УДК 519.7

МИНИМИЗАЦИЯ ФУНКЦИОНАЛОВ, АССОЦИИРОВАННЫХ С ЗАДАЧАМИ КРИПТОГРАФИЧЕСКОГО АНАЛИЗА АСИММЕТРИЧНЫХ ШИФРОВ

© Дулькейт В.И., Файзуллин Р.Т., Хныкин И.Г.

Омский государственный университет им. Ф.М. Достоевского
пр-т Мира, 55а, г. Омск, 644077, Россия

e-mail: *r.t.faizullin@mail.ru*

Abstract. Global optimization problems associated with cryptographic analysis of asymmetric ciphers. The aim of this article is to establish relation between well-known problems of cryptographic analysis and global optimization problems which can be associated with SAT representation of cryptographic algorithms where bits of key is part of SAT solution string. There was constructions SAT forms for factorization problem, SAT forms for logarithmic problem and logarithmic problem on elliptic curves. For numerical solution was adapted some low relaxation algorithms and for example results of numerical experiments give to us strong more then 50% bits for unknowns in SAT factorization form.

Введение

В работе рассматривается модификация метода логического криптоанализа [1], основанная на процедуре минимизации функционалов, ассоциированных с задачами криптографического анализа асимметричных шифров. Предложены алгоритмы сведения криптографических алгоритмов RSA, дискретного логарифмирования и дискретного логарифмирования на эллиптических кривых к задаче ВЫПОЛНИМОСТЬ и последующему сведению к проблеме поиска глобального минимимума ассоциированных функционалов. Показано, что для задачи факторизации подход позволяет получить строго более чем 60% бит определяющих ключ бит.

1. Концепция

Пусть $L(x)$ – пропозициональная формула в конъюнктивной нормальной форме на множестве булевых переменных $x \in B^N\{0,1\}$. Задача ВЫПОЛНИМОСТЬ (SAT) заключается в том, что бы найти решающий набор $x_0 \in B^N\{0,1\}$, такой что $L(x_0) =$ ИСТИНА или доказать, что решающего набора не существует.

Рассмотрим переход от задачи ВЫПОЛНИМОСТЬ к задаче поиска глобального минимума функционала вида (1).

Пусть дана КНФ:

$L(x) = \bigwedge\limits_{i=1}^{M} c_i$, где c_i – дизъюнкты вида $c_i = \bigvee\limits_{j<N} q_{i,j}(x_j)$. Здесь $q_{i,j}(x_j) = x_j$ или $\bar{x}_j$.

Сделаем переход к эквивалентной ДНФ:

$D = \tilde{L}(x) = \bigvee\limits_{i=1}^{M} \tilde{C}_i$, где $\tilde{C}_i$ – конъюнкты вида $\tilde{C}_i = \bigwedge\limits_{j<N} \tilde{Q}_{i,j}(x_j)$. Здесь $\tilde{Q}_{i,j}(x_j) = \bar{q}_{i,j}(x_j)$

Рассмотрим функционал вида:

$$\min_{x\in E^N} F(x) = \sum_{i=1}^{M} C_i(x),$$

где $C_i(x) = \prod_{j=1}^{N} Q_{i,j}(x_j)$ и где

$$Q_{i,j}(x_j) = \begin{cases} (1-x_j)^2 & \text{, если } x_j \in C_i(x) \\ x_j^2 & \text{, если } \overline{x}_j \in C_i(x) \\ 1 & \text{, иначе} \end{cases} \tag{1}$$

Суммирование ведется по всем M конъюнктам ДНФ, эквивалентной исходной КНФ. Соответствие между булевыми и вещественными переменными следующее: ЛОЖЬ→0, ИСТИНА→1.

Переход от булевой формуле к вещественной основан на использовании соответствия:

$$\begin{cases} y_i \vee y_j \to x_i + x_j \\ y_i \wedge y_j \to x_i^2 x_j^2 \\ \overline{y}_i \to (1-x_i) \end{cases} \text{, где } \{y_i \in B, x_i \in R\}$$

Легко заметить, что $\min_{x\in E^N} F(x) = 0$ соответствует достижению значения ИСТИНА на исходной КНФ.

Без потери общности можно рассмотреть 3-ДНФ, эквивалентную исходной КНФ и ассоциированый функционал:

$J(x) = \sum_{\xi} z_i^2 z_j^2 z_k^2,$

$$\text{где} z_i = \begin{cases} 1-x_i, \text{если} x_i \in c_i(x) \\ x_i, \text{если} \overline{x_i} \in c_i(x) \end{cases}, \ c_i(x) - i \tag{2}$$

триплет Дифференцируя функционал по всем переменным x_i, получаем систему уравнений:

$\sum_{\xi\in\Xi} z_j^2 z_k^2 x_i = \sum_{\xi\in\Lambda} z_j^2 z_k^2, i = 1, 2, ..P,$

$$\text{где } \begin{array}{l} \Xi = \{\xi,\ i \in \xi : x_i \in c_i(x)\} \\ \Lambda = \{\xi,\ i \in \xi : \overline{x_i} \in c_i(x)x\} \end{array} \tag{3}$$

Коэффициент при x_i A_i и правая часть B_i связаны соотношением: $A_i(x_1, .., x_{i-1}, x_{i+1}, ..x_n) \geq B_i(x_1, .., x_{i-1}, x_{i+1}, ..x_n)$

Поясним выбор представления исходной КНФ именно в виде эквивалентной 3-ДНФ. Дифференцируя функционал $F(x)$ (см.(1)) по всем переменным x_i, получаем систему уравнений аналогичную (3), но количество вкладов в A_i и B_i определятся длиной скобок. Любая процедура решения этой системы при произвольной длине скобок будет естественным образом приводить к большим ошибкам округления. Ограничивая число переменных в скобках, мы исключаем эту техническую трудность.

Рассмотрим систему (3), как нелинейное операторное уравнение: $\Phi(x) = 0$.

Как показано в [2] применение метода Ньютона к решению данного уравнения неэффективно, т.к. решение принадлежит ядру производного оператора. Как альтернатива был предложен метод последовательных приближений с «инерцией» [3]:

$$(\sum_{p=0}^{K}\sum_{\xi\in\Xi}\alpha_p x_i(t-p)^2 x_j(t-p)^2)x_k(t+1) = \sum_{\xi\in\Lambda} x_j^2(t)x_k^2(t) \sim A\cdot x_k(t+1) = B$$

$$\sum_{p=1}^{K}\alpha_p = 1 \tag{4}$$

Имеется ввиду то, что итерации происходят для вещественных чисел, а итоговый или промежуточный вектор проектируется на $\mathrm{B}^{\mathrm{N}}\{0,1\}$, и уже на булевом векторе проверяется SAT. Ниже мы опишем различные модификации и гибридизации метода последовательных приближений с «инерцией» в применении к решению задачи K-SAT, и покажем способы повышения эффективности алгоритма.

2. Гибридизация алгоритма

Основная процедура состоит из последовательных итераций, которые совмещают метод последовательных приближений и сдвиг по антиградиенту, т.к. дефект (3) это не что иное, как антиградиент исходного функционала.

Итерация состоит из двух блоков. Первый блок, определяется формулой (5), используется схема Зейделя. Суть схемы Зейделя в том, что при нахождении очередного $x_i(t+1)$ на $(t+1)$-й итерации, это значение подставляется вместо $x_i(t)$. Необходимо отметить, что реализация алгоритма допускает использование схемы Якоби, когда найденные $x_i(t+1)$ не используются в текущей итерации. Тесты показали, что схема Зейделя более устойчива в применении к решаемой задаче. Именно, после каждой итерации по схеме Зейделя значение функционала монотонно уменьшается, чего нельзя сказать о схеме Якоби. Кроме того, во всех случаях схема Зейделя быстрее приводила к решению. Отметим, что данное обстоятельство препятствует напрашивающейся простой схеме распараллеливания процесса решения с разделением данных.

Второй блок – реализация сдвига по градиенту. Рассмотрим (3). Пусть $\bar{x}(t)$ является решением, тогда $\Phi(\bar{x}(t)) = 0$.

Уравнение (3) переписывается в виде $A(\bar{x}(t))\cdot\bar{x}(t) - B(\bar{x}(t)) = 0$. Это необходимое условие, которому должен удовлетворять вектор решения. Если текущее t-е приближение $\bar{x}(t)$ не является решением, то $A_i(\bar{x}(t))\cdot x_i(t) - B_i(\bar{x}(t)) = p_i \neq 0$. Для итеративной формулы: $A_i(\bar{x}(t))\cdot x_i(t+1) - B_i(\bar{x}(t)) = p_i$. Следовательно, что бы удовлетворить необходимому условию, необходимо перейти к вектору:

$\bar{x}_i(t+1) = \bar{x}_i(t) + p_i/A_i$

Очевидно, что после реализации (6) возможна ситуация, когда $\bar{x}_i(t+1) \notin R[0,1]$. В этом случае необходимо штрафным способом ограничивать $\bar{x}_i(t+1)$, иначе метод

начинает экспоненциально расходиться. Особенно это проявляется на K-SAT формулах при K>4. При приближении к решению скорость сходимости может сильно уменьшаться, т.е. алгоритм формирует цикл лежащий на некотором плато (поверхность определяемая функционалом) и траектория, образованная последовательными приближениями более не выходит за пределы этого плато. Чтобы сойти с плато и продолжить сходимость к решению применяется т.н. метод смены траектории.

Метод смены траектории заключается в поиске нового вектора приближения, который бы обладал свойствами не худшими, чем текущий вектор приближения, но позволял бы продолжить поиск решения. Суть метода нахождения такого вектора в следующем.

Рассмотрим 3-КНФ, эквивалентную исходной 3-ДНФ:

$$K(x) = \prod_{\xi} (z_i + z_j + z_k), \text{ где } z_i = \begin{cases} x_i, \text{если} x_i \in c_i(x) \\ 1 - x_i, \text{если} \overline{x_i} \in c_i(x) \end{cases}, \ c_i(x) - i \text{ триплет} \quad (5)$$

$$K(x) = 1 \Leftrightarrow (z_i + z_j + z_k) \neq 0, \forall \xi$$

Для данного приближения, рассмотрим множество переменных:

$$E0 = \{ x_i \mid \exists \text{ триплет } c_i : x_i \text{или} \neg x_i \in c_i \& c_i(\bar{x}) = 0 \}$$

С вероятностью m_p поменяем значения x_i на противоположные. При этом, вероятность того, что другие триплеты станут невыполнимыми не высока. Экспериментально установлено, что полученный вектор x_0 обладает свойствами не худшими, чем $\bar{x}$ (количество невыполнимых триплетов до и после операции примерно одинаково). Используя данный вектор x_0 в качестве нового начального приближения, алгоритм очень быстро (в большинстве случаев за 5-10 итераций) находит следующее приближение, на котором функционал $F(x)$ достигает значения не хуже, чем на векторе $\bar{x}$. При этом, очень часто, удается проскочить плато, но при дальнейшем движении по новой траектории метод может зациклиться на другом плато. Тогда метод смены траектории повторяется.

Дополнительно рассмотрим множество:

$$E1 = \{ x_i \mid \exists \text{ триплет } c_i : x_i \text{или} \neg x_i \in c_i \& c_i(\bar{x}) = 1 \}$$

Введем вероятности m_{p0}, m_{p1}. С вероятностью m_{p0} при смене траектории будем использовать множество $E0$. С вероятностью m_{p1} – множество $E1$. Вероятность m_{p0} влияет на величину изменения вектора и при этом количество невыполнимых триплетов не увеличивается. Вероятность m_{p1} влияет на качественное изменение вектора, количество невыполнимых триплетов может увеличиться. В принципе, чем выше m_{p1}, тем меньшую роль играют рестарты. Метод смены траектории применяется при достижении условия $|F(\bar{x}_2) - F(\bar{x}_1)| < \varepsilon_2$

3. Преобразование исходной КНФ методом резолюции

Данное преобразование позволяет получить КНФ с меньшим количеством дизъюнктов и литералов, эквивалентную исходной. «Резольвента» – дизъюнкция конъюнктов, отличающихся знаком по единственной переменной. Все возможные резольвенты добавляются к КНФ и используются для вычисления других резольвент.

Дублирующие конъюнкты и тавтологии удаляются, и используется сокращенная процедура с глубиной рекурсии 1. Вычислительная сложность процедуры $O(n \cdot \log n)$. Метод резолюции в применении к КНФ, ассоциированных с задачами факторизации и дискретного логарифмирования (см. ниже) позволяет уменьшить исходное число конъюнктов до 50% и иногда разрешить до 20% переменных. Подробно о методе резолюций можно найти в [4].

4. Способы распараллеливания алгоритма

Гибридизированный алгоритм допускает целый набор способов распараллеливания. Исходная формула, делится на определенное количество подформул. Для каждой подформулы находится вектор решения. Найденные вектора некоторым образом объединяются в один, который потом используется для поиска решения всей формулы. Были исследованы два способа реализации параллельного алгоритма.

ПРОЦЕДУРА 1

ДНФ, эквивалентная исходной КНФ делится на n подформул. Для каждой из подформул, с помощью основного алгоритма, ищется выполнимый набор. Полученные вектора используются в качестве инициализирующего набора для следующей подформулы. После n-1 итерации вычисляется усредненный набор. Данный набор используется в качестве инициализирующего для итерационной процедуры, которая применяется ко всей формуле. Вычислительные эксперименты показали что, выполнимый набор для подформул, состоящих из 90% (и более) от исходного скобок числа находится во всех случаях и за минимальное количество итераций (не более 20). Данный способ показал неплохие в среднем результаты при решении различных типов примеров.

ПРОЦЕДУРА 2

ДНФ, эквивалентная исходной КНФ делится на две подформулы. Ищется выполнимый набор для каждой из двух подформул. Вычислительные эксперименты показали, что в полученных наборах решений значения литералов совпадают на 60%. Далее применяется следующая процедура. Осуществляется формирование нового набора приближений для итерационной схемы путём усреднения значений переменных двух наборов, полученных на предыдущем этапе:

$$\mathrm{x}_i = \frac{x_{1i} + x_{2i}}{2}.$$

Далее запускается основная схема решений, но уже для функционала J, который ассоциирован со всей формулой. Тесты показали, что данная процедура увеличивает число выполнимых скобок, но не всегда находит выполнимый набор для всей формулы. Для улучшения результатов была рассмотрена следующая естественная модификация. Каждый вектор решений для соответствующей подформулы, определяет точку в n-мерном пространстве. Между полученными точками проводится отрезок прямой. Двигаясь по этой прямой с некоторым шагом l вычисляются новые вектора x_i по формуле: $x_{li} = \min(x_{1i}, x_{2i}) + \frac{|x_{1i} - x_{2i}|}{k} \cdot l$, где l – это номер шага. Для каждого x_l вычисляется значение функционала J. Затем выбирается тот набор значений $x_{li} = \min(x_{1i}, x_{2i}) + \frac{|x_{1i} - x_{2i}|}{k} \cdot l$, где l, значение, при котором значение данного

функционала минимально. Этот вектор и будет являться новым начальным набором приближений для итерационной процедуры, которая запускается для функционала, ассоциированного со всей формулой.

Как показали тесты, данная процедура наиболее эффективно находит набор решений для многих тестовых формул. Конечно, есть примеры, для которых данный метод не вычисляет точного набора значений. Но описанная процедура позволяет максимально приблизиться к решению. В формуле остаётся до 2% скобок невыполнимыми. Кроме того, 2.5% переменных остаются не использованными, то есть независимо от того какое значение они будут принимать, выполнимые скобки будут по-прежнему принимать значение ИСТИНА. Но при изменении их значения на противоположное количество выполнимых скобок увеличивается.

5. Увеличение разрядности вычислений

Была исследована сходимость алгоритма при увеличении разрядности вычислений. Испытания с типами DOUBLE и FLOAT показали преимущество вычислений с двойной точностью. При переходе на тип DOUBLE количество решенных примеров увеличивается на 10%, скорость сходимости в среднем также увеличивается. Дальнейшее увеличение разрядности к значимому эффекту не приводит.

6. Результаты численных экспериментов

После каждой модификации проводилось тестирование алгоритма для определения эффективности проделанных изменений. При тестировании использовалось несколько типов примеров: тесты с соревнований решателей SAT 2005 года www.lri.fr/∼simon/, тесты библиотеки SATLib www.cs.ubc.ca/∼hoos/, тесты, сформированные для задач факторизации и дискретного логарифмирования, тесты для КНФ больших размерностей, сформированные случайным образом. Подробные результаты представлены в [3].

Оказалось, что сдвиг по градиенту хорошо сокращает погрешности и ускоряет сходимость алгоритма. Вычислительные эксперименты со случайными формулами показали заметное уменьшение времени решения тестов. Число решенных примеров увеличилось примерно на 20%. Применение данного приема позволило достаточно эффективно решать тесты uf20-91 (из 1000 тестов решены 703). Однако некоторые тесты решались только после задания определенного начального приближения, что говорит о необходимости рестартов. На примерах uf250-1065 алгоритм показал результат 6% от стандартных трудных тестов (предыдущая версия алгоритма – 1%). Тесты SAT-2005 (OKGenerator10000-42000 – 10000 переменных, 42000 скобок) использовались в сокращенной форме. Максимально удалось решить подформулы из 36000 скобок (предыдущая версия алгоритма – 35000 скобок). На подформулах из более чем 36000 скобок метод зацикливается на некотором плато, что говорит о необходимости смены траектории.

Метод смены траектории существенно увеличил число решаемых примеров. Результаты представлены в таблице1.

Таблица 1. Результаты тестирования алгоритма + метод смены траектории

Наименование теста	Литералы (N)	Дизъюнкты (M)	Тесты	Успех %	Итерации
BSI, 3-SAT					
RTI	100	429	500	98,6	19988
BMS	100	<429	500	79,8	29831
CBSI, 3-SAT					
CBS_b10	100	403	1000	100	38972
CBS_b10	100	449	1000	100	38880
CBS_b90	100	449	1000	98	29738
UF 3-SAT					
uf20-91	20	92	1000	100	448
uf250-1065	250	1065	100	98	9731
"Flat"Graph Colouring Problems					
flat30-60	90	300	100	100	4317

Таблица 2. Результаты тестирования параллельного алгоритма (процедура 1).

Наименование теста	Литералы	Скобки (M)	% решенных тестов	Число итераций
RTI	100	429	85	14
BMS	100	<429	85	14
sat05-1663	2000	8400	55	200
sat05-1676	4000	16800	50	200
sat05-1656	12000	50400	50	200
uf20-91	20	91	90	14
uf250-1065	250	1065	90	21

Результаты тестирования параллельного алгоритма на различных типах примеров приведены в таблицах 2,3. Можно сделать заключение о том, что процедура 2 более эффективна.

7. Сведение к КНФ задачи факторизации

В последнее время наблюдается повышенный интерес к проблеме кодирования криптографических алгоритмов в терминах задачи выполнимости (SAT). В работе [5] эскизно иллюстрируется подход, позволяющий в принципе свести задачу факторизации к SAT. Далее будет представлен алгоритм построения алгоритмов генерации эквивалентных, но различных КНФ и последующей минимизации ассоциированного функционала, для задачи факторизации, что позволит применить процедуры распаралелливания 1 и 2.

Таблица 3. Результаты тестирования параллельного алгоритма (процедура 2).

Наименование теста	Количество литералов (N)	Количество скобок (M)	% решенных тестов	Число итераций (часть формулы)	Число итераций (вся формула)
RTI	100	429	100	10	14
BMS	100	< 429	100	7	14
sat05-1663	2000	8400	99	20	200
sat05-1676	4000	16800	99	20	200
sat05-1656	12000	50400	99	20	200
uf20-91	20	91	100	10	14
uf250-1065	250	1065	100	20	21

Рассмотрим непосредственно алгоритм сведения к КНФ задачи факторизации. Требуется для заданного числа n получить КНФ, решающий набор которой существует тогда, и только тогда когда n составное число. Кроме того, решающий набор должен содержать все биты двоичного представления нетривиальных делителей n. Без потери общности, рассмотрим классический алгоритм умножения "столбиком". Будем отождествлять биты сомножителей и результата с литералами (свободными логическими переменными). Результат умножения первого сомножителя на i-ый бит второго можно представить в виде вектора $\overline{P_i}$. Именно суммирование всех этих векторов представляет основную сложность. Поэтому предлагается выполнять эту операцию последовательно с сохранением результата в промежуточных векторах $\overline{S_k}$.

Весь процесс вычисления можно разбить на три этапа относительно операции сложения:

1. Сложение векторов составленных из произведений двух литералов. Выполняется один раз. В результате этой операции будет заполнен вспомогательный вектор сумм $\overline{S_1}$ и вычислено два младших бита результата. Условно данный этап можно записать так:$\bar{P}_1 + \bar{P}_2 = (\bar{S}_2, r_2, r_1)$.

2. Суммирование вектора $\overline{S_k}$ с вектором произведений . Выполняется N-3 раз. В результате заполняется массив $\overline{S_{k+1}}$ и вычисляется очередной бит результата

3. Последнее суммирование вектора $\overline{S_k}$ с вектором произведений. Выполняется один раз. В результате вычисляются оставшиеся биты результата.

Теперь перейдём к рассмотрению идеи генерации КНФ. Простейший случай – приравнивание одного литерала другому: $x = y$. Данное равенство будет справедливо тогда и только тогда, когда истинна формула $(\bar{x} \vee y)\,(x \vee \bar{y})$.

Другим часто встречающимся выражением является: $x = A \oplus B \oplus C$ (8)

Поступая аналогичным образом, получаем эквивалентную формулу:

$$(\overline{x} \vee (A \oplus B \oplus C))\left(x \vee \overline{(A \oplus B \oplus C)}\right) = (Ax \oplus Bx \oplus Cx \oplus \overline{x})\,(A\overline{x} \oplus B\overline{x} \oplus C\overline{x})$$

Для представления правой части в виде КНФ воспользуемся леммами 1, 2.

Лемма 1.

$\bigoplus\limits_{i=1}^{N} x_i = \prod\limits_{\{\delta_i\}\in M_N} \left(x_1^{\delta_1} \vee x_2^{\delta_{21}} \vee \cdots \vee x_N^{\delta_N}\right)$, где в левой части сумма по модулю 2, M_N – множество двоичных векторов длины N, содержащих чётное число нулей. Операция "возведения в степень" имеет стандартный для булевой алгебры смысл: $x^{\delta} = \begin{cases} \overline{x}, & \delta = 0 \\ x, & \delta = 1 \end{cases}$

Лемма 2.

$$\bigvee_{i=1}^{N} x_i^{\delta_i} \vee \prod_{j=1}^{L} y_j^{\sigma_j} = \prod_{\{\pi_k\}\in 2^L / \{0,0,\ldots,0\}} \left(\bigvee_{i=1}^{N} x_i^{\delta_i} \vee \bigvee_{j=1}^{L} \left(y_j^{\sigma_j}\right)^{\pi_k}\right)$$

После применения леммы 1 будут получены конъюнкты следующего вида:

$\left(d \vee \overline{a\overline{b}c} \vee x\overline{y}c\right)$, т.е. можно выделить 3 вида дизъюнктов внутри каждого конъюнкта:

1. Одиночные литералы (то к чему следует стремиться: в правильной КНФ все дизъюнкту должны быть одиночными литералами).

2. Дизъюнкты вида $\overline{\prod x_i^{\delta_i}}$, которые по правилу де Моргана можно легко свести к одиночным литералам.

3. Дизъюнкты вида $\prod x_i^{\delta_i}$, наиболее трудный случай, сведение скобок с такими дизъюнктами к КНФ иллюстрируется леммой 2.

Отдельного рассмотрения заслуживает операция вычисления переноса в следующий разряд при суммировании трёх слагаемых. Перенос может быть вычислен через соответствующую сумму: $c = carry\,(x, y, z, sum) = \overline{(sum \oplus xyz \oplus \overline{xyz})}$

Выполнение данного равенства эквивалентно истинности следующей формулы: $\left(\overline{c} \vee \overline{(sum \oplus xyz \oplus \overline{xyz})}\right) \cdot (c \vee (sum \oplus xyz \oplus \overline{xyz}))$.

Приведённое выражение можно преобразовать положив: $x = \overline{c}$, $A = sum$, $B = xyz$, $C = \overline{xyz}$. И далее описанной выше процедурой можно построить соответствующую КНФ. Трудоемкость полученного алгоритма оценивается, как $O(n^2)$ в зависимости от количества бит исходного числа. Для факторизации числа, представляемого двоичным вектором длиной 1024 бит получились КНФ с 500 000 переменными 12 000 000 скобок. Отметим, что результат о превышении числа верных бит после нескольких тысяч итераций алгоритма (4) относится именно к числу переменных, например, из 500 000 переменных мы получаем в среднем более чем 300 000 верных, что в силу очевидных соотношений между битами переноса по строке, отвечающей нулевому биту, может существенно облегчить решение основной задачи.

8. Сведение к КНФ других задач криптоанализа

Были получены аналогичные алгоритмы сведения для задач дискретного логарифмирования и дискретного логарифмирования на эллиптической кривой. Предложен алгоритм генерации множества эквивалентных КНФ для задачи факторизации, учитывающий неделимость на малые простые числа. Последнее обстоятельство позволяет строить параллельные версии приведенных выше алгоритмов и методикой "голосования бит" определять верные биты с достаточно большой верятностью.

Таблица 4. Результаты тестирования полного алгоритма для задачи факторизации.

Число бит	Литералы	Дизьюнкты	Время решения	Время RANOV	Время SATz
20	254	4979	0.1 с	0.1 с	0.1 с
32	801	17867	2 м	>1 ч	1.8м
40	990	22333	7 м	>1 ч	12 м
44	1199	27291	36 м	>1 ч	>1 ч
48	1428	32741	3,5 ч	>10ч	>10 ч
56	1946	45141	36 м	>10ч	>10 ч
60	2235	52079	10,2ч	>20 ч	>20 ч
68	2873	67455	79 ч	>100 ч	>100 ч
72	3222	75881	168ч	>200 ч	>200 ч

9. Результаты работы алгоритма для задачи факторизации

Результаты приведены в табл. 4. Для группы задач длины до 72 бит факторизуемого числа были получены точные решения. При этом эффективность предложенного метода превосходит известные нам алгоритмы.

Другой интересный результат в том, что уже после первых нескольких сотен итераций метод находит более 60% верных бит решения. Трудность заключается в определении, какие именно биты были определены верно. При этом при росте числа бит исходного факторизуемого числа, происходит рост процентного отношения числа верных бит для соотношения, определяющего факторизуемое число (см. Рис. 1).

Для тестирования использовалось 50 различных тестов для каждой размерности. В качестве сомножителей выбирались числа, удовлетворяющие всем тестам, гарантирующим криптостойкость RSA. На рисунке представлены усредненные данные. Обратим внимание на то, что вне зависимости от размерности задачи для каждого теста проводилось всего 1000 итераций. Исходя из рисунка можно сделать предположение о том, что рост числа верных бит в зависимости от длины факторизуемого числа имеет логарифмический характер.

Аналогичные результаты получены и для задачи дискретного логарифмирования, но основным препятствием здесь является высокая размерность получающихся задач, так для 1024 бит число переменных в функционале оценивается величиной 1000 000 000, а число дизъюнктов на порядок больше. Результаты точного решения КНФ, эквивалентных задаче дискретного логарифмирования приведены в таблице 5.

Заключение

Разработанный метод не уступает известным методам решения SAT на многих группах тестовых примеров и превосходит их на тестах задачи факторизации больших размерностей. Показан рост относительного числа верно найденных бит для задачи факторизации с ростом размерности задачи, так для рабочего числа

Рис. 1. Рост процентного отношения числа верных бит к числу бит факторизуемого числа.

Таблица 5. Результаты тестирования полного алгоритма для задачи дискретного логарифмирования.

Размерность	Литералы	Дизьюнкты	Время (сек)	RANOV (сек)	SATz (сек)
18	28224	448018	63.57	97.23	81.16
20	38840	623239	108.20	>1800	>1800
22	51832	839032	182.73	>1800	>1800
24	67440	1099630	277.46	>1800	>1800
26	85904	1409250	417.71	>1800	>1800
Знак '>' означает, что за указанное время решение найдено не было					

бит 1024 среднее значение верных бит равно 65%, что больше стартового значения в 62% для 50 бит. Кроме того, показано наличие "слабых" размерностей, для которых метод является эффективным.

Список литературы

1. *Cook S.A.* The Complexity of Theorem Proving Procedures. Proceedings Third Annual ACM Symposium on Theory of Computing, May 1971.
2. *Файзуллин Р.Т., Хныкин И.Г., Дулькейт В.И., Салаев Е.В.* Алгоритм минимизации функционала, ассоциированного с задачей 3-SAT и его практические применения, г.Челябинск, 2007.
3. *Файзуллин Р.Т.* О решении нелинейных алгебраических систем гидравлики // Сибирский журнал индустриальной математики. – 1999. -№2. – С. 176-184.
4. *Хныкин И.Г.* Модификация КНФ, эквивалентных задачам криптоанализа асимметричных шифров
методом резолюции // ИТМУ № 8, 2007.
5. *Беспалов Д.В. Семёнов А.А.* О логических выражениях для задачи 2-ФАКТОРИЗАЦИЯ // Вычислительные технологии. – 2002. – Т.7 – Ч.2.

Статья поступила в редакцию 01.05.2008