6. *Metropolis N. and Ulam S.* The Monte Carlo method // J. Amer. Statistical Association. 1949. V. 44. No. 247. P. 335–341.
7. *Boros E. and Hammer P.* Pseudo-Boolean optimization // Discr. Appl. Math. 2002. V. 123, Iss. 1–3. P. 155–225.
8. *Glover F. and Laguna M.* Tabu Search. Norwell: Kluwer Academic Publishers, 1997.
9. ЦКП Иркутский суперкомпьютерный центр СО РАН. `http://hpc.icc.ru`
10. *Отпущенников И. В., Семенов А. А.* Технология трансляции комбинаторных проблем в булевы уравнения // Прикладная дискретная математика. 2011. № 1(11). С. 96–115.
11. *Otpuschennikov I., Semenov A., Gribanova I., et al.* Encoding cryptographic functions to SAT using TRANSALG system // Frontiers in Artificial Intelligence and Applications. 2016. V. 285. P. 1594–1595.
12. *Anderson R.* A5 (Was: Hacking digital phones). Newsgroup Communication. 1994. `http://yarchive.net/phone/gsmcipher.html`.

УДК 519.719.2 DOI 10.17223/2226308X/12/39

ОБ АППАРАТНОЙ РЕАЛИЗАЦИИ ОДНОГО КЛАССА БАЙТОВЫХ ПОДСТАНОВОК

Д. Б. Фомин, Д. И. Трифонов

Рассмотрены вопросы реализации на ПЛИС и СБИС одного класса подстановок и проведено сравнение с реализациями произвольных байтовых отображений. Изучен способ реализации произвольной подстановки. Показано, что любая подстановка на множестве V_8 может быть реализована с использованием 40 LUT (812 GE). Для одного класса подстановок на множестве V_8, обладающего высокими криптографическими свойствами, показана возможность реализации с использованием 19 LUT (147 GE).

Ключевые слова: *S-Box, подстановка, ПЛИС, СБИС.*

Согласно критерию Шеннона [1], каждая криптографически безопасная функция должна представлять собой композицию функций, реализующих свойство перемешивания и рассеивания. Наиболее широко распространённый способ обеспечить свойство перемешивания — использование нелинейных преобразований, в частности подстановок. Подстановки являются неотъемлемой частью большого класса криптографических функций, таких, как поточные и блочные шифры, хэш-функции. К подстановкам предъявляются требования, позволяющие гарантировать невозможность применимости известных методов криптографического анализа, таких, как линейный, алгебраический и разностный.

Помимо криптографических требований, также предъявляются требования и к реализации подстановок, что порождает подходы к построению подстановок больших размерностей с использованием преобразований меньших размерностей. Это позволяет добиться возможности:

— программной реализации с большими таблицами замен;
— программной реализации с меньшим количеством битовых преобразований (bitslice-реализации [2]);
— использования подстановок для низкоресурсной реализации на ПЛИС и СБИС;
— эффективного аппаратного маскирования [3, 4].

Известно большое количество способов построения подстановок с использованием преобразований меньшей размерности: на основе сети Фейстеля [5 – 7], с использовани-

ем конструкции типа Misty [5, 8, 9], SPN-сети [10 – 12] и др. [13 – 15]. В данной работе рассмотрен ещё один класс подстановок и исследованы вопросы его аппаратной реализации.

Обозначим $\mathbb{F}_{2^n}$ — конечное поле из 2^n элементов и V_n — векторное пространство размерности n элементов поля $\mathbb{F}_2$. Каждый элемент поля $a \in \mathbb{F}_{2^n}$ может быть представлен как n-битовый вектор $a = (a_0, a_1, \ldots, a_{n-1})$, $a_i \in \mathbb{F}_2$, $i = 0, \ldots, n-1$.

Рассмотрим один способ построения $2m$-битовых подстановок, задав подстановку на прямом произведении $\mathbb{F}_{2^m} \times \mathbb{F}_{2^m}$.

Определение 1. Пусть $\overline{x}_1, \overline{x}_2 \in \mathbb{F}_{2^m}$, π_1, π_2, $\widehat{\pi}_1$, $\widehat{\pi}_2$ — подстановки на $\mathbb{F}_{2^m}$. Подстановку $F_A : \mathbb{F}_{2^m} \times \mathbb{F}_{2^m} \to \mathbb{F}_{2^m} \times \mathbb{F}_{2^m}$, определяемую уравнениями

$$\overline{y}_1 = \begin{cases} \pi_2\left((\overline{x}_2)^2 \cdot \pi_1\left(\overline{x}_1\right)\right), & \overline{x}_1 \neq 0; \\ \widehat{\pi}_2\left(\overline{x}_2\right), & \overline{x}_1 = 0, \end{cases}$$

$$\overline{y}_2 = \begin{cases} \pi_1\left(\overline{x}_1\right) \cdot \overline{x}_2, & \overline{x}_2 \neq 0; \\ \widehat{\pi}_1\left(\overline{x}_1\right), & \overline{x}_2 = 0, \end{cases}$$

будем называть подстановкой типа «A».

Данная подстановка предложена в [16] и её криптографические характеристики (как и криптографические характеристики некоторых других классов подстановок) теоретически обоснованы в [17]. В наиболее интересном с точки зрения практического применения случае $m = 4$ такая конструкция при подходящем выборе параметров позволяет построить 6-равномерную подстановку с нелинейностью 20 и алгебраической степенью 7.

Рассмотрим вопрос сложности реализации подстановок типа «A» на ПЛИС, который может быть оценён количеством используемых ресурсов ПЛИС, таких, как количество ячеек памяти и таблиц замены (LUT), которые в современных ПЛИС фирмы «Xilinx» реализуют произвольную булеву функцию от шести переменных. Для этого реализуем подстановки такого типа с использованием системы автоматизированного проектирования (САПР) Xilinx Vivado 2018.2 на ПЛИС Kintex-7 KC705 Evaluation Platform (xc7k325tffg900-2). В качестве стратегии оптимизации в части Synthysis выбрана стратегия «Flow Area Optimized high», а в части Implemetation — стратегия «Area Explore». Исследуем также вопрос эффективности реализации на СБИС, который оценивается в условных вентилях (GE). Количество GE оценивалось в САПР ISE 9.2i на ПЛИС XC5VLX3 семейства Virtex5.

Для корректности сравнения рассмотрим три варианта реализации 8-битовой подстановки типа «A»:

— реализация «в лоб», с помощью которой можно реализовать произвольное отображение $V_8 \to V_8$;
— реализация произвольного отображения $V_8 \to V_8$ с использованием координатных функций, которое позволяет существенно сократить используемые ресурсы;
— реализация подстановки типа «A».

В случае реализации произвольного отображения $V_8 \to V_8$ «в лоб» происходит запись таблицы значений преобразования в память. Экспериментальные исследования показали, что тип памяти, в которой хранится данная таблица, не влияет на оценку GE, необходимых для её хранения. Таким образом, будем рассматривать память

типа BRAM. При реализации на ПЛИС будем использовать встроенные ячейки памяти. Результаты оценки количества GE, необходимых для реализации данной памяти, показали, что для реализации отображения в табличном виде необходимо 65 558 GE.

Для уменьшения количества GE, необходимых для реализации отображения $V_8 \to V_8$, применим следующий подход. Так как LUT рассматриваемых ПЛИС реализуют произвольную булеву функцию от шести переменных, можно разбить входной вектор на две части: первые 2 бита и оставшиеся 6 бит соответственно. Рассмотрим отображение $f : V_8 \to V_8$, а также функции f_i, $i = 1, 2, 3, 4$, $f_i : V_8 \to V_8$, которые существенным образом зависят лишь от шести переменных, причём

$$f(x_1, x_2, x_3, \ldots, x_8) = \begin{cases} f_1(0, 0, x_3, \ldots, x_8), & \text{если } x_1 = 0, x_2 = 0; \\ f_2(0, 1, x_3, \ldots, x_8), & \text{если } x_1 = 0, x_2 = 1; \\ f_3(1, 0, x_3, \ldots, x_8), & \text{если } x_1 = 1, x_2 = 0; \\ f_4(1, 1, x_3, \ldots, x_8), & \text{если } x_1 = 1, x_2 = 1. \end{cases}$$

Таким образом, для реализации каждой функции f_i, $i = 1, 2, 3, 4$, необходимо 6 LUT (ровно по одному LUT для реализации каждой из шести координатных функций). Для реализации мультиплексора (т. е. функции выбора выходной функции) необходимо ещё 8 LUT. Суммарное количество LUT, необходимых для данной реализации функции f, равно 40.

Экспериментальные исследования показали, что для реализации отображения $f : V_8 \to V_8$ потребовалось 812 GE. Это примерно в 80 раз меньше, чем при реализации этого же отображения с использованием памяти.

Для реализации подстановки типа «A» требуется реализовать четыре подстановки на двоичных векторах длины 4, две операции сравнения, две операции сложения и два мультиплексора (см. определение 1). Экспериментальные исследования показали, что для реализации данной конструкции на ПЛИС необходимо 19 LUT. Это более чем в 2 раза меньше по сравнению с реализацией 8-битовой подстановки с использованием координатных функций. Результаты оценки количества GE, необходимых для реализации подстановки типа «A», показали, что необходимо лишь 147 GE — примерно в 5,5 раз меньше, чем требуется для реализации произвольной 8-битовой подстановки при помощи координатных функций, и почти в 446 раз меньше, чем при реализации этого же отображения с использованием памяти.

Полученные результаты позволяют утверждать, что подстановки, рассмотренные в [16], могут быть использованы при синтезе стойких низкоресурсных примитивов. В [17, 18] подстановки, обобщающие конструкции [16], потенциально могут использовать меньше ресурсов ПЛИС и СБИС, как, например, следующая подстановка $S(\overline{x}_1, \overline{x}_2) = (\overline{y}_1, \overline{y}_2)$, $\overline{x}_i, \overline{y}_i \in \mathbb{F}_{2^m}$, $i = 1, 2$, для реализации которой необходимо реализовать две подстановки на двоичных векторах длины 4, по две операции сравнения и сложения и два мультиплексора (подстановка типа «G», рассмотренная в [17]):

$$a = \overline{x}_1^{-1},\ b = \overline{x}_2^{-1},\ c = a \cdot b,\ d = a \cdot \overline{x}_2;$$

$$S(\overline{x}_1, \overline{x}_2) = (\overline{y}_1, \overline{y}_2),$$

$$\text{где} \quad \overline{y}_1 = \begin{cases} a, & c \neq 0, \\ c, & c = 0, \end{cases} \quad \overline{y}_2 = \begin{cases} b, & d \neq 0, \\ d, & d = 0. \end{cases}$$

ЛИТЕРАТУРА

1. *Shannon C.* Communication theory of secrecy systems. // Bell System Technical J. 1949. No. 28. P. 656–715.

2. *Rebeiro C., Selvakumar D., and Devi A. S. L.* Bitslice implementation of AES // Cryptology and Network Security. 2006. P. 203–212. https://link.springer.com/chapter/10.1007/11935070_14.
3. *Boss E., Grosso V., Tim Güneysu T., et al.* Strong 8-bit sboxes with efficient masking in hardware // J. Cryptographic Engineering. 2017. No. 7(2). P. 149–165.
4. *Kutzner S., Nguyen P. H., and Poschmann A.* Enabling 3-share threshold implementations for all 4-bit s-boxes // LNCS. 2013. V. 8565. P. 91–108.
5. *Canteaut A., Duval S., and Leurent G.* Construction of lightweight s-boxes using Feistel and MISTY structures (full version) // Cryptology ePrint Archive. 2015. No. 2015(711).
6. *Lim C. H.* CRYPTON: A New 128-bit Block Cipher — Specification and Analysis. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.52.5771. 1998.
7. *Gérard B., Grosso V., Naya-Plasencia M., and Standaert F.-X.* Block ciphers that are easier to mask: How far can we go? // LNCS. 2013. V. 8086. P. 383–399.
8. *Matsui M.* New block encryption algorithm MISTY // LNCS. 1997. V. 1267. P. 54–68.
9. *Grosso V., Leurent G., Standaert F.-X., and Varici K.* Ls-designs: Bitslice encryption for efficient masked software implementations // LNCS. 2014. V. 8540. P. 18–37.
10. *Standaert F.-X., Piret G., Rouvroy G., et al.* ICEBERG : An involutional cipher efficient for block encryption in reconfigurable hardware // LNCS. 2004. V. 3017. P. 279–299.
11. *Rijmen V. and Barreto P.* The Khazad Legacy-Level Block Cipher. https://www.researchgate.net/publication/228924670_The_Khazad_legacy-level_block_cipher. 2018.
12. *Lim C.-H.* A revised version of Crypton — Crypton v1.0 // LNCS. 1999. V. 1636. P. 31–45.
13. *Stallings W.* The Whirlpool secure hash function // Cryptologia. 2006. No. 30(1). P. 55–67.
14. *Perrin L., Udovenko A., and Biryukov A.* Cryptanalysis of a theorem: Decomposing the only known solution to the big APN problem (full version) // Cryptology ePrint Archive. 2016. No. 2016(539).
15. *De la Cruz Jiménez R. A.* On some methods for constructing almost optimal s-boxes and their resilience against side-channel attacks // Cryptology ePrint Archive. 2018. No. 2018(618).
16. *Fomin D.* New classes of 8-bit permutations based on a butterfly structure // CTCrypt'18. 2018. https://ctcrypt.ru/files/files/2018/09_Fomin.pdf
17. *Fomin D.* On the way of constructing $2n$-bit permutations from n-bit ones // CTCrypt'19. 2019 (в печати).
18. *Фомин Д. Б.* О подходах к построению низкоресурсных нелинейных преобразований // Обозрение прикладной и промышленной математики. 2018. Т. 25(4). С. 379–381.

УДК 519.17 DOI 10.17223/2226308X/12/40

О ПАРАМЕТРАХ ГЕНЕРАТОРА РАУНДОВЫХ КЛЮЧЕЙ АЛГОРИТМА 2-ГОСТ

В. М. Фомичев, А. М. Коренева, А. И. Тулебаев

Необходимость защиты информации в условиях ограниченных ресурсов определяет актуальность построения облегченных реализаций для известных криптографических алгоритмов. В 2014 г. была представлена низкоресурсная реализация ГОСТ 28147-89 под названием 2-ГОСТ. Несмотря на достоинства, схема имела потенциал в части усиления криптографической стойкости, в том числе за счёт модификации ключевого расписания. В 2018 г. предложен новый алгоритм генерации раундовых ключей для 2-ГОСТ на основе регистра сдвига длины 8 над множеством двоичных векторов длины 32. Вместе с тем параметры обратной свя-