МОДИФИКАЦИИ ПОТОЧНОГО ШИФРА RC4

MODIFICATIONS OF STREAM CIPHER RC4

A. Zelenoritskaya
M. Ivanov

Summary. In this paper, we consider possible modifications of the stream cipher RC4, which became widespread due to its properties, such as elegant design and an effective software and hardware implementation. We present an enhancement version of the RC4 algorithm, which obtained by adding parallel LFSR and stochastic transformation boxes (R-boxes). The result of proposed modification is the increased complexity of dependence between the key and the initial state of the cipher, as well as increased bit depth of the internal state.

Keywords: Stream cipher, RC4, Spritz, LFSR, R-box.

Зеленорицкая Анастасия Викторовна
Национальный исследовательский ядерный университет «МИФИ»
Иванов Михаил Александрович
Д.т.н., профессор, РГУ нефти и газа (НИУ) имени И. М. Губкина
fkb-info@gubkin.ru

Аннотация. Рассматриваются возможные модификации поточного шифра RC4, который благодаря своим свойствам, в первую очередь элегантному дизайну и эффективной программной и аппаратной реализации, получил широкое распространение. Анализируется возможность внедрения в структуру алгоритма параллельных LFSR и блоков стохастического преобразования (R-блоков). Результатом таких модификаций является усложнение зависимости между ключом и начальным состоянием шифра, а также увеличение разрядности внутреннего состояния.

Ключевые слова: поточный шифр, RC4, Spritz, R-блок.

Введение

Поточный, байт-ориентированный шифр RC4 был разработан Р. Ривестом в 1987 году. Сначала алгоритм являлся коммерческой тайной, но спустя семь лет шифр был анонимно разослан и выложен в Интернет [1]. С тех пор шифр интенсивно изучался и нашел широкое применение (SSL/TLS, WEP, Oracle secure SQL и др.). Базовая идея автора RC4 понятна, наличие каждого элемента шифра можно легко объяснить. Благодаря элегантному дизайну, эффективной программной и аппаратной реализации шифр получил широкое распространение и до сих пор является самым популярным поточным шифром.

Достоинства алгоритма:

- Большой размер внутреннего состояния (8-разрядная таблица замен S и два 8-разрядных счетчика $Q1$ и $Q2$ ($8 \times 256 + 16$ бит);
- Быстрые функции обновления состояния и получения выходного значения;
- Масштабируемость: шифр хорошо работает при любой разрядности $n > 2$ (не только для $n = 8$);
- Ключ, состоящий из произвольного числа n-разрядных элементов, может быть любой длины вплоть до $2n$;
- Универсальный самодостаточный алгоритм формирования таблицы замен;
- В процессе работы криптоалгоритма состояние шифра меняется: за $2n$ шагов гарантированно полностью изменяется таблица замен.

Как показывает история развития криптографии с течением времени криптоалгоритмы ослабевают, в том числе по причине разработки новых видов атак и обнаружения новых уязвимостей. Кроме того, известны неудачные примеры реализации RC4, например, в протоколе WEP [2]. Со временем были выявлены статистические слабости как алгоритма инициализации S-блока, так и функции генерации псевдослучайных чисел (ПСЧ). Основные найденные уязвимости алгоритма [3]:

- алгоритм инициализации S-блока в некоторых случаях позволяет по состоянию шифра восстановить ключ;
- зависимость начальных значений в таблице замен S-блока от ключа;
- встречаются коллизии ключей: разные ключи переключают шифр в одно и то же состояние;
- в некоторых случаях возможно определение текущего состояния шифра по выходной псевдослучайной последовательности;
- первые байты ключевого потока с выхода генератора псевдослучайных чисел (PRNG) всегда неслучайны, в результате по этим байтам можно делать предположения об используемом ключе. В результате в тех случаях, когда долговременный ключ и nonce просто склеиваются для создания ключа шифра, этот долговременный ключ может быть получен с помощью анализа достаточно большого количества сообщений, зашифрованных с использованием данного ключа.

Таблица 1. Формирование выходного байта и изменение состояния шифра в RC4 и Spritz.

RC4	Spritz	Комментарий
1: $i = i + 1$	1: $i = i + w$	Параметр w является константой. В RC4 $w = 1$, а в Spritz параметр w может принимать любое нечетное значение.
2: $j = j + S[i]$	2: $j = k + S[j + S[i]]$ 2a: $k = i + k + S[j]$	В RC4 для обхода массива S использовались параметры i и j. В Spritz к ним добавляется параметр k.
3: SWAP($S[i]$; $S[j]$)	3: SWAP($S[i]$; $S[j]$)	-
4: $z = S[S[i] + S[j]]$	4: $z = S[j + S[i + S[z + k]]]$	См. пункт 2
5: Return z	5: Return z	-

И хотя RC4 все еще можно использовать, естественно относясь с повышенным вниманием к обнаруженным уязвимостям, ему на замену ищутся новые, более стойкие альтернативы.

Поточный шифр Spritz

Самая серьезная модификации, алгоритм Spritz, была предложена самим автором RC4 [4]. В этой модификации были сохранены основные особенности оригинального шифра, при этом авторы постарались закрыть найденные уязвимости, сохранив главные достоинства алгоритма — высокое быстродействие и простоту реализации. Серьезным нововведением является внедрение в архитектуру Spritz конструкции Sponge. Sponge — это итеративная конструкция на основе преобразования (псевдослучайной перестановки) F для создания функции, принимающей на входе строку данных произвольной длины («впитывание») и дающей на выходе в качестве результата строку данных также произвольной длины («выжимание»). Многие свойства функций Sponge основываются на том, что нет никаких характеристик преобразования F, которые могли бы быть полезны для атакующего. Конструкция имеет внутреннее состояние фиксированного размера b (бит), которое разделено на две части — первая имеет разрядность r, а вторая — разрядность c. Значение r называется битовой скоростью, а значение c — мощностью [5, 6].

Стойкость шифра зависит от значения c: этот параметр показывает, какое количество информации останется без изменения при впитывании новой порции данных и будет изменено только во время преобразования f. Атакующему потребуется количество времени, равное

$$O\left(2^{\frac{c}{2}}\right),$$

для поиска коллизии в пространстве состояний. Мощность Spritz при значении параметра $n = 8$ составляет не менее 112 байт (именно последние 112 байт S-блока остаются нетронутыми функцией «Впитывание»). Помимо повышения криптостойкости, Spritz, в отличие от RC4, можно использовать не только как поточный шифр. По задумке авторов, Spritz может быть использован в качестве хеш-функции и генератора имитовставки.

В табл. 1 приведены операции ядра шифра Spritz (формирование выходного байта и изменение состояния шифра) и его отличия от RC4.

Spritz — не единственная модификация алгоритма RC4. До и после нее появлялись и другие варианты улучшения RC4.

Основные строительные блоки поточных шифров

Существует огромное количество поточных шифров, однако какого-то единого универсального подхода к их синтезу, как это имеет место в мире блочных шифров, нет. Однако все же можно выделить основные строительные блоки поточных криптоалгоритмов, это регистры сдвига с линейной обратной связью (LFSR, Linear Feedback Shift Register), блоки замен (S-блоки) и блоки стохастического преобразования (R-блоки).

LFSR. Исторически первыми поточными криптоалгоритмами были именно шифры на LFSR. LFSR по сути — это простейший PRNG. Однако сейчас эти устройства используются лишь в качестве строительных блоков. Логика работы LFSR описывается системой линейных уравнений и поэтому имея фрагмент выходной последовательности небольшой длины, можно легко восстановить структуру устройства.

Существуют две конструкции последовательных LFSR: схема Галуа и схема Фибоначчи. На рис. 1 показан пример LFSR, построенного по схеме Фибоначчи, где Q_i — состояние i-го элемента памяти LFSR, а $\oplus$ — операция сложения по модулю два. Характер обратных связей устройства определяется видом образующего примитивного многочлена

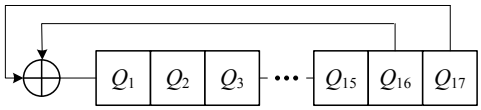

Рис. 1. Схема последовательного LFSR при Φ(x) = x17 + x16 + 1.

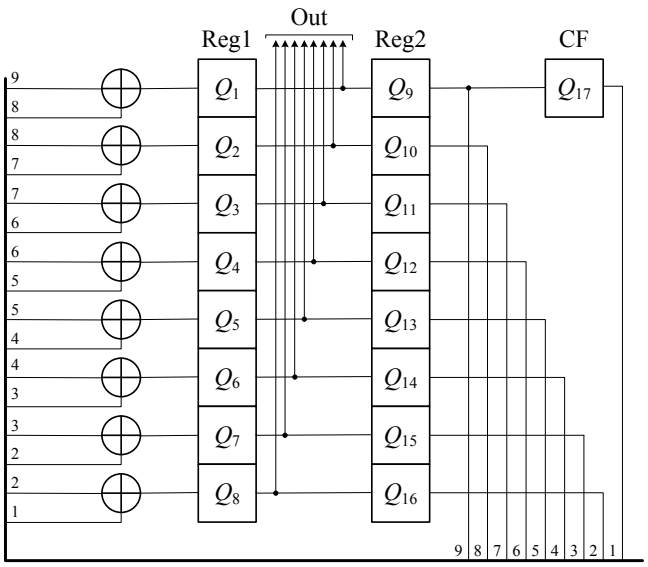

Рис. 2. Схема параллельного LFSR при Φ(x) = x17 + x16 + 1 и n = 8.

$\Phi(x) = \varphi(x\text{-}1)\ x\text{-}N$, где $\varphi(x\text{-}1)$ — характеристический многочлен, N — степень многочленов $\Phi(x)$ и $\varphi(x)$ [7].

Рассмотренные LFSR последовательного типа могут использоваться только для генерации битовой ПСЧ, которая снимается с выхода одного из разрядов регистра. Если необходима n-разрядная последовательность, необходимо использовать LFSR параллельного типа.

Выбираем образующий многочлен степени $N > n$, выбираем схему Фибоначчи или Галуа и синтезируем PRNG, работающий в n раз быстрее исходного LFSR (иначе говоря, выполняющего за один такт своей работы преобразования, которые в исходном LFSR выполняются за n тактов). При этом следует помнить, что если числа n и $S = 2N - 1$ не являются взаимно простыми, генератор ПСЧ вырождается, так как длина формируемой им последовательности оказывается существенно меньше максимально возможной длины, равной S. При программной реализации наибольшее быстродействие достигается в случае использования конструкции Фибоначчи и разреженного многочлена (многочлена с относительно небольшим числом ненулевых коэффициентов) $\Phi(x)$. На рис. 2 приведен пример построения такого параллельного LFSR.

Как показано ниже, один такт работы устройства требует использования всего трех ассемблерных инструкций.

```
; Вход:
; CF ‖ Reg2 ‖ Reg1 = (Q17 Q16 … Q9 Q8 … Q1) — «старое» состояние PRNG
RCR Reg2; Циклический сдвиг вправо через флаг переноса CF, CF = Q9
XCNG Reg1, Reg2; Reg2 = (Q8 Q7 Q6 Q5 Q4 Q3 Q2 Q1)
XOR Reg1, Reg2; CF ‖ Reg2 ‖ Reg1 — "новое" состояние PRNG
```

***S*-блоки.** Блок замен — это важнейший криптографический примитив, базовый элемент всех современных блочных и многих поточных шифров [8–12]. Как уже отмечалось выше, в шифре RC4 специфирован универсальный алгоритм формирования блоков замен приемлемого для большинства приложений качества.

***R*-блоки.** В [13] предложен блок стохастического преобразования (R-блок), который может эффективно использоваться для решения различных задач защиты информации. Схема одного из возможных вариантов построения R-блока, впервые предложенного в работе

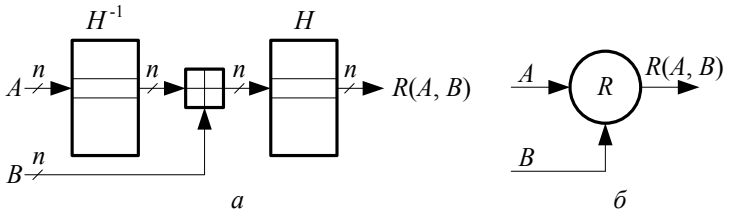

Рис. 3. Логика работы R-блока (а) и его условное графическое обозначение (б).
⊞ — сумматор по модулю 2^n.

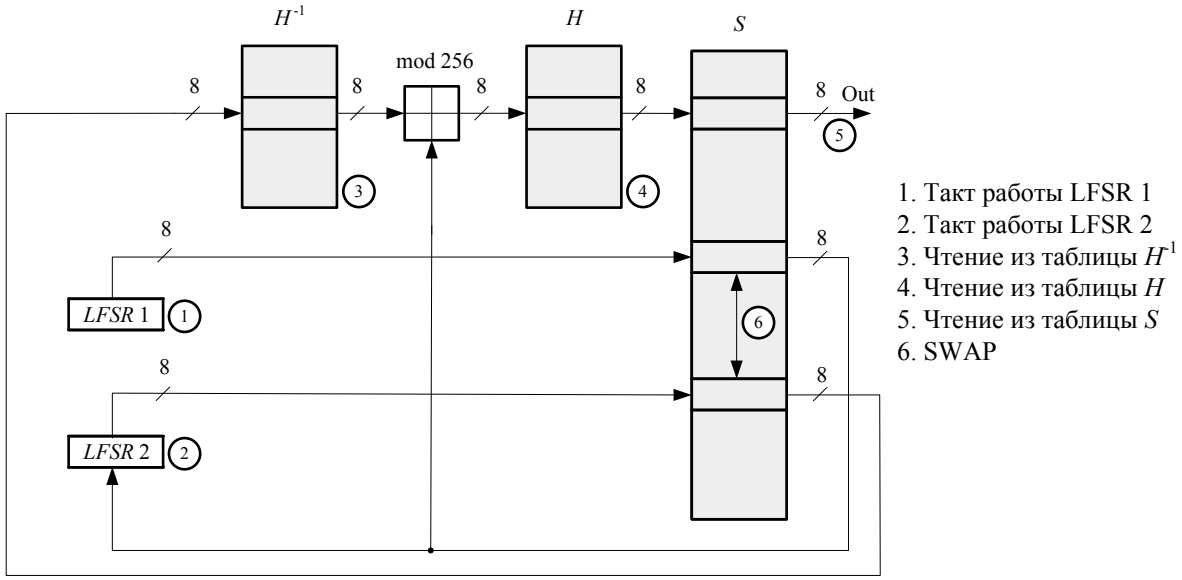

Рис. 4. Вариант ядра модифицированного PRNG RC4.

[14] для обеспечения универсальной защиты информации, пересылаемой по каналу связи, и его условное графическое обозначение показаны на рис. 3. Ключевая информация n-разрядного R-блока — заполнение таблицы $H = \{H(m)\}, m = 0,\ldots,(2^n - 1)$, размерности $n \times 2^n$, содержащей элементы GF(2n), перемешанные случайным образом, т.е. $H(m) \in \mathrm{GF}(2^n)$. Результат $R_H(A,B)$ стохастического преобразования входного n-разрядного двоичного набора A зависит от заполнения таблицы H и параметра преобразования B, задающего смещение в таблице относительно ячейки, содержащей значение A, следующим образом $R_H(A,B) = H\left((m_A + B) \bmod 2^n\right)$, где mA — адрес ячейки таблицы H, содержащей код A, т.е. $H(m_A) = A$. Иначе говоря, результат работы R-блока суть считывание содержимого ячейки таблицы H, циклически смещенной на B позиций в сторону старших адресов относительно ячейки, содержащей код A. Для обеспечения независимости времени преобразования от исходных данных в состав R-блока вводится таблица H-1 = $\{H\text{-}1(j)\}$ размерности $n \times 2^n$, причем $\forall j$ = 0, 1, …, (2n — 1) H-1(j) = mj. Иными словами ячейка с адресом j в массиве H-1 хранит адрес ячейки массива H, содержащей код j. Интересно отметить, что, во-первых, при H-1 = {0, 1, …, (2n — 1)} B = 0 получаем классический S-блок с таблицей замен H; во-вторых, при записи в каждую ячейку массивов H и H-1 ее собственного адреса получаем классический сумматор по модулю 2^n, а значит, R-блок может быть назван *стохастическим сумматором*, т.е. сумматором с непредсказуемым результатом работы, зависящим от заполнения ключевой таблицы H.

Достоинством R-блока является простая программная реализация (4 инструкции Ассемблера, по две инструкции XLAT и ADD).

R-блок может использоваться для реализации шифрования методом гаммирования. В этом случае на вход A подается исходная последовательность, на вход B — гамма шифра (Keystream), а с выхода $R_H(A,B)$ шифртекст. Единственное, что надо учитывать — это необходимость использования преобразования R^{-1} (обратного R) на принимающей стороне. Вторая возможная область

применения R-блоков — замена сумматоров по модулю 2^n при модификации известных алгоритмов стохастического преобразования данных, например, поточных алгоритмов PIKE, RC4 и ряда других. Кстати, R-блоком можно двумя способами заменить сумматор по модулю 2^n на рис. 3, получив при этом две разновидности R^2-блоков.

Варианты модификации алгоритма RC4

Главная цель предлагаемых модификаций — сохранить первоначальный замысел автора RC4, усилив шифр в части конкретной реализации счетчиков и сумматоров по модулю 256.

Предлагаются два направления модификации шифра RC4. Первое, легковесное (Light-Weight) направление связано с реализацией счетчиков i и/или j на параллельных LFSR, формирующих n-разрядные псевдослучайные последовательности. Пример такого решения для $n = 8$ был рассмотрен ранее на рис. 2. При этом следует отметить, что наиболее простая программная реализация имеет место при выборе в качестве образующего многочлена $\Phi(x)$ LFSR трехчлена вида $xN + x^k + 1$, где $k \in \{n, 2n, 3n. \ldots\}$. Для $n = 8$ такими многочленами помимо

$x^{17} + x^{16} + 1$ являются $x^{15} + x^8 + 1$, $x^{39} + x^8 + 1$, $x^{63} + x^{32} + 1$, $x^{65} + x^{32} + 1$, ….

Второе направление связано с заменой сумматора (сумматоров) по модулю 256 в составе RC4 на восьмиразрядные стохастические сумматоры в схеме инициализации таблицы замен и/или в схеме генерации выходной последовательности. В первом случае очевидно усложняется связь между ключом и исходным состоянием алгоритма. При этом исходный ключ имеет вид $K = K_1 \parallel K_2$, где K_1 обеспечивает формирование таблицы стохастического преобразования H, а K_2 обеспечивает формирование таблицы замен S. Во втором случае помимо устранения известных уязвимостей PRNG RC4 мы получаем существенное увеличение (по сути удвоение) разрядности внутреннего состояния шифра. На рис. 4 показан один из вариантов ядра модифицированного PRNG.

Заключение

Рассмотрены достоинства поточного алгоритма RC4. Дано объяснения причинам появления многочисленных модификаций алгоритма. Приведено описание функция ядра алгоритма Spritz, самой существенной модификация RC4. Перечислены основные строительные блоки современных поточных шифров: LFSR, S- и R-блоки. В RC4 и всех его многочисленных модификациях используются только блоки замен, поэтому логичным является попытка встраивания LFSR и R-блоков в структуру алгоритма. Приводится Light-Weight версия RC4 на основе использования параллельных LFSR. Описываются возможные модификации RC4 на основе использования R-блоков. Большое количество вариантов такой модификации позволит в будущем синтезировать полиморфную конструкцию шифра.

Тестирование PRNG модифицированного алгоритма, построенных с использованием R-блоков по методике НИСТ [15], показало статистическую безопасность этих модификаций, так как число пройденных оценочных тестов оказалось сопоставимо с результатами тестирования блочных PRNG на основе ГОСТ 28147–89 и AES-128.

ЛИТЕРАТУРА

1. Ronald L. Rivest. RSA security response to weaknesses in key scheduling algorithm of RC4. Technical note, RSA Data Security, Inc., 2001. [The structure of RC4 was never published officially, it was leaked in 1994 to the Internet. This note confirms that the leaked code is indeed RC4.].
2. Поточные шифры. А. А. Асосков, М. А. Иванов, А. А. Мирский и др.— М.: Кудиц-образ, 2003.
3. Sourav Sen Gupta. Analysis and Implementation of RC4 Stream Cipher. Indian Statistical Institute. Kolkata, West Bengal, India, 2013.
4. Rivest R., Schuldt J.: Spritz — a spongy RC4-like stream cipher and hash function, 2014. [https://people.csail.mit.edu/rivest/pubs/RS14.pdf]
5. G. Bertoni, J. Daemen, M. Peeters, G. Van Assche. Sponge functions. Ecrypt Hash Workshop, Barcelona, Spain, May 2007. http://sponge.noekeon.org/.
6. G. Bertoni, J. Daemen, M. Peeters, G. Van Assche. Permutation-based encryption, authentication and authenticated encryption. http://keccak.noekeon.org/KeccakDIAC2012.pdf
7. Иванов М.А., Чугунков И. В. Криптографические методы защиты информации в компьютерных системах и сетях.— М.: НИЯУ МИФИ, 2012.
8. Carlisle Adams, Stafford E. Tavares. Good S-boxes are easy to find. Advances in cryptology — CRYPTO '89. Proceedings of a conference held at the University of California, Santa Barbara, CA (USA), August 20–24, 1989, pp.612–615.
9. Serge Mister, Carlisle Adams. Practical S-Box Design. Workshop on Selected Areas in Cryptography (SAC'96). Queen's University, 1996, pp. 61–76.
10. El-Sheikh, H.M.; El-Mohsen, O.A.; Elgarf, S.T.; Zekry, A. A new approach for designing key-dependent S-box defined over GF (24) in AES. Int. J. Comput. Theory Eng. 2012, 4, 158.
11. Leander, G.; Poschmann, A. On the classification of 4 bit S-boxes. In Proceedings of the 1st international Workshop on Arithmetic of Finite Fields, Madrid, Spain, 21–22 June 2007; pp. 159–176.

12. Zhang, W.; Bao, Z.; Rijmen, V.; Liu, M. A New Classification of 4-bit Optimal S-boxes and its Application to PRESENT, RECTANGLE and SPONGENT. In International Workshop on Fast Software Encryption; Springer: Berlin/Heidelberg, Germany, 2015; pp. 494–515.
13. Ahmad Albatsha, Michael A. Ivanov. Stochastic data transformation boxes for information security applications. Biologically Inspired Cognitive Architectures (BICA) for Young Scientists. Editors Alexei V. Samsonovich, Valentin V. Klimov. Proceedings of the First International Early Research Career Enhancement School on BICA and Cybersecurity (FIERCES2017) pp. 221–227.
14. Осмоловский С. А. Стохастические методы передачи данных.— М.: Радио и связь, 1991.
15. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. NIST Special Publication 800–22. http://csrc.nist.gov/groups/ST/toolkit/rng/documents/SP800–22b.pdf.