности, трейс-форма). Предлагаемая система предназначена для сокращения этих действий. На данный момент *Boolean Functions* является весьма неполной системой, хотя и содержит некоторые готовые алгоритмы; предполагается дальнейшее её развитие и выпуск новых версий.

Система позволяет работать с двоичными векторами и совершать стандартные операции над ними, такие, как определение веса Хэмминга, сложение двух векторов, вычисление их скалярного произведения, сравнение двух векторов и др.

Библиотека работает с представлениями булевой функции в виде АНФ, таблицы истинности и представлением с помощью следа (трейс-форма), а также умеет переводить одно представление в другое.

Boolean Functions позволяет проверять, являются ли две булевы функции аффинно эквивалентными (используется алгоритм [1]), а также генерировать функции, аффинно эквивалентные заданной.

Основное предназначение данной системы — работа с булевыми функиями специального вида, в частности с бент-функциями. Система позволяет проверять, является ли булева функция бент-функцией. *Boolean Functions* содержит алгоритмы генерации бент-функций от 4, 6 и 8 переменных степени не больше 3, алгоритм генерации

[1]Работа выполнена при финансовой поддержке РФФИ (проект № 11-01-00997) и ФЦП «Научные и научно-педагогические кадры инновационной России» на 2009–2013 гг. (гос. контракт № 02.740.11.0362).

всех бент-функций, находящихся на минимальном расстоянии от заданной (подробнее см. [2]), а также алгоритм построения кодов, состоящих из векторов значений бент-функций, сдвиги которых на заданную бент-функцию являются линейными кодами (подробнее см. [3]).

Работа с достаточно сложными объектами (представления булевых функций и т. д.) базируется на более простых элементах, таких, как векторы, матрицы и подпространства. Реализованы также различные итераторы, позволяющие строить переборные схемы для векторов, матриц и подпространств.

Первая версия системы *Boolean Functions* будет доступна на странице Института математики им. С. Л. Соболева СО РАН `http://math.nsc.ru/~bf` .

ЛИТЕРАТУРА

1. *Meng Q., Yang M., Zhang H., and Liu Y.* Analysis of Affinely Equivalent Boolean Functions // Cryptology ePrint Archive, Report 2005/025.
2. *Коломеец Н. А., Павлов А. В.* Свойства бент-функций, находящихся на минимальном расстоянии друг от друга // Прикладная дискретная математика. 2009. № 4. С. 5–20.
3. *Павлов А. В.* Бент-функции и линейные коды в CDMA // Прикладная дискретная математика. Приложение. 2010. № 3. С. 95–97.

УДК 519.7

ПРИМЕНЕНИЕ SAT-ПОДХОДА В РЕШЕНИИ КОМБИНАТОРНЫХ ЗАДАЧ[1]

А. А. Семенов, И. В. Отпущенников, С. Е. Кочемазов

Рассмотрим использование SAT-подхода в решении различных дискретных задач «переборной» природы. Для многих относительно простых по постановкам дискретных задач не известно алгоритмов, эффективных хотя бы на некоторых практически значимых подклассах рассматриваемых задач.

Простейший пример такого рода — анализ свойств недетерминированного автомата $\tilde{A}$, распознающего некоторый регулярный язык. Может оказаться так, что диагностируемое свойство (либо его отсутствие) может быть установлено эффективно для детерминированного автомата A, который эквивалентен $\tilde{A}$. Однако процедура преобразования $\tilde{A}$ в A обычно крайне трудоемка [1]. Более того, реализация этой процедуры требует работы со структурами данных, представляющими автоматы. Другой подход к этой проблеме состоит в переходе от автомата $\tilde{A}$ к кодирующей его системе булевых уравнений $S(\tilde{A})$. Такой переход осуществляется эффективно [2]. Диагностируемое свойство автомата $\tilde{A}$ либо его отсутствие может быть установлено на основе решения системы $S(\tilde{A})$. Решение системы $S(\tilde{A})$ может быть найдено с применением современных быстрых булевых решателей (SAT-решателей, [3]).

Следующий пример такого рода — автоматные модели в современной вычислительной биологии [4]. На первый взгляд совершенно непонятно, какие вычислительные методы следует использовать для их изучения. Однако с этими моделями естественным образом связываются некоторые дискретные функции (см., например, [5]), и для выявления тех или иных свойств моделей приходится решать задачи обращения таких функций, которые также допускают эффективную сводимость к булевым уравнениям.

[1]Работа поддержана грантом РФФИ, проект № 11-07-00377-а.

Еще одна область применения SAT-подхода связана с задачами комбинаторной оптимизации. Имеются мощные коммерческие пакеты решения оптимизационных задач из семейства 0—1-целочисленного линейного программирования (ЦЛП). Соответствующие алгоритмы комбинируют технику ветвей, границ и отсечений с методами решения задач линейного программирования над полем рациональных чисел. Однако данные методы подходят далеко не ко всем задачам комбинаторной оптимизации. Современные решатели ЦЛП-задач, как правило, не работают с нелинейными и невыпуклыми целевыми функциями. Между тем и такие задачи допускают эффективную сводимость к булевым уравнениям и, в конечном счете, к SAT-задачам.

В работе приведены результаты решения задачи поиска неподвижных точек и циклов автоматных отображений в генных сетях дискретно-автоматной природы со сложными многофакторными механизмами внутреннего взаимодействия и результаты параллельного решения SAT-задач, кодирующих квадратичную задачу о назначениях, в отношении которой коммерческие решатели, основанные на методе ветвей и границ, оказываются низкоэффективными.

ЛИТЕРАТУРА

1. *Хопкрофт Дж., Мотвани Р., Ульман Дж.* Введение в теорию автоматов, языков и вычислений. М.: Вильямс, 2002.
2. *Семенов А. А., Отпущенников И. В., Кочемазов С. Е.* Пропозициональный подход в задачах тестирования дискретных автоматов // Современные технологии. Системный анализ. Моделирование. 2009. № 4. С. 48–56.
3. `http://www.satlive.org` — Up-to-date links for the SATisfability Problem.
4. Системная компьютерная биология / под ред. Н. А. Колчанова, С. С. Гончарова, В. А. Лихошвая, В. А. Иванисенко. Новосибирск: Изд-во СО РАН, 2008.
5. *Евдокимов А. А., Кочемазов С. Е., Семенов А. А.* Применение символьных вычислений к исследованию дискретных моделей некоторых классов генных сетей // Вычислительные технологии. 2011. Т. 16. № 1. С. 30–47.

УДК 519.6

О ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЯХ ПРИ РЕАЛИЗАЦИИ МЕТОДА СОГЛАСОВАНИЯ В КРИПТОАНАЛИЗЕ

В. М. Фомичев

Метод согласования [1 – 4] применяется для определения ключа криптосистемы, как правило, по известным открытому и шифрованному тексту. Он менее трудоемок по сравнению с полным опробованием ключей, если функция шифрования $E(q, x)$ открытого текста x по ключу $q \in V_n = \{0, 1\}^n$ допускает декомпозицию на две функции как $E(q, x) = g(q, g'(q, x))$, где для множеств существенных ключевых переменных K и K' соответственно функций g и g' выполнено $K \setminus K' \neq \varnothing$ и $K' \setminus K \neq \varnothing$. Наибольший эффект от применения метода достигается, если множества K и K' равномощны и $K \cap K' = \varnothing$. При этом опробование ключа выполняется как независимое опробование переменных из множеств K и K' и ключ q определяется с вычислительной сложностью порядка $\mathrm{O}(2^{n/2})$ операций типа зашифрования-расшифрования при использовании памяти, достаточной для хранения порядка $\mathrm{O}(2^{n/2})$ ключей.

Оценим среднее время параллельных вычислений и размер памяти для реализации метода согласования применительно к некоторым итеративным симметричным блоч-

ным шифрам (СБШ) при условии, что ключ выбирается случайно равновероятно из ключевого множества.

Задача: для r-раундового СБШ вычислить n-битовый ключ q по известным t-битовым блокам x и y открытого и шифрованного текстов, где числа r, n, t — натуральные, $t \geqslant n$ и ключ q по блокам x и y определяется однозначно. В условиях метода согласования сделаем следующие предположения и обозначения:

1) ключ q есть конкатенация независимых ключей: $q = v \cdot w$, где $v \in V_m$, $w \in V_{n-m}$ и $m \leqslant n/2$;
2) функции шифрования первых l раундов и остальных $r-l$ раундов шифрования суть подстановки соответственно g_v и z_w, определяемые бинарными ключами v и w, где $l < r$.

Тогда зашифрование блока x в блок y с помощью подстановки E_q имеет вид

$$y = E_q(x) = g_v z_w(x) = z_w(g_v(x)). \tag{1}$$

Корректность предлагаемых алгоритмов следует из (1).

В модели вычислительной системы используются N идентичных вычислителей с неограниченной памятью, с одинаковыми производительностью и скоростью чтения/записи данных и др. Различаются два случая: кластерные вычисления (КВ) и распределенные вычисления (РВ). Преимущество модели КВ заключается в возможности активного обмена данными между вычислителями. Преимущество модели РВ, где координатор распределяет задания между процессорами — участниками вычислений и объединяет результаты выполнения заданий, в том, что число участников РВ может заметно превосходить число процессоров в кластерной системе. Вместе с тем обмен данными участник осуществляет только с координатором.

Кластерные вычисления

Пусть кластерная система имеет 2^k вычислителей, снабженных блоками памяти, $k \leqslant m$. Каждый вычислитель имеет номер, являющийся его адресом (число от 0 до $2^k - 1$). Для реализации алгоритма каждый вычислитель использует адресную память размера 2^{t-k} ячеек, в ячейке могут быть записаны несколько вариантов ключей — элементов V_n. Адреса ячеек суть элементы V_{t-k}.

Для любого двоичного вектора $(\alpha_1, \alpha_2, \ldots)$ размерности больше k обозначим

$$\delta(\alpha_1, \alpha_2, \ldots) = (\alpha_1, \ldots, \alpha_k), \quad \bar{\delta}(\alpha_1, \ \alpha_2, \ldots) = (\alpha_{k+1}, \alpha_{k+2}, \ldots).$$

Для вектора $\alpha = (\alpha_1, \ldots, \alpha_k) \in V_k$ и пространства векторов V_s, где $s \geqslant k$, обозначим

$$V_s(\alpha) = \{\xi \in V_s : \delta(\xi) = \alpha\}.$$

Алгоритм состоит из предварительного и оперативного этапов.

Предварительный этап (заполнение блоков памяти вычислителей).

Вычислитель с номером $\alpha \in V_k$ последовательно опробует ключи v из $V_m(\alpha)$ и вычисляет $g_v(x)$ для блока x. Затем пара $(v, g_v(x))$ направляется вычислителю с номером $\delta(g_v(x))$, который записывает ключ v в свою память по адресу $\bar{\delta}(g_v(x))$.

По завершении этапа множество ключей из V_m распределено по ячейкам памяти всех вычислителей. Обозначим через $Q(\alpha, \beta)$ множество ключей из V_m, записанных в памяти вычислителя с номером α по адресу β.

Оперативный этап (определение ключа).

1) Вычислитель с номером $\alpha \in V_k$ последовательно опробует ключи w из $V_{n-m}(\alpha)$ и вычисляет $(z_w)^{-1}(y)$ для блока y, затем пара $(w, (z_w^{-1})(y))$ направляется вычислителю с номером $\bar{\delta}((z_w^{-1})(y))$.
2) Вычислитель с номером $\delta((z_w^{-1})(y))$ обращается в свою память по адресу $\bar{\delta}((z_w^{-1})(y))$. Конкатенация $v \cdot w$ для каждого ключа v из множества $Q = Q(\delta((z_w^{-1})(y)), \bar{\delta}((z_w^{-1})(y)))$ есть кандидат на значение искомого ключа $q = v \cdot w$. Если $Q \neq \varnothing$, то вычислитель отбраковывает все ключи вида $v \cdot w$ (например, по критерию соответствия известным парам открытого и шифрованного текстов).

Характеристики метода.

Пусть $\tau_з$, $\tau_п$, $\tau_о$ — время реализации зашифрования, пересылки и обращения в память в некоторых условных единицах, и $\max\{\tau_з, \tau_п, \tau_о\} = \tau$. Тогда минимум среднего времени реализации метода $T(m)$ достигается при $m = \lfloor n/2 \rfloor$:

$$T = T(\lfloor n/2 \rfloor) = \mathrm{O}(\tau 2^{n/2-k}). \tag{2}$$

Надёжность метода равна 1. Каждому вычислителю достаточно иметь порядка $2^{n/2-k}$ ячеек, в которые записываются элементы $V_{n/2}$. Таким образом, совокупный объем требуемой памяти $2^{n/2}$ ячеек также распределяется между 2^k процессорами.

Распределенные вычисления

В системе РВ с 2^p участниками (вычислителями), $p \leqslant m$, каждый участник имеет номер, являющийся его адресом (число от 0 до 2^{p-1}). Алгоритм использует 2^t ячеек адресной памяти координатора, в каждую из которых может быть записано несколько вариантов ключей — элементов V_n. Участники могут отправлять данные координатору, но не могут обращаться к его памяти.

Алгоритм состоит из предварительного и оперативного этапов.

Предварительный этап (заполнение памяти координатора).

Вычислитель с номером $\alpha \in V_p$ последовательно при каждом ключе v из $V_m(\alpha)$ вычисляет $g_v(x)$ для блока x и направляет пару $(v, g_v(x))$ координатору, где ключ v записывается в память координатора по адресу $g_v(x)$. По завершении этапа множество ключей из V_m распределено по ячейкам памяти координатора. Обозначим через $Q(\beta)$ множество ключей из V_m, записанных в памяти координатора по адресу β.

Оперативный этап (определение ключа).

1) Вычислитель с номером $\alpha \in V_p$ последовательно при каждом ключе w из $V_{n-m}(\alpha)$ вычисляет $(z_w^{-1})(y)$ для блока y и направляет пару $(w, (z_w^{-1})(y))$ координатору.
2) Координатор обращается в память по адресу $(z_w^{-1})(y)$. Конкатенация каждого ключа v из $Q((z_w^{-1})(y))$ с ключом w есть кандидат на значение искомого ключа $q = v \cdot w$. Если $Q((z_w^{-1})(y)) \neq \varnothing$, то координатор подвергает отбраковке все ключи вида $v \cdot w$ (например, по критерию соответствия известным парам открытого и шифрованного текстов).

Характеристики метода.

Положим, что в каждый такт на 1-м этапе в любую ячейку памяти записывается не более одного варианта ключа v, на 2-м этапе из любой ячейки памяти извлекается не более одного варианта ключа v, т. е. замедления «из-за очередей» в работе вычислителей не происходит.

Тогда среднее время $T(m)$ алгоритма достигает минимума при $m = \lfloor n/2 \rfloor$:

$$T = T(\lfloor n/2 \rfloor) \approx 2^{n/2-p}(\tau_з + \tau_п + 2^p \tau_о). \tag{3}$$

Координатору достаточно иметь $2^{n/2}$ ячеек, в которые записываются элементы $V_{n/2}$. Отсюда среднее время алгоритма согласования по сравнению с полным опробованием ключей сокращается не более чем в 2^p раз и сокращение зависит от соотношения величин $\tau_{\text{о}}$, $\tau_{\text{з}}$ и $\tau_{\text{п}}$. Надёжность метода равна 1.

Комбинирование кластерных и распределенных вычислений

В данной модели РВ система использует 2^p участников, $p \leqslant m$. Каждый участник имеет номер, являющийся его адресом (число от 0 до $2^p - 1$). Координатор располагает кластерной подсистемой 2^k вычислителей, $k \leqslant p$, каждый вычислитель кластерной системы имеет номер, являющийся его адресом (число от 0 до $2^k - 1$), и блок памяти размера 2^{t-k} ячеек (адрес ячейки есть элемент V_{t-k}). В каждую ячейку могут быть записаны несколько вариантов ключей — элементов V_n. Участники РВ могут отправлять данные кластерным вычислителям, но не могут обращаться в память кластерных вычислителей.

Предварительный этап (заполнение памяти координатора).

Участник с номером $\alpha \in V_p$ последовательно при каждом ключе v из $V_m(\alpha)$ вычисляет $g_v(x)$ для блока x и направляет пару $(v, g_v(x))$ кластерному вычислителю с номером $\delta(g_v(x))$, который вычисляет адрес $\bar{\delta}(g_v(x))$ и записывает по этому адресу ключ v. По завершении этапа множество ключей из V_m распределено по блокам памяти кластерных вычислителей координатора. Обозначим через $Q(\alpha, \beta)$ множество ключей из V_m, записанных в блоке памяти вычислителя с номером α по адресу β.

Оперативный этап (определение ключа).

1) Вычислитель с номером $\alpha \in V_p$ последовательно при каждом ключе $w \in \in V_{n-m}(\alpha)$ вычисляет $(z_w^{-1})(y)$ для блока y и направляет пару $(w, (z_w^{-1})(y))$ кластерному вычислителю с номером $\delta((z_w^{-1})(y))$.
2) Кластерный вычислитель с номером $\delta((z_w^{-1})(y))$ обращается к своему блоку памяти по адресу $\bar{\delta}((z_w^{-1})(y))$. Конкатенация каждого ключа v из множества $Q = Q(\delta((z_w^{-1})(y)), \bar{\delta}((z_w^{-1})(y))$ с ключом w есть кандидат на значение ключа. Если $Q \neq \varnothing$, то вычислитель с номером $\delta((z_w^{-1})(y))$ все ключи вида $v \cdot w$ подвергает отбраковке (например, по критерию соответствия известным парам открытого и шифрованного текстов).

Характеристики метода.

Минимум $T(m)$ достигается при $m = \lfloor n/2 \rfloor$, и верны оценки

$$\mathrm{O}((\tau_{\text{з}} + \tau_{\text{п}})2^{n/2-p}) \leqslant \min T(m) \leqslant \mathrm{O}(\tau_{\text{о}} 2^{n/2-k}).$$

Следовательно, $\min T(m)$ может быть сокращен в несколько раз по сравнению с КВ и РВ (ср. с формулами (2), (3)). Коэффициент сокращения определяется соотношением скоростей шифрования, пересылки данных и обращения к памяти. Надёжность метода равна 1.

Кластерному вычислителю достаточно иметь $2^{n/2-k}$ ячеек, в которые записываются элементы $V_{n/2}$.

Выводы

Время определения ключа блочного шифра методом согласования может быть существенно сокращено по сравнению с однопроцессорной вычислительной системой:

1) при использовании КВ с числом процессоров 2^k — примерно в 2^k раз;
2) при использовании РВ с 2^p участниками — до 2^p раз, сокращение определяется соотношением скоростей шифрования, пересылки данных и обращения к памяти координатора;

3) при использовании РВ с 2^p участниками и подсистемы КВ координатора с числом процессоров 2^k, где $k \leqslant p$, — от 2^k до 2^p раз, сокращение определяется соотношением скоростей шифрования, пересылки данных и обращения к памяти кластерных вычислителей.

При использовании КВ память распределяется по вычислителям кластерной системы.

Подробное изложение представленных результатов можно найти в [5].

ЛИТЕРАТУРА

1. *Брассар Ж.* Современная криптология / пер. с англ. М.: Полимед, 1999.
2. *Грушо А. А., Тимонина Е. Е., Применко Э. А.* Анализ и синтез криптоалгоритмов. Курс лекций. Йошкар-Ола: МФ МОСУ, 2000.
3. *Фомичёв В. М.* Методы дискретной математики в криптологии. М.: ДИАЛОГ-МИФИ, 2010.
4. *Шнайер Б.* Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. М.: ТРИУМФ, 2002.
5. *Фомичёв В. М.* О реализации метода согласования в криптоанализе с помощью параллельных вычислений // Прикладная дискретная математика. 2011 (в печати).