УДК 681.3.06

Л.С. Сорока[1], А.А. Кузнецов[2], С.А. Исаев[3]

[1] *Академия таможенной службы Украины, Днепропетровск*
[2] *Харьковский университет Воздушных Сил им. И. Кожедуба, Харьков*
[3] *Харьковский национальный университет им. В.Н. Каразина, Харьков*

ИССЛЕДОВАНИЕ ВЕРОЯТНОСТНЫХ МЕТОДОВ ФОРМИРОВАНИЯ НЕЛИНЕЙНЫХ УЗЛОВ ЗАМЕН

Рассматриваются вероятностные методы формирования нелинейных узлов замен (S-блоков). Анализируются жадный алгоритм и алгоритм ветвей и границ поиска S-блоков по множеству сформированных функций, проводятся экспериментальные исследования их эффективности с учетом различных критериев стойкости S-блоков (нелинейности, автокорреляции и алгебраической степени). Приводятся теоретические оценки эффективностивероятностных методов, которые сопоставляются с полученными экспериментальными результатами исследований.

Ключевые слова: *нелинейный узел замен, побитовый метод формирования S-блоков, биективность, нелинейность, автокорреляция, алгебраическая степень.*

Введение

Нелинейные узлы замен (S-блоки) являются главным компонентом современных блочных симметричных шифров (БСШ), например, таких как DESи AES. S-блок nxmпредставляет собой *нелинейную* функцию отображения nвходных бит в m выходных. Если n = mи каждый выход S-блока равновероятен, такой S-блок называют биективным. Биективные S-блоки получили наибольшее распространение в современных БСШ. Т.к. методы формирования биективных и небиективных S-блоков существенно отличаются друг от друга, в нашей работе мы рассматривали построение только *биективных* узлов замен.

Для того чтобы БСШ был стойким к криптоаналитическим атакам необходимо, чтобы используемые в нем узлы замен удовлетворяли определенным критериям стойкости [1].Методам формирования *стойких биективных* узлов замен посвящено множество работ. В нашей работе мы рассмотрели класс побитовых вероятностных (случайных) методов формирования S-блоков.

В основу наших исследований положена работа [2], в которой приводятся теоретические расчеты, показывающие практические границы размеров S-блоков, которые могут быть эффективно построены, используя вероятностные методы синтеза. Целью нашей работы было проведение экспериментальных исследований по оценке эффективности вероятностных методов формирования нелинейных узлов замен, и дальнейшее сопоставление полученных экспериментальных результатов с теоретическими оценками[2]. В основе проводимых исследований лежала оценка ожидаемого числа формирований

(шаг 1) и отборов (шаг 2) нелинейных функций, до того, как искомая перестановка (биективныйS-блок)была найдена. При этом эффективность случайных методов формирования S-блоков исследовалась с учетом наложенных ограничений по различным критериям стойкости (нелинейности, автокорреляции и алгебраической степени)[1].

1. Вероятностные методы формирования S-блоков

Любой S-блок S может быть представлен в виде совокупности булевых функций. В этом представлении оценка стойкости S-блока S основана на показателях булевых функций, которые его описывают. Пусть f_i^S – булева функция, которая описывает i-й бит S, $1 \le i \le m$. S реализуется через m-кортеж функций $F = \left[f_1^S, f_2^S, ..., f_m^S \right]$. Для того чтобы S-блок был стойким, каждая составляющая его функция и их линейные комбинации должны удовлетворять определенным критериям стойкости: сбалансированности, нелинейности NL, автокорреляции AC, алгебраической степени Deg и строгому лавинному критерию SAC[1].

Пусть S-блок $S \in S_{2^m}$ реализуется через m-кортеж m-битных функций $F^S = \left[f_1^S, f_2^S, ..., f_m^S \right]$. Пусть Φ^m – множество сбалансированных булевых функций, которые удовлетворяют некоторым заданным критериям и пусть $\Phi^{m,m}$ – множество m-битных перестановок (биективных S-блоков) $S \in S_{2^m}$ таких, что f_i^S и все их линейные комбинации принадлежат Φ^m, $1 \le i \le m$.

Маловероятно, что случайный поиск эффективно найдет отображения $S \in \Phi^{m,\,m}$, т.к. большинство перестановок не удовлетворяют критериям, которые определяют $\Phi^{m,m}$. Другим наиболее очевидным подходом после генерации случайной перестановки является нахождение перестановки случайным выбором mm-битных функций, которые будут описывать данную перестановку (это и есть побитовый метод). Данный метод основан на том, что криптографически стойкая m-битная перестановка состоит из m криптографически стойких m-битных функций $f_i : Z_2^m \to Z_2$, $1 \le i \le m$. Таким образом, побитовый метод формирует множество N сбалансированных m-битных функций $\Delta = \{f_1, f_2, ..., f_N\}$, а затем пытается выбрать m функций $F = \left[f_1', f_2', ..., f_m' \right]$ из Δ, которые реализуют перестановку и удовлетворяют накладываемым критериям стойкости. Данный класс методов рассматривался в работах [3 – 7]. Если существует алгоритм A, который производит функции f с заданным показателем P, тогда побитовый метод может быть использован, чтобы объединить эти функции в перестановку.

Очевидно, что побитовый метод формирования S-блоков из-за своей случайной природы имеет определенные ограничения, и нам необходимо определить насколько быстро (как функция от m) этот метод становится невычислимым. В работе [2] 1994 г. приводятся теоретические и экспериментальные расчеты, устанавливающие границы для размеров S-блоков, которые могут быть эффективно построены с использованием данного подхода. Теоретические расчеты работы [2] являются актуальными и на сегодняшний день, однако экспериментальные результаты требуют своей значительной переработки. Далее мы приведем теоретические выкладки из работы [2] и свои экспериментальные результаты, а также сопоставим полученные нами экспериментальные результаты с полученными результатами в работе [2].

Кроме ограничений накладываемых случайной природой метода, побитовый метод может стать неэффективным из-за вычислительно ресурсоемкого поиска по множеству сформированных функций. Как мы увидим далее – нахождение перестановки в множестве функций является вычислительно трудной задачей. По аналогии с работой [2] в экспериментальных исследованиях мы рассмотрели два алгоритма решения этой задачи поиска: жадный алгоритм и алгоритм ветвей и границ.

Анализ из [2] показывает, что для m> 8 задача нахождения любой m-битной перестановки становится невычислимой. Этот результат основан на ожидаемом числе формирований функций, которые должны быть сделаны до того, как перестановка $S \in \Phi^{m,\,m}$ будет найдена. Вместе с этим задача поиска по множеству сформированных функций Δ становится невычислимой уже для m> 6.

2. Оценки эффективности вероятностных методов формирования S-блоков

Начнем наш анализ с определения значения N, для которого $\Delta = \{f_1, f_2, ..., f_N\}$ содержит m функций, которые реализуют перестановку с вероятностью как минимум S. Для этого приведем основные определения и*конечные* формулы расчетов из работы [2], которые будут использованы далее в работе.

Говорят, что m-кортеж $F = [f_1, f_2, ..., f_m]$ реализует перестановку S, если f_i – проекция i-го бита S или $f_i^S = f_i$, $1 \le i \le m$. Кортеж $F = [f_1, f_2, ..., f_m]$ называют *частичной перестановкой*, если F реализует первые k бит некоторой m-битной перестановки $S \in S_{2^m}$.

Обозначим P(N, m) – вероятность того, что множество N сбалансированных m-битных булевых функций содержит подмножество m функций, которые реализуют перестановку. Из [2]:

$$P(N,m) \ge \prod_{0 \le i \le N-m} \left[1 - \binom{m-1+i}{m-1} \cdot \frac{\left(2^{m-1}!\right)^{2m}}{\left(2^m!\right)^{m-1}} \right],$$

откуда следует, что:

$$\ln\left(P(N,m)\right) > -\frac{O(1)\cdot N^{m+\frac{1}{2}}\cdot 2^{\frac{m^2+1}{2}}\cdot \pi^{\frac{m}{2}}}{\left(1-\frac{m}{N}\right)^{(N-m)}\cdot m^{m+\frac{1}{2}}\cdot (N-m)^{\frac{1}{2}}\cdot e^{2^m}}. \quad (1)$$

Используя (1), определим значение N, для которого P(N, m) приблизительно S или $\ln\left(P(N,m)\right) \approx -\ln 2$. Расчетные результаты, приведенные в табл. 1, взяты из работы [2]. В табл. 1 приводится нижняя граница значения N, для которого мы ожидаем, что Δ содержит перестановку с вероятностью S. Значение приводится с округлением к ближайшему значению степени по основанию 2. Табл. 1 ясно показывает, что значение N чрезвычайно велико для m = 8 и невычислимо для m = 10. Невычислимость представляет собой основное ограничение для всех побитовых методов.

Таблица 1

Размер N, для которого P(N, m) > 0.5

m	4	5	6	7	8	9	10
N	2^4	2^7	2^{13}	2^{24}	2^{45}	2^{80}	2^{146}

Теперь приведем основные теоретические результаты для жадного алгоритма поиска. Заранее отметим, что нет гарантий того, что жадный алгоритм найдет перестановку в Δ даже, если известно, что она существует.Предположим, что у нас есть множество сбалансированных функций Δ. Жадный алгоритм, представленный ниже, дает общий побитовый метод для нахождения перестановки в Δ.

Жадный алгоритм G1

случайно выбрать $f_1 \in \Delta; F \leftarrow [f_1]; k \leftarrow 2$;

while $k \le m$ **do**

случайно выбрать $f_k \in \Delta$;

if $\left[f_{i_1}, f_{i_2}, ..., f_{i_{k-1}}, f_k\right]$ – случайная перестановка **then**

$F \leftarrow F \cup \left[f_k\right]$;

$k \leftarrow k+1$;

output $F \leftarrow \left[f_{i_1}, f_{i_2}, ..., f_{i_{m-1}}, f_{k-1}\right]$.

В [4] алгоритм не только проверяет – расширяет ли выбранная функция f_kчастичную перестановку, но проверяет также – удовлетворяет ли она накладываемым криптографическим критериям. Анализ алгоритма G1 будет состоять из определения ожидаемого числа итераций главного цикла **while** для нахождения перестановки (т.е. числа функций, которые будут проверены).

Пусть L_m – ожидаемое число итераций цикла **while** алгоритма G1. Тогда согласно [2]:

$$L_m > O(1)\cdot \pi^{2^{m-2}}\cdot 2^{-\frac{m}{2}}. \quad (2)$$

Расчеты, выполненные в соответствии с (2), представлены в табл. 2.

Таблица 2

Количество функций N, для которых ожидается \нахождение перестановки в Δ жадным алгоритмом

m	4	5	6	7	8
N	2^5	2^{11}	2^{23}	2^{56}	2^{100}

Из табл. 2 видно, что приблизительно 2^{56} функций необходимо проверить, чтобы найти перестановку $S \in S_{2^7}$. Аналогично свыше 2^{100} функций должно быть проверено, чтобы найти 8-битную перестановку, что, очевидно, невычислимо. Тогда можно утверждать, что $m \le 6$ является ограничением для построения S-блоков с использованием жадного алгоритма.

3. Экспериментальные результаты

В данном разделе мы приводим экспериментальные результаты, полученные реализацией побитового метода формирования криптографически стойких перестановок $S \in \Phi^{m,m}$. Экспериментальные исследования проводились по аналогии с работой [2] 1994 года. Хотя приведенные в работе теоретические расчеты остаются актуальными и на сегодняшний день, экспериментальные результаты требуют переработки, т.к. на время написания статьи не существовало развитого математического аппарата для анализа криптостойкостиS-блоков.

Согласно [2] множество $\Phi^{m,m}$ определялось, как множество m-битных перестановок, описываемых функциями, которые максимально нелинейны и удовлетворяют SAC. При этом нелинейность и SAC выражались следующим образом:средняя нелинейность k функций f_1, f_2, …, f_k определялась как $\sum_{i=1}^{k}\delta\left(f, A^m\right)/k$, где $\delta\left(f, A^m\right)$ – расстояние от f до множества m-битных аффинных функций; также среднее SACk переменных x_1, x_2, …, x_k определялась как $\frac{1}{k}\cdot\sum_{i=1}^{k}\sum_{X\in Z_2^m} f(X)\oplus f(X\oplus e_i)$, которое отражает отклонение от оптимального значения 2^{m-1}.

Однако современный подход в анализе стойкости S-блоков в терминах булевых функций предполагает, что криптографические показатели S-блоков определяются не только показателями составляющих его булевых функций, а и показателями всех их линейных комбинаций. При этом за значение показателя S-блока в целом берется не среднее арифметическое показателей отдельных булевых функций, а *наихудший показатель по множеству показателей отдельных булевых функций и всех их линейных комбинаций.*

В наших экспериментах исследуемые криптографические показатели S-блоков в терминах булевых функций были:

1) Нелинейность NL. Для полноты и возможности сопоставления получаемых результатов с работой [2] мы рассмотрели два варианта: нелинейность S-бло-

ка, выраженную как *минимальную* (худшую) нелинейность *толькосоставляющих функций*; нелинейность S-блока – как *минимальную* нелинейность *составляющих функций и всех их линейных комбинаций*.

2) Автокорреляция AC. Аналогично с нелинейностью, автокорреляция вычислялась как *максимум* (худшее значение) автокорреляций составляющих функций – в первом случае, и *максимум*автокорреляций составляющих функций и всех их линейных комбинаций – во втором.

3) Алгебраическая степень Deg. В первом случае вычислялась как *минимальная* (худшая) алгебраическая степень составляющих функций, во втором – как *минимальная* алгебраическая степень составляющих функций и всех их линейных комбинаций.

В наших исследованиях показатель SAC не рассматривался, т.к. согласно работе [8] мы можем через некоторые линейные преобразования привести произвольную булеву функцию либо S-блок к удовлетворению SAC. При этом очевидно, что т.к. преобразования носят линейный характер, сохраняются остальные криптографические показатели преобразуемых функций/S-блоков нелинейность NL, автокорреляция AC и алгебраическая степень Deg. Поэтому побитовый поиск S-блоков по множеству функций, удовлетворяющих SAC, не представляется необходимым. К тому же, как будет показано далее, каждое введение какого-либо критерия отбора функций может существенно сказываться на эффективности поиска.

Мы исследовали два варианта заполнения множества Δ. В первом случае – поисковое множество заполнялось случайными сбалансированными булевыми функциями, во втором – криптографически стойкими. Исследовалось влияние содержимого множества Δ на эффективность нахождения S-блоков в нем.

По аналогии с работой [2] мы рассмотрели два алгоритма поиска по множеству функций Δ: жадный алгоритм и алгоритм ветвей и границ. Жадный алгоритм, как показано в Алгоритме G1, исключает вероятность экспоненциальной работы, проверяя каждую функцию в Δ *ровно один раз*, но при этом нет никаких гарантий, что он найдет перестановку в Δ, даже если она существует. Для того чтобы гарантировать успех нахождения этим алгоритмом *случайной* перестановки, размер Δ должен быть большим в сравнении с приведенным теоретическим в табл. 1. Далее мы покажем, что размер Δ, требуемый для успешного завершения жадного алгоритма, становится еще больше, когда накладываются криптографические ограничения на искомые перестановки. С другой стороны, алгоритм ветвей и границ проверяет *все возможные* m-кортежи из Δ. Поиск направляется граничной функцией, которая удаляет те k-кортежи $F = \left[f_{i_1}, f_{i_2}, ..., f_{i_k} \right]$ из дерева поиска, которые не являются частичными перестановками. Далее мы покажем, что алгоритм ветвей и границ сравним с жадным алгоритмом по времени выполнения, в то время как требуемый размер Δ для гарантии нахождения перестановки значительно меньше.

3.1. Жадный алгоритм

Реализованный нами жадный алгоритм является вариацией основного Алгоритма G1 и представлен ниже, как Алгоритм G2. Пусть накладываемый порядок для $\Delta = \{f_1, f_2, ..., f_N\}$ будет отображен в индексе f_i, так что, если $F = \left[f_{i_1}, f_{i_2}, ..., f_{i_N} \right]$ – перестановка, найденная жадным алгоритмом, то $1 \le i_1 \le i_2 \le$ $\le ... \le i_m \le N$. Заметим, что первый бит, задаваемый вначале функцией f_1, *обязательно* ограничивает множество перестановок, которые могут быть найдены алгоритмом. Данное ограничение является строгим.

Алгоритм G2

сформировать $\Delta \leftarrow \{f_1, f_2, ..., f_N\}$; $F \leftarrow \{0\}$;

for i ← 1 **to**NP**do**

π ← случайная перестановка 1, 2, …, N;

for j ← 1 **to**N**do**

$f_i \leftarrow f_{\pi(i)}$;

k ← 1; numfuncs ← 1;

whilek ≤ m**and**numfuncs ≤ N **do**

if $F \cup [f_k]$ – частичная перестановка **and** $f_k \in \Phi^m$ **then**

$F \leftarrow F \cup [f_k]$;

k ← k + 1;

numfuncs ← numfuncs + 1;

ifk = m + 1 **then**

output $F \leftarrow \left[f_{i_1}, f_{i_2}, ..., f_{i_{m-1}}, f_{k-1} \right]$;

return.

Алгоритм G2 повторяет основной шаг жадного алгоритма NP раз, переставляя элементы Д на каждой итерации для того, чтобы изменить порядок, в котором проверяются функции. Назовем каждую такую перестановку элементов в Δ *пробегом*.

Результаты проведенных экспериментов с помощью реализации Алгоритма G2 с ограничением только на *составляющие* функции для m = 4, 5 приведены в табл. 3, 4. Для m = 4 число пробегов NP задавалось равным 5000, для m = 5 –1000. Колонки таблиц означают следующее: 1) количество функций N, размер Δ; 2) «AI» - среднее число итераций внутреннего цикла **while** в случае, когда перестановка была найдена; 3) «TI» - общее число итераций внутреннего цикла **while** для NP итераций главного цикла; 4) «INC» - число итераций главного цикла, для которых перестановка не была найдена; 5) тип колонки обозначает ограничение, которое накладывалось на отбираемые функции: «L» - высокая нелинейность NL, «A» - низкая автокорреляция AC, «D» - высокая алгебраическая степень Deg. Столбец «Сл.» означает, что на формируемые функции для Δ не накладывалось никаких ог-

раничений (кроме сбалансированности), «Ст.» - множество Δ состояло из *стойких* функций (удовлетворяющих накладываемым критериям стойкости).

Под высокой нелинейностью NL мы понимаем NL = 4 для 4-битных функций и NL≥ 10 для 5-битных функций; под низкой автокорреляцией мы понимаем AC = 8 для обеих размерностей функций; высокая алгебраическая степень означает Deg = 3 и Deg = 4 для 4-х и 5-ти битных функций соответственно.

Таблица 3

Алгоритм G2 для m = 4 cNP = 5000. Ограничения на *составляющие* функции

# ф.	AI		TI		INC		% неудач		тип
	Сл.	Ст.	Сл.	Ст.	Сл.	Ст.	Сл.	Ст.	
100	45.74 ± 0.97	45.31 ± 0.98	277 604	265 147	955	811	19.1	16.2	-
100	47.02 ± 1.01	44.45 ± 0.95	286 437	275 486	974	949	19.4	18.9	L
100	49.95 ± 1.09	45.2 ± 0.98	311 631	271 097	1330	843	26.6	16.8	LA
100	49.66 ± 1.04	45.23 ± 0.97	303 873	260 987	1229	706	24.5	14.1	LAD
200	69.85 ± 1.67	59.28 ± 1.53	367 572	323 338	219	130	4.3	2.5	L
200	70.88 ± 1.71	58.52 ± 1.49	423 610	320 060	405	130	8.1	2.5	LA
200	74.94 ± 2.83	58.36 ± 1.48	362 820	310 764	202	119	4.04	2.3	LAD

Таблица 4

Алгоритм G2 для m = 5 cNP = 1000. Ограничения на *составляющие* функции

# ф.	AI		TI		INC		% неудач		тип
	Сл.	Ст.	Сл.	Ст.	Сл.	Ст.	Сл.	Ст.	
10000	4111.25 ± 205.56	4447.61 ± 206.35	6 463 791	6 299 837	361	345	36.1	34.5	-
10000	4721.25 ± 241.92	4391.92 ± 209.4	7 511 296	6 272 820	530	349	53.0	34.9	L
10000	5371.92 ± 754.38	4246.79 ± 201.7	9 861 911	6 391 897	967	355	96.7	35.5	LA
10000	6673.8 ± 917.12	4391.59 ± 212.98	9 895 423	6 280 032	970	353	97.0	35.3	LAD
20000	8433.57 ± 377.18	7226.49 ± 353.92	11 125 518	7 993 806	255	97	25.5	9.7	L
20000	11580.39 ± 1121.66	7036.79 ± 331.53	19 249 830	8 115 567	897	117	89.7	11.7	LA
20000	11936.32 ± 1079.71	6787.42 ± 327.59	19 434 148	7 958 285	930	107	93.0	10.7	LAD
40000	12288.59 ± 600.26	8899.62 ± 499.99	14 357 709	9 650 750	73	18	7.3	1.8	L
40000	21991.92 ± 1372.66	8888.14 ± 511.75	36 298 294	9 051 221	784	13	78.4	1.3	LA
40000	22814.48 ± 1514.79	9275.04 ± 519.74	37 260 424	9 085 680	836	15	83.6	1.5	LAD
80000	13932.7 ± 865.27	9222.91 ± 524.13	14 522 278	9 402 202	3	0	0.3	0	L
80000	38079.82 ± 2114.65	9747.98 ± 544.63	64 891 000	9 533 935	624	0	62.4	0	LA
80000	41988.4 ± 2224.8	9726.45 ± 602.84	64 405 780	9 844 720	612	2	61.2	0.2	LAD

Таблицы 3-4 показывают, что критерий низкой автокорреляции является значительно ограничивающим эффективность поиска. Например, в табл. 4 из 40 000 функций только 73 из 1000 пробегов потерпели неудачу в нахождении жадным алгоритмом перестановки с максимально нелинейными составляющими функциями. В случаях нахождения перестановокбыли проверены приблизительно 12 338 функций из Δ. Для того же числа функций и пробегов 784 пробега потерпели неудачу в нахождении перестановки, составляющие функции которой удовлетворяли высокой нелинейности и низкой автокорреляции.

Когда такая перестановка находилась, проверялось приблизительно 22 862 функций из Δ. В то же время при наложении еще и показателя алгебраической степени эффективность поиска практически не меняется (как видно из таблиц, разница составляет 1-2%).

В случае же, когда множество Δ состоит из стойких функций, эффективность нахождения S-блоков с заданными показателями практически не меняется при различных накладываемых ограничениях. Как видно из табл. 3-4 разница в эффективности составляет 1-2%.

Следует отметить, что использование множества Δ со стойкими функциями позволило значительно повысить эффективность нахождения перестановок для m = 5. Так, при 30 000 случайных функций в Δ и накладываемых ограничениях по нелинейности и автокорреляции процент неудач поиска составил 86.3%, а для того же числа стойких функций – всего 3.9%. Для размерности m = 4 мы не видим существенной разницы. Это объясняется тем, что с ростом размерности функций вероятность нахождения криптографически стойкой функции среди случайно сформированных функций уменьшается экспоненциально. Мы сформировали 10 000 случайных сбалансированных функций для каждой размерностей m = 4-8 и провели анализ их криптографических показателей. В табл. 5 приведены результаты, наглядно демонстрирующие справедливость сказанного выше. «RG» в таблице означает случайную генерацию.

Таблица 5

Соотношение стойких функций относительно сбалансированных с ростом m.
Число генераций = 10 000

m	NL	RG	(NL, AC)	RG	(NL, AC, Deg)	RG
4	4	8507	(4, 8)	7887	(4, 8, 3)	7887
5	10-12	6432	(10-12, 8)	618	(10-12, 8, 4)	618
			(10-12, 8-16)	5841	(10-12, 8-16, 4)	5841
6	24-26	1282	(24-26, 8-24)	1144	(24-26, 8-24, 5)	1144
			(24-26, 8-16)	211	(24-26, 8-16, 5)	211
			(24-26, 8)	0	(24-26, 8, 5)	0
7	54-56	0	(54-56, 8-24)	0	(54-56, 8-24, 6)	0
8	112-116	0	(112-116, 8-32)	0	(112-116, 8-32, 7)	0

Так, для m = 4 среди 10 000 сформированных случайных булевых функций, 8507 из них обладали высокой нелинейностью (NL = 4), 7887 – высокой нелинейностью и низкой автокорреляцией (NL = 4, AC= 8). Заметим, что число функций с высокой алгебраической степенью осталось таким же. Однако уже для размерности 5 видно, что число криптографически стойких функций с высокой нелинейностью и низкой автокорреляцией существенно сократилось с 7887 функций (для m = 4) до 612 (для m = 5). Начиная с размерности 7, среди 10 000 случайно сгенерированных сбалансированных функций не было найдено ни одной криптографически стойкой функции.

Результаты экспериментов при проверке не только составляющих функций, а и всех их линейных комбинаций приведены в табл. 6-7.

Таблица 6

Алгоритм G2 для m = 4 cNP = 5000. Ограничения на *весь*S-блок

# ф.	AI		TI		INC		% неудач		тип
	Сл.	Ст.	Сл.	Ст.	Сл.	Ст.	Сл.	Ст.	
100	43.89 ± 0.95	44.77 ± 0.97	281 273	283 851	911	1084	18.22	21.68	-
100	58.61 ± 1.93	56.5 ± 1.46	430 695	401 373	3374	2847	67.48	56.94	L
100	59.73 ± 2.06	57.26 ± 1.96	476 453	458 596	4364	4025	87.28	80.5	LA
100	61.3 ± 2.28	60.07 ± 2.23	474 605	463 309	4364	4123	87.28	82.46	LAD
300	124.91 ± 3.19	120.3 ± 3.13	855 241	760 706	1294	971	25.88	19.42	L
300	152.47 ± 4.75	136.13 ± 4.24	1 250 229	1 149 810	3355	2783	67.1	55.66	LA
300	147.88 ± 4.79	135.96 ± 4.28	1 215 500	1 163 484	3137	2899	62.74	57.98	LAD
500	165.93 ± 4.61	148.78 ± 4.33	986 436	886 529	524	405	10.48	8.1	L
500	211.45 ± 6.73	193.05 ± 6.13	1 805 604	1 627 844	2564	2176	51.28	43.52	LA
500	211.93 ± 6.76	195.4 ± 6.31	1 765 171	1 646 712	2418	2192	48.36	43.84	LAD
1000	203.51 ± 6.67	174.29 ± 5.89	1 182 166	1 004 102	196	147	3.92	2.94	L
1000	319.07 ± 11.01	282.45 ± 10.16	2 811 462	2 534 219	1765	1599	35.3	31.98	LA
1000	314.98 ± 10.99	281.62 ± 10.22	2 780 914	2 597 722	1788	1663	35.76	33.26	LAD
1500	215.66 ± 7.58	191.18 ± 7.15	1 247 079	1 098 544	126	124	2.52	2.48	L
1500	375.98 ± 14.12	334.69 ± 13.24	3 685 811	3 324 368	1567	1421	31.4	28.42	LA
1500	378.22 ± 14.17	333.22 ± 13.34	3 485 029	3 272 713	1456	1411	29.12	28.22	LAD

Таблица 7

Алгоритм G2 для m = 5 cNP = 1000. Ограничения на *весь*S-блок. Поиск по стойким функциям

# ф.	AI	TI	INC	% неудач	тип
10000	0	10 000 000	1000	100	L
20000	12 956.5 ± 13142.53	19 985 913	998	99.8	L
30000	0	30 000 000	1000	100	L
40000	30 978 ± 12579.05	39 981 956	998	99.8	L
80000	44 993.25 ± 11855.87	79 719 946	992	99.2	L
100000	50846.66 ± 13520.7	99 557 620	991	99.1	L
200000	124619.23±22773.62	199 020050	987	98.7	L
500000	232165.13±42023.8	487 679596	954	95.4	L

Для случая m = 5 исследование проводилось только для Д, содержащего стойкие функции. Из табл. 7 видно, что процент неудач даже при 500 000 стойких функциях в Δ и единственном накладываемом ограничении по нелинейности составил 95.5%. Заметим, что при рассмотрении аналогичного огра-

ничения по нелинейности только на составляющие функции уже при 40 000 стойких функций процент неудач был равен практически нулю. Аналогичная, хотя немного лучшая, ситуация просматривается и для размерности m = 4. Так, например, из таблиц 3 и 6 видно, что процент неудач поиска при ограничении по нелинейности на весь S-блок и 100 случайных функциях в множестве Δ значительно повысился с 19.4% до 67.48%, для стойкого множества Δ - с 18.9% до 56.94%. Это объясняется тем, что при наложении криптографических ограничений на весь S-блок приходится проверять $2^k - 1$ функций, вместо k, как для случая ограничения только на составляющие функции.

Как и в случае накладываемых ограничений только на составляющие функции, при ограничениях на весь S-блок для m = 4 существенной разницы в эффективности поиска заметить не удалось при использовании стойкого и случайного Δ. Для m = 5 подобное исследование проводить не имело смысла в виду малоэффективности нахождения случайных перестановок даже в стойком множестве Δ.

Заметим, что в работе [7] авторами используется разновидность жадного алгоритма. При этом множество Δ функций не формируется, а сразу фиксируется одна начальная функция, относительно которой идет *весь* дальнейший поиск. Поиск осуществляется по всему множеству Φ^m. Формируется новая функция f_i, проверяется – составляет ли она перестановку/частичную перестановку с уже отобранными функциями. В случае успешной проверки функция f_i, а также все ее линейные комбинации с уже отобранными функциями, проверяются на соответствие заданным критериям нелинейности NL, автокорреляции AC и алгебраической степени Deg. В случае если функция f_i удовлетворяет наложенным ограничениям, она сохраняется, иначе – отбрасывается и формируется следующая. Процесс поиска продолжается до тех пор, пока не будет сформирован S-блок, удовлетворяющий критериям поиска.

3.2. Алгоритм ветвей и границ

Алгоритм ветвей и границ поиска S-блока в Δ упорядочивает k-кортежи функций в дерево, по которому затем происходитпоиск. Каждый узел дерева поиска представляет собой частичную перестановку $\left[f_{i_1}, f_{i_2}, ..., f_{i_k}\right]$, где $1 \le i_1 < i_2 < \ldots < i_k \le N$. Дети узла $\left[f_{i_1}, f_{i_2}, ..., f_{i_k}\right]$ задаются множеством

$$\left\{ \begin{array}{l} \left[f_{i_1}, f_{i_2}, ..., f_{i_k}, f_i\right] \mid i > i_k, \\ \left[f_{i_1}, ..., f_{i_k}, f_i\right] - \text{частичная перестановка} \end{array} \right\}.$$

Узел удаляется из дерева, если кортеж, который он представляет, не является частичной перестановкой либо не удовлетворяет накладываемым криптографическим ограничениям. В нашей работе будем ссылаться на алгоритм ветвей и границ, как на Алгоритм В.

Алгоритм В

сформировать $\Delta \leftarrow \{f_1, f_2, ..., f_N\}$;

$F \leftarrow 0$;

for $i_1 \leftarrow 1$ **to**N**do**

SEARCH(F, i_1, 1);

procedure SEARCH(F, i_k, k)

for $i_{k+1} \leftarrow i_k + 1$ **to** N **do**

if $F \cup \left[f_{i_{k+1}}\right]$ –

частичнаяперестановка**and** $f_{i_k} \in \Phi^m$ **then**

if (k + 1) < m**then**

SEARCH($F \cup \left[f_{i_{k+1}}\right]$, i_{k+1}, k + 1);

else

output $F \cup \left[f_{i_{k+1}}\right]$;

return.

Таблицы 8 – 11 показывают результаты выполнения Алгоритма В для m = 4, 5. Выполнено по 100 пробегов для каждого эксперимента. Под пробегом понимается поиск по одной выборке Δ. Для каждого пробега формируется новая выборка. В табл. 8 и 11 показаны результаты при поиске S-блоков с ограничениями, наложенными только на *компонентные функции*, в табл. 10 и 12 – *на весь S-блок*, т.е. на компонентные функции и их линейные комбинации. Колонки таблицы обозначают следующее: первая колонка N - размер Δ; «AFI» - среднее число функций заданного типа, находимые на каждом пробеге; «IWP» - число пробегов, для которых перестановка была найдена; «API» - среднее число перестановок, находимых на каждом пробеге; последняя колонка – тип искомых функций.

Таблица 8

Алгоритм В с m = 4. NP = 100. Ограничения на *компонентные* функции

# ф.	AFI	IWP		API		тип
	Сл.	Сл.	Ст.	Сл.	Ст.	
20	20	87	85	3.61 ± 0.78	3.87 ± 1.01	-
20	17.28 ± 0.33	77	88	2.4 ± 0.66	3.34 ± 0.62	L
20	15.84 ± 0.41	56	75	1.49 ± 0.45	3.55 ± 0.94	LA
20	15.48 ± 0.41	51	85	1.15 ± 0.42	3.65 ± 0.94	LAD
50	42.42 ± 0.66	100	100	92.68 ± 9.41	178.73 ± 11.85	L
50	39.36 ± 0.71	100	100	67.97 ± 7.46	183.75 ± 12.2	LA
50	39.05 ± 0.85	100	100	69.13 ± 8.59	183.64 ± 12.01	LAD

Таблица 9

Алгоритм B с m = 4. NP = 100. Ограничения на *весь*S-блок

# ф.	AFI	IWP		API		тип
	Сл.	Сл.	Ст.	Сл.	Ст.	
20	17.07 ± 0.39	28	47	0.42 ± 0.21	0.75 ± 0.28	L
20	15.47 ± 0.52	3	17	0.04 ± 0.06	0.19 ± 0.11	LA
20	15.68 ± 0.43	8	11	0.08 ± 0.06	0.14 ± 0.11	LAD
50	41.95 ± 0.57	100	100	17.18 ± 2.36	32.15 ± 3.34	L
50	39.36 ± 0.73	85	99	3.61 ± 0.81	7.52 ± 1.05	LA
50	39.31 ± 0.75	89	99	3.17 ± 0.71	8.88 ± 1.15	LAD
80	62.39 ± 0.89	100	100	21.6 ± 2.76	58.48 ± 4.17	LA
80	62.69 ± 0.98	100	100	22.83 ± 2.65	59.41 ± 4.45	LAD

Таблица 10

Алгоритм B с m = 5. Ограничения на *компонентные* функции

# ф.	AFI	IWP		API		тип
	Сл.	Сл.	Ст.	Сл.	Ст.	
120	120	38	37	0.64 ± 0.006	0.57 ± 0.005	-
120	75.68 ± 0.03	6	44	0.06 ± 0.001	0.61 ± 0.005	L
120	70.31 ± 0.03	4	43	0.04 ± 0.001	0.66 ± 0.005	LA
120	67.57 ± 0.03	6	36	0.06 ± 0.001	0.45 ± 0.004	LAD
500	320.58 ± 2.92	100	100	92.38 ± 5.27	860.22 ± 14.25	L
500	31.22 ± 1.32	0	100	0	898.11 ± 13.31	LA
500	27.64 ± 1.19	0	100	0	905.12 ± 14.48	LAD
1000	62.85 ± 1.82	0	-	0	-	LA
2000	125.57 ± 2.82	54	-	0.98 ± 0.31	-	LA
5000	314.02 ± 3.87	100	-	89.29 ± 6.72	-	LA
5000	269.02 ± 3.86	100	-	40.52 ± 3.77	-	LAD

Таблица 11

Алгоритм B с m = 5. Ограничения на *весь*S-блок

# ф.	AFI	IWP		API		тип
	Сл.	Сл.	Ст.	Сл.	Ст.	
500	318.38 ± 2.57	0	1	0	0.02 ± 0.02	L
1000	639.53 ± 4.63	0	12	0	0.12 ± 0.08	L
2000	1278.12 ± 5.51	38	100	0.52 ± 0.21	5.67 ± 0.63	L
2000	125.12 ± 3.06	0	0	0	0	LA (AC ≤ 8)
2000	1281.49 ± 5.29	0	0	0	0	LA (AC ≤ 16)
5000	312.36 ± 4.38	0	0	0	0	LA (AC ≤ 8)
5000	2898.37 ± 8.11	0	0	0	0	LA (AC ≤ 16)
5000	269.9 ± 4.38	0	0	0	0	LAD (AC ≤ 8)

Прочерки в табл. 10 означают, что выполнять расчеты для 1000 – 5000 функций не имеет смысла, т.к. уже для 500 функций успех нахождения перестановки составил 100%.

Интересно сравнить результаты, полученные жадным алгоритмом и алгоритмом ветвей и границ (табл.3-4 и9-10 соответственно). Так, например, по табл. 10N = 500 стойких функций требуется, чтобы найти 5-битную перестановку, которая удовлетворяет высокой нелинейности и низкой автокорреляции для компонентных функций на каждом пробеге, используя алгоритм ветвей и границ. Из табл.4 видно, что только для N = 30000 стойких функций жадный алгоритм находит практически с такой же вероятностью успеха подобные 5-битные перестановки, удовлетворяющие высокой нелинейности и низкой автокорреляции.

Также интересно сравнить расчетные результаты из табл. 1 с результатами, полученными Алгоритмом B. Табл. 1 указывает, что для m = 4, N = 16 функций требуется, чтобы найти перестановку с вероятностью S. Из табл. 8 видно, что как только найдено свыше 16 функций в Δ, удовлетворяющих ограничениям для данного пробега (при наложении ограничений только на составляющие функции), более половины пробегов находили перестановку. Аналогично, граница из табл. 1 для m = 5 нахождения перестановки с вероятностью S N = 128. Мы видим, что результаты из табл. 10 также согласуются с этой границей.

Рассмотрим теперь случай с m = 6. Из табл. 1 видно, что нам потребуется приблизительно 8000 функций для ожидания содержания 6-битной перестановки в Δ с вероятностью большей S. Чтобы оценить размер дерева поиска, которое проверялось бы Алгоритмом B, используем метод аппроксимации Монте-Карло [9].

АлгоритмМонте-Карло
average ← 0;
for i = 1 **to** NP **do**
  sum ← 0;
  product ← 0;
  k ← 1;
  сформировать $\Delta_k \leftarrow \{f_1, f_2, ..., f_N\}$;
  while $\Delta_k \neq \varnothing$ **do**
    product ← product · $|\Delta_k|$;
    sum ← sum + product;
    f_k ← случайная функция из Δ_k;
    k ← k + 1;
    вычислить Δ_k;
  average ← average + sum;
average ← average/N;
output average;
return.

В Алгоритме B под выборкой Δ_k мы понимаем подмножество функций из Δ, которые составляют частичные решения в совокупности с частичным решением F предыдущего уровня k – 1. Аналитическая оценка редко возможна, потому что тяжело предсказать, как различные ограничения влияют на строение дерева поиска. Здесь мы также применили метод Монте-Карлодля оценки размера дерева при наложении различных криптографических критериев на искомый S-блок.

Идея метода состоит в том, чтобы выполнить определенное число экспериментов, при этом каждый эксперимент состоит в осуществлении поиска по дереву со случайно выбранными значениями a_i. Предположим, что у нас есть частное решение $(a_1, a_2, ..., a_{k-1})$ и число выборов для a_k составляет $x_k = |S_k|$. Если $x_k \neq 0$, то a_k выбирается случайно из S_k, каждый элемент имеет вероятность выбора $1/x_k$.Если $x_k = 0$, эксперимент прекращается. Таким образом, если $x_1 = |S_1|$, $a_1 \in S_1$ выбирается случайно с вероятностью $1/x_1$; если $x_2 = |S_2|$ при выбранномa_1 из S_1, $a_2 \in S_2$ выбирается случайно с вероятностью $1/x_2$ и т.д. Ожидаемое значение

$$x_1 + x_1x_2 + x_1x_2x_3 + x_1x_2x_3x_4 + ...$$

является ожидаемым числом узлов в дереве, т.е. это максимальное число вариантов, которые будут проверены алгоритмом поиска до нахождения решения (в нашем случае S-блока), если оно существует. Алгоритм выполняет NP экспериментов, чтобы аппроксимировать число узлов в дереве. В нашем случае в качествеa_i выступают функции из выборки Δ, в качестве S_k – выборка Δ_k.

Табл. 12 показывает ожидаемое число узлов, которые необходимо проверить Алгоритму B для m = 4, 5, 6 и различного размера Δ. Количество функций в Δ бралось в соответствии с полученными экспериментальными результатами из табл. 8-11 для m = 4, 5, для m = 6 в соответствии с теоретическим расчетом из табл. 1. Предполагается, что Δ - набор сбалансированных функций, удовлетворяющих указанным в таблице ограничениям. Ограничения накладывались, как на основные составляющие функции (F:L, LA, LAD), так и на весь S-блок (S: L, LA, LAD). При накладывании ограничений Δ формировалась только из стойких сбалансированных функций. Заметим, что для m = 5 и m = 6 автокорреляция задавалась AC≤ 16 и AC≤ 24 соответственно, нелинейность для m = 6 задавалась NL≥ 24.

Таблица 12

Оценка методом Монте-Карло числа узлов, проверяемых Алгоритмом B. Число пробегов = 100

m	# ф.	огранич.	ср. размер дерева поиска
4	20	-	136.6 ± 0.79
4	20	F: L	149.8 ± 1.29
4	20	F: LA	136.8 ± 0.83
4	20	F: LAD	127.4 ± 0.87
4	50	S: L	838.5 ± 6.21
4	50	S: LA	570 ± 2.99
4	50	S: LAD	690 ± 4.25
5	120	-	5611.2 ± 40.2
5	500	F: L	321 110 ± 3043.09
5	500	F: LA	402 130 ± 4580.05
5	500	F: LAD	368 525 ± 3534.9
5	2000	S: L	3 251 880 ± 21980.34
5	10000	S: LA	278 475 100 ± 2065128
5	10000	S: LAD	281 436 500 ± 2280479
6	8000	-	355 650 800 ± 3368649
6	8000	F: L	488 942 560 ± 6551134
6	8000	F: LA	566 570 560 ± 6353469
6	8000	F: LAD	407 832 240 ± 4666580
6	8000	S: L	881 840 ± 3123.76
6	8000	S: LA	809 840 ± 3457.16
6	8000	S: LAD	817 600 ± 615.49

Из таблицы видно растущую экспоненциально с возрастанием m размерность дерева поиска. Для 8000 6-битных функций в Δ, число узлов в дереве поиска, проверяемых Алгоритмом B, составило

приблизительно 400 миллионов. Мы делаем вывод, что 6-битные перестановки, которые удовлетворяют высокой нелинейности и низкой автокорреляции, а также имеют высокую алгебраическую степень, могут быть найдены Алгоритмом В, используя большое количество вычислительных ресурсов. Для m = 7, табл. 1 показывает, что Δ должно содержать приблизительно 2^{24} функций прежде, чем будет ожидаться с вероятностью 1/2, что перестановка будет найдена. Для данного числа функций метод Монте-Карло показывает, что число узлов в дереве поиска, которые требуется проверить Алгоритмом В, оказывается больше 2^{40} функций, и мы заключаем, что Алгоритм В не вычислим для m> 6.

Выводы

Наши результаты указывают, что нахождение *m*-битных перестановок (биективных S-блоков), используя побитовые методы, для m> 6, которые удовлетворяют критериям высокой нелинейности, низкой автокорреляции и высокой алгебраической степени – невычислимая задача. Побитовый метод, использующий алгоритм ветвей и границ может быть использован для нахождения 6-битного S-блока, которая удовлетворяет высокой нелинейности и низкой автокорреляции, но потребует нетривиального количества компьютерных ресурсов. Возможно, существуют более эффективные алгоритмы поиска для побитового метода с m≤ 6, но вне зависимости от поискового алгоритма, размер Δ, требуемый для ожидания того, что перестановка будет найдена, чрезвычайно велик для m≥ 8. Таким образом, мы заключаем, что классический метод синтеза стойких узлов замен является непрактичным уже для m = 6 и невычислимым для m > 6.

Список литературы

1. *Кузнецов А.А. Методика исследования эффективности нелинейных узлов замен симметричных криптографических средств защиты информации / А.А. Кузнецов, Ю.А. Избенко, И.В. Московченко // Збірник наукових праць ДонІЗТ. – Донецьк: ДонІЗТ. – 2008. – № 14. – С. 74-81.*
2. *O'Connor L.J. An analysis of a class of algorithms for S-box construction / L.J.O'Connor //Cryptology. – 1994. – 7(3) – P. 133-151.*
3. *Webster A.F. On the design of S-boxes / Webster A.F., Tavares S.E. //CRYPTO 85, H.C. Williams ed.,Advances in Cryptology, Lecture Notes in Computer Science,Springer-Verlag. – 1986. – vol. 218. – P. 523-534.*
4. *Adams C. The structured design of cryptographically good S-boxes / C. Adams, S.Tavares // Journal of Cryptology. – 1990. – 3(1). – P.27-41.*
5. *Forre R. Methods and instruments for designing S-boxes / Forre R. // Journal of Cryptology. – 1990. – 2(3). – P.115-130.*
6. *Dawson M.H. A unified framework for substitution box design based on information theory / M.H. Dawson // Master's thesis, Queen's University, Kingston, Ontario, Canada. – 1991.*
7. *Сорока Л.С. Вероятностная модель формирования нелинейных узлов замен для симметричных криптографических средств защиты информации / Л.С. Сорока, А.А. Кузнецов, И.В. Московченко, С.А. Исаев // Системи обробки інформації, Харківський університет Повітряних Сил, Харків. – 2009. – Вип. 3 (77). – С. 286 – 294.*
8. *Seberry J. Improving the Strict Avalanche Characteristics of Cryptographic Functions / J. Seberry, X. Zhang, Y. Zheng // Information Processing Letters. – 1994. – 50(1). – P. 37-41.*
9. *Reingold E.M. Combinatorial Algorithms: Theory and Practice / E.M. Reingold, J. Nievergeld, N. Deo // Prentice-Hall. – 1976.*

Поступила в редколлегию 2.11.2011

Рецензент: д-р техн. наук, проф. Ю.В. Стасев, Харьковский университет Воздушных Сил им. И. Кожедуба, Харьков.

ДОСЛІДЖЕННЯ ЙМОВІРНІСНИХ МЕТОДІВ ФОРМУВАННЯ НЕЛІНІЙИХ ВУЗЛІВ ЗАМІН

Л.С. Сорока, О.О. Кузнецов, С.О. Ісаєв

Розглядуєтся ймовірнісні методи формування нелінійних вузлів замін (S-блоків). Аналізується жадібний алгоритм і алгоритм гілок і меж пошуку S-блоків по множині сформованих функцій, проводяться експериментальні дослідження їх ефективності з урахуванням різних критеріїв стійкості S-блоків (нелінійності, автокореляції та алгебраїчної ступені). Наводяться теоретичні оцінки ефективності ймовірнісних методів, які зіставляються з отриманими експериментальними результатами досліджень.

Ключові слова: *нелінійний вузол замін, побітовий метод формування S-блоків, бієктивність, нелінійність, автокореляція, алгебраїчна ступінь.*

RESEARCH OFPROBABILISTIC METHODS OF NONLINEAR SUBSTITUTION BOXES CONSTRUCTION

L.S. Soroka, A.A. Kuznetsov, S.A. Isaev

Probabilistic methods for constructing the nonlinear substitution boxes (S-boxes)are considered. The greedy algorithm and branch and bound search for S-boxes over the set of constructed functions are analyzed, experimental research of their effectiveness taking into account different criteria of strength (nonlinearity, autocorrelation and algebraic degree) is carried out. Theoretical evaluation of the effectiveness of probabilistic methods is presented, which is compared with obtained experimental results of research.

Keywords: *nonlinear substitution box, bit-by-bit method of S-box construction, bijectivity, nonlinearity, autocorrelation, algebraic degree.*