SAT-подход в криптоанализе некоторых систем поточного шифрования*

А. А. Семенов, О. С. Заикин, Д. В. Беспалов, А. А. Ушаков
Институт динамики систем и теории управления СО РАН, Иркутск, Россия
e-mail: biclop@rambler.ru, oleg.zaikin@icc.ru, bespalov@irk.ru

A new approach for cryptoanalysis of some stream ciphering systems is considered. This approach is based on heuristic algorithms of SAT problem solving. A technique of reduction of cryptanalysis problems to SAT-problems is described. Examples of successful cryptanalysis of some generators of binary sequences (Geffe, Volfram and Gifford generators) are introduced.

Введение

В работе рассмотрен подход к криптоанализу генераторов ключевого потока, базирующийся на современных эвристических методах решения SAT-задач. Под термином "SAT-задачи" далее понимаются как задачи доказательства выполнимости/невыполнимости пропозициональных выражений (обычно записанных в конъюнктивной нормальной форме), так и задачи поиска выполняющих наборов выполнимых пропозициональных выражений.

Отметим, что базовые определения криптографии, а также описания основных криптографических конструкций, используемых в данной работе, могут быть найдены, например, в книге [1].

По-видимому, впервые постановка задачи криптоанализа как SAT-задачи была приведена в работе [2], здесь же данный тип криптоанализа получил название "логический криптоанализ". Авторы работы [3] независимо сформулировали концепцию логического криптоанализа в контексте общей проблемы обращения одного класса дискретных функций (см. также [4]).

В настоящей статье описана технология логического криптоанализа, доведенная до программной реализации, и приведены результаты успешного логического криптоанализа ряда известных систем поточного шифрования.

Основная цель статьи — демонстрация принципиальной возможности использования эвристического поиска над булевыми структурами в криптоанализе некоторых систем шифрования. Речь идет главным образом о поточных шифрах — системах, в которых жестким противовесом качественному рассеиванию и перемешиванию является требование высокой скорости генерации ключевого потока.

Отметим, что в известных технологиях криптоанализа поточных шифров, в первую очередь в корреляционном криптоанализе, используются весьма существенные объемы ключевого потока, формируемого генератором. Объем ключевого потока, используемо-

*Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (грант № 07-01-00400-а) и гранта Президента РФ НШ-1676.2008.1.

го в логическом криптоанализе, как правило, существенно меньше. Так, в известной атаке Т. Кейна и А. Шермана на генератор Гиффорда (см. [5]) для нахождения 64-битовой инициализирующей последовательности генератора используется 65536 бит (2^{16}) ключевого потока. Логический криптоанализ, реализованный в настоящей работе, позволял (в серии экспериментов) находить инициализирующую последовательность генератора Гиффорда на основе анализа 160 бит ключевого потока.

Второй плюс рассматриваемой технологии состоит в том, что она доведена до законченной программной реализации, чего нельзя сказать о многих методах криптоанализа, эффективность которых оценивается асимптотическими выражениями типа "$O(...)$" для "числа элементарных шагов". В некоторых случаях константы в подобных оценках не ясны, как не ясны и возможности эффективных программных реализаций такого рода атак.

Третий положительный момент данной технологии состоит в том, что она дает возможность построить классы аргументированно трудных тестов для различных задач комбинаторной оптимизации (этот тезис подробно комментируется в заключении).

Минусом описываемой в работе технологии является тот факт, что многослойность структуры используемых алгоритмов не дает возможности получения содержательных верхних оценок трудоемкости криптоанализа.

План статьи следующий. Первый раздел содержит базовые понятия и результаты. Здесь же поясняется техника пропозиционального кодирования алгоритмов и описан программный комплекс (LC-комплекс), функцией которого является сведение реальных задач криптоанализа к SAT-задачам. Во втором разделе кратко сообщается о наиболее эффективных на данный момент алгоритмах решения SAT-задач, а также о некоторых программных SAT-решателях. В третьем разделе приведены результаты успешного криптоанализа некоторых генераторов двоичных последовательностей, используемых в поточном шифровании.

1. Задачи обращения дискретных функций как SAT-задачи

Булевы переменные — это переменные, принимающие значения в множестве $\{0,1\}$. Через $\{0,1\}^n$, $n \in \mathrm{N}$, обозначается множество всех двоичных слов (последовательностей) длины n. Пусть $X = \{x_1, ..., x_n\}$ — множество булевых переменных. Термы x_i, $\overline{x_i}$, $i \in \{1, ..., n\}$, называются литералами над X. Литералы x и $\overline{x}$ называются контрарными. Дизъюнктом над X называется произвольная дизъюнкция литералов над X, в которой нет повторяющихся и контрарных литералов. Конъюнктивной нормальной формой (КНФ) над X называется произвольная конъюнкция различных дизъюнктов над X. Пусть L — произвольная формула алгебры логики (см. [6]) от переменных $x_1, ..., x_n$. Тот факт, что L при подстановке $x_1 = \alpha_1, ..., x_n = \alpha_n$, $\alpha_i \in \{0,1\}$, $i \in \{1, ..., n\}$, принимает значение $\beta \in \{0,1\}$, обозначаем через $L|_{(\alpha_1,...,\alpha_n)} = \beta$.

Пусть C — произвольная КНФ над множеством булевых переменных $X = \{x_1, ..., x_n\}$. Вектор $(\alpha_1, ..., \alpha_n) \in \{0,1\}^n$ называется набором, выполняющим C, если $C|_{(\alpha_1,...,\alpha_n)} = 1$. КНФ над X, для которой существует выполняющий набор, называется выполнимой, в противном случае КНФ называется невыполнимой. К SAT-задачам, рассматриваемым далее, относится задача распознавания выполнимости произвольной КНФ, а также задача поиска выполняющего набора произвольной выполнимой КНФ. Задача распознавания выполнимости произвольной КНФ является исторически первой NP-полной задачей (см. [7]).

Пусть $L(x_1, ..., x_n)$ — произвольная формула алгебры логики. Выражения вида

$$L(x_1, ..., x_n) = 0,$$
$$L(x_1, ..., x_n) = 1$$

называются логическими уравнениями. Решить логическое уравнение $L(x_1, ..., x_n) = \beta$, $\beta \in \{0, 1\}$, означает найти такой набор $(\alpha_1, ..., \alpha_n)$, $\alpha_i \in \{0, 1\}$, $i \in \{1, ..., n\}$, что $L(x_1, ..., x_n)|_{(\alpha_1,...,\alpha_n)} = \beta$. Если такого набора не существует, то рассматриваемое логическое уравнение не имеет решений.

В качестве формальной вычислительной модели далее используется машина Тьюринга с входным алфавитом $\Sigma = \{0, 1\}$. Через $\{0, 1\}^n, n \in \mathrm{N}$, обозначается множество всех двоичных последовательностей длины n.

Дискретной функцией называется произвольная (вообще говоря, частичная) функция вида $f_n : \{0, 1\}^n \to \{0, 1\}^*$, $n \in \mathrm{N}$; здесь $\{0, 1\}^* = \bigcup\limits_{n\in\mathrm{N}} \{0, 1\}^n$. Функции вида $f_n : \{0, 1\}^n \to \{0, 1\}$ называются булевыми. Через $Dom f_n \subseteq \{0, 1\}^n$ обозначается область определения функции f_n, а через $Ran f_n \subset \{0, 1\}^*$ — ее область значений. Дискретную функцию f_n назовем всюду определенной (тотальной, [8]), если $Dom f_n = \{0, 1\}^n$.

Пусть M — программа машины Тьюринга (МТ-программа), которая останавливается на произвольном слове из $\{0, 1\}^*$, причем в заключительной конфигурации на ленте записано некоторое двоичное слово.

Очевидно, что данная программа вычисляет некоторое семейство тотальных дискретных функций, обозначаемое через $f^M = \left\{f_n^M\right\}_{n\in N}$. Согласно тезису Черча, под алгоритмически вычислимыми семействами тотальных дискретных функций следует понимать именно семейства вида f^M. Индекс "M" в дальнейших обозначениях опускается.

Для данных функций могут быть введены понятия временной и пространственной сложности так, как это делается, например, в книге [9]. Далее везде термин "вычислительная сложность" обозначает сложность по времени в смысле [9].

Определение 1 (см. также [4]). *Класс $\Im$ образован семействами тотальных дискретных функций, вычислимых МТ-программами, вычислительная сложность которых ограничена полиномом от длины входа.*

Определение 2. *Проблема обращения дискретной функции f_n семейства $f = \{f_n\}_{n\in N}$, $f \in \Im$, ставится следующим образом: дано двоичное слово $y \in Ran\ f_n$, требуется найти такое слово $x \in \{0, 1\}^n$, что $f_n(x) = y$.*

Проблемы обращения дискретных функций описанного класса будут далее сводиться к задачам решения логических уравнений. Логические уравнения рассматриваются как слова над конечными алфавитами, которые могут быть взаимооднозначно преобразованы (закодированы) в двоичные слова (см. [9] или [10]). При этом под объемом логического уравнения понимается длина двоичного слова, кодирующего данное уравнение в некоторой фиксированной кодировке. Далее все такие кодировки предполагаются "разумными" (в терминологии [9]).

Теорема 1. *Рассмотрим произвольное семейство дискретных функций $f = \{f_n\}_{n\in\mathrm{N}}$, $f \in \Im$. Проблема обращения функций данного семейства за полиномиальное от n время преобразуется в проблему поиска решений логических уравнений вида $L(x_1, ..., x_{p(n)}) = 1$, где $p()$ — некоторый полином, а $L(x_1, ..., x_{p(n)})$ — КНФ над множеством булевых переменных $X = \left\{x_1, ..., x_{p(n)}\right\}$. Функция объема получаемого при этом семейства КНФ есть некоторый полином от n.*

Доказательство данного утверждения аналогично доказательству теоремы Кука (см. [7, 9]) и, ввиду своей громоздкости, здесь не приводится. Отметим лишь характерные моменты. Смысл утверждения состоит в том, что процедуру вычисления каждой функции $f_n, n \in N$, семейства $f \in \Im$ можно рассматривать как применение программы M к произвольным входам из $\{0,1\}^n$. Структура двоичной машины Тьюринга дает возможность описать этот процесс в виде системы логических уравнений, вообще говоря, в достаточно произвольной форме. Последующие преобразования позволяют перейти к одному уравнению вида

$$L(x_1, ..., x_{p(n)}) = 1, \tag{1}$$

где $p()$ — некоторый полином, а $L(x_1, ..., x_{p(n)})$ — формула алгебры логики, имеющая вид КНФ.

Переменных в полученном итоговом уравнении может быть больше, чем в исходном, однако функция роста их числа есть полином. Очень важно, что возможно построить такой переход от исходной системы логических уравнений к уравнению вида (1), при котором не теряются решения исходной системы и не возникает "лишних" решений. Иными словами, можно гарантировать, что тот $x' \in \{0,1\}^n$, который может быть эффективно выделен из некоторого решения (1), действительно является прообразом $y \in Ran f_n$. Данное свойство обозначается как консервативность погружающей редукции. Более точно (в соответствии с [9]) преобразование логического уравнения U_1 над множеством булевых переменных X_1 в логическое уравнение U_2 над множеством булевых переменных X_2 называется консервативным, если множества решений уравнений U_1 и U_2 равномощны. По-видимому, одним из первых вопросы консервативности рассматривал Дж. Саймон в диссертации [11]. Данная работа недоступна. Ниже приведено утверждение (теорема 2), дающее общую схему консервативного перехода от произвольных систем логических уравнений к системам вида (1).

Все рассматриваемые далее булевы функции предполагаются тотальными. Рассмотрим логическое уравнение

$$F\left(h_1\left(x_1^1, ..., x_{r_1}^1\right), ..., h_s\left(x_1^s, ..., x_{r_s}^s\right)\right) = 1. \tag{2}$$

Здесь $h_1, ..., h_s$ — некоторые (в общем случае также сложные) булевы функции. Положим

$$X = \{x_1, ..., x_n\} = \bigcup\nolimits_{i=1}^{s} \left\{x_1^i, ..., x_{r_i}^i\right\}.$$

Таким образом, $F : \{0,1\}^n \to \{0,1\}$. Решениями (2), если они существуют, являются векторы из $\{0,1\}^n$. Рассматриваем проблему поиска решений уравнения (2). Введем новую булеву переменную u_1, зависящую от переменных $x_1^1, ..., x_{r_1}^1$ в том смысле, что u_1 принимает значение "1" тогда и только тогда, когда $h_1(x_1^1, ..., x_{r_1}^1) = 1$. Иными словами, имеет место логическая эквивалентность $u_1 \equiv h_1(x_1^1, ..., x_{r_1}^1)$. Данную эквивалентность представляем в виде КНФ над множеством переменных $\left\{x_1^1, ..., x_{r_1}^1, u_1\right\}$. Полученную КНФ обозначаем через $C(u_1 \equiv h_1(x_1^1, ..., x_{r_1}^1))$. Рассмотрим логическое уравнение

$$C\left(u_1 \equiv h_1\left(x_1^1, ..., x_{r_1}^1\right)\right) F\left(u_1, ..., h_s\left(x_1^s, ..., x_{r_s}^s\right)\right) = 1. \tag{3}$$

Теорема 2. *Множество решений уравнения (2) пусто тогда и только тогда, когда пусто множество решений уравнения (3). Если множества решений уравнений (2) и (3) не пусты, то существует биекция между этими множествами, причем от*

любого решения уравнения (3) за линейное время осуществим переход к соответствующему решению уравнения (2).

Подробное доказательство данного утверждения приведено в [12]. Следующий пример иллюстрирует важные в практическом отношении свойства описанных преобразований.

Пример 1. Требуется найти решение произвольного логического уравнения вида

$$x_1 \oplus ... \oplus x_n = 1. \tag{4}$$

Хорошо известно (см., например, [13]), что сложность КНФ-представлений булевых функций вида $f(x_1, ..., x_n) = x_1 \oplus ... \oplus x_n$ растет как экспонента от n. Однако можно от уравнения (4) перейти к уравнению

$$(y_1 \equiv x_1 \oplus x_2)...C(y_{n-1} \equiv y_{n-2} \oplus x_n)y_{n-1} = 1, \tag{5}$$

где формулы $(y_i \equiv y_{i-1} \oplus x_{i+1}), y_0 = x_1, i \in \{1, ..., n-1\}$, обозначают КНФ эквивалентностей, записанных в скобках. Несложно видеть, что логическое уравнение (5) — это уравнение над множеством переменных $Y = X \cup \{y_1, ..., y_{n-1}\}$, левая часть которого имеет вид КНФ, состоящей из $4n-3$ дизъюнктов (последний дизъюнкт однолитеральный). В соответствии с теоремой 2 множества решений (4) и (5) равномощны и от любого решения (5) можно эффективно перейти к соответствующему решению (4).

В соответствии с [14] генератор псевдослучайной двоичной последовательности может быть определен таким образом.

Определение 3 (см. [14]). *Генератор псевдослучайной двоичной последовательности — это детерминированный алгоритм, который, получая на входе случайную двоичную последовательность длины n (выбранную из $\{0,1\}^n$ в соответствии с равномерным распределением), выдает на выходе последовательность длины m, $m >> n$. Данная последовательность по некоторым признакам должна вести себя как случайная последовательность. Последовательность на входе генератора называется инициализирующей, последовательность на выходе генератора называется выходной.*

В некоторых других источниках (см., например [15]) выходная последовательность генератора называется ключевым потоком. Несложно понять, что детерминированный алгоритм, упоминаемый в определении 3, задает некоторую дискретную функцию, определенную всюду на $\{0,1\}^n$. С другой стороны, время работы генератора по очевидным практическим соображениям должно ограничиваться полиномом от длины входа. В этом случае дискретная функция, задаваемая алгоритмом генератора, принадлежит некоторому семейству из класса $\Im$. Именно такого рода генераторы двоичных последовательностей рассматриваются далее.

В общем случае под криптоанализом генератора понимается поиск инициализирующей последовательности по некоторому известному фрагменту выходной последовательности. В соответствии с [2 и 3], криптоанализ, рассматриваемый как SAT-задача, называем логическим криптоанализом. Схема логического криптоанализа генератора в контексте описанного выше подхода (теоремы 1 и 2) состоит в следующем. Алгоритм вычисления соответствующей дискретной функции представляется в виде некоторой системы логических уравнений. Данная система при помощи консервативных преобразований приводится к виду (1). В левую часть полученного уравнения подставляются биты известного фрагмента ключевого потока. В итоге имеем уравнение вида (1), в левой части которого находится выполнимая КНФ. Находим ее выполняющий набор, из которого выделяем биты инициализирующей последовательности.

Простейшими примерами генераторов двоичных последовательностей являются регистры сдвига с линейной обратной связью (РСЛОС, см., например, [14, 16]). Напомним, что регистром сдвига с линейной обратной связью (РСЛОС) $< n, P(X) >$ длины n и полиномом связи

$$P(X) = c_n X^n + c_{n-1} X^{n-1} + ... + c_1 X + 1$$

над полем $GF(2)$ называется устройство, состоящее из n ячеек (регистров) $r_1, ..., r_n$, в каждой из которых может находиться один бит информации. Информация в ячейках модифицируется при помощи специального тактового механизма. Моменты (такты), когда происходят запись, модификация и считывание информации, обозначаем через τ ($\tau \in \{0, 1, 2, ...\}$). В начальном такте (при $\tau = 0$) в ячейки $r_1, ..., r_n$ заносится инициализирующая последовательность. В каждый такт $\tau, \tau \in \{1, 2, ...\}$, РСЛОС выдает (считывает) в качестве 1 бита информации текущее значение ячейки r_n. Кроме этого, происходит следующая модификация данных в ячейках:

$$r_i \to r_{i+1},\ i \in \{1, ..., n-1\},$$
$$f(r_1, ..., r_n) \to r_1,$$

где

$$f(r_1, ..., r_n) = c_1 r_1 \oplus ... \oplus c_n r_n,$$

$c_i,\ i \in \{1, ..., n\}$, — коэффициенты из поля $GF(2)$. Запись $r_j \to r_k$ означает, что данные из ячейки r_j переносятся в ячейку r_k.

Очевидно, что РСЛОС представляет собой генератор двоичной последовательности в смысле определения 3. Выходная последовательность длины m данного генератора — это последовательность битов, являющаяся результатом первых m тактов работы РСЛОС ($\tau \in \{1, ..., m\}$). РСЛОС используются в качестве примитивов во многих системах поточного шифрования.

Пример 2. Логический криптоанализ РСЛОС $< 3, X^3 + X^2 + 1 >$. Схема работы данного генератора представлена на рис. 1. Пусть в начальный момент времени в ячейках регистра находится инициализирующая последовательность $s = (s_1, s_2, s_3) = (0, 0, 1)$. Предположим, что данная последовательность не известна. Предположим также, что известны четвертый, пятый и шестой биты выходной последовательности. А именно, полагаем, что в выходной последовательности $t_4 = 0,\ t_5 = 1,\ t_6 = 1$. Используем информацию об алгоритмической природе РСЛОС, а именно выписываем следующие логические уравнения:

$$s_1 \oplus s_2 \oplus t_4 = 0,\ s_2 \oplus s_3 \oplus t_5 = 0, s_3 \oplus t_4 \oplus t_6 = 0.$$

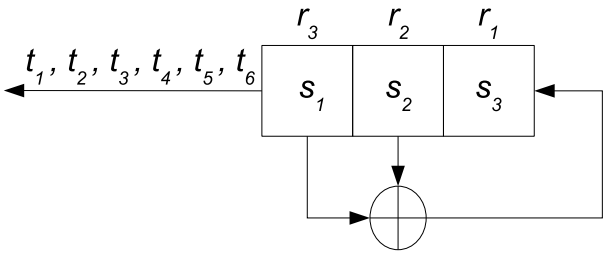

Рис. 1. Схема работы РСЛОС

Применение консервативных преобразований описанного выше типа к данной системе уравнений дает следующую КНФ:

$$C = (s_1 \vee s_2 \vee \overline{t_4})\,(s_1 \vee \overline{s_2} \vee t_4)\,(\overline{s_1} \vee s_2 \vee t_4)\,(\overline{s_1} \vee \overline{s_2} \vee \overline{t_4}) \times$$
$$\times\,(s_2 \vee s_3 \vee \overline{t_5})\,(s_2 \vee \overline{s_3} \vee t_5)\,(\overline{s_2} \vee s_3 \vee t_5)\,(\overline{s_2} \vee \overline{s_3} \vee \overline{t_5}) \times$$
$$\times\,(s_3 \vee t_4 \vee \overline{t_6})\,(s_3 \vee \overline{t_4} \vee t_6)\,(\overline{s_3} \vee t_4 \vee t_6)\,(\overline{s_3} \vee \overline{t_4} \vee \overline{t_6})\,.$$

Подставим известные значения переменных t_4, t_5, t_6 в КНФ C:

$$C|_{t_4=0,t_5=1,t_6=1} = (s_1 \vee s_2 \vee 1)\,(s_1 \vee \overline{s_2})\,(\overline{s_1} \vee s_2)\,(\overline{s_1} \vee \overline{s_2} \vee 1) \times$$
$$\times\,(s_2 \vee s_3)\,(s_2 \vee \overline{s_3} \vee 1)\,(\overline{s_2} \vee s_3 \vee 1)\,(\overline{s_2} \vee \overline{s_3})\,s_3\,(s_3 \vee 1 \vee 1) \times$$
$$\times\,(\overline{s_3} \vee 1)\,(\overline{s_3} \vee 1) = (s_1 \vee \overline{s_2})\,(\overline{s_1} \vee s_2)\,(s_2 \vee s_3)\,(\overline{s_2} \vee \overline{s_3})\,s_3.$$

Таким образом, задача поиска инициализирующей последовательности сведена к задаче поиска выполняющего набора полученной КНФ. Такой набор существует и единствен: $s_1 = 0,\ s_2 = 0,\ s_3 = 1$.

Описанная в данном пункте техника сведения систем логических уравнений в произвольной форме к уравнениям в форме (1) была программно реализована в виде так называемого LC-комплекса (см. [17]). Данный комплекс состоит из высокоуровневого языка программирования (LC-язык), на котором записываются алгоритмы вычисления рассматриваемых функций, и специального транслятора (LC-транслятор), транслирующего тексты на LC-языке в логические выражения в формате КНФ. Получаемые SAT-задачи решаются при помощи специальных программ, называемых SAT-решателями.

2. Архитектура современных эффективных SAT-решателей

В данном разделе кратко описываются основные особенности архитектуры наиболее эффективных SAT-решателей.

Ядро всякого SAT-решателя — некоторый алгоритм логического вывода. Подавляющее большинство современных SAT-решателей, показывающих высокие результаты на специализированных конкурсах (см. [18]), использует в качестве ядра алгоритм DPLL (см. [19]). Данный алгоритм — это модифицированный вариант процедуры Девиса и Патнема (см. [20]). Основное отличие DPLL от алгоритма Девиса—Патнема состоит в стратегии распространения булевых ограничений (Boolean constraint propagation), использующей правило единичного дизъюнкта (unit clause).

Пример 3. Рассматриваем КНФ

$$(x_1 \vee \overline{x_2})\,(\overline{x_1} \vee x_2)\,(\overline{x_1} \vee \overline{x_2} \vee x_3)\,(\overline{x_2} \vee \overline{x_3} \vee x_4)\,(x_3 \vee \overline{x_4})\,.$$

Угадываем $x_4 = 1$. После подстановки угаданного значения в исходную КНФ получаем КНФ

$$(x_1 \vee \overline{x_2})\,(\overline{x_1} \vee x_2)\,(\overline{x_1} \vee \overline{x_2} \vee x_3)\,(x_3)\,.$$

Очевидно, что если данная КНФ выполнима, то в любом выполняющем ее наборе переменная x_3 принимает значение единицы. В подобных случаях говорят, что угадывание $x_4 = 1$ индуцирует присвоение $x_3 = 1$ по правилу единичного дизъюнкта. Итак, можно заключить, что в любом наборе, выполняющем исходную КНФ, в котором $x_4 = 1$, переменная x_3 также принимает значение единицы, а значения переменных x_1 и x_2 дают набор, выполняющий КНФ

$$(x_1 \vee \overline{x_2})\,(\overline{x_1} \vee x_2)\,.$$

На данном этапе требуется осуществлять новое угадывание. Например, угадывание $x_1 = 1$ и последующее применение правила единичного дизъюнкта дает набор $x_1 = x_2 = 0$, выполняющий КНФ $(x_1 \vee \overline{x_2})\,(\overline{x_1} \vee x_2)$. После этого относительно исходной КНФ заключаем, что она выполнима на наборе (1111).

В некоторых случаях результатами применения правила единичного дизъюнкта могут быть так называемые конфликты. Конфликтом называется ситуация, когда по правилу единичного дизъюнкта (из угаданных на текущий момент присвоений) выводятся одновременно $x = 0$ и $x = 1$ (для некоторой булевой переменной x). Например, для КНФ

$$(x_1 \vee x_3)\,(\overline{x_1} \vee x_2)\,(x_1 \vee \overline{x_3})\,(\overline{x_1} \vee \overline{x_2})$$

угадывание $x_1 = 1$ дает вывод из второго дизъюнкта — $x_2 = 1$, а из четвертого дизъюнкта — $x_2 = 0$. Если такое происходит, то вступает в действие процедура разрешения конфликта. В рассматриваемом примере следует изменить присвоение $x_1 = 1$ на противоположное, т. е. положить $x_1 = 0$.

Переменные, значения которых угадываются, называют переменными уровней решений (decision level), уровни решений при этом нумеруются. На самом деле иерархию уровней решений можно представить в виде бинарного дерева, корнем которого служит первый уровень, помечаемый соответствующей булевой переменной. Потомки корня соответствуют переменным, значения которых угадываются и индуцируются на последующих уровнях. Ветви, выходящие из произвольной вершины, помечаются значениями соответствующей переменной. Листья дерева помечаются как sat либо как conflict. Путь из корня в лист, помеченный как sat, определяет (совместно с индуцированными на данном пути присвоениями) набор, выполняющий исходную КНФ. Путь из корня в лист, помеченный как conflict, определяет последовательность угадываний, из которой по правилу единичного дизъюнкта был выведен конфликт. Несложно понять, что если КНФ выполнима, то для любой альтернативы корня данного дерева всегда найдется путь в лист, помеченный как sat. Если же КНФ невыполнима, то все пути из любой альтернативы корня будут оканчиваться листьями, помеченными как conflict. Очевидно, что для данного дерева можно естественным образом определить процедуру отсечения его поддеревьев, в которых любой путь из корня оканчивается листом, помеченным как conflict. Возможности нетривиальных отсечений приводят к сокращению перебора.

Первоначально в DPLL был принят элементарный механизм разбора конфликтов — хронологический бэктрекинг. При хронологическом бэктрекинге разрешение конфликта происходит за счет изменения последнего (перед конфликтом) уровня решения — значение угаданной на этом уровне переменной меняется на противоположное. Позже была сформулирована концепция нехронологического бэктрекинга (или бэкджампинга), которая привела к созданию по-настоящему скоростных SAT-решателей. Ключевой механизм бэкджампинга — процедура Clause Learning (далее — CL-процедура), позволяющая запоминать информацию о конфликтах. Далее мы иллюстрируем основные перечисленные моменты, используя обозначения, принятые в [21].

Пример 4. Рассмотрим КНФ

$$(\overline{x_1} \vee x_2 \vee x_4)\,(\overline{x_1} \vee \overline{x_2} \vee \overline{x_3})\,(x_1 \vee x_2 \vee x_3)\,(\overline{x_1} \vee x_3 \vee \overline{x_4})\,(x_1 \vee \overline{x_2} \vee x_3)\,. \qquad (6)$$

Следуя [21], обозначаем угаданные присвоения через $x = \alpha@s$; здесь $\alpha \in \{0, 1\}$ — угаданное значение переменной x, а $s \in N$ — номер уровня решения; индуцированные

присвоения обозначаются через $x = \beta,\ \beta \in \{0,1\}$. Процесс вывода индуцированных присвоений по правилу единичного дизъюнкта можно представить в виде специального ориентированного графа, называемого графом вывода (Implication Graph).

На рис. 2 изображен граф вывода для двух уровней решения: на первом уровне угадывается $x_3 = 0$, на втором уровне — $x_2 = 0$, после чего из третьего дизъюнкта по правилу единичного дизъюнкта выводится индуцированное присвоение $x_1 = 0$ (номера дизъюнктов, на которых срабатывает правило единичного дизъюнкта, приписываются дугам графа вывода). Далее видим, что правило единичного дизъюнкта выводит из четвертого дизъюнкта $x_4 = 0$, а из первого — $x_4 = 1$, т. е. имеем конфликт. CL-процедура запоминания информации о конфликте заключается в конъюнктивном приписывании к исходной КНФ нового ограничения — дизъюнкта, запрещающего присвоения, приведшие к конфликту. В рассматриваемом случае к конфликту привели одновременные присвоения $x_3 = 0$ и $x_2 = 0$, таким образом, от исходной КНФ переходим к КНФ

$$(\overline{x_1} \vee x_2 \vee x_4)\,(\overline{x_1} \vee \overline{x_2} \vee \overline{x_3})\,(x_1 \vee x_2 \vee x_3)\,(\overline{x_1} \vee x_3 \vee \overline{x_4})\,(x_1 \vee \overline{x_2} \vee x_3)\,(x_2 \vee x_3)\,. \qquad (7)$$

Несложно видеть, что данная КНФ выполнима на тех и только тех наборах, на которых выполнима исходная КНФ. Построим граф вывода для данной КНФ, предположив, что на первом уровне происходит угадывание $x_3 = 0$ (рис. 3). Таким образом, исходная КНФ выполнима на наборе $x_1 = 1,\ x_2 = 1,\ x_3 = 0,\ x_4 = 0$.

В общем случае использование CL-процедуры позволяет точно выявлять присвоения, ответственные за конфликт, вследствие чего возможны откаты не к последнему уровню решения (как в хронологическом бэктрекинге), а к более ранним (в иерархии угадываний) уровням. В этом и состоит основной конструктивный момент нехронологического бэктрекинга (бэкджампинг). Следует отметить, что использование CL-процедуры приводит к росту объема памяти, задействуемой решателем в ходе решения SAT-задачи.

Помимо перечисленных, важнейшими компонентами современных эффективных SAT-решателей являются эвристики выбора переменных и специальные структуры данных, позволяющие совершать откаты, оперируя при этом минимальным объемом информации.

Эвристики выбора позволяют при угадывании очередной переменной уровня решения делать выбор, исходя из некоторых “разумных” предположений. Первые эвристики выбора не использовали информации о ходе поиска и получили название статических. Позже были введены динамические эвристики, используемые в настоящее

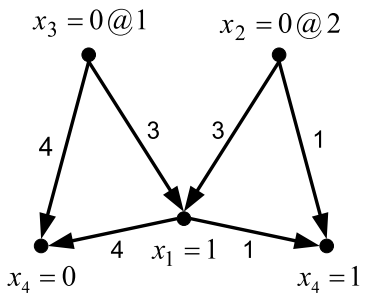

Рис. 2. Граф вывода для КНФ (6)

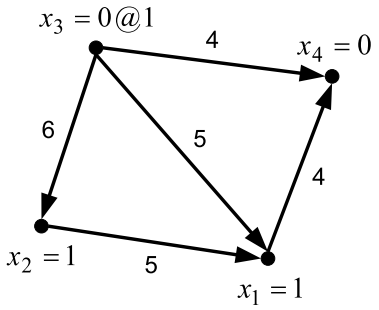

Рис. 3. Граф вывода для КНФ (7)

время в большинстве эффективных SAT-решателей. Впервые динамическая эвристика выбора переменных уровней решения была описана в работе [22] и получила название VSIDS (Variable State Independent Decaying Sum). Основная ее идея заключается в присвоении переменным специальных индексов активности — активность переменной тем выше, чем чаще данная переменная принимает участие в конфликтах. Обычно динамические эвристики выбора переменных применяются в сочетании с рестартами. Впервые рестарты были использованы в SAT-решателе zchaff (см. [23]), хотя теоретическое обоснование эффективности рестартов в DPLL-поиске было дано несколько позднее (см. [24]). Согласно данной концепции, процесс поиска разбивается рестартами на фрагменты (этапы). После каждого рестарта угадывание переменных уровней решения происходит в соответствии с построенной на предыдущих этапах поиска таблицей активности переменных (таблица активности в каждом фрагменте поиска модифицируется). Описанный подход позволяет сохранять и эффективно использовать предысторию поиска.

Быстрые структуры данных, описанные в [23, 25], предназначены для эффективной реализации стратегии распространения булевых ограничений и организации откатов с наименьшими вычислительными затратами.

Проблематика построения эффективных программных SAT-решателей становится в последние годы весьма интенсивно развивающимся направлением. В Internet регулярно проводятся конкурсы SAT-решателей и обновляются библиотеки тестовых примеров (см. [18]). Наиболее эффективные на данный момент SAT-решатели — это minisat (см. [26]), zchaff (см. [27]), berkmin (см. [28]).

3. Логический криптоанализ некоторых генераторов ключевого потока (успешные атаки)

Далее рассматриваются некоторые генераторы, описанные в источниках [15, 29]. В работе [30] детально разобран логический криптоанализ генератора Геффе (обычного). В данном разделе мы приводим результаты успешного логического криптоанализа усиленного варианта генератора Геффе, клеточно-автоматного генератора Вольфрама и генератора Гиффорда. Во всех численных экспериментах использовалась версия LC-комплекса, дополненная блоком минимизации булевых функций в классе КНФ (см. [31]). Минимизация осуществлялась при помощи пакета ESPRESSO (см. [32]). Для каждого из рассматриваемых генераторов были построены серии из 10 тестов (в каждом тесте инициализирующая последовательность генерировалась случайным образом). Задачи поиска выполняющих наборов КНФ решались при помощи SAT-решателя minisat 2.0 (см. [26]).

Для успешного криптоанализа генераторов Геффе и Вольфрама оказалось достаточно ресурсов персонального компьютера. Для криптоанализа генератора Гиффорда использовался вычислительный кластер Blackford ИДСТУ СО РАН (см. [33]). Данный ресурс имеет следующие характеристики:

— четыре вычислительных узла;

— восемь четырехъядерных процессоров Intel E5345 Xeon Quad Core 2.33 GHz;

— суммарный объем оперативной памяти на узлах — 16 GB;

— суммарный объем дисковой памяти на узлах — 1280 GB;

— интерконнект — Gigabit Ethernet.

В целях единообразного представления мы приводим здесь результаты криптоанализа генераторов Геффе и Вольфрама на одном ядре процессора Intel E5345 Xeon Quad Core 2.33 GHz (в каждом эксперименте были доступны 4 GB оперативной памяти).

3.1. Генератор Геффе

Данный генератор (см. [14]) представляет собой один из простейших типов композиционных генераторов, основной принцип работы которых — смешивание выходов нескольких РСЛОС посредством некоторой нелинейной булевой функции. Общая схема генератора Геффе представлена на рис. 4.

Здесь z_i^τ — биты, получаемые на выходе РСЛОС, с номером $i \in \{1,2,3\}$, в моменты времени $\tau \in \{1,2,...,n,...\}$ (момент $\tau = 0$ соответствует начальному заполнению регистров, сдвиги регистров происходят синхронно). Для каждого натурального $\tau \in \{1,2,...,n,...\}$ значением $\varphi\left(z_1^\tau, z_2^\tau, z_3^\tau\right)$ является 1 бит. Для обычного генератора Геффе перемешивающая функция имеет вид $\varphi(z_1^\tau, z_2^\tau, z_3^\tau) = z_1^\tau z_2^\tau \oplus z_2^\tau z_3^\tau \oplus z_3^\tau$, в усиленном генераторе Геффе — это $\varphi(z_1^\tau, z_2^\tau, z_3^\tau) = z_1^\tau z_2^\tau \oplus z_2^\tau z_3^\tau \oplus z_1^\tau z_3^\tau$. В [29] отмечается, что генератор Геффе демонстрирует высокую линейную сложность и хорошие статистические свойства генерируемой последовательности. Тем не менее он является слабым по отношению к корреляционным атакам, требующим значительного объема ключевого потока. В [15] описана линеаризационная атака на генератор Геффе, сложность которой ниже сложности известных вариантов корреляционных атак. Логический криптоана-

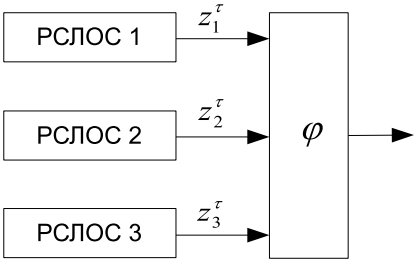

Рис. 4. Схема генератора Геффе

Т а б л и ц а 1. Результаты логического криптоанализа генератора Геффе

Обычный генератор Геффе		Усиленный генератор Геффе	
№ КНФ	Время решения	№ КНФ	Время решения
1	1 мин 31 с	1	3 мин 42 с
2	24 с	2	1 мин 22 с
3	39 с	3	39 с
4	32 с	4	9 мин 51 с
5	1 мин 19 с	5	2 мин 2 с
6	2 мин 59 с	6	2 мин 39 с
7	3 мин 52 с	7	9 мин 50 с
8	29 с	8	29 мин 23 с
9	11 с	9	4 мин 14 с
10	2 мин 42 с	10	3 мин 14 с
В среднем	1 мин 28 с	*В среднем*	6 мин 42 с

лиз применялся к усиленному генератору Геффе с РСЛОС, задаваемыми следующими примитивными полиномами над полем $GF(2)$:

$$\text{РСЛОС 1: } x^{31}+x^7+1, \quad \text{РСЛОС 2: } x^{32}+x^7+x^5+x^3+x^2+x+1,$$

$$\text{РСЛОС 3: } x^{33}+x^{16}+x^4+x+1.$$

Таким образом, суммарная длина инициализирующей последовательности составляла 96 бит. Анализируемый фрагмент выходной последовательности имел длину 192 бита. Размерность КНФ по обычному генератору Геффе — 768 переменных, 14 388 дизъюнктов. Размерность КНФ по усиленному генератору Геффе — 768 переменных, 14 772 дизъюнкта (табл. 1).

3.2. Клеточно-автоматный генератор Вольфрама

Данный генератор (см. [29]) представляет собой клеточный автомат с обновляемой синхронно рабочей областью. Рабочая область генератора — это лента, разбитая на n ячеек (клеток), в каждую из которых занесены биты инициализирующей последовательности. Далее значения всех ячеек обновляются в соответствии с основными принципами теории клеточных автоматов — новое значение ячейки с номером i есть значение некоторой булевой функции от текущих значений ячеек с номерами $i-1$, i и $i+1$. При вычислении значений в граничных ячейках принимается циклическая нумерация, т. е. нулевая ячейка — это ячейка с номером n, а ячейка с номером $n+1$ — это первая ячейка. Новые состояния всех ячеек вычисляются одновременно (синхронное обновление). Биты ключевого потока снимаются с некоторой фиксированной ячейки, например, с первой. Общая схема генератора Вольфрама представлена на рис. 5.

Здесь $a_{i-1}^{\tau}, a_i^{\tau}, a_{i+1}^{\tau}$ — окрестность ячейки с номером i на шаге с номером τ (момент $\tau = 0$ соответствует начальному заполнению ленты). Показано, как содержимое i-й ячейки меняется на $a_i^{\tau+1}$ при помощи функции переходов f.

В оригинальной работе Вольфрама (см. [34]) была предложена следующая функция переходов:

$$a_i^{\tau+1} = f\left(a_{i-1}^{\tau}, a_i^{\tau}, a_{i+1}^{\tau}\right) = a_{i-1}^{\tau} \oplus \left(a_i^{\tau} \vee a_{i+1}^{\tau}\right), \quad i = \{1,2,...,n\}, \quad \tau \in \{1,2,3,...\}.$$

Генератор Вольфрама интересен прежде всего тем, что его криптоанализ не укладывается в разработанные “классические” схемы для генераторов на основе сдвиговых регистров. Тем не менее, В. Майером и О. Штаффельбахом (см. [29]) была предложена корреляционная атака, демонстрирующая криптографическую слабость функции Вольфрама. Данный генератор оказался нестойким по отношению и к логическому криптоанализу.

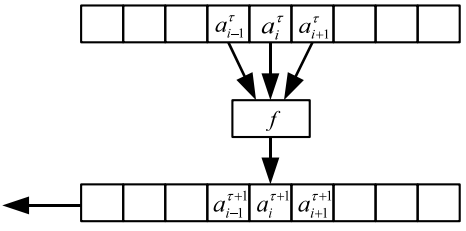

Рис. 5. Схема генератора Вольфрама

Т а б л и ц а 2. Результаты логического криптоанализа генератора Вольфрама

№ КНФ	Длина ленты 64 бита	№ КНФ	Длина ленты 128 бит
1	1 с	1	2 ч 58 мин
2	4 с	2	4 ч 33 мин
3	3 с	3	1 ч 33 мин
4	1 с	4	5 ч 45 мин
5	2 с	5	4 ч 46 мин
6	5 с	6	49 мин
7	9 с	7	1 ч 30 мин
8	7 с	8	6 ч 56 мин
9	2 с	9	43 мин
10	2 с	10	47 мин
В среднем	4 с	*В среднем*	3 ч 2 мин

В первой серии экспериментов анализировался генератор Вольфрама с лентой, состоящей из 64 бит (табл. 2). Анализируемый фрагмент выходной последовательности имел длину 128 бит. Размерность получаемых КНФ — 8384 переменных, 49 536 дизъюнктов.

Во второй серии экспериментов рассматривался генератор Вольфрама (с той же функцией переходов) с рабочей лентой, состоящей из 128 бит. Анализируемый фрагмент выходной последовательности имел длину 256 бит. Размерность КНФ — 33 408 переменных, 197 888 дизъюнктов.

3.3. Параллельный логический криптоанализ генератора Гиффорда

Очень перспективное направление в логическом криптоанализе — использование суперкомпьютеров. Далее будет рассмотрен простейший подход к реализации логического криптоанализа на вычислительном кластере. Данный подход был применен к логическому криптоанализу генератора Гиффорда.

Генератор Гиффорда (см. [29, 35]) был разработан в 1984 году и довольно долго использовался на практике для передачи текстовой информации. Первая успешная атака на данный генератор описана в статье [5]. Авторы этой работы применили для криптоанализа шифра Гиффорда весьма изощренный математический аппарат. Общая схема генератора Гиффорда представлена на рис. 6.

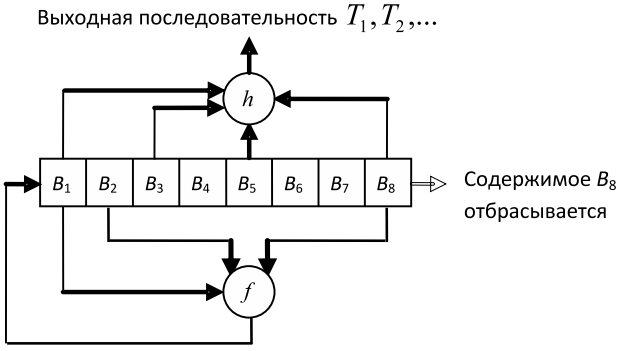

Рис. 6. Схема генератора Гиффорда

Входная последовательность размещается в регистре генератора, образованного восемью ячейками, каждая из которых имеет емкость 1 байт. Таким ообразом, в начальный момент времени $\tau = 0$ в ячейках B_1, B_2,..., B_8 записаны байты $b_1^0, ..., b_8^0$ инициализирующей последовательности (общая длина 64 бита). Ключевой поток представляет собой последовательность байтов $T_1, T_2, ...$, выдаваемых генератором в моменты времени $\tau \in \{1, 2, ...\}$. На каждом шаге новое состояние регистра формируется с помощью сдвига текущего состояния регистра на 1 байт вправо. Содержимое ячейки B_8 при этом отбрасывается, а содержимое ячейки B_1 обновляется при помощи функции обратной связи:

$$b_1^{\tau+1} = f\left(b_1^\tau, b_2^\tau, b_8^\tau\right) = b_1^\tau \oplus \left(>>_1^* \left(b_2^\tau\right)\right) \oplus \left(<<_1 \left(b_8^\tau\right)\right).$$

Операции $>>_1^*$ и $<<_1$ обозначают соответственно операции сдвига вправо на 1 бит с сохранением старшего бита (так называемый sticky right-shift) и сдвига влево на 1 бит с обнулением младшего бита. То есть для $b = (x_1, x_2, ..., x_8)$, $x_i \in \{0, 1\}, i \in \{1, ..., 8\}$, имеем:

$$>>_1^* (b) = (x_1, x_1, x_2, ..., x_7), \quad <<_1 (b) = (x_2, x_3, ..., x_8, 0).$$

Для получения байта ключевого потока применяется фильтрующая нелинейная функция $h : \{0,1\}^{32} \to \{0,1\}^8$. Данная функция получает на входе 4 байта (содержимое ячеек B_1, B_3, B_5, B_8), ее выходом является 1 байт:

$$h\left(B_1, B_3, B_5, B_8\right) = Extract\_Byte_3\left((B_1||B_3) \times (B_5||B_8)\right).$$

Здесь $||$ — конкатенация содержимого соответствующих ячеек, а $\times$ — целочисленное умножение. Таким образом, аргумент функции $Extract\_Byte_3$ понимается как натуральное число. Собственно функция $Extract\_Byte_3\,(x)$ возвращает третий байт слева 32-битного вектора, являющегося двоичным представлением натурального аргумента x.

Далее мы описываем общую схему крупноблочного распараллеливания SAT-задач, используемую в логическом криптоанализе генератора Гиффорда.

Рассматривается произвольная КНФ C над множеством булевых переменных $X = \{x_1, ..., x_n\}$, состоящая из m дизъюнктов. Выбираем в множестве X некоторое подмножество:

$$X^d = \{x_{i_1}, ..., x_{i_d}\}, \quad \{i_1, ..., i_d\} \subseteq \{1, ..., n\}, \quad d \in N.$$

Положим $k = 2^d$. Обозначим через $Y_1, ..., Y_k$ различные двоичные векторы длиной d, каждый из которых является набором значений булевых переменных из множества X^d; соответствующее множество обозначаем через Y^d. Осуществим декомпозицию КНФ C на семейство КНФ $\Delta_d\,(C) = \{C_1, ..., C_k\}$ по следующему правилу: $C_j, j \in \{1, ..., k\}$, есть результат подстановки в C значений переменных $x_{i_v}, v \in \{1, ..., d\}$, из вектора Y_j. В этом случае говорим, что C_j — это КНФ, полученная подстановкой в C вектора Y_j, обозначая данный факт через $C_j = C|_{Y_j}$. Семейство $\Delta_d\,(C)$ назовем семейством, порожденным из C множеством X^d. Дополнительно полагаем, что семейство, порожденное из C множеством X^d при $d = 0$, состоит из единственной КНФ C.

Пусть семейство КНФ $\Delta_d\,(C) = \{C_1, ..., C_k\}$ порождено из КНФ C множеством X^d ($k = 2^d$). Можно показать, что C выполнима тогда и только тогда, когда выполнима хотя бы одна КНФ семейства $\Delta_d\,(C)$. Таким образом, решение исходной SAT-задачи для КНФ C сводится к решению k SAT-задач для КНФ $C_1, ..., C_k$.

Данная схема распараллеливания была применена в логическом криптоанализе генератора Гиффорда. Анализируемый фрагмент выходной последовательности имел длину 160 бит. Размерность получаемых КНФ — 7784 переменных, 39700 дизъюнктов. В

Т а б л и ц а 3. Результаты параллельного логического криптоанализа генератора Гиффорда, длина регистра 64 бита

№ КНФ	Время решения
1	1 ч 3 мин
2	1 ч 31 мин
3	49 мин
4	1 ч 21 мин
5	2 ч 12 мин
6	2 ч 11 мин
7	1 ч 53 мин
8	25 мин
9	2 ч 15 мин
10	2 ч 14 мин
В среднем	1 ч 35 мин

качестве множества X^d при $d = 5$ использовалось множество $X^5 = \{x_1, x_2, x_3, x_4, x_5\}$, образованное переменными, кодирующими первые 5 бит инициализирующей последовательности. В табл. 3 результаты успешного параллельного логического криптоанализа генератора Гиффорда на вычислительном кластере Blackford ИДСТУ СО РАН (см. [33]).

Заключение

Следует особо отметить, что исследованные в настоящей работе задачи логического криптоанализа могут рассматриваться как прототипы тестов для различных методов и подходов к решению комбинаторных задач. В качестве примера приведем известную (см., например, [36]) сводимость SAT-проблемы к проблеме безусловной непрерывной оптимизации. По известной КНФ

$$C = D_1 ... D_m,$$

являющейся конъюнкцией дизъюнктов $D_j,\ j \in \{1, ..., m\}$, над множеством литералов $L = \{x_1, ..., x_n, \overline{x_1}, ..., \overline{x_n}\}$, построим полином от n действительных переменных по следующим правилам:

1) литералу $z \in \{x, \overline{x}\}$ поставим в соответствие выражение

$$f(z) = \begin{cases} (1-x)^2, & z = x, \\ x^2, & z = \overline{x}, \end{cases}$$

где x — действительная переменная;

2) произвольному дизъюнкту $D_j,\ j \in \{1, ..., m\}$, поставим в соответствие выражение

$$F(D_j) = \prod\nolimits_{z \in D_j} f(z);$$

3) КНФ C сопоставляем полином $P(C) = \sum_{j=1}^{m} F(D_j)$ от n действительных переменных $x_1, ..., x_n$.

Несложно понять, что КНФ C выполнима тогда и только тогда, когда глобальный минимум полинома $P(C)$ на R^n равен 0. От данной задачи легко перейти к задаче

глобального поиска на единичном кубе в R^n, если ввести ограничения $x_i \in [0,1]$, $i \in \{1,...,n\}$. В этом случае вместо сопоставлений $x \to (1-x)^2$ и $\overline{x} \to x^2$ можно использовать соответственно $x \to (1-x)$ и $\overline{x} \to x$. При помощи консервативной редукции задачи ВЫПОЛНИМОСТЬ к задаче 3-ВЫПОЛНИМОСТЬ можно получить в итоге задачу глобального поиска на $[0,1]^n$ для полинома третьей степени. Следовательно, всякая задача обращения дискретных функций из рассматриваемого в работе класса может быть переформулирована в задачу глобального поиска для полинома третьей степени на единичном евклидовом кубе.

Известно множество подобных редукций к различным комбинаторным задачам (задачи на графах, задачи целочисленного программирования и др.). Таким образом, описанные в работе технологии сводимости задач криптоанализа к SAT-задачам в конечном счете позволяют строить аргументированно сложные тесты для разнообразных алгоритмов комбинаторной оптимизации.

Список литературы

[1] Рябко Б.Я., Фионов А.Н. Криптографические методы защиты информации. М.: Горячая линия — Телеком, 2005.

[2] Massacci F., Marraro L. Logical Cryptoanalysis as a SAT Problem: the Encoding of the Data Encryption Standard. Dipartimento di Imformatica e Sistemistica, Universita di Roma "La Sapienza", 1999. Preprint.

[3] Семенов А.А., Буранов Е.В. Погружение задачи криптоанализа симметричных шифров в пропозициональную логику // Матер. конф. "Вычислительные и информационные технологии в науке, технике и образовании". Новосибирск; Алматы; Усть-Каменогорск, 2003. Ч. 3. С. 118–126.

[4] Семенов А.А. О сложности обращения дискретных функций из одного класса // Дискретный анализ и исследование операций. 2004. Сер. 1. Т. 11, № 4. С. 44–55.

[5] Cain T.R., Sherman A.T. How to break Gifford's Cipher // Cryptologia. 1997. Vol. 21, N 3. P. 237–286.

[6] Яблонский С.В. Введение в дискретную математику. М.: Наука, 1986. 384 с.

[7] Кук С.А. Сложность процедур вывода теорем // Кибернетический сборник. 1975. Новая серия. Вып. 12. С. 5–15.

[8] Катленд Н. Вычислимость. Введение в теорию рекурсивных функций. М.: Мир, 1983. 256 с.

[9] Гэри М., Джонсон Д. Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982. 416 с.

[10] Схрейвер А.А. Теория линейного и целочисленного программирования. М.: Мир, 1991. Т. 1. 360 с.

[11] Simon J. On some central problems in computational complexity: PhD thesis. Cornell Univ., Ithaca, N.Y., 1975.

[12] Семенов А.А. Логико-эвристический подход в криптоанализе генераторов двоичных последовательностей// Тр. Междунар. науч. конф. ПАВТ'07. Челябинск, ЮУрГУ, 2007. Т. 1. С. 170–180.

[13] Нигматуллин Р.Г. Сложность булевых функций. М.: Наука, 1991. 240 с.

[14] Menezes A., Van Oorschot P., Vanstone S. Handbook of Applied Cryptography. CRC Press, 1996. 657 с.

[15] Агибалов Г.П. Логические уравнения в криптоанализе генераторов ключевого потока // Вестн. Томского гос. ун-та. Прилож. 2003. № 6. С. 31–41.

[16] Лидл Р., Нидеррайтер Г. Конечные поля: в 2 т. М.: Мир, 1988. Т. 2. 822 с.

[17] Буранов Е.В. Программная трансляция процедур логического криптоанализа симметричных шифров // Вестн. Томского гос. ун-та. Прилож. 2004. № 9 (1). С. 60–65.

[18] SATLive [http://www.satlive.org]

[19] Davis M., Longemann G., Loveland D. A machine program for theorem proving // Communicat. of the ACM. 1962. Vol. 5. P. 394–397.

[20] Devis M., Putnam H. A computing procedure for quantification theory // ACM. 1960. Vol. 7. P. 201–215.

[21] Marques-Silva J.P., Sakallah K.A. GRASP: A search algorithm for propositional satisfiability // IEEE Transactions on Computers. 1999. Vol. 48, N 5. P. 506–521.

[22] Zhang L., Madigan C., Moskewicz M., Malik S. Efficient conflict driven learning in a Boolean satisfiability solver // Proc. Intern. Conf. on Computer Aided Design (ICCAD), San Jose, California. 2001. P. 279–285.

[23] Moskewicz M., Madigan C., Zhao Y. et al. Chaff: Engineering an Efficient SAT Solver // Proc. Design Automation Conf. (DAC). 2001. P. 530–535.

[24] Beame P., Kautz H., Sabharwal A. Understanding the power of clause learning // Proc. of 18^{th} Intern. Joint Conf. on Artificial Intelligence (IJCAI). 2003. P. 1194–1201.

[25] Lynce I., Marques-Silva J. Efficient data structures for backtrack search SAT solvers // Annals of Mathematics and Artificial Intelligence. 2005. Vol. 43. P. 137–152.

[26] MiniSat [http://minisat.se/MiniSat.html]

[27] Zchaff [http://www.princeton.edu/ chaff/zchaff.html]

[28] Goldberg E., Novikov Y. BerkMin: A Fast and Robust SAT Solver // Automation and Test in Europe (DATE). 2002. P. 142–149.

[29] Поточные шифры. Результаты зарубежной открытой криптологии. М., 1997. 389 с.

[30] Ушаков А.А. Логический криптоанализ генераторов двоичных последовательностей // Вестн. Томского гос. ун-та. Прилож. 2005. № 14. С. 92–95.

[31] Буранов Е.В. Оптимизация редукций в логическом криптоанализе // Вестн. Томского гос. ун-та. Прилож. 2005. № 14. С. 50–53.

[32] Espresso [http://www.csc.uvic.ca/∼csc485c/espresso/espresso.exe]

[33] Суперкомпьютерный центр ИДСТУ СО РАН [http://www.mvs.icc.ru]

[34] Wolfram S. Cryptography with cellular automata // Lecture Notes in Computer Science 1986. Vol. 218. P. 429–432.

[35] Gifford D.K., Lucassen J.M., Berlin S.T. The application of digital broadcast communication to large scale information systems // IEEE J. on Selected Areas in Communications. 1985. Vol. 3, N 3. P. 457–467.

[36] Gu J., Purdom P., Franco J., Wah B.W. Algorithms for the satisfiability (SAT) problem: A Survey // DIMACS Series in Discrete Mathematics and Theoretical Computer Science. 1997. Vol. 35. P. 19–152.

Поступила в редакцию 21 марта 2008 г.,
в переработанном виде — 30 мая 2008 г.