Решение задач криптоанализа поточных шифров в распределенных вычислительных средах *

М. А. Посыпкин[1], О. С. Заикин[2],
Д. В. Беспалов[2], А. А. Семенов[2]

[1] *Центр Грид-технологий и распределенных вычислений Института системного анализа РАН*
[2] *Институт динамики систем и теории управления Сибирского отделения РАН*

В статье приведены результаты криптоанализа ряда систем поточного шифрования в распределенных вычислительных средах (РВС). Работа представляет собой практическое развитие идей логического криптоанализа — задачи криптоанализа ставятся и решаются как SAT-задачи. Основной акцент сделан на задаче криптоанализа генератора ключевого потока шифра A5/1, которая решалась в распределенной вычислительной среде, состоящей из четырех высокопроизводительных кластеров.

Введение

Задача поиска набора, выполняющего произвольную конъюнктивную нормальную форму (КНФ), либо констатация факта отсутствия такового набора известна как SAT-задача. Несмотря на то, что SAT-задача в такой общей постановке NP-трудна, многочисленные ее частные случаи, возникающие в практических приложениях, допускают весьма эффективные процедуры решения посредством разнообразных эвристических алгоритмов. Разработка специализированных программных решателей SAT-задач выделилась за последние 10 лет в самостоятельное активно развивающееся направление. Среди создаваемых SAT-решателей регулярно проводятся конкурсы в Internet (см. [1]). Основную массу проблем, в отношении кото-

* Работа выполнена при поддержке фонда РФФИ (№ 07–01–00400–а), Аналитической ведомственной программы «Развитие научного потенциала высшей школы» и Совета по грантам Президента Российской Федерации (№ НШ–1676.2008.1 и НШ–5511.2008.9).

рых оправдан SAT-подход, составляют задачи верификации дискретных автоматов и моделей программ (model checking).

В настоящей работе SAT-подход используется для решения задач криптоанализа некоторых систем поточного шифрования в распределенных вычислительных средах (РВС). Применение SAT-подхода к задачам криптоанализа известно как логический криптоанализ. Первой публикацией по данному направлению следует считать, по-видимому, работу [2]. Авторы статьи [3] независимо обосновали базовые концепции логического криптоанализа. Основу логического криптоанализа составляют процедуры трансляции алгоритмов шифрования в логические выражения, представленные в КНФ, а также алгоритмы и технологии решения получаемых SAT-задач (задачи поиска секретного ключа ставятся как SAT-задачи). В работе [2] идеология логического криптоанализа была применена к широко известному алгоритму блочного шифрования DES (см. [4]). Однако получившиеся КНФ оказались сложными для всех известных на тот момент SAT-решателей. В работах [5–7] было предложено использовать SAT-подход в решении задач криптоанализа генераторов ключевого потока, применяемых в поточных системах шифрования. В качестве аргумента перспективности таких исследований приводились соображения о традиционно меньшей стойкости поточных шифров в сравнении с блочными (при одинаковых длинах ключей), что является следствием существенного превосходства поточного шифрования перед блочным по скорости. В статье [7] приведены результаты успешного логического криптоанализа ряда генераторов ключевого потока, реализованного на обычном ПК. Для некоторых генераторов «последовательный» логический криптоанализ не дал приемлемых результатов. В связи с этим в работах [8–10] была развита параллельная технология решения SAT-задач, базирующаяся на идеологии крупноблочного распараллеливания «по данным». Данная технология была успешно апробирована на вычислительных кластерах малой мощности при решении задач криптоанализа некоторых генераторов (суммирующий, пороговый, генератор Гиффорда). Однако для решения задачи криптоанализа одного из наиболее интересных с практической точки зрения генераторов поточного шифрования и генератора A5/1 (см. [11]) вычислительных мощностей обычных кластеров оказалось недостаточно. Основным результатом настоящей статьи является полный криптоанализ генератора ключевого потока A5/1, осуществленный в распределенной среде. Особый акцент мы делаем на том факте, что при решении данной задачи, в отличие от некоторых недавно анонсированных исследований, не использовались специальные вычислители на ПЛИС и дополнительные накопители данных. Также подчеркнем, что главная мотивация предпринятых исследований состояла не столько в желании осуществить криптоанализ реально используемой системы шифрования, сколько в желании аргументиро-

вать эффективность предлагаемой методики решения SAT-задач в распределенных вычислительных средах.

Приведем краткий план статьи. В первом разделе в общих чертах описывается техника сведения задач криптоанализа генераторов ключевого потока к SAT-задачам. Здесь же описаны основные особенности решения SAT-задач в распределенной вычислительной среде. Во втором — приведено описание модификации неоднократного победителя специализированных конкурсов — SAT-решателя minisat (см. [12]) и его модифицированного варианта — решателя dminisat, ориентированного на применение в среде параллельных и распределенных вычислений. В третьем разделе приведено детальное описание процесса решения задачи криптоанализа генератора ключевого потока A5/1, сведенной к задаче SAT, в распределенной вычислительной среде в рамках программного комплекса BNB-Grid.

1. Постановки задач криптоанализа в форме SAT-задач и технологии их решения в РВС

1.1. Сведение задач криптоанализа генераторов ключевого потока к SAT-задачам

Пусть $X = \{x_1,...,x_n\}$ — множество, образованное булевыми переменными и пусть $L(x_1,...,x_n)$ — произвольная формула исчисления высказываний, содержащая переменные из множества X. Подстановка в $L(x_1,...,x_n)$ произвольного вектора значений переменных $(\alpha_1,...,\alpha_n)$, $\alpha_i \in \{0,1\}$, $i \in \{1,...,n\}$, дает в результате либо 0, либо 1. Данный факт будем записывать как $L(\alpha_1,...,\alpha_n) = \beta, \beta \in \{0,1\}$. Выражения вида

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

называются логическими уравнениями. Решением логического уравнения $L(x_1,...,x_n) = \beta$, $\beta \in \{0,1\}$, называется такой вектор $(\alpha_1,...,\alpha_n)$, $\alpha_i \in \{0,1\}$, $i \in \{1,...,n\}$, что $L(\alpha_1,...,\alpha_n) = \beta$. Важнейший класс логических уравнений образован уравнениями вида

$$C(x_1,...,x_n) = 1, \qquad (1)$$

где $C(x_1,...,x_n)$ — конъюнктивная нормальная форма (КНФ) над X. Если уравнение (1) имеет решения, то КНФ $C(x_1,...,x_n)$ называется выполни-

мой, а решения (1) называются выполняющими данную КНФ наборами. В противном случае $C(x_1,...,x_n)$ называется невыполнимой. К SAT-задачам относятся задачи определения выполнимости произвольных КНФ и поиска наборов, выполняющих выполнимые КНФ.

Обозначим через $f=\{f_n\}_{n\in\mathrm{N}}$ натуральное семейство дискретных функций вида

$$f_n:\{0,1\}^n\to\{0,1\}^*,$$

определенных всюду на $\{0,1\}^n$ ($dom\ f_n=\{0,1\}^n$) и алгоритмически вычислимых за полиномиальное от n время. Проблемой обращения произвольной функции f_n из такого семейства называется следующая задача: по произвольному $y\in range\ f_n\subset\{0,1\}^*$ и известному алгоритму вычисления f (программе для выбранной вычислительной модели) требуется найти такой $x\in\{0,1\}^n$, что $f_n(x)=y$. Данную проблему будем называть проблемой обращения функции f_n в точке $y\in range\ f_n$.

Перенося известную идею С. А. Кука о пропозициональном кодировании алгоритмов (см. [13]) на проблему обращения функций из рассматриваемого класса, можно показать справедливость следующего факта.

Теорема 1. *Для любого семейства f дискретных функций из определенного выше класса существует алгоритм с полиномиально от n ограниченной сложностью, который, получая на входе n и $y\in range\ f_n$, преобразует проблему обращения f_n в точке y в проблему поиска решений логического уравнения вида $C(x_1,...,x_{q(n)})=1$, где $q(\cdot)$ — некоторый полином, а $C(x_1,...,x_{q(n)})$ — выполнимая КНФ над множеством булевых переменных $\{x_1,...,x_{q(n)}\}$.*

Детальное доказательство данного факта в контексте бинарных машин с неограниченными регистрами (МНР) приведено в [14]. Там же подробно описана концепция транслятора алгоритмов вычисления дискретных функций из рассматриваемого класса в SAT-задачи. Программная реализация данного транслятора дала возможность рассматривать задачи криптоанализа некоторых систем шифрования в форме SAT-задач. Наиболее успешным этот подход оказался в отношении генераторов ключевого потока, используемых в поточном шифровании.

Определение. *Генератором ключевого потока* (*генератором*) *называется такое натуральное семейство функций вида* $f_{k,m}:\{0,1\}^k \to \to \{0,1\}^*, m \in \mathrm{N}$, *определенных всюду на* $\{0,1\}^k$, *что для произвольного* $x \in \{0,1\}^k$ *и произвольного* $m \in \mathrm{N}$, *существует* $y \in \{0,1\}^m \cap \cap range\, f_{k,m}(x)$, *и данное значение может быть вычислено детерминированным алгоритмом* $A(f_{k,m})$ *с полиномиальной от* m *трудоемкостью. При этом вектор* x *называется инициализирующей последовательностью длины* k, *а вектор* y — *фрагментом ключевого потока длины* m. *Задача криптоанализа генератора заключается в восстановлении по некоторому вектору* $y \in range\, f_{k,m}$ (*фрагменту ключевого потока*) *и известному алгоритму* $A(f_{k,m})$ *инициализирующей последовательности* $x \in \{0,1\}^k : f_{k,m}(x) = y$.

Обратим внимание на тот факт, что генераторы, строго говоря, не относятся к классу дискретных функций, определенному выше: параметр k — длина инициализирующей последовательности является константой. Однако идея трансляции алгоритмов вычисления функций в КНФ к генераторам полностью применима.

1.2. Технология логического криптоанализа в РВС

Описываемая далее технология впервые была представлена в [8]. Основана она на следующем простом факте. Пусть $C(x_1,...,x_{q(n)})$ — КНФ, кодирующая (в смысле теоремы 1) задачу обращения функции f_n из определенного выше класса в точке $y \in range\, f_n$. Переменные из множества $X = \{x_1,...,x_n\}$ называем переменными входа функции f_n (если функцию f_n реализовать схемой из функциональных элементов в произвольном полном базисе, то входные полюса данной схемы будут помечены переменными из множества X). Все переменные в $\{x_1,...,x_{q(n)}\} \setminus X$ функционально зависят от переменных из X, и поэтому любая подстановка значений всех переменных из X индуцирует вывод остальных переменных из $C(x_1,...,x_{q(n)})$ по правилу единичного дизъюнкта (данный факт строго доказан в [15]). Данный факт дает основу для построения декомпозиции исходной SAT-задачи в отношении КНФ $C(x_1,...,x_{q(n)})$ на семейство, со-

стоящее из 2^d SAT-задач ($d : 1 \le d \le n$) для КНФ, построенных следующим образом: среди переменных входа выбираются некоторые d переменных $x_{i_1},....,x_{i_d}$, после чего рассматривается семейство, образованное КНФ вида

$$C(x_1,...,x_{q(n)})|_{x_{i_1}=\alpha_1,...,x_{i_d}=\alpha_{i_d}},$$

причем вектор $\alpha = \left(\alpha_{i_1},...,\alpha_{i_d}\right)$ пробегает все множество $\{0,1\}^d$. Полученное семейство КНФ обозначается через Δ_d и называется декомпозиционным семейством, а множество $\left\{x_{i_1},....,x_{i_d}\right\}$-декомпозиционным множеством. Для решения SAT-задач в отношении КНФ из декомпозиционного семейства можно использовать РВС.

Вопрос выбора декомпозиционного множества далеко не всегда является тривиальным: если величина d будет большой, то в декомпозиционном семействе окажутся простые КНФ, однако их количество будет слишком велико. При малых значениях параметра d КНФ в декомпозиционном семействе, как правило, слишком сложны для отдельных вычислительных узлов. Для решения проблемы наилучшего (с точки зрения общей трудоемкости вычислительного процесса) выбора значения параметра d в [8] было предложено использовать специальные прогнозные функции. Прогнозные функции позволяют делать оценочные предположения относительно общей трудоемкости процесса решения SAT-задач в РВС на основе обработки случайных выборок КНФ из декомпозиционных семейств, построенных для различных значений параметра d.

Предложенная в [8] прогнозная функция имеет следующий вид:

$$T\left(\Theta_d\right) = \begin{cases} \dfrac{2^d}{q_d} \cdot \tau_S\left(\Theta_d\right), 2^d > R_0, \tau_S\left(\Theta_d\right) < g(C); \\ \tau_S\left(\Theta_d\right), 2^d \le R_0, \tau_S\left(\Theta_d\right) < g(C); \\ \infty, \tau_S\left(\Theta_d\right) \ge g(C). \end{cases}$$

Здесь через Θ_d обозначена случайная выборка, состоящая из q_d КНФ декомпозиционного семейства Δ_d. Через $\tau_S\left(\Theta_d\right)$ обозначено суммарное время работы фиксированного SAT-решателя S на всех КНФ выборки Θ_d. Число R_0 — некоторая константа, разделяющая ситуации: когда требуется формировать случайную выборку, а когда нет (данная величина определяется числом вычислительных узлов распределенной вычисли-

тельной системы). Функция $g(C)$ — некоторый полином от длины кодировки исходной КНФ $C(x_1,...,x_{q(n)})$, определяющий границы «разумного времени» проведения процедуры прогнозирования. Глобальный минимум функции $T(\cdot)$ на множестве выборок, получаемом в результате изменения параметра d на множестве $\{1,...,n\}$, можно рассматривать как прогноз наилучших (с позиции трудоемкости решения) параметров декомпозиции исходной SAT-задачи.

В [8] приведены результаты успешного криптоанализа (с использованием техники прогнозных функций) на вычислительном кластере Blackford (см. [16]) суммирующего и порогового генераторов, а также генератора Гиффорда.

1.3. Декомпозиция SAT-задачи, кодирующей криптоанализ генератора A5/1

Данный генератор интересен, прежде всего, тем, что он используется в действующем протоколе шифрования трафика в сетях сотовой телефонии стандарта GSM. В литературе описано довольно много атак на данный генератор, однако нам не удалось обнаружить убедительные результаты его криптоанализа в форме программной реализации и серий тестов. Относительно недавно был анонсирован криптоанализ A5/1 с использованием специальных вычислителей на ПЛИС, однако и в этом случае публикаций, в которых были бы приведены результаты успешных численных экспериментов (в форме серий тестов), на текущий момент нет. В описании генератора A5/1 мы следуем статье [11]. В соответствии с этим описанием в генераторе A5/1 используются три регистра сдвига с линейной об-

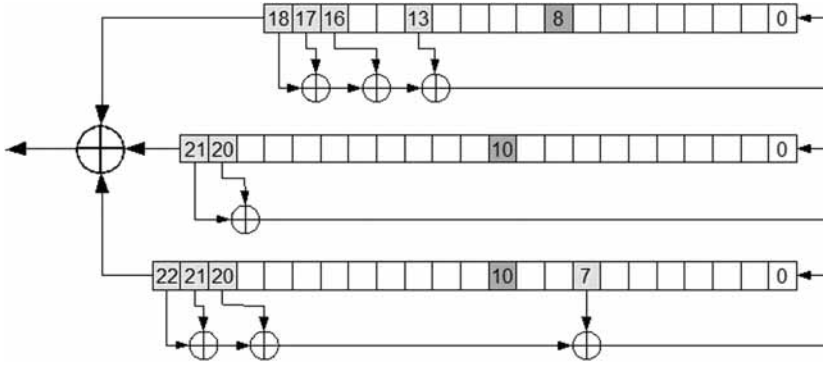

Рис. 1. Схема работы генератора ключевого потока шифра A5/1

ратной связью (РСЛОС, LFSR, см. [17]), задаваемые следующими полиномами обратной связи: РСЛОС 1: $X^{19}+X^{18}+X^{17}+X^{14}+1$; РСЛОС 2: $X^{22}+X^{21}+1$; РСЛОС 3: $X^{23}+X^{22}+X^{21}+X^{8}+1$.

В каждом такте могут сдвигаться не все регистры. Сдвиг регистра с номером m, $m\in\{1,2,3\}$, происходит, если значение функции $\chi_m\left(b_s^1,b_s^2,b_s^3\right)$ равно 1, и не происходит, если значение данной функции равно 0. Через b_s^1, b_s^2, b_s^3 здесь обозначены значения так называемых «серединных битов» текущего шага, то есть битов, находящихся в данный момент в девятой ячейке первого РСЛОС, и в ячейках с номером «11» РСЛОС2 и РСЛОС3 (данные ячейки на рис. 1 отмечены более темным цветом, нумерация ячеек ведется справа налево). Функция $\chi_m(\cdot)$ определяется следующим образом:

$$\chi_m\left(b_s^1,b_s^2,b_s^3\right)=\begin{cases}1, b_s^m=\text{majority}\left(b_s^1,b_s^2,b_s^3\right),\\ 0, b_s^m\neq\text{majority}\left(b_s^1,b_s^2,b_s^3\right),\end{cases}$$

где $\text{majority}(x,y,z)=x\cdot y\vee x\cdot z\vee y\cdot z$ (функция большинства).

Проблема построения оптимальной декомпозиции для решения задачи криптоанализа A5/1 в форме SAT-задачи оказалась нетривиальной. Были исследованы различные стратегии формирования декомпозиционных множеств. Первые схемы основывались на идеях, ранее успешно примененных в параллельном криптоанализе ряда генераторов (пороговый, суммирующий, Гиффорда, см. [8]). При этом, исходя из некоторых «разумных» предположений, сначала строилось базовое декомпозиционное множество, а затем предпринимались попытки его усечения на основе значений соответствующих прогнозных функций.

На рис. 2 представлена схема построения декомпозиционного множества X', состоящего из 31 переменной.

В соответствии с данной схемой предлагается включить в X' переменные, кодирующие начальные состояния ячеек регистров, начиная с первых ячеек, до ячеек, содержащих серединные биты (соответствующие ячейки на рисунке — темные). Иными словами, имеем декомпозиционное множество

$$X'=\{x_1,...,x_9,x_{20},...,x_{30},x_{42},...,x_{52}\}. \quad (2)$$

Данный выбор продиктован следующими естественными соображениями. Произвольная фиксация битов из построенного таким образом множества X' индуцирует точные значения серединных битов для значительного числа последующих состояний всех трех регистров. Но серединные биты являются наиболее информативными битами, поскольку именно они

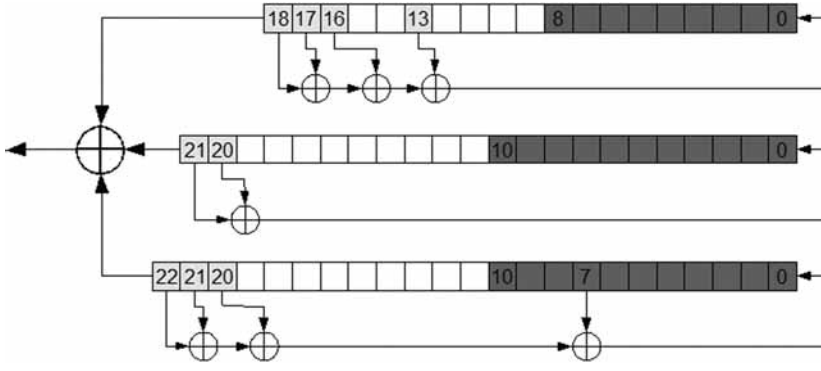

Рис. 2. Схема построения декомпозиционного множества из 31 переменной (X')

определяют, какая ветвь соответствующего условия (значение функции большинства) имеет место.

Довольно неожиданным оказался тот факт, что построенное в соответствии с (2) множество X' не удалось уменьшить за счет технологии прогнозных функций. В проводимых статистических экспериментах для каждого варианта декомпозиционного множества и начального фрагмента ключевого потока некоторой фиксированной длины (от 128 до 256 бит) строились случайные выборки объемом 1000 КНФ. Для каждой такой выборки считалось значение прогнозной функции — среднее значение времени решения SAT-задач по выборке, умноженное на мощность пространства перебора. Прогнозы по схемам формирования, отличным от схемы (2), оказались хуже. Во всех экспериментах использовался SAT-решатель dminisat, представляющий собой модификацию известного решателя minisat. Подробное описание dminisat приведено ниже.

Итак, на основании большого числа экспериментов по прогнозированию параметров наилучшей в смысле общей трудоемкости декомпозиции было принято решение остановиться на декомпозиции, определяемой декомпозиционным множеством вида (2). Тем самым возникла проблема организации в РВС вычислительной процедуры, в ходе которой требуется решить (в худшем случае) 2^{31} SAT-задач.

2. Модификация SAT-решателя с ориентацией на задачи криптоанализа в РВС

Во всех вычислительных экспериментах по логическому криптоанализу в РВС генератора A5/1 использовался решатель dminisat. Данный

решатель является модификацией SAT-решателя minisat (см. [12]) и ориентирован на решение SAT-задач из декомпозиционного семейства, полученного в результате применения к КНФ, кодирующей работу A5/1, декомпозиции, определяемой множеством (2).

Отметим, что модификация minisat была необходима, поскольку в оригинальных версиях (как minisat1.14.1, так и minisat 2.0) данный решатель на получаемых в результате декомпозиции формулах не находил решение за разумное время (вычисления прерывались после 10 минут работы).

Модифицировалась версия minisat-Cv1.14.1 (см. [12]). Основной этап модификации заключался в незначительном изменении процедуры выбора уровней решения (см. [18]), реализованной в minisat. А именно, была добавлена процедура присвоения начальной активности (отличной от нулевой) для 64 переменных, кодирующих исходное заполнение РСЛОС 1–3 (искомый ключ). Данный факт позволил на начальной стадии процесса решения выделить 64 переменные, соответствующие инициализирующей последовательности, в качестве приоритетных при выборе очередной переменной уровня решения. Это простое изменение привело к значительному росту производительности SAT-решателя на рассматриваемых КНФ.

Кроме того, были изменены некоторые базовые константы решателя. Подобно большинству известных SAT-решателей, minisat периодически понижает активность всех переменных и дизъюнктов с целью увеличения приоритета в угадывании для переменных, повышающих свою активность на более поздних шагах поиска. Кроме этого, в 2 % случаев minisat присваивает значение переменной, выбранной случайным образом, а не переменной, обладающей максимальной активностью. Эти эвристики в среднем показывают хорошие результаты на широком наборе тестовых примеров, используемых в конкурсах SAT-решателей. Однако для КНФ, кодирующих шифр A5/1, они оказываются неэффективными. Запрещение периодического понижения активности и случайного выбора переменных дает увеличение скорости работы модифицированного в описанном выше смысле решателя еще на 20–30 %.

Еще один аспект модификации SAT-решателя обусловлен необходимостью решения в РВС большого числа однотипных задач (SAT-задач из декомпозиционного семейства). Передача по сети отдельных SAT-задач из декомпозиционного семейства снижает общую эффективность вычислительного процесса в РВС. Для решения данной проблемы была организована передача на вычислительные узлы РВС «пакетов заданий». Если для простоты предположить, что РВС состоит из 2^k узлов ($k < 31$), то на каждом узле можно организовать решение 2^{31-k} SAT-задач для КНФ из декомпозиционного семейства. Тем самым пакет заданий для каждого узла может быть «описан» двоичным вектором длины k (всего имеем 2^k различ-

Таблица 1

Сравнительные характеристики процессоров Intel E8400 и Intel E5472

Название процессора	Intel E8400	Intel E5472
Число ядер	2	4
Частота ядра	3,0 ГГц	3,0 ГГц
Частота шины	1333 МГц	1600 МГц
Технология производства	45 нанометров	45 нанометров
Кэш L2	6 Мб	12 Мб

ных пакетов заданий). Получая конкретный двоичный вектор α длины k, решатель на произвольном узле самостоятельно порождает 2^{31-k} векторов длины 31. Первые k компонент этих векторов образуют вектор α, оставшиеся компоненты изменяются произвольным образом. Описанная процедура генерации пакетов заданий на рабочих узлах PBC была встроена в dminisat.

Далее мы приводим прогноз времени решения задачи криптоанализа генератора A5/1 в рамках описанной технологии (с использованием решателя dminisat) на вычислительном кластере СКИФ-МГУ «Чебышев» [19]. Данный кластер состоит из 1250 четырехъядерных процессоров E5472. Все численные эксперименты по прогнозированию проводились на процессоре Intel E8400. В табл. 1 приведены сравнительные характеристики данного процессора и процессора Intel E5472.

Из таблицы видно, что мощность одного ядра процессора Intel E8400 сопоставима с мощностью одного ядра процессора Intel E5472 (незначительное отличие лишь в частоте шины). Исходя из этого факта, мы сочли возможным напрямую экстраполировать результаты проведенных численных экспериментов применительно к процессу решения задачи логического криптоанализа генератора A5/1 на кластере СКИФ-МГУ «Чебышев». Итоговый прогноз времени логического криптоанализа A5/1 на данном кластере составил 19–21 час.

3. Решение задачи криптоанализа генератора A5/1 в распределенной вычислительной среде

3.1. Необходимость и общая схема применения технологий распределенных вычислений для задачи криптоанализа генератора A5/1

Прогноз времени решения данной задачи на суперкомпьютере СКИФ-МГУ показывает, что даже при полной загрузке этого кластера, решение задачи криптоанализа требует значительного времени. Монопольное

использование многопроцессорных вычислительных комплексов, находящихся в коллективном доступе, невозможно. Вместе с тем, в распоряжении исследователей зачастую имеются ресурсы различных кластеров, грид-систем, высокопроизводительных серверов. Программный комплекс BNB-Grid [20] позволяет эффективно задействовать подобные разнородные распределенные вычислительные ресурсы для решения сложных вычислительных задач. Примером использования BNB-Grid может служить решение задачи поиска энергетически-оптимальной конформации атомного кластера [21].

Структура приведенного выше вычислительного алгоритма для решения задачи криптоанализа шифра A5/1 допускает эффективную реализацию для параллельных и распределенных систем, так как подзадачи из декомпозиционного семейства могут обрабатываться независимым образом. Поэтому, учитывая высокую вычислительную сложность решения рассматриваемой задачи, было принято решение применить систему BNB-Grid для решения данной задачи, задействовав вычислительные ресурсы нескольких многопроцессорных вычислительных комплексов.

3.2. Вычислительный комплекс BNB-Grid

Программный комплекс BNB-Grid предназначен для решения задач оптимизации на распределенных системах, состоящих из рабочих станций, небольших многопроцессорных комплексов, доступных в монопольном режиме, суперкомпьютеров коллективного доступа, сегментов грид. BNB-Grid запускает и организует взаимодействие параллельных и последовательных приложений, решающих задачу с помощью библиотеки BNB-Solver [22] на вычислительных узлах. В результате формируется иерархическая распределенная система: на верхнем уровне части работы распределяются между параллельными приложениями, а далее они распределяются по процессорам средствами библиотеки BNB-Solver.

Ядро системы реализовано на языке программирования Java с использованием промежуточного программного обеспечения Internet Communication Engine (ICE)[23] — аналога CORBA. Каждый вычислительный узел представлен в системе распределенным экземпляром объекта типа CE-Manager — Computing Element Manager (рис. 3). Этот объект предоставляет интерфейс для запуска и остановки приложений на вычислительном узле. Координация работы объектов типа CE-Manager осуществляется с помощью другого распределенного объекта типа CS-Manager — Computing Space Manager. Графический интерфейс пользователя также реализован как ICE-объект, обозначенный на рисунке GUI Manager.

ICE-объекты размещаются либо на одном, либо на нескольких компьютерах в пределах локальной сети. Для обеспечения длительной автономной работы этих компонентов в фоновом режиме на сервере применя-

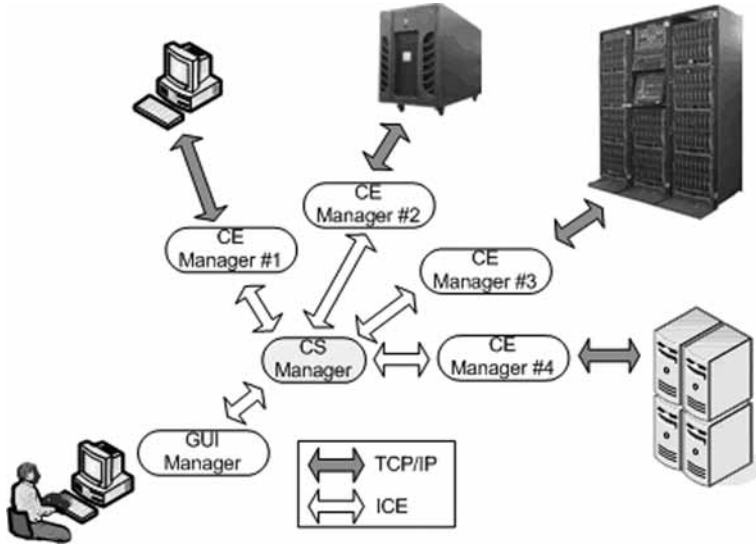

Рис. 3. Организация вычислений в BNB-Grid

ется технология ICE-Grid, автоматизирующая запуск сервисов ICE и унифицирующая доступ к удаленным объектам.

Доступ к удаленному вычислительному узлу осуществляется по схеме, показанной на рис. 4. Объект CE-Manager устанавливает SSH-соединение с внешним управляющим модулем узла, запускает на нем процесс CE-Server и устанавливает с ним TCP/IP соединение. Если удаленный узел защищен сетевыми экранами, то применяется механизм туннелирования сетевого соединения через SSH-канал. Процесс CE-Server информирует CE-Manager о состоянии вычислительного узла. При обрыве соединения или перезагрузке узла CE-Server перезапускается. При получении запроса на запуск параллельного приложения CE-Manager создает объект APP-Manager, CE-Server создает процесс APP-Proxy, который запускает приложение BNB-Solver на узле. При этом CS-Manager взаимодействует с APP-Manager средствами ICE, а APP-Manager, в свою очередь, устанавливает TCP/IP соединение с BNB-Solver через APP-Proxy. Процесс APP-Proxy необходим, так как на суперкомпьютерах общего доступа непосредственный доступ из внешней сети к вычислительным модулям, как правило, невозможен.

На одном вычислительном узле может быть запущено несколько экземпляров приложения BNB-Solver. Такой подход часто оказывается целесообразным для суперкомпьютеров коллективного доступа, работающих под управлением систем пакетной обработки. На таких системах приложение, запросившее достаточно большое число процессоров может быть

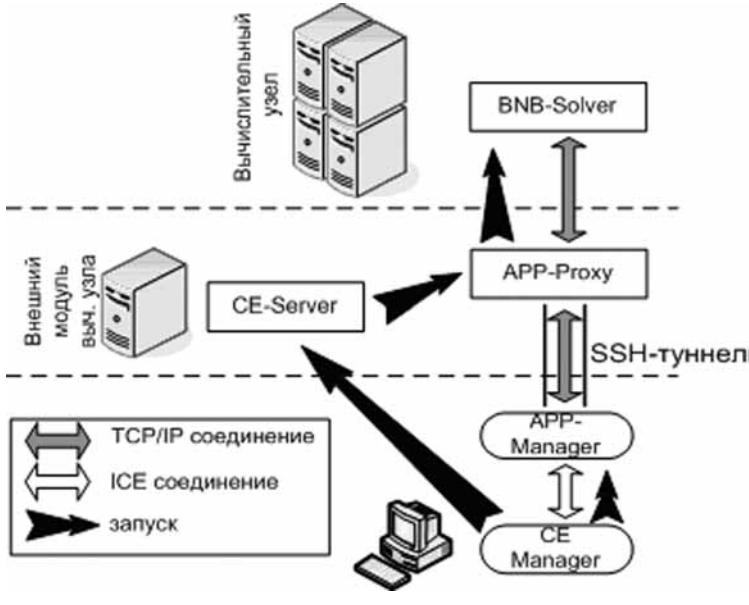

Рис. 4. Запуск приложений на удаленном вычислительном узле и установление соединения

надолго помещено в очередь в ожидании соответствующего «окна» в расписании заданий. В то же время, приложение, запросившее существенно меньшее число процессоров, может быть запущено ранее. Этот эффект объясняется тем, что системе пакетной обработки проще эффективно размещать небольшие задания.

Управляющие модули BNB-Grid и запущенных экземпляров библиотеки BNB-Solver образуют иерархическую систему управления распределением вычислительной нагрузки. На верхнем уровне части работы распределяются ICE-объектом CS-Manager между экземплярами приложений BNB-Solver, а на нижнем — управляющим процессом (Master Process) приложения BNB-Solver между его рабочими процессами (Worker Process). Понятие части работы зависит от применяемого алгоритма: для методов ветвей и границ частями будут подмножества пространства допустимых решений, для эвристических алгоритмов — допустимые решения.

Программный комплекс BNB-Grid предназначен для решения ресурсоемких задач, требующих длительного времени вычислений. На практике проводились расчеты в течение нескольких суток. Из-за принудительного завершения приложения системой пакетной обработки, сетевого или аппаратного сбоя, экземпляр BNB-Solver может не закончить обработку вы-

деленной ему части работы. Поэтому была разработана система, обеспечивающая устойчивость к аппаратным сбоям вычислительных и управляющего узлов. Порция работы, передаваемая экземпляру параллельного приложения, сохраняется на CS-Manager. В случае аварийного завершения любого из них, соответствующая резервная копия возвращается в общий пул. Общий пул подмножеств и допустимых решений периодически сохраняется на диске управляющим модулем BNB-Grid. В случае аварийного сбоя управляющего модуля вычисления могут быть возобновлены с сохраненного состояния.

3.3. Реализация декомпозиционного алгоритма решения задачи SAT в BNB-Grid

В библиотеку BNB-Solver был добавлен модуль, решающий набор подзадач SAT на параллельной вычислительной системе с распределенной памятью. Этот модуль работает в соответствии с известной парадигмой «управляющий-рабочие»: один из процессов (управляющий) распределяет подзадачи по рабочим процессам. Этот модуль используется на многопроцессорных вычислительных узлах, входящих в состав распределенной системы.

Система BNB-Grid загружает на управляющий модуль описание задачи в формате XML, в состав которого входит набор КНФ из декомпозиционного семейства. Управляющий модуль в дальнейшем распределяет эти КНФ по различным вычислительным узлам. Параллельному приложению передается порция задач, существенно превосходящая количество рабочих процессов, это нужно для оптимизации загрузки процессоров и максимального снижения простоя. Эффективность работы параллельного приложения тем выше, чем больше число переданных подзадач, так как в этом случае нивелируются негативные эффекты, связанные с разбалансировкой нагрузки. Однако, при слишком большом числе подзадач требуемое время работы параллельного приложения сильно увеличивается. Это является нежелательным из-за ограничений систем пакетной обработки, установленных на кластерах: параллельное приложение, запрашиваемое время работы которого велико, может быть не запущено или будет простаивать в очереди очень длительное время. Экспериментально было определено компромиссное число КНФ в порции задач, обеспечивающее приемлемые показатели эффективности и, вместе с тем, не приводящее к чрезмерно длительному ожиданию в очереди. Это число составляет $4 \cdot p$, где p — число процессов параллельного приложения.

Расчеты останавливаются, когда хотя бы на одном из вычислителей найден выполняющий набор. Это набор передается на управляющий модуль BNB-Grid и сохраняется пользователем с помощью графического интерфейса.

3.4. Вычислительные эксперименты

Для расчетов были выбраны три тестовых примера для генератора A5/1 с длиной инициализирующей последовательности 64 бита. Для каждой из полученных SAT-задач было сгенерировано декомпозиционное семейство, содержащее 2^{18} наборов. Расчеты осуществлялись на четырех многопроцессорных вычислительных комплексах, характеристики которых представлены в табл. 2, взятых из 11-й редакции списка самых мощных суперкомпьютеров стран СНГ[24].

В результате расчетов, на проведение которых потребовалось несколько дней, были решены все тестовые задачи криптоанализа A5/1. На каждом кластере на выполнение отправлялось по четыре приложения BNB-Solver. При этом из-за загруженности кластеров одновременно выполнялась только часть приложений. Количество одновременно работающих вычислительных ядер изменялось в процессе расчетов от 0 до 5568, составляя в среднем приблизительно 2–3 тысячи. Выполняющий набор (инициализирующая последовательность генератора) для первого теста был найден за 56 часов, для второго — за 25 часов, для третьего теста потребовалось 122 часа непрерывных расчетов. Время нахождения выполняющего набора в основном определяется следующими факторами: в каком сегменте декомпозиционного множества он расположен и загруженностью вычислителей.

Таблица 2

Характеристики многопроцессорных вычислительных комплексов, участвующих в вычислениях

Название	**Ведомственная принадлежность**	**Архитектура процессора***	**Число вычислительных ядер**
MVS-100k	Межведомственный суперкомпьютерный центр РАН	Xeon E5450 3 GHz	7920
СКИФ-МГУ	МГУ им. М. В. Ломоносова	Xeon E5472 3 GHz	5000
Кластер РНЦ	РНЦ «Курчатовский институт»	Xeon 5345 2.333 GHz	3456
BlueGene/P	ВМиК МГУ им. М. В. Ломоносова	Power PC 850 MHz	8192

* Приведена архитектура самого мощного процессорного узла.

Заключение

В статье рассмотрен параллельный вариант алгоритма решения задачи SAT и его реализация в программном комплексе BNB-Grid, предназначенном для решения задач оптимизации в среде распределенных вычислений. Результаты проведенных вычислительных экспериментов показывают, что при условии вовлечения достаточной вычислительной мощности, удается выполнить криптоанализ генератора A5/1 в течение нескольких дней. Данный результат показывает, с одной стороны, потенциальную применимость предложенного подхода для решения задач криптоанализа, с другой, аргументирует использование параллельных вычислительных технологий в решении SAT-задач, возникающих в практических приложениях. Высокая степень параллелизма использованного алгоритма позволяет надеяться на существенное снижение времени его работы с увеличением числа вычислительных ядер.

В дальнейшем планируется провести вычислительные эксперименты с целью нахождения всех выполняющих наборов для КНФ, кодирующих криптоанализ генератора ключевого потока A5/1. Данное исследование направлено на поиск всех возможных вариантов инициализирующих последовательностей, порождающих некоторый ключевой поток бесконечной длины — существование такого рода «коллизий» несложно показать теоретически, однако вопрос их полного перечисления представляется интересной вычислительной задачей.

В заключение авторы хотели бы выразить благодарность руководству Межведомственного суперкомпьютерного центра РАН, НИВЦ МГУ, ВМК МГУ, РНЦ «Курчатовский институт» за предоставленную возможность проведения длительных расчетов на высокопроизводительной вычислительной технике этих организаций.

Литература

1. SatLive: http://www.satlive.org
2. *Massacci F., Marraro L.* Logical Cryptoanalysis as a SAT Problem: the Encoding of the Data Encryption Standard. Preprint. Dipartimento di Imformatica e Sistemistica, Universita di Roma «La Sapienza». 1999.
3. *Семенов А. А., Буранов Е. В.* Погружение задачи криптоанализа симметричных шифров в пропозициональную логику // Вычислительные технологии (совместный выпуск с журналом «Региональный вестник Востока»). 2003. Т. 8. С. 118–126.
4. *Шнайер Б.* Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. М.: Триумф, 2002. 816 с.
5. *Семенов А. А., Буранов Е. В., Ушаков А. А.* Эвристический поиск в криптоанализе генераторов двоичных последовательностей // Вестник Томского гос. ун-та. Приложение № 17. 2006. С. 127–133.

6. *Семенов А. А.* Логико-эвристический подход в криптоанализе генераторов двоичных последовательностей // Труды международной научной конференции ПАВТ'07. Челябинск: ЮУрГУ, 2007. Т. 1. С. 170–180.
7. *Семенов А. А., Заикин О. С., Беспалов Д. В., Ушаков А. А.* SAT-подход в криптоанализе некоторых систем поточного шифрования // Вычислительные технологии. 2008. Т. 13. № 6. С. 134–150.
8. *Заикин О. С., Семенов А. А.* Технология крупноблочного параллелизма в SAT-задачах // Проблемы управления. 2008. № 1. С. 43–50.
9. *Семенов А. А., Заикин О. С.* Неполные алгоритмы в крупноблочном параллелизме комбинаторных задач // Вычислительные методы и программирование. 2008. Т. 9. С. 108–118.
10. *Семенов А. А., Заикин О. С., Беспалов Д. В., Буров П. С., Хмельнов А. Е.* Решение задач обращения дискретных функций на многопроцессорных вычислительных системах // Труды Четвертой международной конференции «Параллельные вычисления и задачи управления» PACO'2008 (Москва 26–29 октября 2008). С. 152–176.
11. *Biryukov A., Shamir A., Wagner D.* Real Time Cryptanalysis of A5/1 on a PC // Fast Software Encryption Workshop. 2000. P. 1–18.
12. MiniSat: http://minisat.se/MiniSat.html.
13. *Cook S. A.* The complexity of theorem-proving procedures, Proc. 3rd Ann. ACM Symp. on Theory of Computing, Association for Computing Machinery, Ohio, 1971, P. 151–159. [Перевод: Кук С. А. Сложность процедур вывода теорем. Кибернетический сборник. Новая серия. Вып. 12. С. 5–15. М.: Мир, 1975.]
14. *Семенов А. А.* Трансляция алгоритмов вычисления дискретных функций в выражения пропозициональной логики // Прикладные алгоритмы в дискретном анализе. Серия: Дискретный анализ и информатика, Вып. 2. Иркутск: Изд-во ИГУ, 2008. С. 70–98.
15. *Семенов А. А., Отпущенников И. В.* Об алгоритмах обращения дискретных функций из одного класса // Прикладные алгоритмы в дискретном анализе. Серия: Дискретный анализ и информатика, Вып. 2. Иркутск: Изд-во ИГУ, 2008. С. 127–156
16. Blackford: http://www.mvs.icc.ru/.
17. *Menezes A., Van Oorschot P., Vanstone S.* Handbook of Applied Cryptography. CRC Press, 1996. 657 p.
18. *Marqeus-Silva J. P. Sakallah K. A.* GRASP: A search algorithm for propositional satisfiability // IEEE Trans. on Computers, 1999. Vol. 48. № 5. P. 506–521.
19. НИВЦ МГУ: http://parallel.ru/cluster/.
20. *Афанасьев А. П.*, *Посыпкин М. А.*, *Сигал И. Х.* Проект BNB-Grid: решение задач глобальной оптимизации в распределенной среде // Труды Второй международной конференции «Системный анализ и информационные технологии» САИТ-2007. Т. 2. С. 177–181.
21. *Посыпкин М. А.* Методы и распределенная программная инфраструктура для численного решения задачи поиска молекулярных кластеров с минимальной энергией // Параллельные вычислительные технологии (ПаВТ'2009): Труды

Международной научной конференции (Нижний Новгород, 30 марта–3 апреля 2009 г.). Челябинск: Изд. ЮУрГУ, 2009. 599 с. С. 269–281.

22. *Evtushenko Y., Posypkin M., Sigal I.,* A framework for parallel large-scale global optimization // Computer Science — Research and Development 23(3). 2009. P. 211–215.
23. *Henning M.* A New Approach to Object-Oriented Middleware // IEEE Internet Computing, Jan 2004.
24. Одиннадцатая редакция списка top50 самых мощных компьютеров СНГ: http://www.supercomputers.ru.