2008 Математические методы криптографии № 2(2)
DOI 10.17223/20710410/2/10
УДК 004.4

О ПРЯМОМ ОПЕРАЦИОННОМ АНАЛИЗЕ СИММЕТРИЧНЫХ ШИФРОВ

А.М. Кукарцев, А.М. Попов, В.С. Шестаков

Сибирский государственный аэрокосмический университет им. академика М.Ф. Решетнева, г. Красноярск

E-mail: amkukarcev@mail.ru, alexey_m_popov@newmail.ru, shockk@mail.ru

Рассматривается методика исследования криптографических алгоритмов и их ключей с позиции их математического аппарата. Уточняется понятие сложности криптоалгоритма как относительной оценки криптостойкости. Описывается реализация метода прямого операционного анализа.

Ключевые слова: *криптография*, *композиционные криптоалгоритмы*, *анализ криптостойкости.*

Проблема безопасности криптоалгоритмов является многогранной. В ней можно выделить организационную и техническую стороны. Ошибки могут быть заложены и в самом алгоритме. Даже алгоритмы, которые признаны стойкими, при неправильном использовании будут слабыми. Ещё более важным является правильное использование ключевой информации. Практически к каждому алгоритму поставляется перечень заведомо слабых ключей. Но слабость таких ключей заключается в том, что они не стойки к уже известным методам криптоанализа. Наибольшую актуальность проблема стойкости ключей приобретает в алгоритмах, для которых требуется частая плановая их смена. Возможно ли проанализировать алгоритм перед его использованием? Ещё более интересен вопрос: а можно ли проектировать заведомо безопасный шифр (либо дать исчерпывающий перечень рекомендаций по его безопасному использованию) и его (что немаловажно) реализацию? Ведь каждый новый алгоритм шифрования требует детального и тщательного изучения. В настоящей работе рассматривается методика детального исследования некоторых криптографических алгоритмов.

1. Метод операционного анализа и изоалгоритмы

Для любого криптоалгоритма можно определить изоморфизм, который ставит в соответствие объектам, преобразуемым алгоритмом, булевы векторы. В качестве отображения можно использовать перевод значений из различных позиционных систем счисления в двоичную. Используя изоморфизм, можно перейти от алгоритма как последовательности операций к системе булевых функций и вести анализ последней [1].

Детальное исследование полученных булевых функций позволяет анализировать основную составляющую криптостойкости – математическую сложность самих алгоритмов и их ключей. Цель данной работы – описать теоретическую часть этой методики, уточнить понятие сложности алгоритмов, реализовать средства анализа и апробировать методику на шифре простой замены – сложении чисел в двоичном позиционном коде.

Так как сам метод анализа является вычислительно сложным, то следует также рассмотреть и недостатки методики. Такие недостатки возникают при её практической реализации. В работе рассмотрены варианты, позволяющие реализовать описываемую методику анализа.

Для начала опишем основные понятия, применяемые для компактности изложения метода. На рис. 1 показана модель «чёрного» ящика рассматриваемых преобразований.

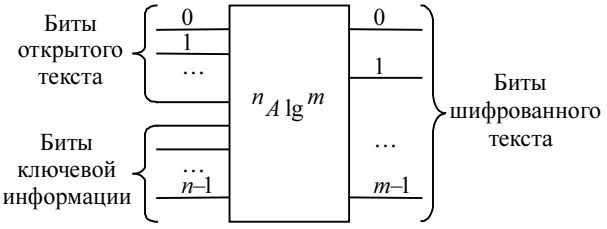

Рис. 1. «Чёрный ящик» алгоритмов шифрования

Определим следующие понятия, применяемые при описании метода. Далее будем называть *аргументом-вектором* булев вектор, который является элементом множества, изоморфного множеству входных объек-

тов. Будем называть *альфа-вектором* булев вектор, который является элементом множества, изоморфного множеству выходных объектов. Будем называть *изоалгоритмом* и обозначать ${}^{n}A\,\mathbf{lg}^{m}$ систему булевых функций, отображающих аргумент-вектор в альфа-вектор:

$$ {}^{n}A\,\mathbf{lg}^{m}:\begin{cases} y_0 = f_0(x_{n-1}, x_{n-2}, \ldots, x_0), \\ y_1 = f_1(x_{n-1}, x_{n-2}, \ldots, x_0), \\ \quad\vdots \\ y_{m-1} = f_{m-1}(x_{n-1}, x_{n-2}, \ldots, x_0). \end{cases} \tag{1} $$

Так как все изоалгоритмы имеют одинаковую математическую природу (системы булевых функций), то возможно ввести общие характеристики, по которым можно будет сравнивать различные исходные алгоритмы.

Для вычисления значения конкретного бита зашифрованного текста необходимо проделать какое-то количество логических операций, которые будут отражать вычисление значения булевой функции. Для различных булевых функций такое количество операций различно. С позиции анализа «грубой силы» необходимо минимальное количество операций для перебора возможных вариантов. Так как функции булевы, то такими операциями являются функции аристотелева базиса.

Булева функция может быть представлена в виде двух нормальных форм – дизъюнктивной и конъюнктивной. Тогда определим сложность как минимально необходимое количество логических операций для преобразования вектора-аргумента в компоненту альфа-вектора среди всех возможных КНФ и ДНФ. Единицей измерения считается операция на бит, оп/бит (operation per bit, opb).

Под сложностью всего изоалгоритма будем понимать среднее значение сложности всех булевых функций, входящих в его состав. Можно считать сложностью изоалгоритма вектор сложностей булевых функций. Так как важно значение каждого бита, то возможно проработать более эффективную методику расчёта сложности для всего изоалгоритма. Метод прямого операционного анализа основывается на непосредственном переходе от таблицы булевых векторов к системе булевых функций (1).

2. Интерпретация значений сложности

Расчёт значений сложности является технически трудной задачей. Но даже если такие значения получены, то встаёт вопрос о том, что они выражают. Проведённые аналитические исследования и практическая проверка показали, что среди форм булевых функций можно выделить идеальные нормальные формы.

Идеальная нормальная форма (*ИНФ*) – это совершенная нормальная форма, которая также является минимальной нормальной формой. Такую форму имеют, например, симметрические разности различного числа переменных.

Есть предположение, что ИНФ имеют максимальную сложность среди всех остальных булевых функций той же разрядности. Это позволяет определить *верхнюю границу* сложности булевой функции от n переменных как

$$ d_{\max} = 2^{n-1}(n+1). \tag{2} $$

Минимальную границу сложности считаем равной нулю. Такой сложности соответствуют булевы функции, тождественно равные константам (0 или 1).

Для криптографии максимальная сложность (2) не интересна для рассмотрения, так как важна сложность алгоритма с фиксированным ключом, и если алгоритм с фиксированным ключом будет иметь максимальную сложность, то ему будут соответствовать очень мало различных функций. В этом случае при известном преобразовании, тем более в форме булевых функций, построить обратное, не зная ключа, не составляет труда. Малая сложность также плоха, так как не маскирует зависимость выходных бит от входных. Таким образом, можно принять, что *оптимальная сложность* находится посредине между максимальной и минимальной и определяется как

$$ d_{\text{opt}} = 2^{n-2}(n+1). \tag{3} $$

3. Прямой операционный анализ шифра простой замены – сложения чисел заданной разрядности

В качестве примера операционного анализа алгоритмов преобразований рассмотрим алгоритм классического поразрядного сложения с переносом. Тогда под вектором-аргументом будем понимать пару слагаемых, причём первая половина соответствует первому слагаемому, вторая – второму (ключу). Старшинство бит в половинах вектора-аргумента соответствует старшинству бит в слагаемом. Так как присутствует единица переноса, то длина альфа-вектора будет равна половине длины аргумента-вектора плюс единица.

Для эксперимента возьмём длины векторов аргументов 2, 4, 6, 8 и 10 бит. Таким длинам соответствуют алгоритмы сложения: ${}^{1+1}A\,\mathbf{lg}^{1+1}_{(\mathbf{mod}\,2)}$, ${}^{2+2}A\,\mathbf{lg}^{2+1}_{(\mathbf{mod}\,4)}$, ${}^{3+3}A\,\mathbf{lg}^{3+1}_{(\mathbf{mod}\,8)}$, ${}^{4+4}A\,\mathbf{lg}^{4+1}_{(\mathbf{mod}\,16)}$ и ${}^{5+5}A\,\mathbf{lg}^{5+1}_{(\mathbf{mod}\,32)}$. В значениях

длин аргументов-векторов выделена ключевая и информационная часть, в значении длин альфа-векторов – единица переноса.

Также проанализируем изменение сложности при фиксации ключевых бит (указание ключа). Это позволит анализировать слабые и сильные ключи.

Важно отметить, что изоалгоритм, соответствующий алгоритму шифрования и имеющий сложность, близкую к оптимальной, не может считаться безопасным. Это связано с тем, что при фиксации бит ключа сложность может изменяться достаточно сильно. Таким образом, если рассматривать алгоритм совместно с конкретным ключом, то ключ можно считать настройкой алгоритма. Ключ перестраивает внутренние связи алгоритма, и соответственно получается преобразователь с длиной вектора-аргумента соответствующей разности общей длины и ключа. По этой же длине необходимо оценивать верхнюю и оптимальную границы сложности.

На базе компонентов CryptoLab была разработана утилита operationAnalyzerSum, которая анализирует сложение. Данная утилита базируется на системе классов CryptoLab. Система классов CryptoLab содержит расширение для средств операционного анализа. Эти средства представлены реализацией алгоритма «Вариационное дерево»[1], алгоритма расчёта сложности и алгоритма фиксации ключевых бит.

Общее время работы утилиты при расчёте сложностей с возможностью фиксации ключа составило 27 мин на машине класса PC AMD Athlon 2500+, RAM 512 Мбайт, Microsoft Windows XP Professional выпуска апреля 2005 г. Исходные данные анализа – значение n. Утилита последовательно перебирает все целые значения от 2 до n включительно. Утилита формирует также основной протокол работы и выводит данные об изоалгоритмах в промежуточные файлы. Создаются как текстовые файлы, так и бинарные. Результаты работы утилиты показаны на графиках.

На рис. 2 представлены значения сложности алгоритма шифрования, на рис. 3 – значения средней сложности по ключам.

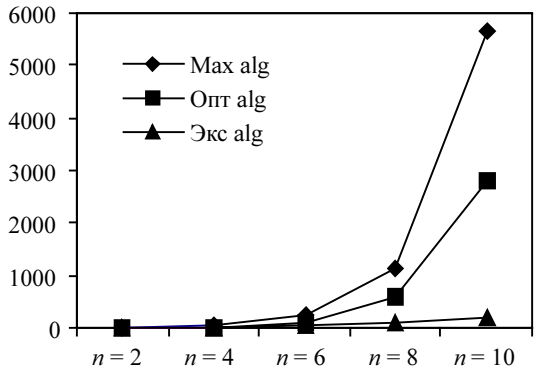

Рис. 2. Значения сложности алгоритма

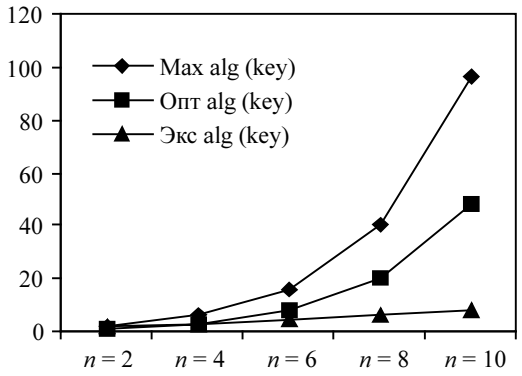

Рис. 3. Средние значения сложности алгоритма с фиксированными ключами

Для проведения анализа сложности с фиксированным ключом использовались средние оценки. Возможность применения средних оценок объясняется тем, что если присутствуют ключи, дающие требуемое значение сложности, но их мало (в процентном отношении к общему числу ключей), то их нельзя использовать. Это связано с тем, что криптоаналитик, скорее всего, сначала проверит именно очень сильные (сложность, близкая к максимальной) ключи и очень слабые ключи (сложность, близкая к нулю). Таким образом, интересно большое множество ключей, которые и попадают под среднюю оценку.

Из рис. 2 и 3 можно сделать вывод о том, что при увеличении длины вектора-аргумента средняя сложность изоалгоритма как с ключом, так и без него резко падает по сравнению с оптимальной границей. Средняя сложность с ростом длины аргумента-вектора растёт почти линейно, в отличие от оптимальной границы.

Таким образом, сложность можно использовать для анализа криптографических алгоритмов и производить следующие сравнения и оценки:

- сопоставлять различные алгоритмы с различной длиной ключа;
- сопоставлять различные ключи в рамках одного алгоритма;
- оценка сложности в зависимости от длины ключа;
- оценка эффективности использования композиций алгоритмов.

Актуальна, однако, проблема возможности анализа для больших длин векторов-аргументов. Так, например, утилита, использованная в эксперименте, не закончила анализ исходного преобразования с длиной вектора-аргумента, равной 12 бит, за сутки.

4. Возможность операционного анализа

Операционный анализ позволяет исследовать достаточно большой класс алгоритмов. Но тогда зададимся вопросом о том, насколько возможен в принципе такой анализ.

Рассмотрим методику расчёта сложности. Так как для расчёта сложности необходима система булевых функций (1), то необходимы все значения альфа-векторов. Определим соотношения, связывающие количество строк таблицы альфа-векторов, полный её объём и примерное время для её построения. Пусть τ_0 – время, необходимое для преобразования одного аргумента-вектора в альфа-вектор (фактически, это есть время зашифрования одного блока открытого текста), τ – время построения таблицы, c – количество строк в таблице, V – полный объём таблицы в битах. Тогда

$$c = 2^n, \ \tau = c\tau_0 = 2^n\tau_0. \tag{4}$$

Если принять во внимание, что перебираются все значения аргумента-вектора, то можно упорядочить таблицу и убрать из неё данные об аргументе-векторе. Тогда объём таблицы будет

$$V = cm = 2^n m. \tag{5}$$

Так, для DES по (5) имеем $V = 2^{120} \times 2^6 = 2^{126}$ бит. То есть расчёт сложности для DES технически невозможен уже потому, что негде хранить экспериментальный материал, необходимый для анализа. Даже если его удастся разместить, то время, необходимое на генерацию данных по (4) при условии, что шифрование одного блока реализуется одной (хотя в действительности их несколько сотен) операцией для процессора на частоте 2,5 ГГц, составит $\tau = 2^{120}/(2{,}5\cdot 10^9\ \text{Гц}) \approx 10^{26}$ с $> 10^{18}$ лет.

Будем называть *прямым операционным анализом* расчёт сложности через построение таблицы альфа-векторов. Любой алгоритм с шириной входа больше 50 бит технически не может быть подвергнут прямому операционному анализу. Следовательно, необходимы пути обхода этой границы.

Большинство алгоритмов шифрования не являются монолитными преобразованиями и, следовательно, могут быть представлены как композиции более мелких (с меньшей сложностью и меньшими длинами аргументов-векторов и альфа-векторов) преобразований. Так, например, DES может быть представлен как композиция преобразований, соответствующих одному раунду. В свою очередь, раунд DES также может быть представлен как композиция более мелких преобразований – сумм, сдвигов, таблиц подстановки и перестановки, симметрической разности. Аналогично может быть представлен алгоритм шифрования ГОСТ 28147-89.

Разбиение алгоритма на блоки удобно делать, если получаются преобразования со сравнительно малой длиной вектора-аргумента. Здесь и далее под длиной будем понимать длину вектора-аргумента, так как многие технические характеристики зависят экспоненциально от длины вектора-аргумента, а от длины альфа-вектора линейно. Например, в алгоритмах DES и ГОСТ 28147-89 длины S-блоков будут равны соответственно 6 и 4 бит. Преобразования такой длины уже поддаются прямому операционному анализу.

Изоалгоритмы некоторых преобразований могут быть получены непосредственно из их записи. В частности, такими преобразованиями являются таблицы перестановки бит. Таблицы перестановки в большинстве случаев используются в симметричных блочных криптоалгоритмах (DES, AES, ГОСТ 28147-89). Фактически, булевы функции таких преобразований не будут содержать никаких операций, а будут только присваивать функции тот или иной компонент вектора-аргумента. Будем называть *транспозиционным операционным анализом* расчёт сложности, который производится в булевых функциях, полученных из таблиц перестановок.

Изоалгоритмы, соответствующие некоторым элементарным преобразованиям для больших длин, могут быть получены путём экстраполяции тех же преобразований с малыми длинами. Так, например, сложение 4-битных чисел с единицей переноса может быть получено из сложения 2-битных с единицей переноса по классической пирамидальной схеме соединения сумматоров [2, с. 332]. Отдельно стоит выделить побитовые операции, например побитовую симметрическую разность. Булевы функции таких преобразований полностью повторяют друг друга с точностью до индексов компонент вектора-аргумента. Будем называть *экстраполяционным операционным анализом* расчёт сложности, который производится в булевых функциях, полученных из аналогичных булевых функций малой длины.

Если хранить булевы функции как композицию ИНФ и выполнять все операции в такой форме, то проблема хранения и обработки решается. Это позволяет в ряде случаев осуществлять операционный анализ практически.

Выводы

1. Произведено уточнение понятий сложности криптоалгоритма и операционного анализа.
2. Определены границы и формулы расчёта минимальной, максимальной и оптимальной сложности.
3. Произведён прямой операционный анализ шифра простой замены – сложения по модулю степени 2 заданной разрядности. Представлены значения сложности алгоритма, в том числе с фиксированными ключами.

4. Показана практическая невозможность прямого операционного анализа произвольного преобразования.

5. Введено понятие идеальной нормальной формы, позволяющее обойти алгоритмическую трудность прямого анализа частными методами – транспозиционными, экстраполяционными и композиционными.

ЛИТЕРАТУРА

1. *Кукарцев А.М.*, *Старовойтов С.А.*, *Шестаков В.С.* Операционный анализ криптоалгоритмов // Актуальные проблемы безопасности информационных технологий: Материалы I Междунар. науч.-практич. конф. (Красноярск, 27 марта 2007 г.) / Под общ. ред. О.Н. Жданова, В.В. Золоторёва. Красноярск: Сиб. гос. аэрокосмич. ун-т, 2007. С. 66 – 72.
2. *Титце У.*, *Шенк К.* Полупроводниковая схемотехника: Справочное руководство. М.: Мир, 1982. 512 с.