**Лекция 9.**  Алгебраический анализ стойкости криптографических алгоритмов. Булевы функции.

**Алгебраический анализ стойкости криптографических алгоритмов.**

Задача анализа надежности используемых криптографических алгоритмов является одним из актуальных направлений в информационной безопасности. При выборе алгоритма для анализа стойкости авторы руководствовались следующими соображениями:

* Стандарт симметричного шифрования ГОСТ 28147-89 используется в большинстве российских средств защиты конфиденциальной информации.
* Алгоритм ГОСТ 28147-89 рассматривается в качестве международного стандарта шифрования в ISO 18033.

Алгоритм шифрования ГОСТ 28147-89 в режиме простой замены представляет собой 32 раунда зашифрования, построенного по принципу сети Фейстеля. Длина блока открытого текста (Т) и шифротекста (С) равна 64 бита (8 байт), секретный ключ шифрования (К) -  случайная последовательность длиной 256 бит. Блок открытого текста разбивается на две равные части по 32 бита каждая. Над правой частью открытого текста выполняется раундовое преобразование (F), состоящее из трех операций:

* Сложение с раундовым ключом по модулю 232;
* Замена в восьми секретных S-блоках;
* Циклический сдвиг влево на 11 позиций.

Левая часть открытого текста складывается по модулю два с результатом раундового преобразования. После чего производится обмен местами правой и левой частей текстов. Схема алгоритма шифрования ГОСТ 28147-89 приведена на рис. 1.
Раундовые ключи шифрования вычисляются из исходного секретного ключа путем разбиения его на восемь 32-битных блоков: K1, К2,К3, К4, К5, К6, К7, К8. С 1 по 24 раунд ключи используются в прямом порядке: К1, К2, К3, К4, К5, К6, К7, К8, К1, К2, К3, К4, К5 и так далее. С 25 по 32 раунды  ключи берутся в обратном порядке: К8, К7, К6, К5, К4, К3, К2, К1.


Рисунок 1 - Алгоритм шифрования ГОСТ 28147-89

Российский стандарт симметричного шифрования ГОСТ 28147-89 является стойким к большинству криптографических атак, например, методу полного перебора на ключевом пространстве, дифференциальному и линейному криптоанализам. В тоже время существует вероятность, что алгоритм ГОСТ 28147-89 может быть уязвим к алгебраическим атакам. Опираясь на методы алгебраического взлома алгоритма Advanced Encryption Standard, проведен анализ возможности применения алгебраических методов криптоанализа для взлома ГОСТ 28147-89, в частности в данной статье рассмотрен метод Extended Linearization (XL).

Учитывая, что алгебраические атаки в основе своей используют представление нелинейных преобразований шифрования в виде системы уравнений, необходимо знать таблицы замен. Блоки замены, используемые в конкретной реализации алгоритма ГОСТ 28147-89, являются дополнительным секретным элементом.. Метод основан на использовании «накрывающего» свойства сети Фейстеля. Данное свойство заключается в том, что при идентичных раундах шифрования прохождение текста через четное число раундов сети Фейстеля повлечет изменение только половины выходного блока (шифротекста). Для соблюдения требования идентичности раундов используется нулевое значение секретного ключа (атака на выбранных ключах), при этом все раундовые ключи будут также равны нулю. Первый этап атаки -  нахождение нулевого вектора (z), который равен раундовому преобразованию шифрования от нулевого значения z = F(0). Второй этап - восстановление таблицы блока замены по «накрывающему» свойству. Для алгоритма
ГОСТ 28147-89 атака требует выполнения не более 232 операций зашифрования для однозначного определения таблиц замены.

Рассмотрим один раунд алгоритма ГОСТ. Необходимо составить систему уравнений для 8-ми параллельно используемых S-блоков. Выполним составление уравнений для одного блока, заданного таблицей 1. Аналогичным образом выполняет поиск линейно независимых уравнений для оставшихся 7 блоков замены.

Таблица 1. Таблица замены S-блока

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| x | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| y=S(x) | 4 | 10 | 9 | 2 | 13 | 8 | 0 | 14 | 6 | 11 | 1 | 12 | 7 | 15 | 5 | 3 |

Сначала составляем все уравнения вида (1). Число таких уравнений равно 237=137438953472, число одночленов в них – 37, число переменных – 8.

  (1)
Затем выполним выбор уравнений, верных для исследуемого S-блока, для этого составлена таблица истинности, представленная на рис. 2.

Для данного S-блока верными оказались 2097151 уравнений. Из них можно выбрать ≈37-24=21 линейно независимых уравнений. Предположим, что получено минимально возможное число линейно независимых уравнений – 21. Для решения системы обратимся к алгоритму метода eXtended Lineranization. Вычислим параметр d. Так как отношение , то принимаем d=3. Тогда уравнения системы умножаются на одночлены в первой степени: {x1, x2, x3, x4, y1, y2, y3, y4}. Следовательно, получим 21∙8=168 дополнительных уравнений. Результирующая система будет содержать 189 уравнений, 75 одночленов, которые после приведения к линейному виду рассматриваются как новые переменные.


Рисунок 2 – Таблица истинности для исследуемого блока замены

 Таким образом, для одного раунда должна быть получена система из 8∙189=1512 уравнений, связывающая вход и выход блока замены. Число переменных составит 64, число одночленов в данной системе равно 75∙8=600. Даже если часть уравнений, полученная после умножения, окажется линейно зависимой, оставшихся уравнений будет достаточно для решения методом линеаризации. После составления системы нужно перейти от рассмотрения входов и выходов блока замены к раундовым ключам. Для этого представим i-тый бит входного значения блока замены в виде суммы по модулю 2 бита правой части открытого текста и раудового ключа. Исходя из структуры одного раунда ГОСТ, выразим выход блока замены через известные данные по формуле (2).

(2)
 Таким образом, при работе системой для одного раунда число переменных в нелинейной системе можно сократить в два раза, так как выходы блока замены будут однозначно определены.

При исследовании полнораундного алгоритма ГОСТ система уравнений второго порядка до применения метода XL будет содержать 21∙8∙32=5376 квадратных уравнения, 32∙64=2048 переменных и 37∙8∙32=9472 одночленов. В результате умножения системы на одночлены в первой степени получим систему с 48384 кубическими уравнениями и 19200 одночленами. В первом раунде замена входных битов блоков замены будет аналогична атаке на однораундовую версию, а выход блока замены останется без изменения неизвестным y1,i. Во втором раунде, используя свойство сети Фейстеля, входные биты S-блока будут представлены формулой (3).
(3)
Выход блока также задается неизвестным y2,i. В последующих раундах связь входных битов блока замены и открытого текста задана формулой (4).

 (4)
Для последнего раунда ГОСТ можно выполнить замену по формулам (5) и (6).
(5)
(6)

## **Булевы функции**

В курсе математического анализа изучаются функции, определённые на числовой прямой или на отрезке числовой прямой или на (гипер-) плоскости и т.п. Так или иначе область определения – *непрерывное* множество. В курсе дискретной математики изучаться должны функции, область определения которых – *дискретное* множество[\*](http://www.univer.omsk.su/departs/compsci/kursi/disc/bools.htm). Простейшим (но нетривиальным) таким множеством является множество, состоящее из двух элементов.[\*](http://www.univer.omsk.su/departs/compsci/kursi/disc/bools.htm) Так мы и приходим к понятию булевой функции.

**Определение 1 (Булева функция).** *Булевой функцией* от *n* аргументов называется функция *f* из *n*-ой степени множества { 0, 1 } в множество { 0, 1 }.

Иначе говоря, булева функция - это функция, и аргументы и значение которой принадлежит множеству {0,1}. Множество {0,1} мы будем в дальнейшем обозначать через *B*.

Булеву функцию от *n* аргументов можно рассматривать как *n*-местную алгебраическую операцию на множестве *B*. При этом алгебра *<B;*W>, где W – множество всевозможных булевых функций, называется *алгеброй логики*.

Конечность области определения функции имеет важное преимущество – такие функции можно задавать перечислением значений при различных значениях аргументов. Для того, чтобы задать значение функции от *n* переменных, надо определить значения для каждого из 2*n* наборов. Эти значения записывают в таблицу в порядке соответствующих двоичных чисел. В результате получается таблица следующего вида:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***x*1**  | ***x*2**  | **...**  | ***xn-*1**  | ***xn***  | ***f***  |
| 0  | 0  | ...  | 0  | 0  | f(0,0,...,0,0)  |
| 0  | 0  | ...  | 0  | 1  | f(0,0,...,0,1)  |
| 0  | 0  | ...  | 1  | 0  | f(0,0,...,1,0)  |
| 0  | 0  | ...  | 1  | 1  | f(0,0,...,1,1)  |
| ... | ... | ...  | ... | ... | ...  |
| 1  | 1  | ...  | 0  | 0  | f(1,1,...,0,0)  |
| 1  | 1  | ...  | 0  | 1  | f(1,1,...,0,1)  |
| 1  | 1  | ...  | 1  | 0  | f(1,1,...,1,0)  |
| 1  | 1  | ...  | 1  | 1  | f(1,1,...,1,1)  |
|  |  |  |  |  |  |

Раз у нас есть стандартный порядок записывания наборов, то для того, чтобы задать функцию, нам достаточно выписать значения *f*(0,0,...,0,0)*, f*(0,0,...,0,1)*, f*(0,0,...,1,0)*, f*(0,0,...,1,1),..., *f*(1,1,...,0,0)*, f*(1,1,...,0,1)*, f*(1,1,...,1,0)*, f*(1,1,...,1,1). Этот набор называют *вектором значений функции*.

Таким образом, различных функций *n* переменных столько, сколько различных двоичных наборов длины 2*n*[\*](http://www.univer.omsk.su/departs/compsci/kursi/disc/bools.htm). А их 2 в степени 2*n*.

Множество *B* содержит два элемента – их можно рассматривать как булевы функции от нуля (пустого множества) переменных – *константу 0* и *константу 1*.

Функций от одной переменной четыре: это константа 0, константа 1, *тождественная функция*, т.е. функция, значение которой совпадает с аргументом и так называемая функция ``*отрицание*''. Отрицание будем обозначать символом *¬* как унарную операцию. Приведём таблицы этих четырёх функций:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***x***  | **0**  | ***x***  | ***¬ x***  | **1**  |
| **0**  | 0  | 0  | 1  | 1  |
| **1**  | 0  | 1  | 0  | 1  |
|  |  |  |  |  |

Как видим, функции от некоторого числа переменных можно рассматривать как функции от большего числа переменных. При этом значения функции не меняется при изменении этих ``добавочных'' переменных. Такие переменные называются *фиктивными*, в отличие от остальных – *существенных*.

**Определение 2 (Фиктивные и существенные переменные).** Переменная *xi* называется *фиктивной* (несущественной) переменной функции *f*(*x*1*,···,xn*), если

*f*(*x*1*,···,xi-*1,0*,xi+*1*,···,xn*) *= f*(*x*1*,···,xi-*1,1*,xi+*1*,···,xn*)

для любых значений *x*1*,···,xi-*1*,xi+*1*,···,xn*. Иначе переменная *xi* называется *существенной*.

Функций от двух аргументов шестнадцать. Наиболее употребимые из этих функций (только те, которые существенно зависят от обеих переменных) мы приводим в следующей таблице:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ***x*1** | ***x*2** | ***x*1 *& x*2**  | ***x*1 Ъ *x*2**  | ***x*1 Й*x*2**  | ***x*1 Е *x*2**  | ***x*1 є *x*2**  | ***x*1 *| x*2**  |
| **0**  | **0**  | 0  | 0  | 1  | 0  | 1  | 0  |
| **0**  | **1**  | 0  | 1  | 1  | 1  | 0  | 1  |
| **1**  | **0**  | 0  | 1  | 0  | 1  | 0  | 1  |
| **1**  | **1**  | 1  | 1  | 1  | 0  | 1  | 1  |
|  |  |  |  |  |  |  |  |

Эти функции записываются как бинарные операции в инфиксной нотации. *x*1 *& x*2 называется *конъюнкцией*, *x*1 Ъ *x*2 – *дизъюнкцией*, *x*1 Й *x*2 – *импликацией*, *x*1 є *x*2 – *эквивалентностью*, *x*1 Е *x*2 – *суммой по модулю 2*, *x*1 *| x*2 – *штрихом Шеффера*.

Значения 0 и 1 часто интерпретируют как ``ложь'' и ``истину''. Тогда понятным становится название функции ``отрицание'' – она меняет ``ложь'' на ``истину'', а ``истину'' на ``ложь''. Отрицание читается как ``не''. Конъюнкция читается обычно как ``и'' – действительно, конъюнкция равна 1 тогда и только тогда, когда равны 1 и первая *и* вторая переменная.[\*](http://www.univer.omsk.su/departs/compsci/kursi/disc/bools.htm) Кроме *x*1*& x*2 часто используют обозначение *x*1 Щ *x*2 или *x*1 *· x*2 или *x*1*x*2 или min(*x*1*,x*2). Дизъюнкция читается ``или'' – дизъюнкция равна 1 тогда и только тогда, когда равны 1 первая *или* вторая переменная.[\*](http://www.univer.omsk.su/departs/compsci/kursi/disc/bools.htm) Импликация выражает факт, что из *x*1 следует *x*2.[\*](http://www.univer.omsk.su/departs/compsci/kursi/disc/bools.htm) Импликацию часто также обозначают *x*1 ® *x*2.