УДК 004.056.5 **DOI:10.30837/rt.2020.1.200.07**

К.Е. ЛИСИЦКИЙ, А.А. КУЗНЕЦОВ, д-р техн. наук

ВЫЧИСЛИТЕЛЬНЫЕ АЛГОРИТМЫ РАСЧЕТА АЛГЕБРАИЧЕСКОГО ИММУНИТЕТА НЕЛИНЕЙНЫХ УЗЛОВ ЗАМЕНЫ СИММЕТРИЧНЫХ ШИФРОВ

1. Введение

Современный мир информационных технологий, в котором постоянно обмениваются информацией, в том числе конфиденциальной, информацией с ограниченным доступом, секретной и т.д., трудно представить без использования криптографических преобразований, таких как алгоритмы шифрования, криптографические протоколы, алгоритмы хеширования, алгоритмы электронной цифровой подписи и пр. Важное место среди алгоритмов шифрования занимают симметричные шифры. Благодаря своей простоте, скорости, они нашли широкий спектр применения во многих областях человеческой деятельности. Поэтому изучение и анализ симметричных криптосистем, а также их отдельных компонентов, таких как линейные и нелинейные преобразования и структуры смешивания блоков данных, безусловно, является актуальной научно-технической задачей. Наряду с совершенствованием методов проектирования и разработки симметричных блочных шифров рассматриваются проблемы совершенствования методов их криптоанализа и, в частности, вопросы дальнейшего повышения показателей их стойкости.

Исследования последних лет в этом направлении характеризуются повышенным интересом к алгебраическим методам криптоанализа.

Основная идея построения алгебраических атак основана на поиске возможности описания шифрования с использованием системы уравнений, позволяющей связать биты открытого текста, ключа и шифртекста. Стойкость к алгебраическим атакам во многих работах связывается с показателем алгебраической иммунности входящих в большинство шифров S-блоков (нелинейных узлов замены). Показатель алгебраической иммунности (алгебраического иммунитета) связан со степенью одночлена аннигилирующего полинома к булевой функции (векторной булевой функции), который определяет возможность понижения начального порядка системы уравнений (их числа) с помощью специально разработанного для этого математического аппарата.

Как показывает анализ, вычисление алгебраического иммунитета является нетривиальной задачей. Это связано с решением системы уравнений большой размерности, требующим значительных вычислительных ресурсов и объема памяти, которые уже для байтовых S-блоков выходят за рамки практически приемлемых затрат.

2. Цели и задачи работы

Задача состоит в том, чтобы разработать ускоренный алгоритм расчета АИ, на основе метода Арса – Фожера. Алгоритм должен вычислять АИ байтовых S-box за приемлемые временные сроки, в отличие от неоптимизированного алгоритма, при этом используя минимум ресурсов ОЗУ на ПК. Это будет универсальный алгоритм, позволяющий вычислять АИ для S-блока как можно большей степени, используя минимум временных затрат и ресурсов ОЗУ на ПК.

3. Обзор литературы

Интерес, возникший в свое время к алгебраическим методам криптоанализа блочных и потоковых шифров в 2003 году благодаря работам Н. Куртуа и В. Майера [1], не ослабевает даже сегодня. Результатом повышенного внимания к этим методам стало появление нового криптографического показателя пригодности S-блоков, показателя в виде алгебраического

иммунитета. В [2] следует напомнить, что понятие алгебраического иммунитета для булевых функций было введено в 2004 году В. Мейером, Э. Пасаликом и К. Карле в [3].

Алгебраическим иммунитетом *AI*(*f*) (АИ) булевой функции *f* называется минимальное число *d* такое, что существует булева функция *g* степени *d*, не тождественно равная нулю, для которой $fg = 0$ или $(f \oplus 1)\ g = 0$. Для любой булевой функции выполняется условие $d \le \lfloor n/2 \rfloor$, и существуют функции, имеющие $d \le \lfloor n/2 \rfloor$. Далее приведем еще одну выдержку из [2].

Понятие алгебраического иммунитета различными способами было обобщено на случай векторной булевой функции. Так, в [5] Ф. Армкнехт и М. Краузе, а также Г. Арс и Ж.-Ш. Фожер в [6] рассмотрели алгебраический иммунитет S-блоков и ввели понятия базового *AI*(*F*) и графического *AIgr*(*F*) алгебраического иммунитета векторных булевых функций. Базовый алгебраический иммунитет используется в потоковых шифрах. Графический алгебраический иммунитет используется для изучения устойчивости к алгебраическим атакам блочных шифров. Следующим обобщением, которое принято многими исследователями как одно из наиболее естественных с криптографической точки зрения, является компонентная алгебраическая иммунность $AI_{\text{comp}}(F)$. Компонентный алгебраический иммунитет AIcomp (F) векторной булевой функции F: $Z_2^n \to Z_2^m$ называется минимальным алгебраическим иммунитетом компонентных функций $b\ F$ ($b \in Z_2^m$, $b \neq 0$) i.e., $AI_{\text{comp}}(F) = \min\{AI(b\ F) : b \in Z_2^m, b \neq 0\}$,, where $b\ F = b_1 f_1 \oplus \ldots \oplus b_m f_m$ [7]. В случае компонентного алгебраического иммунитета было также получено в [6], что $AI_{\text{comp}}(F) \le \lfloor n/2 \rfloor$.

Другой метод определения алгебраической иммунности нелинейных узлов замены по Арсу – Фожеру построен с использованием базисов Грёбнера. Он разработан в [9] и основан на утверждении, приведенном ниже.

Утверждение. Пусть $s : V_n \to V_n$ будет булево отображение (S-блок) с координатной функцией $s_1, \ldots, s_n$. Рассмотрим булеву функцию $s : V_n \to V_n$ $f_s : V_{2n} \to \{0,1\}$, которая определяется с помощью следующего отношения:

$$f_s(x, y) = 1, \text{ если } s(x) = y;\ f_s(x, y) = 0 - \text{в противном случае } x, y \in V_n.$$

Тогда алгебраический иммунитет отображения s (по Арсу – Фожеру) совпадает с минимальной степенью ненулевых многочленов, принадлежащих аннулятору функции $f_s : AI(s) = \min \deg A_{nn}(f_s)$.

Следует отметить, что результаты расчета алгебраического иммунитета S-блоков в методе Арса – Фожера [7] и в расчетах компонентного алгебраического иммунитета [8] не совпадают. Далее мы рассмотрим наиболее правильный метод, при котором можно однозначно построить аннигилирующий полином меньшей степени. Это метод Арса – Фожера, основанный на построении базисов Грёбнера.

В [9] эта проблема решается с помощью стороннего программного обеспечения, а именно, программного пакета приложений Magma [10], который реализует широкий спектр функций, связанных с алгеброй, теорией групп, кольцами и полями, теорией чисел и многими другими разделами математики. Используется стороннее программное обеспечение, так как вычисления таких больших систем уравнений требуют значительных временных затрат. В нашей работе будет представлена оптимизированная реализация метода вычисления АИ, которая позволит нам получить результат в приемлемые сроки.

4. Материалы и методы. Вычисление алгебраической иммунности булевого отображения

Прежде чем привести алгоритм вычисления алгебраической иммунности, введем некоторые обозначения. Следуя работе [4]:

Пусть $GF(2)$ – двоичное поле и $GF(2)^n$ – n-мерное векторное пространство над $GF(2)$.

Булева функция $f(x)$ от n переменных – это отображение $f(x): GF(2)^n \to GF(2)$, где $x = (x_1, ..., x_n)$.

Таблица истинности булевой функции $f(x)$ от n переменных – это двоичный выходной вектор значений функции, который содержит 2^n элементов, каждый элемент принадлежит множеству {0, 1}.

Алгебраическая нормальная форма (*полином Жегалкина*) булевой функции $f(x)$ от n переменных записывается в виде

$$f(x) = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus ... \oplus a_n x_n \oplus a_{12} x_1 x_2 \oplus a_{13} x_1 x_3 \oplus ... \oplus a_{(n-1)n} x_{n-1} x_n \oplus ... \oplus a_{123...n} x_1 x_2 x_3 ... x_n,$$

где коэффициенты $a_i \in \{0,1\}$ и каждая булева функция реализуются полиномом Жегалкина единственным образом, т.е. каждое представление $f(x)$ соответствует уникальной таблице истинности.

Алгебраическая степень $Deg(f)$ булевой функции $f(x)$ – число переменных в самом длинном слагаемом алгебраической нормальной формы функции, имеющем ненулевой коэффициент a_i. При этом считаем $Deg(0) = 0$.

Обозначим через V_n множество всех отображений $GF(2)^n \to GF(2)$, т.е. это множество всех возможных булевых функций $f(x)$ от *n* переменных.

Множество V_n будем рассматривать и как кольцо булевых функций и как векторное (линейное) пространство над двоичным полем, т.е. $V_n = GF(2)^{2^n}$.

Булева функция $g \in V_n$ называется *аннигилятором* функции $f \in V_n$, если

$$f \cdot g = 0$$

или

$$(f + 1) \cdot g = 0.$$

Множество различных аннигиляторов булевой функции $g(x)$ образует линейное пространство, которое обозначим

$$Ann(f) = \{g \in V_n \mid f \cdot g = 0\}.$$

Линейное пространство аннигиляторов степени $\le d$ обозначим

$$A_d^n(f) = \{g \in V_n \mid f \cdot g = 0, Deg(g) \le d\} \subset Ann(f).$$

Понятие аннигиляторов булевых функций тесно связано с оценкой эффективности алгебраического криптоанализа поточных шифров [1]. В частности, при использовании фильтрующего генератора (см. рис. 1) псевдослучайных последовательностей (ПСП) поиск начального состояния регистра сдвига с линейной обратной связью (РСЛОС) сопряжен с понижением степени совместной системы полиномиальных булевых уравнений.

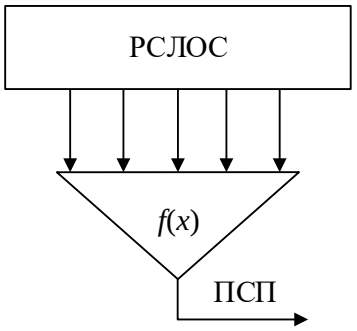

Рис. 1. Структурная схема фильтр-генератора ПСП

Алгоритм алгебраического криптоанализа, предложенный в [1], позволяет, при определенных условиях, по части перехваченной выходной последовательности (ПСП) находить начальное состояние РСЛОС с временно*й* сложностью $O\left((S_n^d)^3\right)$, где

$$S_n^d = \sum_{i=0}^{d} \frac{n!}{i!(n-i)!}$$

и d – наименьшая степень ненулевого аннигилятора фильтрующей булевой функции $f(x)$ или ее инверсии $f(x)+1$.

Таким образом, задачей алгебраического криптоанализа является поиск ненулевых аннигиляторов или, по крайней мере, оценка их минимальной степени. С этой целью в работе [3] введено определение *алгебраической иммунности* $AI(f)$ булевой функции $f \in V_n$ в виде

$$AI(f) = \min\{Deg(g) \mid g \in Ann(f) \text{ или } g \in Ann(f+1)\}.$$

Величина $AI(f)$ численно равна минимальной степени такой булевой функции $g \in V_n$, что $f \cdot g = 0$ или $(f+1) \cdot g = 0$.

Используя введенное выше понятие линейного пространства аннигиляторов степени $\le d$, запишем:

$$AI(f) = \min\{d \mid A_d^n(f) \neq 0 \text{ или } A_d^n(f+1) \neq 0\}, \quad (1)$$

т.е. для оценки алгебраической иммунности булевой функции $f \in V_n$ достаточно найти ненулевой базис пространства аннигиляторов наименьшей степени d.

Величина d позволяет количественно оценить сложность алгебраического криптоанализа и, при достаточно большом d, гарантировать устойчивость поточного криптоалгоритма к алгебраической атаке.

Алгоритм вычисления алгебраической иммунности булевых функций. Один из алгоритмов расчета алгебраической иммунности булевых функций представлен в диссертационной работе [11]. Он основан на построении базиса линейного пространства аннигиляторов $A_d^n(f)$ заданной степени d. Итеративно увеличивая d и повторяя построение базиса пространства $A_d^n(f)$, оценку $AI(f)$ получим по формуле (1), т.е. через ненулевой базис аннигиляторов наименьшей степени.

Для изложения сути алгоритма необходимо ввести следующие дополнительные обозначения.

Моном (одночлен) относительно переменных $x_1,...,x_n$ будем записывать в виде

$$x^u = \prod_{i=1}^{n} x_i^{u_i} = \begin{cases} x_i, u_i = 1, \\ 1, u_i = 0, \end{cases}$$

где $x, u \in V_2^n, x = (x_1,...,x_n), u = (u_1,...,u_n)$ – векторы переменных

Степень одночлена x^u определяется весом Хемминга (числом ненулевых координат) $w_h(u)$ вектора $u = (u_1,...,u_n)$, т.е.

$$Deg(x^u) = w_h(u).$$

С учетом этих обозначений булеву функцию $f(x)$ в алгебраической нормальной форме (в форме полинома Жегалкина) запишем в виде

$$f(x) = \sum_{u \in GF(2)^n} a_u x^u, \; a_u \in GF(2). \quad (2)$$

Функцию (аннигилятор) $g \in A_d^n(f)$ также представим в виде полинома Жегалкина

$$g(x) = \sum_{v \in GF(2)^n : w_h(v) \le d} b x \ , \tag{3}$$

где $b_v \in GF(2)$ – неизвестные коэффициенты аннигилятора, $w_h(v)$ – вес Хемминга вектора $v = (v_1, ..., v_n)$.

Функция g принадлежит пространству аннигиляторов $A_d^n(f)$ только в том случае, если для любого $x \in GF(2)^n$ выполняется равенство $f(x) \cdot g(x) = 0$.

Подставив (2) и (3) получим:

$$f(x) \cdot g(x) = \left(\sum_{u \in GF(2)^n} a_u x^u \right) \left(\sum_{v \in GF(2)^n : w_h(v) \le d} b_v x^v \right) = \sum_{u \in GF(2)^n} \left(\sum_{v \in GF(2)^n : w_h(v) \le d} a_u b_v x^{u \vee v} \right) = 0,$$

где $u \vee v = (u_1 \vee v_1, ..., u_n \vee v_n)$, $\vee$ – дизъюнкция (логическая операция ИЛИ).

После группировки слагаемых по общему множителю, получим равенство

$$\sum_{w \in GF(2)^n} \left(\sum_{a_u, b_v : a_u \vee b_v = w} a_u b_v \right) x^w = 0, \tag{4}$$

которое выполняется для любого $w \in GF(2)^n$. Следовательно, имеем систему линейных однородных уравнений

$$\begin{cases} \sum\limits_{a_u, b_v : a_u \vee b_v = w} a_u b_v = 0, \ \forall w \in GF(2)^n \end{cases} \tag{5}$$

относительно неизвестных коэффициентов b_v аннигилятора $g(x)$.

Поиск пространства аннигиляторов осуществляется путем решения системы линейных уравнений (5), полученных на основе группирования неизвестных коэффициентов по всем различным мономам.

Приведем пример из работы [9] построения базиса пространства $A_d^n(f)$.

В этом примере $f(x)$ – исходный многочлен в виде полинома Жегалкина; $g(x)$ – искомый аннулирующий многочлен.

Пример 1. Для $n = 2$ и $d = 1$ имеем:

$$f(x) = a_{00} + a_{10} x_1 + a_{01} x_2 + a_{11} x_1 x_2 ,$$
$$g(x) = b_{00} + b_{10} x_1 + b_{01} x_2 .$$

После подстановки в $f(x) \cdot g(x) = 0$ получим

$$\begin{aligned} f(x) \cdot g(x) = {} & a_{00} b_{00} + (a_{00} b_{10} + a_{10} b_{10} + a_{10} b_{00}) x_1 + \\ & + (a_{00} b_{01} + a_{01} b_{01} + a_{01} b_{00}) x_2 + \\ & + (a_{10} b_{01} + a_{01} b_{10} + a_{11} b_{00} + a_{11} b_{10} + a_{11} b_{01}) x_1 x_2 = 0, \end{aligned}$$

откуда имеем систему линейных однородных уравнений

$$\begin{cases} a b = 0, \\ a b + a b + a b = 0, \\ a b + a b + a b = 0, \\ a b + a b + a b + a b + a b = 0 \end{cases} \tag{6}$$

относительно неизвестных $b_{00}, b_{10}, b_{01}, \ldots$ – коэффициентов функции $g(x)$.

Тогда, например, для функции $f(x)=x_1+x_2$ (т.е. при $a_{00}=a_{11}=0$ и $a_{10}=a_{01}=1$) получим систему

$$\begin{cases} b_{10}+b_{00}=0, \\ b_{01}+b_{00}=0, \\ b_{01}+b_{10}=0, \end{cases}$$

которой удовлетворяют только два решения:

$$b_{00}=b_{10}=b_{01}=0\text{, т.е. } g(x)=0,$$

$$b_{00}=b_{10}=b_{01}=1\text{, т.е. } g(x)=1+x_1+x_2.$$

Непосредственная проверка показывает, что $g(x)=1+x_1+x_2$ действительно является аннигилятором функции $f(x)=x_1+x_2$:

$$f(x)\cdot g(x)=(x_1+x_2)(1+x_1+x_2)=x_1+x_2+x_1+x_1x_2+x_1x_2+x_2=0.$$

В общем случае для решения системы уравнений (5), например методом Гаусса, строится таблица размером $2^{2n}\times 2^{2n}$, где n – степень S-блока. Ячейки таблицы – сгруппированный результат умножения всех мономов полинома $f(x)$ на полином $g(x)$. Группировка происходит таким образом, что в итоге строчки представляются как все возможные варианты мономов (их 2^{2n}), а столбцы – это неизвестные коэффициенты (их тоже 2^{2n}). Таким образом, значение ячейки таблички может быть = 1, если коэффициент b_i присутствует в конкретном мономе и 0, если такой коэффициент отсутствует. Табл. 1 иллюстрирует этот процесс для примера $f(x)=x_1+x_2$.

Таблица 1

Результирующая таблица сгруппированных коэффициентов

b_{ij} / x	b_{00}	b_{01}	b_{02}	b_{03}
1	0	0	0	0
x_1	1	1	0	0
x_2	1	0	1	0
$x_1 x_2$	0	1	1	0

Подробное описание построения базиса Грёбнера, понятие которого было использовано Жаном Шарлем Фожером для определения алгебраической иммунности S-блоков, приводится в работе [9].

Приведем алгоритм вычисления алгебраической иммунности булевой функции, предлагаемый в работе [9] и оценим его возможную стандартную реализацию по объему вычислений.

4.1. Алгоритм вычисления алгебраической иммунности булевой функции

Вход: $n\in N$, функция $f(x)$ (заданная списком одночленов x^u с ненулевыми коэффициентами a_u в (3)).

Выход: Значение алгебраической иммунности $AI(f)$.

Шаг 1. Присваиваем $d=1$.

Шаг 2. Вычисляем пространство аннигиляторов $A_d^n(f)$ и $A_d^n(f+1)$.

Шаг 3. Если $A_d^n(f)=0$ и $A_d^n(f+1)=0$ присваиваем $d=d+1$ и переходим к шагу 2.

Шаг 4. Если $A_d^n(f)\neq 0$ и/или $A_d^n(f+1)\neq 0$ присваиваем $AI(f)=d$ и подаем на выход алгоритма.

Представленный алгоритм вычисляет алгебраическую иммунность для булевой функции $f(x)$. Для вычисления алгебраической иммунности S-блока по Арсу – Фожеру необходимо перевести (отобразить) нелинейный узел замены в булеву функцию $f(x)$ (таблицу истинности) в соответствии с (1). Чтобы подать на вход алгоритма булеву функцию $f(x)$, ее необходимо представить в виде полинома Жегалкина (3).

Далее обсуждаются возможности оптимизации алгоритма построения полинома Жегалкина.

4.2. Алгоритм приведения булевой функции, заданной в виде таблицы истинности, к алгебраической нормальной форме и оптимизация вычислений

Поиск значения коэффициентов мономов полинома Жегалкина происходит последовательно по всем мономам $a_u x^u$ из (2). В дальнейших рассуждениях вектор будем также представлять в виде целых чисел $U \in \left[0,..,2^n-1\right]: u=(u_1,...,u_n),\ u \in V_2^n$

$$U=\sum_{i=0}^{n-1} u_{i+1} 2^i, \tag{7}$$

а коэффициент $a_{(u_1,...,u_n)}$ одночлена $a_u x^u$ записывать в виде a_U.

Например, одночлену $x^u = x_1 x_2 x_4$ в (2) соответствует вектор $u=(1,1,0,1)$, который запишем как целое число $U=11$, коэффициент $a_{(1,1,0,1)}$ будем обозначать также как a_{11}.

Для заданного одночлена x^u введем определение *компонентных мономов*, под которыми будем понимать все такие ненулевые одночлены x^v, соответствующие вектора $v=(v_1,...,v_n)$ которых могут быть получены по правилу

$$v = u \wedge h = (u_1 \wedge h_1,...,u_n \wedge h_n), \tag{8}$$

где $\wedge$ – конъюнкция (логическая операция И), $h=(h_1,...,h_n)$ – произвольный вектор из V_2^n.

Очевидно, что целочисленное представление H векторов $h=(h_1,...,h_n)$ удовлетворяет условию $0<H<U$ и для поиска всех компонентных мономов необходимо по формуле (7) проверить $U-1$ одночленов. Например, для $x^u = x_1 x_2 x_4$ необходимо выполнить 10 проверок (для всех $0<H<U=11$), компонентными являются одночлены x^v: x_1, x_2, $x_1 x_2$, x_4, $x_1 x_4$, $x_2 x_4$ – им соответствуют целые числа $H=1,2,3,4,5,6$. Очевидно также, что целочисленные представления компонентных мономов также удовлетворяют условию $0<V<U$.

Пусть последовательность булевой функции $f(x)=\sum_{U=0}^{2^n-1} a_U x^U$ над V_2^n представлена в виде вектора двоичных значений $c=(c_0,...,c_{2^n-1})$. Тогда задача нахождения полинома Жегалкина состоит в решении системы линейных уравнений:

$$\begin{pmatrix} a_0 & 0 & 0 & 0 & ... & 0 \\ a_0 & a_1 & 0 & 0 & ... & 0 \\ a_0 & 0 & a_2 & 0 & ... & 0 \\ ... & ... & ... & ... & ... & ... \\ a_0 & a_1 & a_2 & a_3 & ... & a_{2^n-1} \end{pmatrix} = \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ ... \\ c_{2^n-1} \end{pmatrix}. \tag{9}$$

Например, для $n=2$ имеем систему

$$\begin{pmatrix} a_0 & 0 & 0 & 0 \\ a_0 & a_1 & 0 & 0 \\ a_0 & 0 & a_2 & 0 \\ a_0 & a_1 & a_2 & a_3 \end{pmatrix} = \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix},$$

решение которой имеет вид

$$a_0 = a_{(0,0)} = c_0;\ a_1 = a_{(1,0)} = c_0 + c_1;\ a_2 = a_{(0,1)} = c_0 + c_2;\ a_3 = a_{(1,1)} = c_0 + c_1 + c_2 + c_3.$$

В общем случае для решения системы линейных уравнений методом Гаусса необходимо порядка $(2^n)^3$ операций сложений и умножений или, формально, $O\left((2^n)^3\right)$. Однако в данном случае алгоритм восстановления АНФ можно существенно упростить. Действительно, из предыдущего примера видно, что задачу нахождения коэффициентов $a_0, a_1 ..., a_{2^n-1}$ можно реализовать последовательно. Коэффициент $a_0 = c_0$ задан однозначно, а для нахождения каждого следующего коэффициента a_U необходимо решить линейное уравнение от одной неизвестной:

$$a_0 + a_1 + ... + a_U = c_U, \tag{10}$$

где в уравнение входят найденные на предыдущих шагах коэффициенты a_V компонентных мономов.

Например, для нахождения значения коэффициента a_9 монома второй степени x_1x_4 в соответствии с (10) имеем уравнение

$$a_0 + a_1 + a_8 + a_9 = c_9$$

где коэффициенты a_0, a_1, a_8 определены на предыдущих шагах алгоритма; c_9 – значение таблицы истинности булевой функции для монома x_1x_4.

С учетом (10) алгоритм приведения булевой функции к алгебраической нормальной форме запишем в следующем виде.

Алгоритм формирования полинома Жегалкина:

Вход: последовательность $c = (c_0, ..., c_{2^n-1})$ булевой функции $f(x)$;

Выход: строка коэффициентов $a_0, a_1 ..., a_{2^n-1}$ полинома Жегалкина булевой функции $f(x)$

Шаг 1. Принять $a_0 = c_0$;

Шаг 2. Принять $U = 1$, где U – десятичное представление одночлена в виде (3);

Шаг 3. Для одночлена U найти все компонентные мономы V, т.е. такие вектора $v = (v_1, ..., v_n)$, для которых выполняется равенство (4);

Шаг 4. Решить уравнение (6), т.е. найти a_U.

Шаг 5. Если $U \le 2^n - 1$ принять $U = U + 1$ и перейти к шагу 2. Иначе вывести строку коэффициентов $a_0, a_1 ..., a_{2^n-1}$.

Оценим сложность приведенного алгоритма.

Всего выполняется $2^n - 1$ проходов по внешнему циклу (для всех $U = 1, 2, ..., 2^n - 1$). На каждом проходе выполняется $U - 1$ операций логического И (для формирования компонентных мономов) и U операций двоичного сложения (для нахождения коэффициента a_U).

Число выполняемых операций двоичного сложения определяется как сумма членов арифметической прогрессии, для которой:

- первый член прогрессии равен 1 (число операций для вычисления коэффициента a_1);

- последний член прогрессии равен $2^n - 1$ (число операций для вычисления коэффициента a_U).

Сумма $2^n - 1$ членов прогрессии

$$\frac{1+2^n-1}{2}(2^n-1) = 2^{2n-1} - 2^{n-1}$$

операций двоичного сложения или, формально, сложность алгоритма – $O\left(2^{2n-1}\right)$.

Для подсчета числа операций логического И имеем арифметическую прогрессию:

- первый член которой равен 0 (для вычисления коэффициента a_1);

- последний член равен $2^n - 2$ (для вычисления коэффициента a_U), т.е. сумма $2^n - 1$ членов прогрессии

$$\frac{0+2^n-2}{2}(2^n-1) = 2^{2n-1} + 2^n - 2^{n-1} - 1$$

или, формально, сложность алгоритма равна $O\left(2^{2n-1}\right)$.

Очевидно, что рассмотренный алгоритм приведения булевой функции к алгебраической нормальной форме значительно проще общего случая вычисления коэффициентов $a_0, a_1 ..., a_{2^n-1}$ через решение системы линейных уравнений (5).

Возвращаясь к общему алгоритму вычисления алгебраической иммунности, зная, как привести булеву функцию к алгебраической нормальной форме, запишем алгоритм вычисления алгебраической иммунности S-блока:

Алгоритм вычисления алгебраической иммунности S-блока:

Вход: Нелинейный узел замены (S-блок) степени $n \in N$.

Выход: Значение алгебраической иммунности нелинейного узла замены.

Шаг 1. Преобразуем нелинейный узел замены (S-блок) степени n в булеву функцию $f(x)$ от $2n$ переменных.

Шаг 2. Приведем булеву функцию $f(x)$ к виду полинома Жегалкина.

Шаг 3. Присваиваем $d = 1$.

Шаг 4. Вычисляем пространство аннигиляторов $A_d^n(f)$ и $A_d^n(f+1)$.

Шаг 5. Если $A_d^n(f) = 0$ и $A_d^n(f+1) = 0$ – присваиваем $d = d + 1$ и переходим к шагу 2.

Шаг 6. Если $A_d^n(f) \neq 0$ и/или $A_d^n(f+1) \neq 0$ – присваиваем $AI(f) = d$ и подаем на выход алгоритма.

Для реализации такого алгоритма подсчета АИ байтового S-блока, то есть булевой функции 16-ти переменных (n = 16), необходимо иметь таблицу размерами $2^{16} \times 2^{16}$. Для хранения одной такой таблицы, которая содержит результат умножения всех возможных мономов полинома $f(x)$ на мономы аннигилирующего полинома $g(x)$, необходимо приблизительно 8.5 Гб ОЗУ на ПК, с учетом того что каждое число в таблице может занимать 2 байта, это без учета вспомогательных таблиц, которые могут также понадобится в ходе вычислений. Ресурсы ОЗУ, которые занимает таблица, нетрудно вычислить, так как это произведение трех сомножителей. Количество строк таблицы, количество столбов таблицы и размер одной ячейки таблицы.

Если говорить о количестве операций, нам понадобятся неоднократные проходы по всей таблице (2^{32} операций на один проход) для решения систем линейных уравнений.

Оптимизированный алгоритм подсчета АИ

Известно, что значение алгебраической иммунности S-блока не может быть больше чем, $n/2$ [2], следовательно, для байтового S-блока значение алгебраической иммунности не мо-

жет быть больше 4. Если вспомнить, что алгебраическая иммунность определяется минимальной степенью ненулевых полиномов $AI(s) = \min \deg Ann(f_s)$, то можно сказать что в результирующей группировке неизвестных коэффициентов b_v, не может оказаться мономов степени выше 4, где b_v – неизвестные коэффициенты аннигилятора. Следовательно, при построении таблицы перемножения всех мономов полинома $f(x)$ на мономы полинома $g(x)$ мы можем брать только те мономы полинома $g(x)$, у которых степень будет меньше, либо равна $n/2$ (для байтового S-блока это четверка).

Количество подходящих нам мономов полинома $g(x)$ в худшем случае, нетрудно посчитать по формуле

$$k = \sum_{i=1}^{n/2} C_{2n}^{i}.$$

Зная, что максимальная АИ не может быть больше $n/2$ и не получив решения системы уравнений для максимальной степени монома $n/2-1$, можно сделать вывод, что АИ будет равным $n/2$. То есть формула подсчета количество подходящих нам мономов полинома $g(x)$ на самом деле будет выглядеть так:

$$k = \sum_{i=1}^{n/2-1} C_{2n}^{i}. \qquad (11)$$

Для байтового S-блока по Арсу – Фожеру получим:

$$k = C_{16}^{1} + C_{16}^{2} + C_{16}^{3}.$$
$$k = 696.$$

Итого, 696 мономов вместо 65536 для байтового S-блока.

Далее все вычисления при расчете максимально возможного значения алгебраической иммунности будут происходить уже с таблицей размерами (65536×696), а это уже $< 2^{26}$ операций для прохода по всей таблице, вместо 2^{32}. И около 0.1 Гб ОЗУ на ПК, вместо 8.5 Гб для хранения одной такой таблицы.

Если рассматривать расчет АИ для блоков нелинейной замены различных степеней, то размер ячейки таблицы должен быть равен 1 байт при подсчете АИ S-box степени меньше либо равной 2^4, 2 байта для S-box степени меньше, либо равной 2^8 и 4 байта для подсчета АИ S-box степени от 2^8 до 2^{16}. Для универсальности расчетов и наглядности примем размер ячейки таблицы равным 4 байта, для любых степеней блока нелинейной замены, до 2^{16}.

Ниже приведена табл. 2, показывающая эффективность оптимизированного алгоритма подсчета АИ S-box по сравнению со стандартным, размер ячейки таблицы принят равным 4 байта для любых степеней блока нелинейной замены.

Таблица 2

Затраты ОЗУ для построения таблицы перемножения мономов, используемой в алгоритмах подсчета АИ для S-box

Алгоритм / Степень S-box	Стандартный	Оптимизированный
2^4	0,26 Мб ОЗУ	0,008 Мб ОЗУ
2^6	67,1 Мб ОЗУ	1,3 Мб ОЗУ
2^8	17,2 Гб ОЗУ	182 Мб ОЗУ
2^9	275 Гб ОЗУ	1,03 Гб ОЗУ
2^{10}	4398 Гб ОЗУ	26 Гб ОЗУ
2^{11}	70369 Гб ОЗУ	153 Гб ОЗУ
2^{12}	1126 Тб ОЗУ	3721 Гб ОЗУ

Получившиеся результаты показывают очень весомую разницу затрат ресурсов ОЗУ на ПК между оптимизированным алгоритмом и обычным.

Возвращаясь к способу оптимизации, отметим, что возникает только одна проблема – мономы полинома никак не упорядочены по степеням и хранятся в произвольном порядке относительно степеней, а точнее в порядке, который задает арифметика цифр (0...65536). Например, моном первой степени можно встретить в позициях (1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768), где позиции большие, чем 696 никак не вписываются в нашу оптимизацию. Для решения этой проблемы, необходимо сгруппировать полином $g(x)$ таким образом, чтобы мономы были упорядочены по степеням. Например, можно хранить позиции интересующих нас мономов отдельно и обращаться только к нужным позициям. Это позволит работать с таблицей (65536×696), вместо (65536×65536)

Пример 2.

Дан S-блок 2-й степени и соответствующая ему булева функция от 4-х переменных, в виде полинома Жегалкина. $f(x) = x_1 + x_1x_2 + x_3 + x_2x_3 + x_1x_4 + x_3x_4$.

Запишем аннигилирующий полином с искомыми коэффициентами b в общем виде

$$g(x) = b_0 + b_1x_1 + b_2x_2 + b_3x_1x_2 + b_4x_3 +$$
$$+b_5x_1x_3 + b_6x_2x_3 + b_7x_1x_2x_3 + b_8x_4 +$$
$$+b_9x_1x_4 + b_{10}x_2x_4 + b_{11}x_1x_2x_4 + b_{12}x_3x_4 +$$
$$+b_{13}x_1x_3x_4 + b_{14}x_2x_3x_4 + b_{15}x_1x_2x_3x_4.$$

Для подсчета АИ исходного S-блока необходимо построить таблицу перемножения всех мономов полинома $f(x)$ на мономы полинома $g(x)$. Для нашего примера она имеет вид, представленный в табл. 3 (число столбцов ограничено из-за недостаточности места в таблице, они не влияют на результаты, так как имеют нулевые значения ячеек). На входы по столбцам таблицы подаются мономы полинома $f(x)$, на входы по строкам подаются мономы полинома $g(x)$. Ячейки таблицы – результат умножения монома строки на моном столбца.

Таблица 3

Результаты умножения мономов полиномов $f(x)$ и $g(x)$

$g(x)$ \ $f(x)$	0	x_1	0	x_1x_2	x_3	0	x_2x_3	0	0	x_1x_4	0	0	x_3x_4
b_0	0	x_1	0	x_1x_2	x_3	0	x_2x_3	0	0	x_1x_4	0	0	x_3x_4
$b_1\ x_1$	0	x_1	0	x_1x_2	x_1x_3	0	$x_1x_2x_3$	0	0	x_1x_4	0	0	$x_1x_3x_4$
$b_2\ x_2$	0	x_1x_2	0	x_1x_2	x_2x_3	0	x_2x_3	0	0	$x_1x_2x_4$	0	0	$x_2x_3x_4$
$b_3\ x_1x_2$	0	x_1x_2	0	x_1x_2	$x_1x_2x_3$	0	$x_1\ x_2x_3$	0	0	$x_1x_2x_4$	0	0	$x_1x_2x_3x_4$
$b_4\ x_3$	0	x_1x_3	0	$x_1x_2x_3$	x_3	0	x_2x_3	0	0	$x_1x_3x_4$	0	0	x_3x_4
$b_5\ x_1x_3$	0	x_1x_3	0	$x_1x_2x_3$	x_1x_3	0	$x_1x_2x_3$	0	0	$x_1x_3x_4$	0	0	$x_1x_3x_4$
$b_6\ x_2x_3$	0	$x_1\ x_2x_3$	0	$x_1x_2x_3$	x_2x_3	0	x_2x_3	0	0	$x_1x_2x_3x_4$	0	0	$x_2x_3x_4$
$b_7\ x_1x_2x_3$	0	$x_1x_2x_3$	0	$x_1x_2x_3$	$x_1x_2x_3$	0	$x_1x_2x_3$	0	0	$x_1x_2x_3x_4$	0	0	$x_1\ x_2x_3x_4$
$b_8\ x_4$	0	x_1x_4	0	$x_1x_2x_4$	x_3x_4	0	$x_2x_3x_4$	0	0	x_1x_4	0	0	x_3x_4
$b_9\ x_1x_4$	0	x_1x_4	0	$x_1x_2x_4$	$x_1x_3x_4$	0	$x_1x_2x_3x_4$	0	0	x_1x_4	0	0	$x_1x_3x_4$
$b_{10}\ x_2x_4$	0	$x_1x_2x_4$	0	$x_1x_2x_4$	$x_2x_3x_4$	0	$x_2x_3x_4$	0	0	$x_1x_2x_4$	0	0	$x_2x_3x_4$
$b_{11}\ x_1x_2x_4$	0	$x_1x_2x_4$	0	$x_1x_2x_4$	$x_1x_2x_3x_4$	0	$x_1x_2x_3x_4$	0	0	$x_1x_2x_4$	0	0	$x_1x_2x_3x_4$
$b_{12}\ x_3x_4$	0	$x_1x_3x_4$	0	$x_1x_2x_3x_4$	x_3x_4	0	$x_2x_3x_4$	0	0	$x_1x_3x_4$	0	0	x_3x_4
$b_{13}\ x_1x_3x_4$	0	$x_1x_3x_4$	0	$x_1x_2x_3x_4$	$x_1x_3x_4$	0	$x_1x_2x_3x_4$	0	0	$x_1x_3x_4$	0	0	$x_1x_3x_4$
$b_{14}\ x_2x_3x_4$	0	$x_1x_2x_3x_4$	0	$x_1x_2x_3x_4$	$x_2x_3x_4$	0	$x_2x_3x_4$	0	0	$x_1x_2x_3x_4$	0	0	$x_2x_3x_4$
$b_{15}\ x_1x_2x_3x_4$	0	$x_1x_2x_3x_4$	0	$x_1x_2x_3x_4$	$x_1x_2x_3x_4$	0	$x_1x_2x_3x_4$	0	0	$x_1x_2x_3x_4$	0	0	$x_1x_2x_3x_4$

Далее происходит группировка неизвестных коэффициентов по всем возможным мономам от 0 до 2^n. Строится таблица сгруппированных неизвестных коэффициентов b_v (см. табл. 4) На входы по строкам подаются все возможные варианты мономов, входы по столб-

цам соответствуют неизвестным коэффициентам b_v (единица обозначает присутствие коэффициента соответствующего монома, а ноль – его отсутствие). Значение ячейки может быть равно нулю, если в табл. 3 нет мономов таких, как моном текущей строки, или если таких мономов четное количество в конкретной строке табл. 3. Значение ячейки таблицы может быть равно единице для некоторой строки *x* и некоторого столбца *y*, если в строке *y* табл. 3 есть нечетное количество мономов *x*.

Практически, заполнение каждой строки табл. 4. соответствует множеству неизвестных коэффициентов (их позиции обозначены единицами) и соответствует одному из уравнений (его левой части) системы линейных уравнений, задаваемых таблицей. Для вычисления АИ далее необходимо решить полученную систему уравнений, например, методом Гаусса. Полученные решения – это пространство аннигиляторов. Воспользовавшись алгоритмом вычисления алгебраической иммунности S-блока начиная с шага 3, находим АИ.

Для оптимизированного варианта вычисления АИ при построении аннигилирующего полинома $g(x)$ необходимо найти количество мономов, которые будут входить в оптимизированный вариант полинома $g(x)$ по формуле (11). Далее найти все мономы, степень которых будет не больше *n*/2 и построить оптимизированный полином $g(x)$, который будет состоять из найденных подходящих мономов. Напомним, что в общем виде мономы никак не упорядочены по степеням, поэтому взять нужные мономы на позициях от 0 до $2^{2n}-1$ не получится. Необходимо найти на каких именно позициях находятся мономы степени не выше *n*/2. Далее аналогично стандартному варианту вычислений строятся табл. 3 и 4, только размер этих таблиц будет значительно меньше. В оптимизированном варианте будут использоваться только строки и столбцы, выделенные в табл. 3 и 4 цветом. Видно, что объем обрабатываемых данных, участвующих в вычислениях, существенно уменьшается. Для вычисления АИ также решается система линейных уравнений, например, методом Гаусса и ищется АИ алгоритмом вычисления алгебраической иммунности S-блока.

Таблица 4

Группировка мономов по неизвестным коэффициентам
(формирование левых частей уравнений)

x \ b_i	b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}	b_{11}	b_{12}	b_{13}	b_{14}	b_{15}
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
x_1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
x_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
x_1x_2	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
x_3	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
x_1x_3	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
x_2x_3	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
$x_1x_2x_3$	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
x_4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
x_1x_4	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
x_2x_4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$x_1x_2x_4$	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	0
x_3x_4	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
$x_1x_3x_4$	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
$x_2x_3x_4$	0	0	1	0	0	0	1	0	1	0	1	0	1	0	1	0
$x_1x_2x_3x_4$	0	0	0	1	0	0	1	0	0	1	0	1	1	0	1	0

Для еще большей оптимизации мы можем строить аннигилирующий полином $g(x)$ не из всех степеней мономов, не превышающих *n*/2, а из степеней мономов, не превышающих $i \in 1\ldots n/2$. То есть на первой итерации будет строиться полином $g(x)$, состоящий из мономов,

не превышающих 1-й степени, количество таких мономов будет C_{2n}^1. Если результат решения системы получившихся уравнений не даст ни одного ненулевого решения, то d принимается равным $d\_+ 1$, и строится аннигилирующий полином размером уже $C_{2n}^1 + C_{2n}^2$, степень мономов которого не превышает 2. Так происходит до тех пор, пока в результате решения системы линейных уравнений не будет найдено хотя бы одно ненулевое решение. Выход из алгоритма $AI(f) = d$.

Дополнительная оптимизация затрат ОЗУ на ПК при реализации поиска алгебраической иммунности блока нелинейной замены по Арсу – Фожеру.

Отметим, что основные затраты ОЗУ на ПК занимала табл. 3 перемножения полиномов $f(x)$ и $g(x)$, и именно о сжатии этой таблицы шла речь как об основном результате оптимизации. Этот результат впоследствии позволил посчитать АИ для S-box степени 2^9, требуя всего около 1 Гб ОЗУ, в то время как неоптимизированный алгоритм для подсчета S-box той же степени требовал бы около 275 Гб ОЗУ для хранения одной только такой таблицы. Но вычислить АИ для S-box степени 2^{10} и выше уже не представлялось возможным, по крайней мере, на обычном ПК. Напомним, что для хранения табл. 3 для S-box степени 2^{10} оптимизированный алгоритм требовал уже 26 Гб ОЗУ. Весомость затрат таблицы обусловлена необходимостью содержать в каждой ячейке 4 байта для возможности хранения монома максимальной степени 2^{2n}.

Первым делом можно попытаться сократить количество столбцов этой таблицы. Как говорилось выше, мы должны брать все возможные мономы $f(x)$ и не можем уменьшить количество столбцов. На самом деле это не совсем так, мы можем уменьшить количество столбцов, исключив нулевые столбцы, заведомо зная, что они всегда будут получаться в результате умножения на нулевой моном полинома $f(x)$. Но, сколько таких мономов будут иметь различные полиномы $f(x)$ мы знать, конечно же, не можем. В худшем случае $f(x)$ может иметь 2^{2n} ненулевых мономов, поэтому существенной оптимизации это нам не даст.

Более весомая оптимизация будет строиться на возможности не хранить в ОЗУ сразу не всю табл. 3 перемножения полиномов $f(x)$ и $g(x)$, а хранить только одну строку этой таблицы. Одна строка табл. 3 позволяет полностью построить один столбец табл. 4, таблицы группировки мономов по неизвестным коэффициентам (формирование левых частей уравнений). Таким образом, храня в ОЗУ одну строку табл. 3, затем удаляя ее из ОЗУ и заменяя ее на следующую строку табл. 3, мы можем полностью построить табл. 4. Отсюда следует, что затраты ОЗУ на хранения табл. 3 практически отсутствуют и основные затраты ресурсов ОЗУ на ПК для подсчета АИ блока нелинейной замены будет теперь занимать табл. 4.

Таблица 5

Затраты ОЗУ для построения таблицы, используемой в алгоритмах подсчета АИ для S-box при различных методах оптимизации

Алгоритм / Степень S-box	Стандартный	Оптимизированный	Дополнительная оптимизация
2^4	0,26 Мб ОЗУ	0,008 Мб ОЗУ	0,002 Мб ОЗУ
2^6	67,1 Мб ОЗУ	1,3 Мб ОЗУ	0,3 Мб ОЗУ
2^8	17,2 Гб ОЗУ	182 Мб ОЗУ	45 Мб ОЗУ
2^9	275 Гб ОЗУ	1,03 Гб ОЗУ	250 Мб ОЗУ
2^{10}	4398 Гб ОЗУ	26 Гб ОЗУ	6,5 Гб ОЗУ
2^{11}	70369 Гб ОЗУ	153 Гб ОЗУ	38 Гб ОЗУ
2^{12}	1126 Тб ОЗУ……	3721 Гб ОЗУ	930 Гб ОЗУ

5. Эксперименты

Для проведения тестов использовался ПК с Windows 10, Intel Core i7-3630QM 2.4 Ггц, 8Гб ОЗУ

Первая часть экспериментов была посвящена изучению быстродействия различных алгоритмов вычисления АИ на примере блоков нелинейной замены степени 2^8, использовались узлы нелинейной замены шифров AES и Калина (см. рис. 2, 3).

Алгебраическая иммунность S-блока шифра AES (см. рис. 2) получается равной 2. Воспользовавшись выражением (16), имеем: $k = C_{16}^1 + C_{16}^2$, k = 136.

Алгебраическая иммунность S-блока шифра Калина (см. рис. 3) получается равной 3. Воспользовавшись выражением (16), имеем: $k = C_{16}^1 + C_{16}^2 + C_{16}^3$, k = 696.

Именно такие значения АИ получены с помощью комплекса Magma в работе [9].

Время выполнения вычислений для стандартного алгоритма выходит за приемлемые сроки. Оптимизированный алгоритм вычислений выполнил поставленную задачу за 4 с для S-блока AES и за 20 с для S-блока Калина-2 соответственно. Алгоритм на основе базисов Грёбнера с помощью комплекса Magma [9] справился с поставленной задачей за 5 с.

```
0x63 0x7c 0x77 0x7b 0xf2 0x6b 0x6f 0xc5 0x30 0x01 0x67 0x2b 0xfe 0xd7 0xab 0x76
0xca 0x82 0xc9 0x7d 0xfa 0x59 0x47 0xf0 0xad 0xd4 0xa2 0xaf 0x9c 0xa4 0x72 0xc0
0xb7 0xfd 0x93 0x26 0x36 0x3f 0xf7 0xcc 0x34 0xa5 0xe5 0xf1 0x71 0xd8 0x31 0x15
0x04 0xc7 0x23 0xc3 0x18 0x96 0x05 0x9a 0x07 0x12 0x80 0xe2 0xeb 0x27 0xb2 0x75
0x09 0x83 0x2c 0x1a 0x1b 0x6e 0x5a 0xa0 0x52 0x3b 0xd6 0xb3 0x29 0xe3 0x2f 0x84
0x53 0xd1 0x00 0xed 0x20 0xfc 0xb1 0x5b 0x6a 0xcb 0xbe 0x39 0x4a 0x4c 0x58 0xcf
0xd0 0xef 0xaa 0xfb 0x43 0x4d 0x33 0x85 0x45 0xf9 0x02 0x7f 0x50 0x3c 0x9f 0xa8
0x51 0xa3 0x40 0x8f 0x92 0x9d 0x38 0xf5 0xbc 0xb6 0xda 0x21 0x10 0xff 0xf3 0xd2
0xcd 0x0c 0x13 0xec 0x5f 0x97 0x44 0x17 0xc4 0xa7 0x7e 0x3d 0x64 0x5d 0x19 0x73
0x60 0x81 0x4f 0xdc 0x22 0x2a 0x90 0x88 0x46 0xee 0xb8 0x14 0xde 0x5e 0x0b 0xdb
0xe0 0x32 0x3a 0x0a 0x49 0x06 0x24 0x5c 0xc2 0xd3 0xac 0x62 0x91 0x95 0xe4 0x79
0xe7 0xc8 0x37 0x6d 0x8d 0xd5 0x4e 0xa9 0x6c 0x56 0xf4 0xea 0x65 0x7a 0xae 0x08
0xba 0x78 0x25 0x2e 0x1c 0xa6 0xb4 0xc6 0xe8 0xdd 0x74 0x1f 0x4b 0xbd 0x8b 0x8a
0x70 0x3e 0xb5 0x66 0x48 0x03 0xf6 0x0e 0x61 0x35 0x57 0xb9 0x86 0xc1 0x1d 0x9e
0xe1 0xf8 0x98 0x11 0x69 0xd9 0x8e 0x94 0x9b 0x1e 0x87 0xe9 0xce 0x55 0x28 0xdf
0x8c 0xa1 0x89 0x0d 0xbf 0xe6 0x42 0x68 0x41 0x99 0x2d 0x0f 0xb0 0x54 0xbb 0x16
```

Рис. 2. S-блок шифра AES

```
0xA8 0x43 0x5F 0x06 0x6B 0x75 0x6C 0x59 0x71 0xDF 0x87 0x95 0x17 0xF0 0xD8 0x09
0x6D 0xF3 0x1D 0xCB 0xC9 0x4D 0x2C 0xAF 0x79 0xE0 0x97 0xFD 0x6F 0x4B 0x45 0x39
0x3E 0xDD 0xA3 0x4F 0xB4 0xB6 0x9A 0x0E 0x1F 0xBF 0x15 0xE1 0x49 0xD2 0x93 0xC6
0x92 0x72 0x9E 0x61 0xD1 0x63 0xFA 0xEE 0xF4 0x19 0xD5 0xAD 0x58 0xA4 0xBB 0xA1
0xDC 0xF2 0x83 0x37 0x42 0xE4 0x7A 0x32 0x9C 0xCC 0xAB 0x4A 0x8F 0x6E 0x04 0x27
0x2E 0xE7 0xE2 0x5A 0x96 0x16 0x23 0x2B 0xC2 0x65 0x66 0x0F 0xBC 0xA9 0x47 0x41
0x34 0x48 0xFC 0xB7 0x6A 0x88 0xA5 0x53 0x86 0xF9 0x5B 0xDB 0x38 0x7B 0xC3 0x1E
0x22 0x33 0x24 0x28 0x36 0xC7 0xB2 0x3B 0x8E 0x77 0xBA 0xF5 0x14 0x9F 0x08 0x55
0x9B 0x4C 0xFE 0x60 0x5C 0xDA 0x18 0x46 0xCD 0x7D 0x21 0xB0 0x3F 0x1B 0x89 0xFF
0xEB 0x84 0x69 0x3A 0x9D 0xD7 0xD3 0x70 0x67 0x40 0xB5 0xDE 0x5D 0x30 0x91 0xB1
0x78 0x11 0x01 0xE5 0x00 0x68 0x98 0xA0 0xC5 0x02 0xA6 0x74 0x2D 0x0B 0xA2 0x76
0xB3 0xBE 0xCE 0xBD 0xAE 0xE9 0x8A 0x31 0x1C 0xEC 0xF1 0x99 0x94 0xAA 0xF6 0x26
0x2F 0xEF 0xE8 0x8C 0x35 0x03 0xD4 0x7F 0xFB 0x05 0xC1 0x5E 0x90 0x20 0x3D 0x82
0xF7 0xEA 0x0A 0x0D 0x7E 0xF8 0x50 0x1A 0xC4 0x07 0x57 0xB8 0x3C 0x62 0xE3 0xC8
0xAC 0x52 0x64 0x10 0xD0 0xD9 0x13 0x0C 0x12 0x29 0x51 0xB9 0xCF 0xD6 0x73 0x8D
0x81 0x54 0xC0 0xED 0x4E 0x44 0xA7 0x2A 0x85 0x25 0xE6 0xCA 0x7C 0x8B 0x56 0x80
```

Рис. 3. S-блок шифра Калина

Вторая часть экспериментов была посвящена изучению возможностей различных алгоритмов поиска АИ относительно ресурсов ОЗУ на ПК, а именно – максимально возможной степени нелинейного узла замены, алгебраическую иммунность которого позволяют вычислить рассмотренные алгоритмы. Именно здесь сыграет свою роль алгоритм по методу Арсу – Фожера с дополнительной оптимизацией относительно ресурсов ОЗУ на ПК.

Результаты экспериментов показали, что неоптимизированный алгоритм по методу Арсу – Фожера не смог справиться с поставленной задачей из-за нехватки ресурсов ОЗУ на ПК уже при блоке нелинейной замены степени 2^9. Оптимизированный алгоритм по методу Арсу – Фожера преодолел эту сложность, но не смог справиться с блоком нелинейной замены степени 2^{10}. Алгоритм с дополнительной оптимизацией ОЗУ и алгоритм, реализованный с помощью комплекса Magma [9], справились с S-блоком степени 2^{10}. Таким образом, S-блок степени 2^{10} стал максимально возможным для вычисления АИ этими алгоритмами. Далее были проведен ряд тестов с использованием случайно сгенерированных S-блоков степени 2^{10}, а также детерминированного S-блока степени 2^{10}, состоящего из мультипликативно-обратных элементов поля.

Алгебраическая иммунность ряда случайно сгенерированных S-блоков степени 2^{10} получилась равной 3, что подтвердили оба алгоритма. При этом алгоритм по методу Арсу – Фожера с дополнительной оптимизацией выполнил задачу за 15 мин. Алгоритм, реализованный с помощью комплекса Magma [9], справился с задачей за 30 мин.

Алгебраическая иммунность S-блока, состоящего из мультипликативно-обратных элементов поля, получилась равной 2, что подтвердили оба алгоритма. При этом алгоритм по методу Арсу – Фожера с дополнительной оптимизацией выполнил задачу за 1 мин и 40 с. Алгоритм, реализованный с помощью комплекса Magma [9], справился с задачей за 30 мин.

6. Обсуждение

Рассмотрены детали реализации алгоритма расчета алгебраической иммунности по методу Арсу – Фожера. Проведены тесты, показавшие невозможность расчета АИ для байтовых S-блоков в связи с нехваткой вычислительных ресурсов. Время выполнения известного алгоритма вычисления АИ для байтовых S-блоков превышает приемлемые сроки.

Предложен ускоренный алгоритм расчета алгебраической иммунности, байтовых S-блоков по методу Арсу – Фожера, в котором используются реально существующие практические и теоретические ограничения, характерные для оцениваемого показателя. Он позволяет существенно сократить объемы обрабатываемой информации. В качестве таких ограничений использовано то, что в соответствии с теоретическими расчетами алгебраическая иммунность для байтовых S-блоков не может превышать значения 4, т.е. степень аннулирующего многочлена не может быть выше четвертой.

Последующая операция приведения сокращенной по числу столбцов матрицы коэффициентов к диагональному виду позволяет сократить размеры матрицы коэффициентов и по числу строк (появляются строки из одних нулей). В результате действительно удается существенно ускорить процедуру выполнения вычислений.

Предложен также ускоренный алгоритм расчета алгебраической иммунности по методу Арсу – Фожера с дополнительной оптимизацией ресурсов ОЗУ на ПК, что позволяет вычислять блоки нелинейной замены степени до 2^{10} включительно.

Результаты вычислений были проверены алгоритмом, реализованным с помощью комплекса Magma [9], также проведены сравнительные тесты быстродействия.

Предложена усовершенствованная процедура формирования полиномов Жегалкина по заданному значению таблицы истинности булевой функции S-блока.

7. Выводы

Основным результатом статьи является разработка ускоренного метода расчета алгебраической иммунности байтовых S-блоков, а также ускоренного метода расчета алгебраической иммунности с дополнительной оптимизацией относительно ресурсов ОЗУ на ПК.

Исключение при формировании программы множеств данных, не участвующих на каждом из этапов ее работы, а также учет априорно известных данных относительно конечного результата позволили существенно сократить объемы промежуточных вычислений и добиться повышения производительности программы в сотни раз. Усовершенствована также про-

цедура формирования полиномов Жегалкина по заданному значению таблицы истинности булевой функции S-блока.

Время работы программы при расчете алгебраической иммунности байтового S-блока сосоставляет от 4 до 20 с в зависимости от S-блока. Удалось вычислить АИ для S-блока степени 2^{10}, время работы программы при этом со значением АИ равной 3 составляет около 15 мин и со значением АИ равной 2 – около 2 мин. Таким образом, самым перспективным алгоритмом расчета АИ можно считать предложенный ускоренный алгоритм с дополнительной оптимизацией ресурсов ОЗУ, что в сотни раз быстрее, чем неоптимизированный алгоритм по методу Арсу – Фожера и в несколько раз быстрее алгоритма, реализованного в комплексе Magma [9] для максимальной степени S-блока 2^{10}.

Список литературы:

1. Courtois N. and Meier W. Algebraic attacks on stream ciphers with linear feedback // Eurocrypt'2003. LNCS. 2003. V. 2656. P. 345-359.

2. Покрасенко Д.П. Об алгебраической иммунности векторных булевых функций // Прикладная дискретная математика. Приложение. 2014. № 7. С. 43-48.

3. Meier W., Pasalic E., and Carlet C. Algebraic attacks and decomposition of Boolean functions // Eurocrypt'2004. LNCS. 2004. V. 3027. P. 474-491.

4. Armknecht F. and Krause M. Constructing single- and multi-output Boolean functions with maximal immunity // ICALP'2006. LNCS. 2006. V. 4052. P. 180-191.

5. Armknecht F. and Krause H. Constructing single- and multi-output Boolean functions with maximal immunity // ICALP'2006. V.4052. P. 180-191.

6. Ars G. and Faug`ere J.-C. Algebraic immunities of functions over finite fields // Proc. Conf. BFCA. 2005. P. 21-38.

7. Carlet C. On the algebraic immunities and higher order nonlinearities of vectorial Boolean functions // Enhancing Cryptographic Primitives with Techniques from Error Correcting Codes. Amsterdam: IOS Press, 2009. P. 104-116.

8. Faugère J.-C. (June 1999). A new efficient algorithm for computing Gröbner bases (F4) // Journal of Pure and Applied Algebra. Elsevier Science. 139 (1): 61-88.

9. Покрасенко Д.П. Компонентная алгебраическая иммунность S-блоков, использующихся в некоторых блочных шифрах // Прикладная дискретная математика. Приложение. 2017. № 10. С. 49-51.

10. Кузнецов О.О. Алгебраїчний імунітет нелінійних вузлів симетричних шифрів / О.О. Кузнецов, Ю.І. Горбенко, І.М. Білозерцев та інші // Радиотехника. 2017. Вып. 189. С. 47-58.

11. Magma Computational Algebra System. Available at: http://magma.maths.usyd.edu.au/magma.

12. Баев В. В. Эффективные алгоритмы получения оценок алгебраической иммунности булевых функций : дис. … канд. физ.-мат. наук : 01.01.09 / Баев Владимир Валерьевич; Место защиты: Моск. гос. ун-т им. М.В. Ломоносова. Фак. вычислит. математики и кибернетики. Москва, 2008. 101 с.

13. Gw´enol´e Ars, Jean-Charles Faug`ere. Algebraic Immunities of functions over finite fields // Research Report. RR-5532, INRIA. 2005. P.17.

14. Аржанцев И.В. Базисы Грёбнера и системы алгебраических уравнений. Летняя школа. Современная математика. Дубна, июль 2002. Москва : МЦНМО, 2003. 68 с

15. Злобин А.И., Соколова О.В. Компьютерная алгебра в системе Sage : учеб. пособие. Москва : МГТУ им. Баумана, 2011. 55 с.

16. Faugère, J.-C. (June 1999). A new efficient algorithm for computing Gröbner bases (F4). Journal of Pure and Applied Algebra. Elsevier Science. 139 (1): 61-88.

17. Faugère, J.-C. (July 2002). A new efficient algorithm for computing Gröbner bases without reduction to zero (F5) // Proceedings of the 2002 international symposium on Symbolic and algebraic computation (ISSAC). ACM Press. P.75-83.

18. Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone. Handbook of Applied Cryptography CRC Press, 1997. 794 p.

19. Горбенко І.Д., Горбенко Ю.І. Прикладна криптологія. Теорія. Практика. Застосування : підручник для вищих навч. закладів. Харків : Форт, 2013. 880 с.

20. Bart Preneel. Analysis and Design of Cryptographic Hash Functions. Электронный ресурс. Режим доступа: homes.esat.kuleuven.be/~preneel/phd_preneel_feb1993.pdf

21. Carlet C. Vectorial Boolean functions for // Cambridge Univ. Press, Cambridge. 95 p. Электронный ресурс. Режим доступа: www.math.univ-paris13.fr/~carlet/chap-vectorial-fcts-corr.pdf

22. Carlet C. Boolean functions for cryptography and error correcting codes // Cambridge Univ. Press, Cambridge. 2007. 148 p. Электронный ресурс. Режим доступа: www1.spms.ntu.edu.sg/~kkhoongm/chap-fcts-Bool.pdf.

23. Zhuo Zepeng, Zhang Weiguo On correlation properties of Boolean functions // Chinese Journal of Electronics. 2011. Vol.20. №1. P. 143-146.

24. O'Connor L. An analysis of a class of algorithms for S-box construction // J. Cryptology. 1994. P. 133-151.

25. Clark J.A., Jacob J.L., Stepney S. The Design of S-Boxes by Simulated Annealing // New Generation Computing. 2005. 23(3). P.219-231.

26. Кузнецов А.А., Белозерцев И.Н., Андрушкевич А.В. Анализ и сравнительные исследования нелинейных узлов замены современных блочных симметричных шифров // Прикладная радиоэлектроника. Харьков : ХНУРЭ. 2015. Т. 14. №4. С.343-350.

27. Massimiliano Sala, Teo Mora, Ludovic Perret, Shojiro Sakata, Carlo Traverso Gröbner Bases, Coding, and Cryptography. Springer-Verlag Berlin Heidelberg. 426 p.

Харьковский национальный
университет имени В.Н. Каразина

Поступила в редколлегию 18.01.2020