УДК 004.4 DOI 10.17223/2226308X/14/44

DPLL-ПОДОБНЫЙ РЕШАТЕЛЬ ЗАДАЧИ ВЫПОЛНИМОСТИ НАД СИСТЕМОЙ УРАВНЕНИЙ В АНФ[1]

А. В. Ткачев, К. В. Калгин

Описаны SAT-решатель, использующий системы булевых уравнений в алгебраической нормальной форме (АНФ) для внутреннего представления задачи, и особенности реализации типичных для SAT-решателей методик для работы с таким представлением. Приводится сравнение данного решателя с рядом классических SAT-решателей при решении некоторых задач криптоанализа (как, например, атака «guess-and-determine» на потоковый шифр Grain).

Ключевые слова: *SAT-решатель, АНФ, криптоанализ, потоковые шифры.*

Сложные задачи из совершенно различных областей, таких, как проектирование электронных схем или криптоанализ, могут быть представлены в виде задач выполнимости булевых формул (SAT) в конъюнктивной нормальной форме (КНФ). Для эффективного решения SAT существуют и активно развиваются специализированные инструменты — SAT-решатели.

SAT в своей изначальной формулировке подразумевает представление в КНФ, и ещё в 1971 г. было доказано, что данная задача является NP-полной [1]. Современные SAT-решатели обрабатывают задачи, представленные в КНФ (чаще всего записанные в текстовом формате DIMACS).

Однако для некоторых задач — например криптоанализа шифров — «естественным» представлением является АНФ. А потому не менее естественным кажется желание использовать инструмент, напрямую работающий с АНФ. Авторам не удалось обнаружить такого инструмента, равно как и обоснования, что он был бы неэффективным, и потому было решено разработать его самостоятельно. Ближайшим аналогом можно назвать проект Bosphorus [2], использующий представление в АНФ в дополнение к решению задачи в КНФ.

Для удобства решатель работает не с одной большой формулой в АНФ, а с системой уравнений в АНФ, то есть конъюнкцией множества отдельных АНФ с общим набором неизвестных.

1. Конструкция решателя

Постановка задачи: для данной системы уравнений в АНФ выяснить, имеет ли она решение, и если да, то найти одно из них.

Разработанная программа разделена на модули, которые можно отключать независимо друг от друга, чтобы можно было исследовать, какие алгоритмы, методики и эвристики дают наибольший вклад в ускорение решения и насколько эффективно они работают вместе (подобное исследование для классических SAT-решателей проводилось в [3]). По структуре решатель напоминает классический SAT-решатель, основанный на алгоритме DPLL, а реализованные модули — адаптированные под АНФ модификации этого алгоритма.

Распространение констант. Как и в классическом DPLL, одним из базовых этапов работы решателя является вывод значений переменных после означивания очередной переменной. В ходе этого этапа находятся уравнения, которые после подстановки

[1]Работа выполнена в рамках госзадания ИМ СО РАН (проект № 0314-2019-0017) и лаборатории криптографии JetBrains Research.

означенных переменных становятся тривиальными, и вычисляются значения новых переменных. Просмотр уравнений продолжается до тех пор, пока удаётся делать выводы значений переменных. Если в ходе этого этапа выявляется конфликт, это означает, что текущие значения переменных неверны, и происходит откат всех выводов и предшествующего им означивания переменной в DPLL, а затем DPLL продолжает работу, пытаясь означить переменные по-другому. Ввиду представления ограничений в АНФ многие уравнения могут быстро становиться тривиальными — если хотя бы одна переменная в мономе равна нулю, то и весь моном равен нулю, и нужно рассматривать только ненулевые мономы, часть переменных в которых уже может быть означена.

Распространение синонимов. Помимо простого вывода значений отдельных переменных, из ограничений в АНФ при определённых обстоятельствах можно сделать вывод, что две переменные должны иметь одинаковые или противоположные значения. Это может сокращать количество неизвестных в обрабатываемых уравнениях, что позволяет быстрее обнаружить конфликт. Такие выводы не требуют каких-либо сложных структур или вычислений. Рассмотрим такое ограничение: $abx + bcx + acx + y = 1$. Если $a = b = c = 1$, то после упрощения ограничение будет иметь вид $x + x + x + y = 1$, что эквивалентно $x + y = 1$, и несложно сделать вывод, что значения x и y должны быть противоположными. Тогда можно объявить y «антонимом» x и при просмотре других ограничений подставлять вместо y отрицание x, снижая таким образом число неизвестных в этом ограничении и ещё больше упрощая его.

Отслеживаемые мономы (2WM). По аналогии с Watched Literals (2WL) при работе с КНФ, при обработке ограничений в АНФ можно поддерживать структуру «отслеживаемых» частей уравнений. Если в КНФ это отдельные литералы, изменение значений которых может приводить к изменению значения всего дизъюнкта, то в АНФ такой частью является отдельный моном, от значения которого зависит значение всего полинома. Данная эвристика ограничивает число полиномов, которые нужно просмотреть при изменении значения переменной, что позволяет сократить общее время работы решателя, но при этом гарантирует, что «важные» полиномы (которые действительно могут измениться при изменении значения переменной) не будут пропущены.

Упрощение уравнений перед обработкой. Прежде чем запустить DPLL, решатель подставляет известные значения переменных (если таковые имеются, то есть заданы с помощью уравнений вида $x = \text{const}$) во все уравнения и выводит значения других переменных, а также находит синонимы, если это возможно. Эта процедура повторяется до тех пор, пока не останется переменных, значения которых можно вывести и подставить. Упрощенные таким способом уравнения становятся короче и потому обрабатываются немного быстрее. В некоторых случаях возможно доказать невыполнимость или даже вывести значения всех переменных исключительно в ходе этой процедуры, без запуска DPLL.

Порядок выбора переменной. Порядок, в котором выбирается следующая переменная для означивания в алгоритме DPLL, может иметь значительное влияние на время работы решателя. Хотя в разработанном решателе пока можно задать только один из нескольких простых порядков выбора — например, отсутствует выбор переменной на основе «активности», которая подсчитывается в ходе работы решателя на основе частоты встречаемости переменной, возможность смены порядка в него заложена. По умолчанию решатель выбирает неозначенную переменную с наименьшим порядковым номером.

2. Результаты сравнения

Приведём результаты тестирования современных решателей lingeling, cryptominisat5 и представленного в работе решателя на задаче восстановления части ключа (атака «guess-and-determine» [4]) потокового шифра Grain [5]. Атака заключается в том, что часть неизвестных бит фиксируется и эти биты означиваются тем или иным образом («угадываются»), а затем с помощью SAT-решателя вычисляются значения оставшихся неизвестных бит или выявляется, что при таких значениях зафиксированных бит шифр не мог сгенерировать известную гамму, и тогда выбираются другие значения зафиксированных бит. В общем случае при фиксации k бит из n неизвестных потребуется рассмотреть 2^k задач, а время такой атаки можно оценить как $2^k T$, где T — среднее время работы SAT-решателя на одной задаче, в которой k из n бит означены.

Обычно рассматриваются два варианта восстановления начального состояния — задача восстановления ключа, когда известна вся гамма, и задача восстановления состояния регистров, когда гамма известна с какого-то произвольного момента времени. Первую задачу будем обозначать «init=yes», а вторую — «init=no». В шифре Grain из-за фазы инициализации в случае «init=yes» получаются более сложные уравнения, описывающие зависимости значений генерируемых бит ключевого потока от исходных бит ключа, но неизвестными считаются только 80 бит ключа. В случае «init=no» биты гаммы генерируются сразу из некоторого состояния и уравнения получаются более простыми, но неизвестными считаются все 160 бит состояния регистров. Атаки на подобные шифры в различных режимах проводились в [6, 7].

Эксперименты показали, что в случае «init=no» классические и разработанный решатель ведут себя схожим образом, и разработанный решатель лишь незначительно проигрывает классическим по времени решения задач. Однако в случае «init=yes» классические решатели серьёзно уступают разработанному. Некоторые решатели не справляются с задачами такого типа, даже если неизвестно всего 1–2 бита — вероятно, решатель «запутывается» среди множества промежуточных переменных и начинает выбирать их вместо «важных» переменных, соответствующих битам ключа.

На рис. 1 показаны оценки времени атаки для разных решателей. Видно, что при использовании разработанного решателя время остаётся более-менее стабильным при увеличении числа неизвестных бит, в то время как для классических решателей время атаки поначалу снижается, но после некоторого значения начинает возрастать, при этом оценка времени выше. В случае «init=no» общее число неизвестных больше и время полной атаки гораздо выше (10^{30} ч против 10^{17} в случае «init=yes»), поэтому проводить атаку с применением SAT-решателей в этом режиме нецелесообразно.

В дальнейшем планируется провести исследование на большем числе шифров и SAT-решателей. Однако уже из проведённого сравнения понятно, что разработка решателя, использующего представление уравнений в АНФ, перспективна.

Планируется выяснить, с чем связано то, что современные решатели уступают по времени представленному в работе на порядок и больше. Вероятно, замедление классических решателей может быть связано с неудачными эвристиками выбора порядка, в котором означиваются биты. Это означает, что перспективной является разработка новых эвристик как для решателей, использующих представление уравнений в КНФ, так и для представленного в работе. Другая версия — неэффективность работы эвристики clause learning (обучение и пополнение базы конъюнкций) современных решателей на рассматриваемой задаче.

Помимо этого, необходимо изучить применимость других алгоритмов и методик, использующихся в классических решателях, к решателю, основанному на АНФ. На-

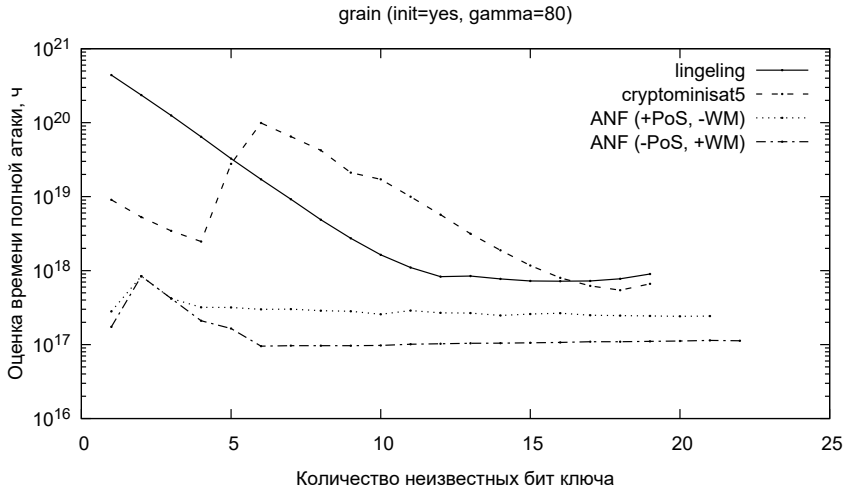

Рис. 1. Оценка времени полной атаки в зависимости от количества неизвестных бит

пример, насколько эффективен clause learning при работе с КНФ и можно ли его адаптировать к АНФ. Возможно, следует использовать другие методы — например, добавление новых уравнений, подобно тому, как это делается в алгебраических атаках.

ЛИТЕРАТУРА

1. *Cook S.* The complexity of theorem-proving procedures // 3rd Ann. ACM Symp. Theory Comput. 1971. P. 151–158.
2. *Choo D., Soos M., Chai K. M. A., and Meel K. S.* BOSPHORUS: Bridging ANF and CNF solvers // Proc. DATE Conf. Exhibition. 2019. P. 468–473.
3. *Katebi H., Sakallah K., and Silva J.* Empirical study of the anatomy of modern SAT Solvers // LNCS. 2011. V. 6695. P. 343–356.
4. *Bard G. V.* Algebraic Cryptanalysis. Springer, 2009.
5. *Hell M., Johansson T., and Meier W.* Grain: A stream cipher for constrained environments // Intern. J. Wireless Mobile Comput. 2007. No. 2. P. 86–93.
6. *Yeo S., Le D. P., and Khoo K.* Improved algebraic attacks on lightweight block ciphers // J. Cryptogr. Eng. 2011. No. 11. P. 1–19.
7. *Semenov A. A. and Zaikin O. S.* Algorithm for finding partitionings of hard variants of Boolean satisfiability problem with application to inversion of some cryptographic functions // SpringerPlus. 2016. Art. No. 554. P. 1–16.