Применение параллельных алгоритмов решения проблемы булевой выполнимости для криптоанализа сжимающего и самосжимающего генераторов ключевого потока

О.С. Заикин

***Аннотация*—Исследуются два генератора ключевого потока – сжимающий и самосжимающий, которые могут быть использованы для поточного шифрования. Для каждого из генераторов рассматривается задача криптоанализа, в которой по известному фрагменту ключевого потока необходимо найти 64-битный секретный ключ. Обе задачи были сведены к проблеме булевой выполнимости, а затем для их решения были использованы современные параллельные решатели, основанные на алгоритме CDCL. Были применены решатели двух видов. В решателях первого вида распараллеливается сам базовый CDCL-алгоритм. В решателях второго вида осуществляется распараллеливание по данным, в итоге решение исходной задач сводится к решению семейства независимых подзадач. В результате экспериментов, проведенных на вычислительном кластере, ряд экземпляров указанных задач криптоанализа удалось успешно решить. Оказалось, что параллельные решатели первого вида лучше подходят для решения рассмотренных задач криптоанализа.**

***Ключевые слова*—проблема булевой выполнимости, параллельные вычисления, криптоанализ, генератор ключевого потока.**

I. Введение

При передаче больших объемов данных (графики, звука, видео) их зачастую приходится шифровать в режиме онлайн. Для этого используются поточные шифры [1]. Главные требования, которые предъявляются к поточным шифрам – это криптостойкость (т.е. стойкость к криптоанализу) и высокая скорость работы. Можно отметить, что разработка и анализ поточных шифров – это бурно развивающееся направление современной криптографии.

В поточном шифровании используется пседослучайная последовательность символов, которая перемешивается с открытым текстом. Обычно эта псевдослучайная последовательность называется ключевым потоком, а для его получения используются генераторы ключевого потока. Анализ стойкости поточного шифра зачастую сводится к анализу стойкости соответствующего генератора.

В проблеме булевой выполнимости (SAT) требуется определить выполнимость булевой формулы [2]. В настоящем исследовании рассматривается поисковый вариант SAT: по данной булевой формуле найти ее выполняющий набор, т.е. набор значений переменных, который обращает ее в значение «истина», либо доказать, что такого набора не существует. Обычно булева формула представляется в виде конъюнктивной нормальной формы (КНФ), которая является конъюнкцией дизъюнктов. Под SAT-подходом к решению некоторой задачи понимается сведение этой задачи к SAT с последующим ее решением с помощью SAT-решателей. Большинство SAT-решателей основаны на алгоритме CDCL (Conflict Driven Clause Learning), который, в свою очередь основан на алгоритме DPLL (Davis–Putnam–Logemann–Loveland). Обзор указанных алгоритмов сделан, например, в статье [3].

SAT-подход активно используется для решения задач из различных предметных областей: комбинаторика (см., например, [4]), криптоанализ (см., например, [5]) и др. В настоящей работе с помощью SAT-подхода решаются задачи криптоанализа сжимающего и самосжимающего генераторов ключевого потока.

В последние 10 лет для решения задач криптоанализа при помощи SAT-подхода все чаще применяются высокопроизводительные вычисления: параллельные MPI-решатели, предназначенные для работы на вычислительных кластерах [6]; распределенные решатели, работающие в гридах из кластеров [7]; распределенные решатели, работающие в проектах добровольных вычислений [8]; распределенные решатели, работающие в гибридных гридах, состоящих из проектов добровольных вычислений и вычислительных кластеров [9].

Статья имеет следующую структуру. Во втором разделе описаны сжимающий и самосжимающий генераторы ключевого потока, приведены формулировки рассматриваемых задач криптоанализа

Статья получена 22 августа 2018. Работа поддержана советом по грантам Президента Российской федерации (грант МК-4155.2018.9) и Российским фондом фундаментальных исследований (грант 16-07-00155-а).

Олег Сергеевич Заикин, старший научный сотрудник Института динамики систем и теории управления им. В.М. Матросова СО РАН (e-mail: zaikin.icc@gmail.com).

этих генераторов, а также описано сведение этих задач к SAT. В третьем разделе приведены результаты вычислительных экспериментов, в рамках которых с помощью современных параллельных SAT-решателей был решен ряд экземпляров исследуемых задач криптоанализа.

II. Сжимающий и самосжимающий генераторы ключевого потока

Генератор ключевого потока – это дискретная функция, которая по данной на вход двоичной последовательности фиксированной длины выдает ключевой поток заданной длины. При этом вход генератора – это исходное внутреннее состояние его регистров, а длина ключевого потока совпадает с длиной открытого текста, который необходимо зашифровать.

Атака на основе открытых текстов (англ. known plaintext attack) на генератор ключевого потока формулируется следующим образом: по известному алгоритму генератора и известному фрагменту ключевого потока найти исходное внутреннее состояние генератора. Именно такой вид криптоанализа будет рассматриваться в настоящей статье. Отметим, что задачу криптоанализа в такой постановке можно рассматривать как задачу обращения дискретной функции, где дискретная функция – это генератор [10].

Регистр сдвига с линейной обратной связью (РСЛОС) - это линейная функция, которая каждый такт работы выдает один бит. РСЛОС не может быть использован в качестве генератора ключевого потока, т.к. обязательным требованием к генератору является его нелинейность. При этом РСЛОС — это удобный криптографический примитив, на основе которого построено большое количество генераторов ключевого потока. В некоторых генераторах, состоящих из нескольких РСЛОС, нелинейность достигается путем смешивания выходов РСЛОС с помощью нелинейной функции.

Сжимающий генератор был описан в статье [11]. Он основан на двух РСЛОС, при этом первый РСЛОС управляет вторым РСЛОС. Именно таким образом достигается нелинейность сжимающего генератора. Данный генератор работает следующим образом. Каждый такт оба РСЛОС выдают один бит. Если бит управляющего РСЛОС равен 1, то на выход генератора в этот такт выдается выходной бит управляемого РСЛОС. Если же он равен 0, то выход управляемого РСЛОС отбрасывается, и на выход генератора в этот такт ничего не выдается. Таким образом, ключевой поток генерируется с нерегулярной скоростью. В настоящей работе был исследован вариант сжимающего генератора, состоящего из РСЛОС со следующими полиномами обратной связи:

- РСЛОС № 1 (31-битный): $X^{31}+X^{7}+1$;
- РСЛОС № 2 (33-битный): $X^{33}+X^{16}+X^{4}+X+1$.

Здесь первый РСЛОС – управляющий, а второй – управляемый. Следует отметить, что использовались первый и третий РСЛОС из широко известного генератора A5/1, используемого для шифрования данных в мобильной связи [7].

Самосжимающий генератор был описан в статье [12]. Он базируется на той же идее, что и сжимающий генератор. Генератор, состоящий из одного РСЛОС, работает следующим образом. РСЛОС тактируется два раз. Если он выдал 00, то на выход генератора в этот такт выдается 0; если 11, то выдается 1; если 01 или 10, то на выход генератора в этот такт ничего не выдается. Был исследован вариант самосжимающего генератора, основанного на следующем РСЛОС: $X^{64}+X^{63}+X^{61}+X^{60}+1$.

Для каждого из описанных генераторов была рассмотрена задача криптоанализа, в которой по известному фрагменту ключевому потоку длиной 96 бит необходимо найти 64-битный секретный ключ, который соответствует начальному заполнению регистров генератора. Чтобы применить криптоанализ, основанный на SAT-подходе, для каждого из двух генераторов была построена SAT-кодировка. Это было сделано при помощи программного средства Transalg [13], предназначенного для сведения к SAT задач обращения дискретных функций. Характеристики КНФ, которые сгенерировал Transalg, приведены в Таблице 1.

Таблица 1. Характеристики КНФ, кодирующие исследуемые генераторы ключевого потока.

	Сжимающий	Самосжимающий
Число переменных	121 017	148 554
Число дизъюнктов	396 003	481 990
Размер, мб.	7.49	9.26

III. SAT-криптоанализ сжимающего и самосжимающего генераторов

Следует отметить, что обе КНФ, описанные в предыдущем разделе, кодируют алгоритмы работы самих генераторов, а не задачи их криптоанализа. На основе каждой из этих КНФ было сделано семейство из 10 КНФ, каждая из которых кодирует один экземпляр задачи криптоанализа соответствующего генератора. Под экземпляром задачи криптоанализа понимается задача поиска секретного ключа по конкретному известному фрагменту ключевого потока. Формирование этих семейств КНФ было сделано путем генерации ключевых потоков, и добавлении в исходные КНФ их значений в виде однолитеральных дизъюнктов. Таким образом, было построено 20 КНФ, для каждой из которых необходимо было решить SAT-задачу. Следует отметить, что каждая из этих КНФ является выполнимой, т.к. для каждого ключевого потока, значение которого закодировано в КНФ, точно существует не менее одно соответствующее исходное заполнение генератора (секретный ключ).

Все эксперименты проводились на вычислительном кластере «Академик В.М. Матросов» Иркутского научного центра РАН [16]. Каждый вычислительный узел этого кластера оснащен 128 Гб оперативной памяти и 2-мя 18-ядерными процессорами Intel Xeon E5-2695.

На первом этапе для решения описанных 20 SAT-задач были применены 4 многопоточных SAT-решателя: Plingeling, Treeengeling, Painless и Syrup. Каждый из них

является призером соревнований SAT Competition последних лет. Во всех этих решателях реализовано распараллеливание самого алгоритма CDCL. Каждый из этих решателей был запущен на каждой из 20 КНФ на одном вычислительном узле кластера (т.е. на 36 процессорных ядрах) с лимитом 1 сутки. Результаты расчетов представлены в Таблицах 2 и 3. Символ "-" в таблицах означает, что работа решателя была прервана из-за достижения временного лимита.

Таблица 2. Результаты решения задач 10 экземпляров задачи криптоанализа сжимающего генератора.

№	Plingeling	Treengeling	Painless	Syrup
1	-	-	-	-
2	-	-	-	-
3	4 ч. 37 м.	-	-	-
4	-	-	-	-
5	-	-	-	-
6	-	-	-	-
7	46 мин.	-	22 ч. 49 м.	-
8	-	-	-	-
9	-	-	-	-
10	37 мин.	-	-	-

Таблица 3. Результаты решения задач 10 экземпляров задачи криптоанализа самосжимающего генератора.

№	Plingeling	Treengeling	Painless	Syrup
1	-	-	8 ч. 59 м.	-
2	-	-	13 ч. 15 м.	-
3	-	-	12 ч.	-
4	-	-	14 ч. 43 м.	-
5	-	-	4 ч. 25 м.	-
6	-	-	-	-
7	-	-	-	-
8	16 ч. 8 м.	-	16 ч. 35 м.	-
9	-	-	23 ч. 8 м.	-
10	22 ч. 51 м.	-	9 ч. 15 м.	-

На втором этапе к семействам SAT-задач были применены многопоточные SAT-решатели, построенные с помощью программного средства ALIAS [14]. Данное средство позволяет конструировать многопоточные SAT-решатели на основе последовательных решателей и библиотеки IPASIR [15]. При этом используется распараллеливание по данным – исходная SAT-задача разбивается на независимые подзадачи. Данный вид распараллеливания позволяет использовать метод Монте-Карло для оценки трудоемкости решения SAT-задачи [6]. ALIAS-подобные решатели сначала работают в режиме прогнозирования, в котором с помощью метода Монте-Карло находится множество переменных, по которому и осуществляется распараллеливание SAT-задачи на подзадачи с их последующим решением. Были использованы следующие последовательные решатели: Rokk, Glucose, Abcdsat, Riss7. На их основе было построено 4 многопоточных SAT-решателя: ALIAS-Rokk, ALIAS-Glucose, ALIAS-Abcdsat и ALIAS-Riss7. Как было отмечено в ряде предыдущих статей (например, в [6]), прогноз трудоемкости, построенный для одной SAT-задачи из семейства SAT-задач, кодирующих экземпляры одной и той же задачи криптоанализа, подходит и для остальных задач из семейства. Поэтому каждый из ALIAS-подобных решателей был запущен в режиме прогнозирования на двух КНФ – по одной из каждого анализируемого семейства. При этом эти решатели также работали на 36 ядрах в течение суток. В Таблицах 4 и 5 представлены результаты расчетов. Кроме времени прогнозов указаны мощности соответствующих множеств переменных, на которых был получен прогноз. Следует отметить, что фактически находилось подмножество множества из 64 переменных КНФ, кодирующих секретный ключ.

Таблица 4. Прогноз трудоемкости (в секундах) решения задачи криптоанализа сжимающего генератора на 36 процессорных ядрах.

Решатель	Прогноз	Переменных
ALIAS-Rokk	15 суток, 7 ч.	25
ALIAS-Glucose	10 суток, 17 ч.	21
ALIAS-Abcdsat	10 суток, 9 ч.	20
ALIAS-Riss7	19 суток, 5 ч.	22

Таблица 5. Прогноз трудоемкости (в секундах) решения задачи криптоанализа самосжимающего генератора на 36 процессорных ядрах.

Решатель	Прогноз	Переменных
ALIAS-Rokk	319 063 927 лет	53
ALIAS-Glucose	252 945 839 лет	56
ALIAS-Abcdsat	243 717 973 лет	58
ALIAS-Riss7	472 710 553 лет	54

Исходя из больших прогнозных значений, решение SAT-задач на найденных множествах переменных запущено не было. И если в случае криптоанализа сжимающего генератора SAT-задачи можно было бы решить за относительно разумное время, то для самосжимающего сделать это было бы невозможно. Следует отметить, что для сжимающего генератора все найденные множества в основном (или даже полностью) состоят из переменных, кодирующих управляющий регистр генератора.

На основе результатов вычислительных экспериментов можно сделать следующие выводы. SAT-решатели на основе ALIAS плохо подходят для решения исследуемых задач криптоанализа. Если говорить о стандартных многопоточных решателях, то рассмотренный вариант сжимающего генератора является более стойким к SAT-криптоанализу, чем соответствующий вариант самосжимающего генератора. При этом лучший результат на сжимающем генераторе показал решатель Plingeling (решено 3 задачи из 10), а на самосжимающем генераторе – решатель Painless (решено 8 из 10 задач). Treengeling, как и Syrup, не смог решить ни одной из 20 SAT-задач. Следует отметить, что на тех экземплярах задач криптоанализа, где SAT-решателям удалось найти решение, реализация метода грубой силы показала бы худшие результаты, т.к. пришлось бы перебирать 2^{64} вариантов значений секретного ключа.

Новизна представленных результатов исходит из того, что ранее с помощью SAT-подхода осуществлялся криптоанализ сжимающего генератора в гораздо более простой формулировке – с секретным ключом всего 45 бит [17]. В статье [18] SAT-решатель использовался для решения полиномиальных уравнений, которые появляются путем угадывания части заполнения РСЛОС. При этом были построены только оценки сложности данной SAT-атаки, а реальные задачи криптоанализа решены не были. В статьях [5-9] SAT-подход был применен для криптоанализа генераторов Bivium и A5/1. В статье [19] с помощью SAT-подхода был осуществлен криптоанализ генератора переменного шага, который идейно очень близок к сжимающему генератору.

IV. Заключение

В данном исследовании с помощью параллельных вычислений и SAT-подхода успешно решен ряд экземпляров криптоанализа сжимающего и самосжимающего генераторов ключевого потока. При этом сжимающий генератор оказался более стойким к данному виду криптоанализа.

Благодарности

Автор благодарит Степана Евгеньевича Кочемазова за полезные советы и плодотворные обсуждения. Автор также признателен Илье Владимировичу Отпущенникову и Александру Анатольевичу Семенову за разработку программного средства Transalg, с помощью которого были сделаны все SAT-кодировки, использованные в статье.

Библиография

[1] Menezes A. J., van Oorschot P. C., Vanstone S.A. Handbook of applied cryptography. CRC Press. 1996. 810 p.

[2] Biere A., Heule M., van Maaren H., Walsh T. Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications. Vol. 185. IOS Press. 2009.

[3] Семенов А.А., Беспалов. Д.В. Технологии решения многомерных задач логического поиска // Вестник Томского государственного университета. 2005. № 14. С. 61–73.

[4] Белей Е.Г., Семенов А.А., О вычислительном поиске квазиортогональных систем латинских квадратов, близких к ортогональным системам // International Journal of Open Information Technologies. 2018. Vol. 6. No. 2. С. 22-30.

[5] Zaikin O.S., Otpuschennikov I.V., Semenov A.A. Solving weakened cryptanalysis problems for the Bivium cipher in the volunteer computing project SAT@home // International Journal of Open Information Technologies. 2015. Vol. 3. No. 12, pp. 1-3.

[6] Заикин О.С., Семенов А.А. Применение метода Монте-Карло к прогнозированию времени параллельного решения проблемы булевой выполнимости // Вычислительные методы и программирование: новые вычислительные технологии. 2014. Т. 15. № 1. С. 22-35.

[7] Посыпкин М.А., Заикин О.С., Беспалов Д.В., Семенов А.А. Решение задач криптоанализа поточных шифров в распределенных вычислительных средах // Труды Института системного анализа Российской академии наук. 2009. Т. 46. С. 119-137.

[8] Заикин, О.С., Семенов А.А., Посыпкин М.А. Процедуры построения декомпозиционных множеств для распределенного решения SAT-задач в проекте добровольных вычислений SAT@home // Управление большими системами. М.: ИПУ РАН. 2013. Вып. 43. С. 138–156.

[9] Afanasiev A.P., Bychkov I.V., Zaikin O.S., Manzyuk M.O., Posypkin M.A., Semenov A.A. Concept of a multitask grid system with a flexible allocation of idle computational resources of supercomputers // Computer and Systems Sciences International. 2017. Vol. 56. Issue 4. pp. 701–707.

[10] Семенов А.А. Декомпозиционные представления логических уравнений в задачах обращения дискретных функций // Известия Российской академии наук. Теория и системы управления. 2009. № 5. С. 47-61.

[11] Coppersmith D., Krawczyk H., Mansour Y. The shrinking generator // Proceedings of CRYPTO'93. Lecture Notes in Computer Science. Springer-Verlag. 1993. Vol. 773. pp. 23–39.

[12] Staffelbach O., Meier W. The Self-shrinking generator // Proceedings of EUROCRYPT'94. Lecture Notes in Computer Science. Springer-Verlag. 1994. Vol. 950. pp. 205–214.

[13] Otpuschennikov I.V., Semenov A.A., Gribanova I.A., Zaikin O.S., Kochemazov S.E. Encoding cryptographic functions to SAT using Transalg system // In Proceedings of ECAI'2016. Frontiers in Artificial Intelligence and Applications. 2016. Vol 285. pp. 1594-1595.

[14] Kochemazov S., Zaikin O. ALAIS: a Modular Tool for Finding Backdoors for SAT // Proceedings of SAT'2018. Lecture Notes in Computer Science. Springer-Verlag. 2018. Vol. 10929. pp. 419–427.

[15] Balyo T., Biere A., Iser M., Sinz C. SAT Race 2015. // Artificial Intelligence. 2016. Vol. 241. pp. 45-65.

[16] Иркутский суперкомпьютерный центр СО РАН. URL: http://hpc.icc.ru.

[17] Эли А.Н. Исследование стойкости генераторов ключевого потока к логическому криптоанализу // Новые информационные технологии в автоматизированных системах. 2015 № 18. С. 108-117.

[18] Debraize B., Goubin L. Guess-and-determine Algebraic Attack on the Self-Shrinking Generator // Proceedings of Fast Software Encryption 2008. Lecture Notes in Computer Science. Springer-Verlag. 2008. Vol. 5086. pp. 235–252.

[19] Zaikin O. Kochemazov S. An Improved SAT-Based Guess-and-Determine Attack on the Alternating Step Generator // Proceedings of International Conference on Information Security 2017. Lecture Notes in Computer Science. Springer-Verlag. 2017. Vol. 10599. pp. 21–38.

Application of parallel SAT solving algorithms for cryptanalysis of the shrinking and self-shrinking keystream generators

Oleg Zaikin

***Abstract*—In this study, shrinking and self-shrinking keystream generators are analyzed. These generators can be used in stream ciphering. For each generator the following cryptanalysis problem was considered: given a known keystream fragment, find the corresponding 64-bit secret key. Both problems were reduced to Boolean satisfiability problem (SAT), and then parallel CDCL solvers were used to solve them. Solvers of two types were employed. In solvers of the first type the base CDCL algorithm itself is parallelized. Solvers of the second type decompose an original SAT instance into a family of simpler independent subproblems. Computational experiments were held on a computing cluster. As a result, several cryptanalysis instances of the mentioned type were solved. It turned out, that the parallel solvers of the first type suit better for the considered cryptanalysis problems.**

***Keywords*—SAT, parallel computing, cryptanalysis, keystream generator.**