Ежегодная международная научно-практическая конференция
«РусКрипто'2022»

Высокопроизводительная псевдослучайная функция pCollapserARX256-32x2

Поликарпов Сергей Витальевич
к.т.н., доцент каф. ИБТКС, ИКТИБ ЮФУ

Румянцев Константин Евгеньевич
д.т.н., заведующий каф. ИБТКС, ИКТИБ ЮФУ

Прудников Вадим Александрович
аспирант, ассистент каф. ИБТКС, ИКТИБ ЮФУ

Институт компьютерных технологий и информационной безопасности,
Южный федеральный университет

Цель работы:

создание высокопроизводительной псевдо-случайной функции на основе псевдо-динамических подстановок с использованием ARX-операций.

Задачи исследования:

1. Определить возможность применения ARX-функций в качестве основного элемента псевдо-динамической подстановки. Для этого требуется подобрать ARX-функцию, оценить её криптографические свойства и свойства получаемых псевдо-динамических подстановок.

2. Интегрировать псевдо-динамические подстановки на основе ARX-функций в структуру псевдо-случайной функции pCollapser (pCollapserARX).

3. Создать последовательную и параллельную программные реализации pCollapserARX и оценить их производительность.

Актуальность:

Существующий набор отечественных симметричных криптоалгоритмов (Кузнечик, Магма, Стрибог) позволяет обеспечить качественную защиту информации при межсетевом взаимодействии в рамках протоколов EPS IPSec и TLS. Однако существенной проблемой при массовом внедрении отечественных криптоалгоритмов может стать недостаточная производительность их программных реализаций [1,2].

Одним из направлений современной криптографии является создание и исследование криптоалгоритмов на основе ARX-операций (Addition, Rotate, Xor). ARX-конструкции наиболее полно используют возможности современных процессоров и обеспечивают высокую скорость преобразования информации (как, например, алгоритм ChaCha20 [3]).

Основным недостатком таких алгоритмов является относительно слабые нелинейные свойства операции арифметического сложения, что потенциально может привести к нахождению эффективных методов криптоанализа таких алгоритмов.

Предпосылки:

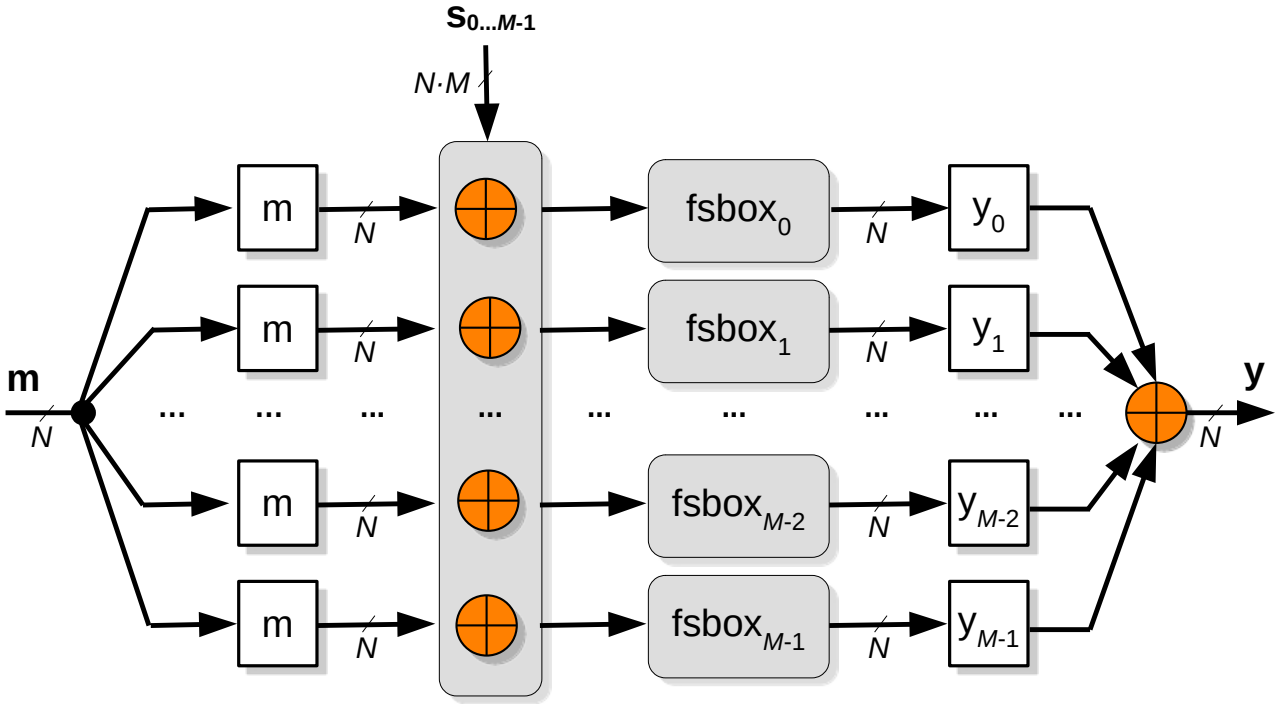
псевдо-динамические подстановки — позволяют эффективно разрушать статистические зависимости между входными и выходными значениями за счёт динамической трансформации её криптографических свойств, а также позволяют обеспечить параллельную работу входящих в её состав фиксированных подстановок.

псевдо-случайная функция pCollapser — позволяет эффективно реализовать возможности псевдо-динамических подстановок *PDSbox*, в том числе, обеспечить динамическую работу *PDSbox* путём формирования и распределения управляющих значений, а также обеспечить экстремальный параллелизм обработки информации за счёт независимой работы группы *PDSbox* в рамках одного раунда.

операции ARX (Addition, Rotate, Xor) — позволяют наиболее полно использовать возможности современных процессоров и обеспечивают высокую скорость преобразования информации.

Псевдо-динамические подстановки *PDSbox*

Структура псевдо-динамической таблицы подстановки основана на базе обычных фиксированных подстановок (*fixed sbox* или *fsbox*). Входное значение каждой фиксированной подстановки параметризуется своим индивидуальным значением состояния s_i (*control state*), где *i* – номер фиксированной подстановки (от 0 до *M*-1).



Параметры:

N – размерность входа/выхода;

M – количество подстановок;

m – входное значение;

y – выходное значение;

$s_{0...M-1}$ – значения состояния;

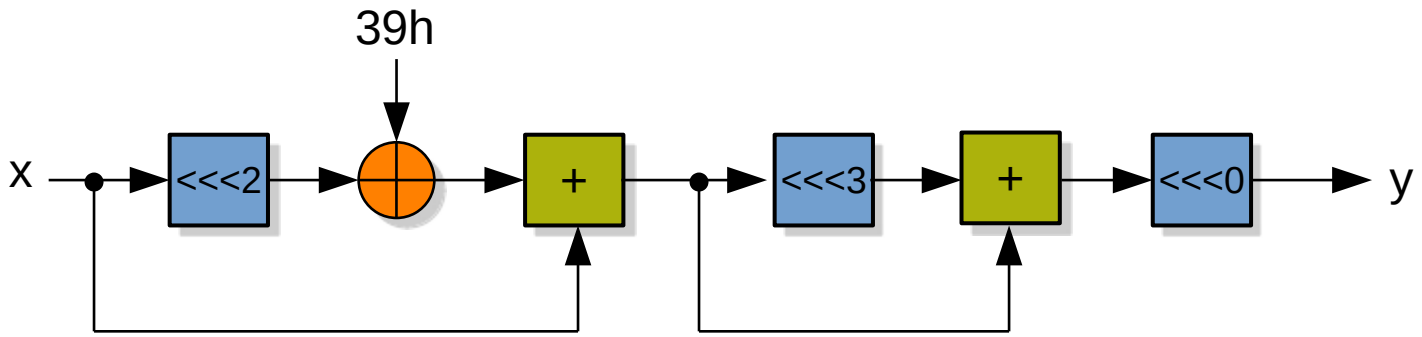
$fsbox_{0...M-1}$ – подстановки.

$$y = \bigoplus_{i=0}^{M-1} fsbox_i(m \oplus s_i) \qquad (1)$$

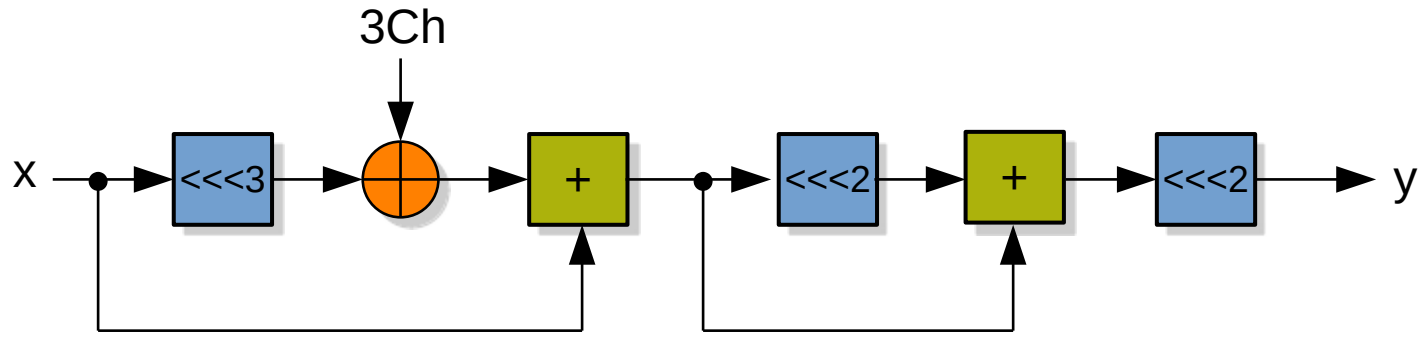
Возможности псевдо-динамических подстановок на примере PDSbox_2x6x6

Для примера выберем в качестве двух фиксированных 6-битовых подстановок *fsbox1* и *fsbox2* две ARX-функции, обладающих заведомо плохими линейными и дифференциальными свойствами:

fsbox1:



fsbox2:



6-битовые подстановки fsbox1 и fsbox2:

Полученные небиективные подстановки:

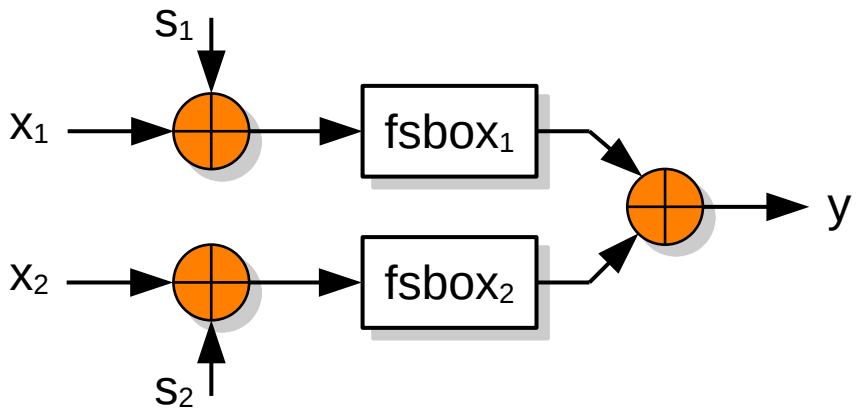
fsbox1: [8, 53, 17, 63, 26, 8, 35, 17, 45, 26, 54, 36, 63, 45, 8, 54, 9, 54, 18, 63, 35, 9, 44, 26, 54, 35, 63, 44, 8, 54, 17, 63, 54, 36, 63, 45, 8, 54, 18, 63, 27, 9, 44, 18, 53, 35, 63, 44, 63, 44, 8, 54, 17, 63, 27, 8, 36, 17, 45, 27, 54, 36, 0, 45]

fsbox2: [62, 48, 34, 20, 10, 59, 45, 31, 37, 39, 25, 15, 60, 50, 32, 22, 24, 14, 0, 2, 55, 37, 27, 9, 19, 5, 54, 40, 42, 28, 14, 0, 39, 25, 11, 60, 62, 48, 34, 20, 30, 12, 2, 51, 37, 39, 25, 15, 17, 7, 52, 42, 24, 14, 0, 2, 8, 57, 47, 29, 19, 5, 54, 40]

Криптографические свойства подстановок:

Параметр	fsbox1	fsbox2
Максимальное значение преобладания	ls = **0.437500**	ls = **-0.328125**
Максимальное значение в таблице распределения дифференциалов (диапазон: 0...64)	Δs = **28** (Δx = 5, Δy = 0)	Δs = **32** (Δx = 36, Δy = 0)
Алгебраическая степень	λs = **4** [4, 5, 6, 5, 6, 6]	λs = **4** [5, 6, 4, 4, 5, 6]
Алгебраический иммунитет	**2**	**2**

Пример типового применения фиксированных подстановок:



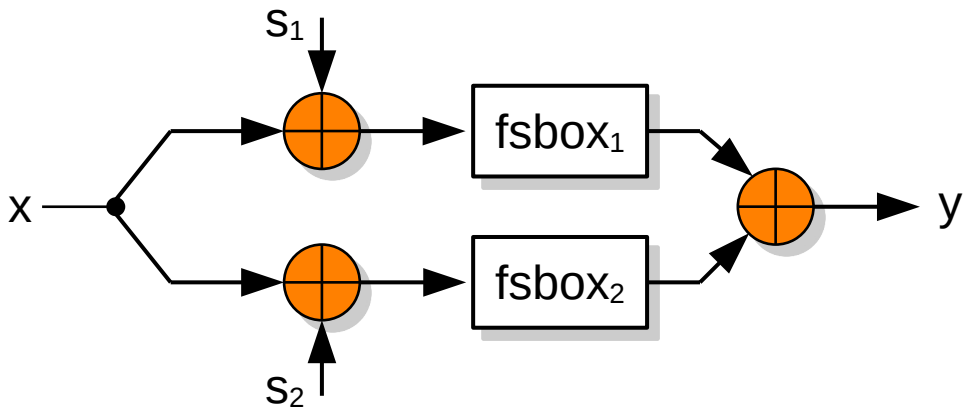
где:

x_1, x_2 – 6-битовые входы;

y – 6-битовый выход;

s_1, s_2 – 6-битовые значения состояний.

Пример псевдо-динамической подстановки PDSbox_2x6x6:



где:

x – 6-битовый вход;

y – 6-битовый выход;

s_1, s_2 – 6-битовые значения состояний.

Дифференциальные свойства представленных конструкций:

Типовое включение подстановок:

Для входной разности $\Delta x = (\Delta x_1 | \Delta x_2)$, $\Delta x_1 = 7$, $\Delta x_2 = 8$:

1) значения состояний $s_1 = 1, s_2 = 1$:

$\Delta x/\Delta y$: [228, 88, 156, 12, 76, 16, 72, 40, 32, 52, 24, 24, 88, 60, 92, 48, 128 ...]

2) значения состояний $s_1 = 1, s_2 = 2$:

$\Delta x/\Delta y$: [228, 88, 156, 12, 76, 16, 72, 40, 32, 52, 24, 24, 88, 60, 92, 48, 128 ...]

PDSbox_2x6x6:

Для входной разности $\Delta x = 7$

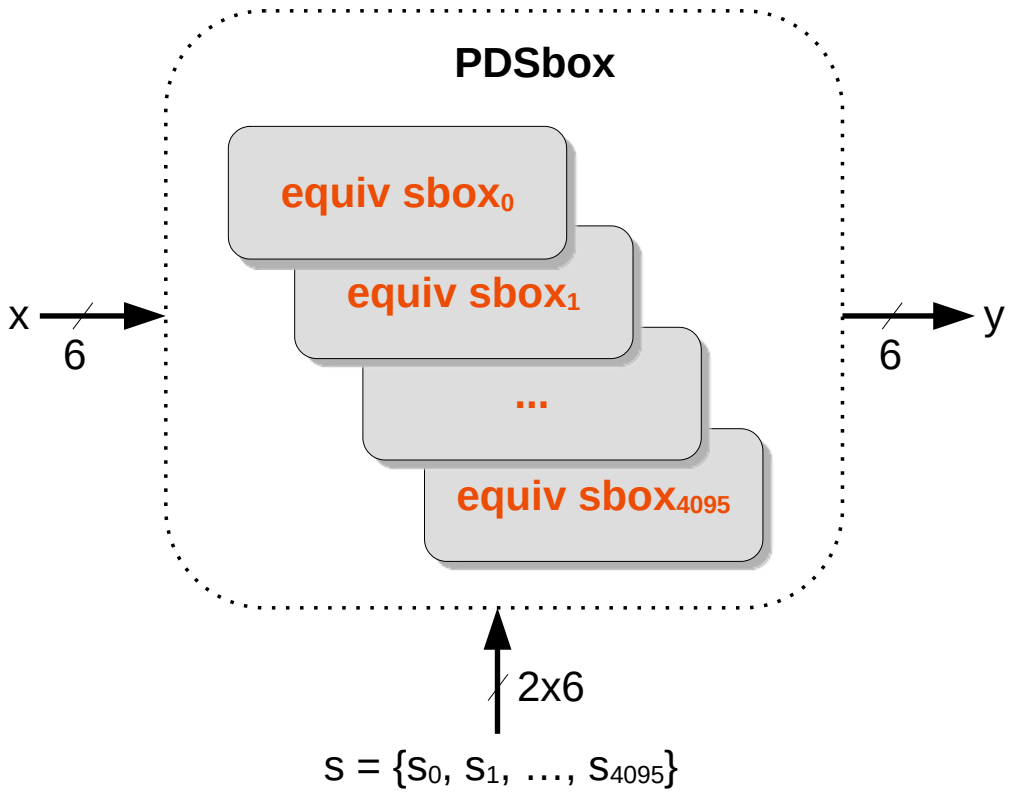
1) значения состояний $s_1 = 1, s_2 = 1$:

$\Delta x/\Delta y$: [2, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 4, 0, 2, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 2, 0 ...]

2) значения состояний $s_1 = 1, s_2 = 2$:

$\Delta x/\Delta y$: [0, 0, 2, 0, 4, 2, 2, 4, 0, 0, 0, 4, 0, 8, 2, 2, 0, 0, 0, 2, 0, 0, 2, 0, 2, 0, 2, 0, ...]

Представление псевдо-динамических подстановок в виде набора эквивалентных подстановок:



Усреднение дифференциальных свойств по множеству эквивалентных подстановок:

1. Разности Δx проходят через одну эквивалентную подстановку (случай, когда значения *s* — фиксированы):

$$\overline{N_D}(\Delta X, \Delta Y) = \frac{\sum_{i=0}^{4095} N_{Di}(\Delta X, \Delta Y)}{4096} \qquad (2)$$

$$\Delta Y = equivsbox_i(x') \oplus equivsbox_i(x'')$$

2. Разности Δx проходят через разные эквивалентные подстановки (случай, когда значения *s* — динамически изменяются):

$$\overline{N_D}(\Delta X, \Delta Y) = \frac{\sum_{j=0}^{4095} \sum_{i=0}^{4095} N_{Dji}(\Delta X, \Delta Y_{ji})}{4096^2} \qquad (3)$$

$$\Delta Y_{ji} = equivsbox_j(x') \oplus equivsbox_i(x'')$$

Усреднённые дифференциальные свойства представленных конструкций:

Типовое включение подстановок:

Для входной разности $\Delta x = (\Delta x_1 | \Delta x_2)$, $\Delta x_1 = 7$, $\Delta x_2 = 8$:

значения состояний $s_1 = 1$, $s_2 = 0...63$:

$\Delta x/\Delta y$: [228, 88, 156, 12, 76, 16, 72, 40, 32, 52, 24, 24, 88, 60, 92, 48, 128 ...]

PDSbox_2x6x6:

Для входной разности $\Delta x = 7$

значения состояний $s_1 = 1$, $s_2 = 0...63$:

$\Delta x/\Delta y$: [0.8125, 0.5625, 1.4375, 0.4375, 1.5, 1.3125, 0.9375, 1.1875, 1.1875, 0.375, 0.875, 1, 1.25, 1.375, 1.5625, 0.875, 0.875, 0.4375, 0.9375, 0.5, 1.3125 ...]

Выводы по псевдо-динамическим подстановкам:

Если обратиться к дифференциальному криптоанализу, то он основан на определении таблицы переходов разностей (DDT) $\Delta x \rightarrow \Delta y$ и дальнейшего поиска в ней и использования таких разностей $\Delta x \rightarrow \Delta y$, для которых вероятности появления наиболее отличаются от идеального (равновероятного) значения $P_{\Delta x \rightarrow \Delta y} = 1$. При этом, для использования найденной неидеальности обычно требуются значительный объём статистических данных.

Существенный момент — таблица DDT обычно определяется для фиксированной функции (фиксированной подстановки).

В случае динамически изменяемой подстановки такой подход в общем случае не применим, так как для каждого нового входного значения будет псевдослучайно изменяться значения на управляющем входе *pdsbox*, т.е. каждый раз будет равновероятно использована одна из эквивалентных подстановок (из всего множества эквивалентных подстановок 2^M).

Выводы по псевдо-динамическим подстановкам:

Каждая отдельная эквивалентная подстановка не обладает идеальными дифференциальными/линейными свойствами. Но при попытке набора статистических данных (в условиях равновероятности эквивалентных подстановок) вероятности появления разностей $P_{\Delta x \to \Delta y}$ будут усредняться по всему множеству эквивалентных подстановок.

Как показали экспериментальные исследования, если входящие в состав *pdsbox* фиксированные подстановки обладают определённой минимальной нелинейностью/сложностью, то при усреднении значения вероятностей в DDT или LAT для *pdsbox* стремятся к идеальным.

Аналогичная ситуация возникает и для других статистических методов криптоанализа (линейный, на основе усечённых и невозможных дифференциалов, дифференциально-линейный, циклический (rotational), циклически-дифференциальный).

Свойства PRF pCollapser

Рассмотрим свойства PRF pCollapser на примере его миниверсии. Для этого, выберем 4 6-битовые небиективные подстановки *fsbox1...4*, полученные из приведённой выше ARX-функции и обладающие заведомо плохими линейными и дифференциальными свойствами:

fsbox1: [8, 53, 17, 63, 26, 8, 35, 17, 45, 26, 54, 36, 63, 45, 8, 54, 9, 54, 18, 63, 35, 9, 44, 26, 54, 35, 63, 44, 8, 54, 17, 63, 54, 36, 63, 45, 8, 54, 18, 63, 27, 9, 44, 18, 53, 35, 63, 44, 63, 44, 8, 54, 17, 63, 27, 8, 36, 17, 45, 27, 54, 36, 0, 45]

fsbox2: [62, 48, 34, 20, 10, 59, 45, 31, 37, 39, 25, 15, 60, 50, 32, 22, 24, 14, 0, 2, 55, 37, 27, 9, 19, 5, 54, 40, 42, 28, 14, 0, 39, 25, 11, 60, 62, 48, 34, 20, 30, 12, 2, 51, 37, 39, 25, 15, 17, 7, 52, 42, 24, 14, 0, 2, 8, 57, 47, 29, 19, 5, 54, 40]

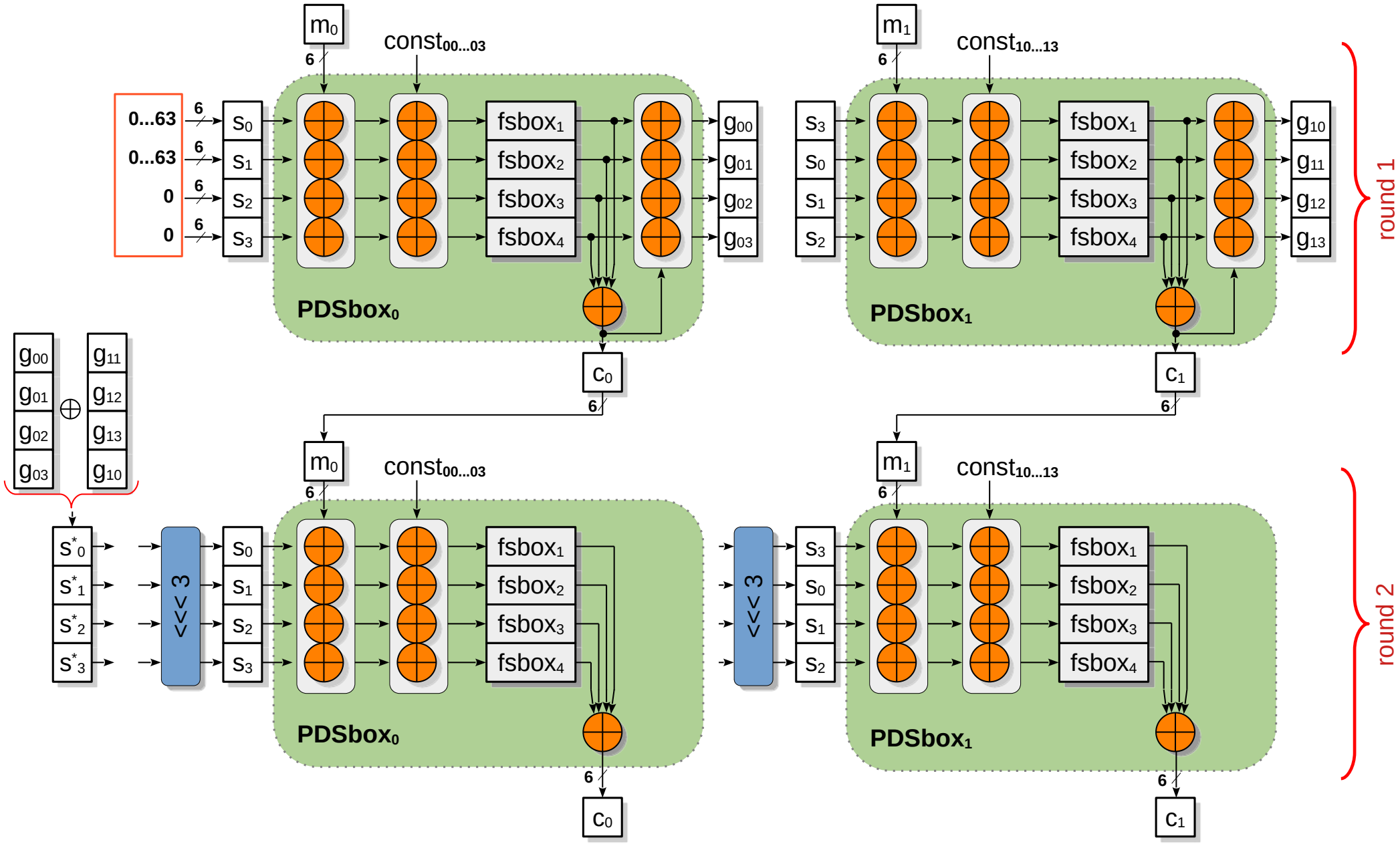
fsbox3: [28, 19, 10, 1, 10, 1, 63, 54, 55, 46, 37, 28, 36, 27, 18, 9, 10, 1, 63, 54, 63, 54, 45, 36, 37, 28, 19, 10, 19, 10, 1, 63, 54, 45, 36, 27, 36, 27, 18, 1, 10, 1, 63, 54, 63, 54, 45, 28, 37, 28, 19, 10, 18, 9, 0, 55, 63, 54, 45, 36, 45, 36, 27, 10]

fsbox4: [30, 5, 51, 42, 41, 0, 15, 53, 35, 42, 9, 63, 62, 21, 20, 11, 9, 63, 30, 5, 51, 42, 41, 0, 30, 5, 35, 42, 9, 63, 62, 21, 41, 0, 15, 53, 20, 27, 57, 32, 62, 21, 20, 11, 57, 32, 15, 53, 51, 42, 41, 0, 15, 53, 20, 27, 9, 63, 62, 21, 20, 11, 57, 32]

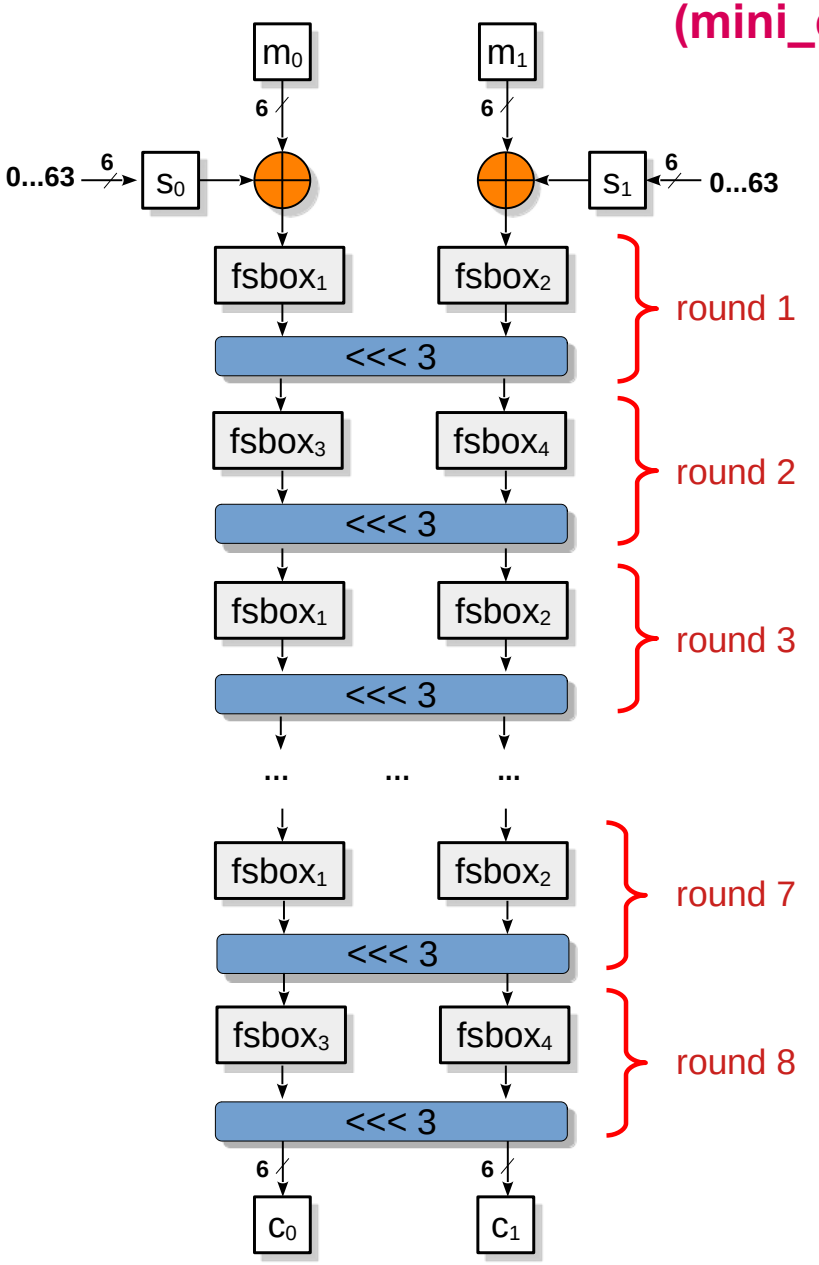
Криптографические свойства подстановок fsbox1...4:

Параметр	fsbox1	fsbox2	fsbox3	fsbox4
Максимальное значение преобладания	ls = **0.437500**	ls = **-0.328125**	ls = **-0.437500**	ls = **-0.359375**
Максимальное значение в таблице распределения дифференциалов (диапазон: 0...64)	Δs = **28** (Δx = 5, Δy = 0)	Δs = **32** (Δx = 36, Δy = 0)	Δs = **24** (Δx = 6, Δy = 0)	Δs = **32** (Δx = 18, Δy = 0)
Алгебраическая степень	λs = **4** [4, 5, 6, 5, 6, 6]	λs = **4** [5, 6, 4, 4, 5, 6]	λs = **4** [6, 5, 5, 4, 5, 6]	λs = **3** [3, 4, 6, 5, 4, 3]
Алгебраический иммунитет	**2**	**2**	**2**	**2**

Миниверсия PRF pCollapser (mini_pCollapser_12x12)



Миниверсия типовой функции на основе SP-сети (mini_conventional_SPN_12x12)



Параметры mini_conventional_SPN_12x12:

1. Размерность входа/выхода – **12** бит.
2. Количество раундов – **8**.
3. Критический путь – **8** фикс. подстановок.
4. Кол-во подстановок – **16**.

Параметры mini_pCollapser_12x12:

1. Размерность входа/выхода – **12** бит.
2. Количество раундов – **2**.
3. Критический путь – **2** фикс. подстановок.
4. Кол-во подстановок – **16**.
5. Используемые константы:

   $const_{00...03} = \{13, 23, 37, 43\}$

   $const_{10...13} = \{7, 17, 31, 47\}$

Криптографические свойства PDSbox_4x6x6

Параметр	PDSbox_4x6x6 (анализ по 100 экв. подстановкам)
Максимальное значение преобладания	ls = **0.265625**
Максимальное значение в таблице распределения дифференциалов (диапазон: 0...64)	Δs = **12**
Алгебраическая степень	λs = **5**
Алгебраический иммунитет	**2**

Криптографические свойства предложенных миниверсий

Параметр	mini_conventional_SPN_12x12	mini_pCollapser_12x12
Максимальное значение преобладания	ls = **0.42627**	ls = **0.04834**
Максимальное значение в таблице распределения дифференциалов (диапазон: 0...4096)	Δs = **956**	Δs = **16**
Алгебраическая степень	λs = **11**	λs = **11**
Алгебраический иммунитет	**2**	**4**

Криптографические свойства миниверсий

Гистограмма распределения усреднённых значений в таблице DDT, s1 = 0...63, s2 = 0...63

mini_conventional_SPN_12x12 наихудший случай: Δx1 = 5, Δx2 = 36			**mini_pCollapser_12x12** наихудший случай: Δx1 = 50, Δx2 = 18		
[minval = 0, maxval = 44, step = 2.2, counted values = 4095]			[minval = 0.932129, maxval = 1.12451, step = 0.00961914, counted values = 4095]		
num bin	hist_x	hist_y	num bin	hist_x	hist_y
0	0.000000	3652	0	0.932129	5
1	2.200000	123	1	0.941748	25
2	4.400000	116	2	0.951367	65
3	6.600000	44	3	0.960986	168
4	8.800000	47	4	0.970605	293
5	11.000000	46	5	0.980225	473
6	13.200000	12	6	0.989844	628
7	15.400000	20	7	0.999463	678
8	17.600000	14	8	1.009082	662
9	19.800000	6	9	1.018701	523
10	22.000000	4	10	1.028320	297
11	24.200000	1	11	1.037939	164
12	26.400000	3	12	1.047559	81
13	28.600000	1	13	1.057178	22
14	30.800000	4	14	1.066797	8
15	33.000000	1	15	1.076416	2
16	35.200000	0	16	1.086035	0
17	37.400000	0	17	1.095654	0
18	39.600000	0	18	1.105273	0
19	41.800000	0	19	1.114893	0
20	44.000000	1	20	1.124512	1

Выводы по миниверсиям рассмотренных функций:

mini_conventional_SPN_12x12:

Несмотря на существенное количество раундов (Nr = 8) и активного межраундового перемешивания бит за счёт циклического сдвига, результирующая функция обладает крайне плохими дифференциальными свойствами и малым значением алгебраического иммунитета. Вывод – при заданных подстановках данная функция не смогла эффективно разрушить статистические зависимости между входами/выходами.

mini_pCollapser_12x12:

Несмотря на малое количество раундов (Nr = 2) результирующая функция обладает свойствами, аналогичными свойствам 12-битовых случайных подстановок.

Усреднённые дифференциальные свойства приближаются к идеальным, что наглядно показывает динамический характер изменения криптографических свойств PDSbox при изменении управляющих значений на входе (s_1 и s_2) и за счёт формирования управляющих значений для второго раунда.

Вывод – при заданных подстановках данная функция обладает высокой нелинейностью/сложностью преобразования и может эффективно разрушать статистические зависимости между входами/выходами.

Псевдо-случайная функция pCollapserARX256-32x2

Для раскрытия возможностей псевдо-динамических подстановок на основе ARX-функций и обеспечения динамического режима их работы предложена структура псевдо-случайной функции pCollapserARX256-32x2, использующая в каждом из 4 раундов по 16 параллельно работающих 64 битных ARX-функций. Каждая из 16 ARX-функций параметризована значениями сдвига $t_0...t_7$ и констант.

Структура pCollapserARX256-32x2 является упрощённой версией PRF pCollapser, предложенной в [4] и имеет:

- размер обрабатываемого блока – 256 бит;
- размер раундового ключа – 256 бит;
- размер управляющего состояния для переключения подстановок – 256 бит;
- суммарное значение внутреннего состояния – 512 бит.

Выражение, описывающее структуру ARX-функции:

(для 64-битной ARX-функции применяются 32-битные значения переменных *a* и *b*):

$$a_2 = ((a_1 + (a_1 <<< t_0) \wedge b_1) <<< t_2) \wedge const_0;$$
$$b_2 = ((b_1 + (b_1 <<< t_1) \wedge a_1) <<< t_3) \wedge const_1;$$
$$a_3 = (a_2 + ((a_2 <<< t_4) \wedge b_2) <<< t_6);$$
$$b_3 = (b_2 + ((b_2 <<< t_5) \wedge a_2) <<< t_7);$$

где « ^ » – сложение по модулю 2 (операция XOR);

« $a <<< t$ » – циклический сдвиг на t бит влево в двоичном слове a;

a_1 – младшие 32 бит входного 64-битного значения;

b_1 – старшие 32 бит входного 64-битного значения;

a_3 и b_3 – соответственно младшие и старшие 32 бит выходного 64-битного значения;

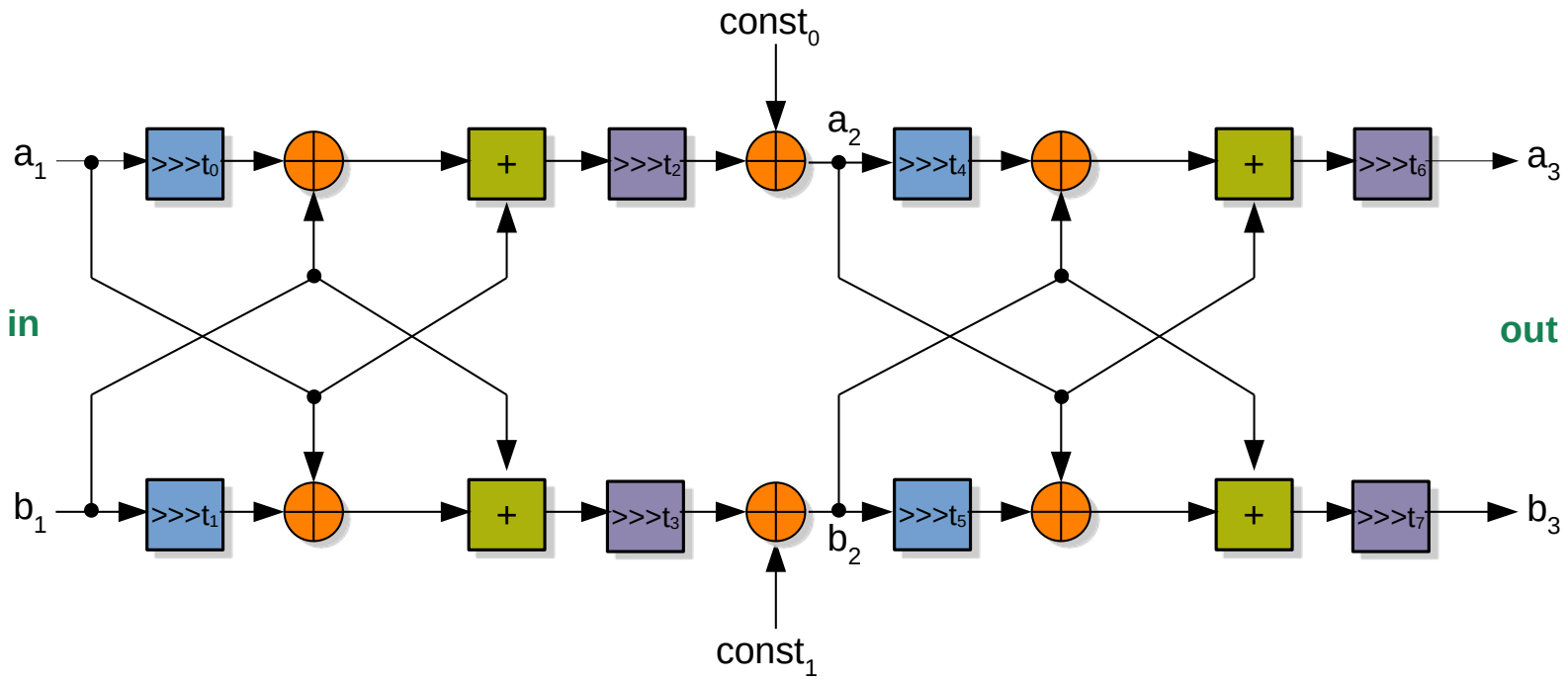
$t_0 \ldots t_7$ – значения циклических сдвигов, задающие конкретную ARX-функцию;

$const$ – 32-битные значения констант, задающие конкретную ARX-функцию.

Структура ARX-функций

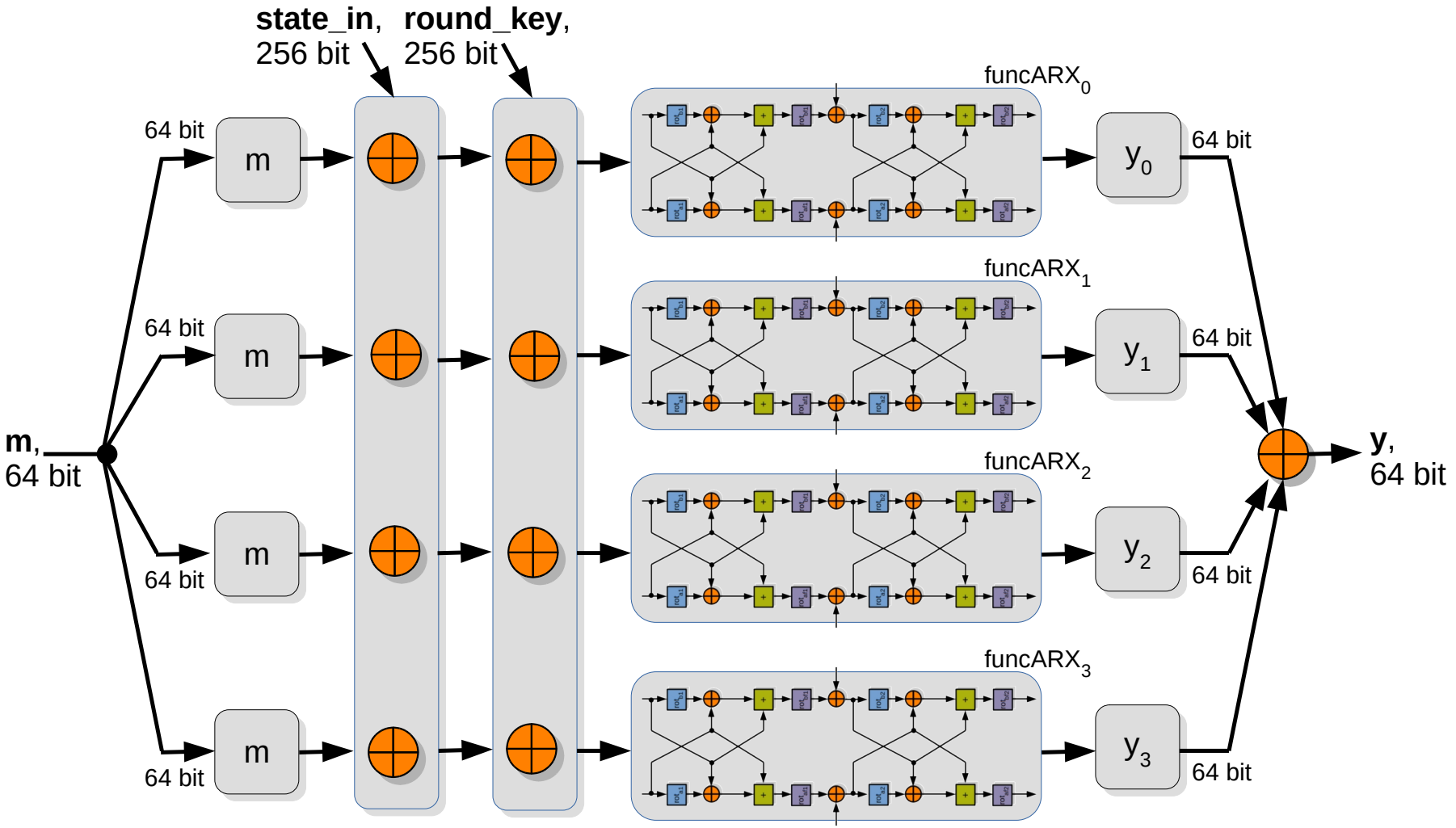
Значения параметров ARX-функций:

	t0	t1	t2	t3	t4	t5	t6	t7
funcARX0:	8	16	16	8	8	16	0	0
funcARX1:	8	16	8	16	16	8	8	8
funcARX2:	16	8	8	16	8	16	16	16
funcARX3:	16	8	16	8	16	8	24	24

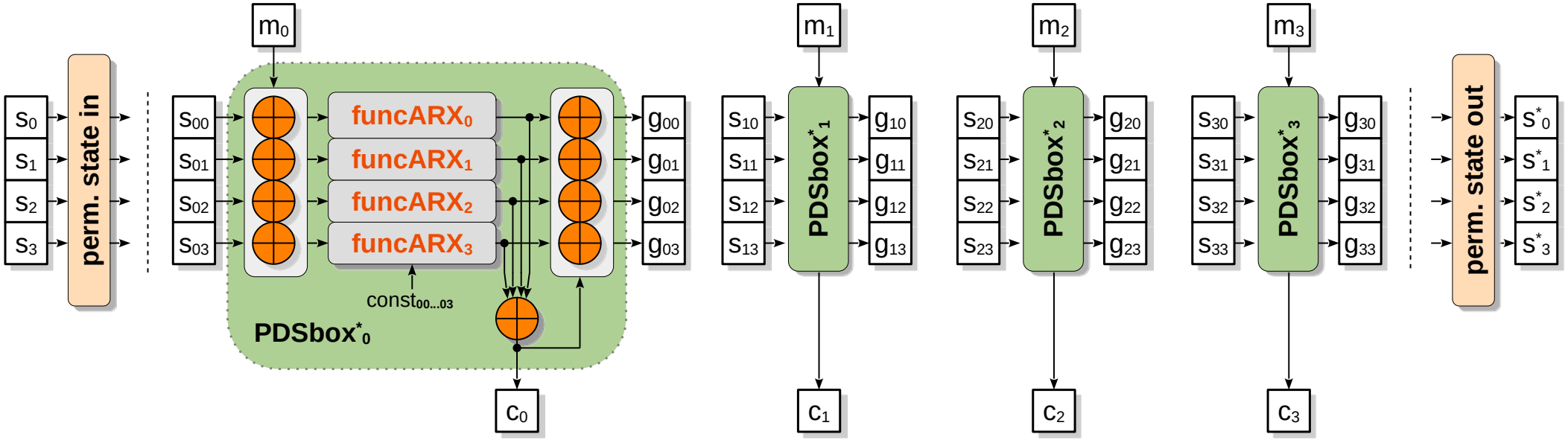


Структура ARX-функций подобрана исходя из обеспечения криптографических свойств (в составе PDSbox) и обеспечения оптимального использования возможностей процессоров и аппаратных платформ.

Псевдо-динамическая подстановка PDSbox_4x64x64



Раунд pCollapserARX256-32x2



где:

m_0 … m_3 – входные значения (размерность всех слов – 64 бит);

c_0 … c_3 – выходные значения;

s_0 … s_3 – входное управляющее состояние;

s^*_0 … s^*_3 – выходное управляющее состояние;

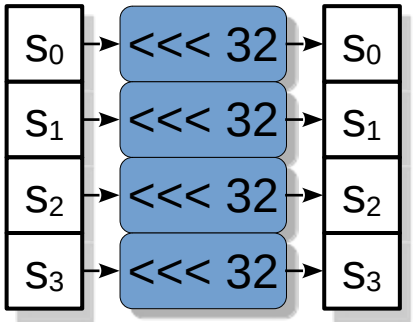
$funcARX_0$ … $funcARX_3$ – ARX-функции (размерностью 64 бит);

perm. state in – операции формирования входных состояний для псевдо-динамических подстановок $PDSbox_{0...3}$.

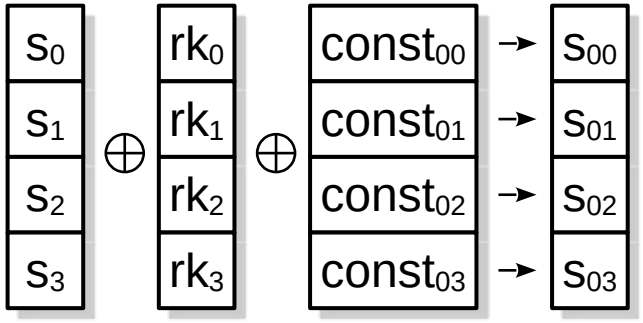
Формирование входных состояний для псевдо-динамических подстановок $PDSbox_{0...3}$

(perm. state in)

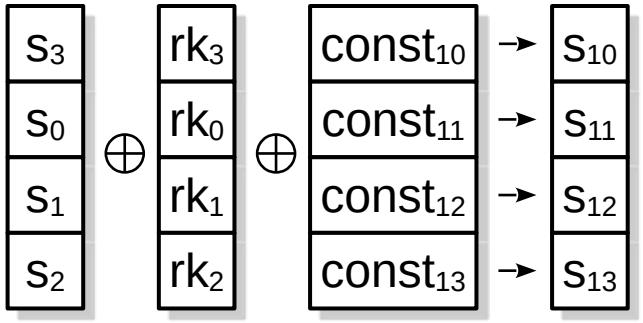
in shuffle:



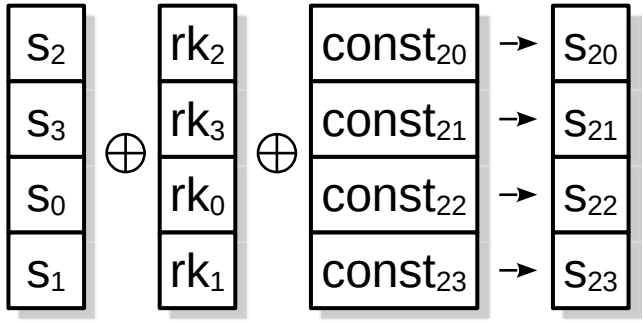
for $PDSbox_0$:



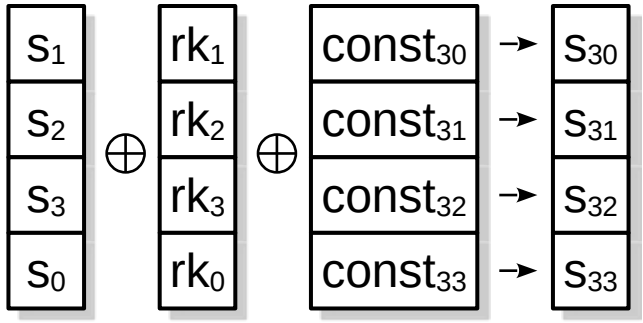
for $PDSbox_1$:



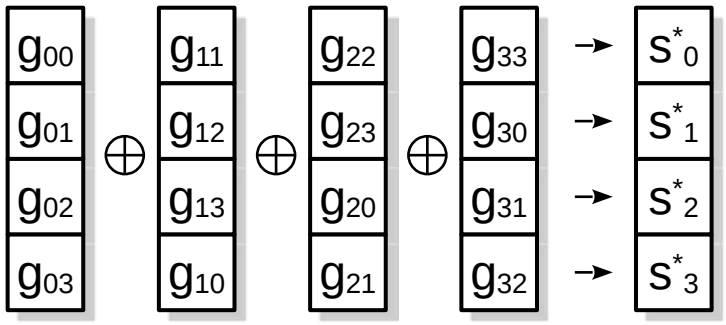
for $PDSbox_2$:



for $PDSbox_3$:



Операции формирования выходных состояний
(perm. state out)



где:

rk_0 … rk_3 – слова раундового ключа;

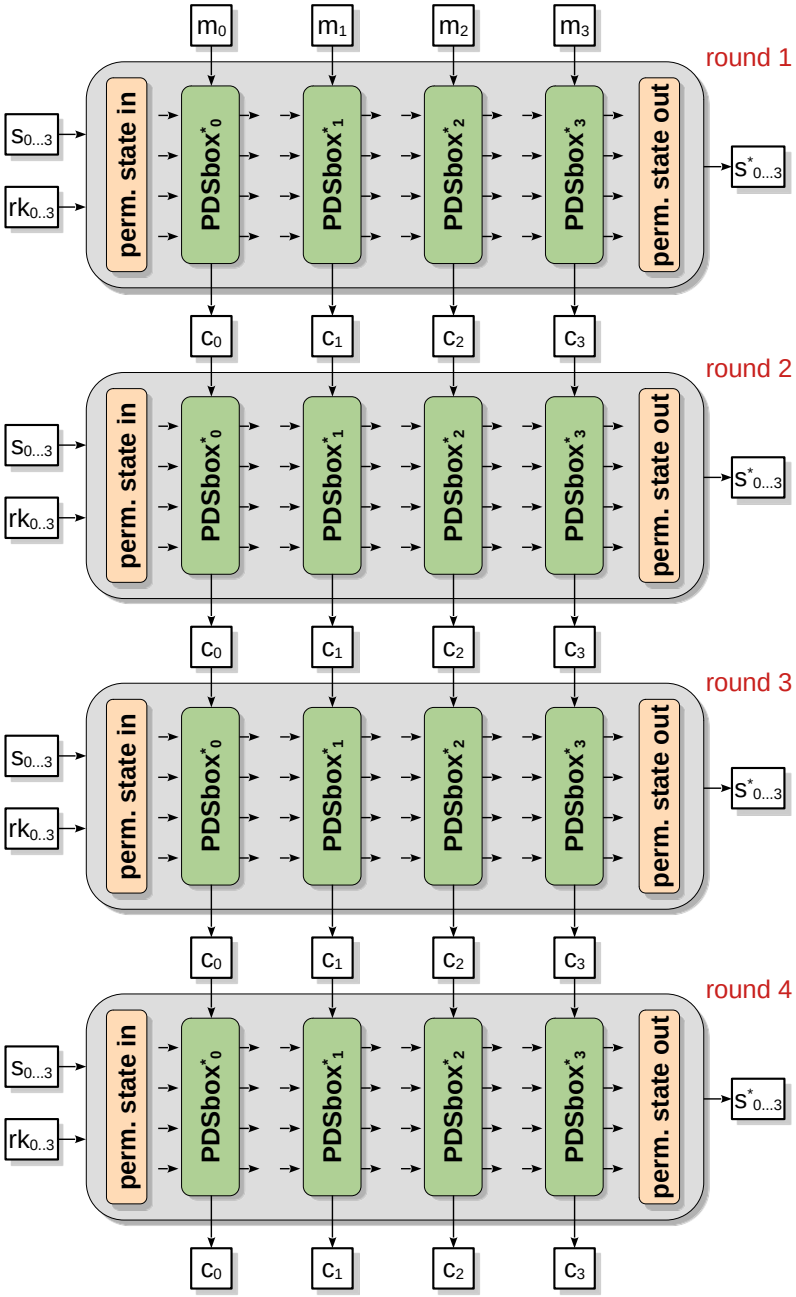
s_{n0} … s_{n3} – входное управляющее состояние для *n*-ой PDSbox;

$const_{n0}$ … $const_{n3}$ – значения констант для *n*-ой PDSbox;

g_{n0} … g_{n3} – выходное значение с *n*-ой PDSbox для формирования нового управляющего состояния;

s^*_0 … s^*_3 – управляющее состояние для следующего раунда.

PRF pCollapserARX256-32x2



где:

m_0 … m_3 – входные значения (4x64 бит);

c_0 … c_3 – выходные значения (4x64 бит);

s_0 … s_3 – входное управляющее состояние (4x64);

s^*_0 … s^*_3 – выходное управляющее состояние (4x64);

rk_0 … rk_3 – раундовый ключ (4x64 бит);

perm. state in – операции формирования входных состояний для псевдо-динамических подстановок;

perm. state out – операции формирования выходных состояний для псевдо-динамических подстановок следующего раунда;

$PDSbox_0$ … $PDSbox_3$ – псевдо-динамические подстановки (4x64x64).

Исследование производительности pCollapserARX256-32x2:

В приложении Б приведён исходный код последовательной реализации pCollapserARX256-32x2 на языке C, а также результаты моделирования и тестирования производительности.

Для тестирования производительности использовался компьютер на базе процессора Intel Core i5-7500@3.40GHz (Max 3.80 GHz), компиляторы gcc 9.3.0, usuba (https://github.com/usubalang/usuba) и операционная система GNU/Linux Ubuntu 20.04.

Таблица 1. Результаты тестирования производительности pCollapserARX256-32x2

Алгоритм	Тип реализации	Скорость на одном ядре, Мбайт/с	Производи-тельность, такт/байт
Кузнечик	последовательная, SSE, язык Си [1]	170	22,4
	параллельная, SSE (4 экземпляра) [1]	360	10,6
Магма	параллельная, AVX2 (16 экземпляров)[5]	1030	3,6
ChaCha20	параллельная, usuba (32 экземпляра)[3]	3300	**1,02**
pCollapserARX, 256 бит	тривиальная, последовательная, язык Си	134	28,4
	последовательная, usuba std, язык Си	182	20,9
	последовательная, usuba std, clang, язык Си	590	6,4
	последовательная, AVX2, язык Си	**798**	**4,8**
	параллельная (8 экземпляров), компилятор usuba, AVX2, язык Си	**1400**	**2,8**

Исследования свойств псевдо-динамических подстановок ARX:

Учитывая доступные вычислительные возможности, были проведены экспериментальные исследования линейных и дифференциальных свойств для ARX-функций размерностью 12 и 24 бит. В данном случае, таблицы линейных аппроксимаций (LAT) и таблицы распределения дифференциалов (DDT) рассчитывались частично, для небольшого количества элементов этих таблиц.

Это вызвано тем, что целью был не поиск максимальных значений в таблицах LAT и DDT, а демонстрация возможностей получаемых псевдо-динамических подстановок – когда объединение ARX-функций, имеющих откровенно слабые криптографические свойства, в структуру псевдо-динамической подстановки позволяет получать свойства эквивалентных подстановок (Equivalent sbox), близкие к свойствам случайно сформированных подстановок аналогичной размерности.

Кроме этого, показано наличие эффекта усреднения значений LAT и DDT при динамическом изменении состояния псевдо-динамических подстановок (Dynamic sbox) [6]. Предполагается, что свойства ARX-функций и получаемых псевдо-динамических подстановок будут сохраняться при масштабировании до 64 бит. Результаты вычислений показаны в таблице 2.

Таблица 2. Результаты исследования свойств псевдо-динамических подстановок ARX

Тип подстановки	#tested sbox	max bias (LAT)	max N_D (DDT)	min degree	algebraic immunity
n = 12 bit, *ARX-function with: 4 add, 8 rol*					
Random sbox 12x12 (non-bijective)	100	0.0432129	20	11	4
ARX_2x6_4a8r	**4**	**0.127**	**268**	**5**	**2**
Equivalent sbox 12x12	100	0.0507812	20	11	4
Dynamic sbox4x12x12	**100**	**0.0035986**	**1.84**	**>11**	-
– «» –	**1000**	**0.0001137**	**1.24**	**>11**	-
n = 24 bit, *ARX-function with: 4 add, 8 rol* (partial computations)					
Random sbox 24x24 (non-bijective)	100	0.000479341	20	23	-
ARX_2x12_4a8r	**4**	**0.0012017**	**74918**	**9**	-
Equivalent sbox 24x24	100	0.0006589	20	23	-
Dynamic sbox4x24x24	**100**	**0.0000498**	**2.56**	**>23**	-
– «» –	**1000**	**0.0000098**	**2.174**	**>23**	-
n = 64 bit (estimated values)					
Equivalent sbox 64x64	-	~ random_sbox64	~ random_sbox64	63	-
Dynamic sbox4x64x64	**-**	**~ ideal bias**	**~ ideal N_D**	**>63**	**-**

Выводы:

Показано, что объединение ARX-функций, имеющих откровенно слабые криптографические свойства, в структуру псевдо-динамической подстановки позволяет получать свойства эквивалентных подстановок, близкие к свойствам случайно сформированных подстановок аналогичной размерности.

Исходя из перемешивающих свойств и обеспечения динамического режима работы псевдо-динамических подстановок, а также свойств ARX-функций и 2-кратный запас по раундам, стойкость pCollapserARX256-32x2 оценивается не ниже 256 бит.

Содержит простые операции и имеет заложенные возможности экстремальной параллелизации обработки данных, что позволяет делать эффективные программные и аппаратные реализации для различных процессоров и аппаратных платформ.

Основное назначение – применение в качестве высокопроизводительной PRF в режимах, где не требуется обратимость преобразования: AEAD, CTR, Sponge-конструкции и др.

Для задач легковесной криптографии pCollapserARX легко масштабируется путём изменения размерности слов (например, ARX64-8x2, ARX128-16x2).

Контактные данные

Поликарпов Сергей Витальевич

polikarpovsv@sfedu.ru

Румянцев Константин Евгеньевич

rumyancev@sfedu.ru

Прудников Вадим Александрович

prudnikov@sfedu.ru, +7 919 896 14 27

???

Приложение А.
Список источников

1. A.S. Rybkin, On software implementation of Kuznyechik on Intel CPUs, Mat. Vopr. Kriptogr., 2018, Volume 9, Issue 2, 117–127. DOI: https://doi.org/10.4213/mvk255

2. Нестеренко А.Ю. О программной реализации алгоритмов шифрования с аутентификацией. https://www.ruscrypto.ru/resource/archive/rc2021/files/02_nesterenko.pdf

3. Darius Mercadier and Pierre-Évariste Dagand. 2019. Usuba: high-throughput and constant-time ciphers, by construction. In Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2019). Association for Computing Machinery, New York, NY, USA, 157–173. DOI:https://doi.org/10.1145/3314221.3314636

4. Поликарпов С.В., Кожевников А.А., Румянцев К.Е., Прудников В.А. Псевдослучайная функция PCOLLAPSER, обеспечивающая экстремальный параллелизм обработки информации // Известия ЮФУ. Технические науки. – 2019. – N5 (207). – С. 88-100.

5. Русев А.А., Сонина Л.А., Щербаков Д.А. Высокопроизводительные программные реализации алгоритмов шифрования Кузнечик и Магма на российских процессорах Эльбрус с учётом архитектурных особенностей. https://www.ruscrypto.ru/resource/archive/rc2021/files/12_rusev_sonina_sherbakov.pdf

6. Поликарпов С.В., Румянцев К.Е., Кожевников А.А. Исследование линейных характеристик псевдо-динамических подстановок // Известия ЮФУ. Технические науки. 2015. Май. Т. 166, N5. С. 111-123. URL: http://old.izv-tn.tti.sfedu.ru/wp-content/uploads/2015/5/11.pdf

Приложение Б.
Исходный код последовательной реализации (на языке C)

```
void pCollapserARX256_32x2_encrypt_block_std(uint64_t *m_block, uint64_t *roundkey, uint64_t *c_block)
{
    const uint64_t const [16] = {
        0x072286acdd632df6, 0x039143566eb196fb, 0x81c8a1ab3758cb7d, 0xc0e450d59bac65be,
        0x6072286acdd632df, 0xb039143566eb196f, 0xd81c8a1ab3758cb7, 0xec0e450d59bac65b,
        0xf6072286acdd632d, 0xfb039143566eb196, 0x7d81c8a1ab3758cb, 0xbec0e450d59bac65,
        0xdf6072286acdd632, 0x6fb039143566eb19, 0xb7d81c8a1ab3758c, 0x5bec0e450d59bac6
    };

    uint64_t in_state[4], out_state[4], m[4], s[4], y[4], g[4], c[4], w;

    in_state[0] = 0;    // init state values
    in_state[1] = 0;
    in_state[2] = 0;
    in_state[3] = 0;

    m[0] = m_block[0];  // load input words
    m[1] = m_block[1];
    m[2] = m_block[2];
    m[3] = m_block[3];

    for(int round=0; round<4; round++)    // do rounds
    {
          ...
    }

    c_block[0] = c[0];  // output words
    c_block[1] = c[1];
    c_block[2] = c[2];
    c_block[3] = c[3];
}
```

Приложение Б. (продолжение)

```
for(int round=0; round<4; round++)
{ // words
    in_state[0] = rol64( in_state[0], 32);  // shuffle (rotate) in state
    in_state[1] = rol64( in_state[1], 32);
    in_state[2] = rol64( in_state[2], 32);
    in_state[3] = rol64( in_state[3], 32);

    w = 0;
    { // rows
        s[0] = m[w] ^ in_state[0] ^ const[0] ^ roundkey[0];
        s[1] = m[w] ^ in_state[1] ^ const[1] ^ roundkey[1];
        s[2] = m[w] ^ in_state[2] ^ const[2] ^ roundkey[2];
        s[3] = m[w] ^ in_state[3] ^ const[3] ^ roundkey[3];

        funcARX0 (&s[0], &y[0], &const[0]);
        funcARX1 (&s[1], &y[1], &const[1]);
        funcARX2 (&s[2], &y[2], &const[2]);
        funcARX3 (&s[3], &y[3], &const[3]);

        c[w] = y[0]^y[1]^y[2]^y[3];

        g[0] = c[w] ^ y[0];
        g[1] = c[w] ^ y[1];
        g[2] = c[w] ^ y[2];
        g[3] = c[w] ^ y[3];

        out_state[0] = g[0];
        out_state[1] = g[1];
        out_state[2] = g[2];
        out_state[3] = g[3];
    }
```

```
    w = 1;
    { // rows
        s[0] = m[w] ^ in_state[0] ^ const[4] ^ roundkey[0];
        s[1] = m[w] ^ in_state[1] ^ const[5] ^ roundkey[1];
        s[2] = m[w] ^ in_state[2] ^ const[6] ^ roundkey[2];
        s[3] = m[w] ^ in_state[3] ^ const[7] ^ roundkey[3];

        funcARX0 (&s[3], &y[0], &const[4]);
        funcARX1 (&s[0], &y[1], &const[5]);
        funcARX2 (&s[1], &y[2], &const[6]);
        funcARX3 (&s[2], &y[3], &const[7]);

        c[w] = y[0]^y[1]^y[2]^y[3];

        g[0] = c[w] ^ y[0];
        g[1] = c[w] ^ y[1];
        g[2] = c[w] ^ y[2];
        g[3] = c[w] ^ y[3];

        out_state[3] ^= g[0];
        out_state[0] ^= g[1];
        out_state[1] ^= g[2];
        out_state[2] ^= g[3];
    }
```

Приложение Б. (продолжение)

```
    w = 2;
  { // rows
    s[0] = m[w] ^ in_state[0] ^ const[8 ] ^ roundkey[0];
    s[1] = m[w] ^ in_state[1] ^ const[9 ] ^ roundkey[1];
    s[2] = m[w] ^ in_state[2] ^ const[10] ^ roundkey[2];
    s[3] = m[w] ^ in_state[3] ^ const[11] ^ roundkey[3];

    funcARX0 (&s[2], &y[0], &const[8 ]);
    funcARX1 (&s[3], &y[1], &const[9 ]);
    funcARX2 (&s[0], &y[2], &const[10]);
    funcARX3 (&s[1], &y[3], &const[11]);

    c[w] = y[0]^y[1]^y[2]^y[3];

    g[0] = c[w] ^ y[0];
    g[1] = c[w] ^ y[1];
    g[2] = c[w] ^ y[2];
    g[3] = c[w] ^ y[3];

    out_state[2] ^= g[0];
    out_state[3] ^= g[1];
    out_state[0] ^= g[2];
    out_state[1] ^= g[3];
  }
    w = 3;
  { // rows
    s[0] = m[w] ^ in_state[0] ^ const[12] ^ roundkey[0];
    s[1] = m[w] ^ in_state[1] ^ const[13] ^ roundkey[1];
    s[2] = m[w] ^ in_state[2] ^ const[14] ^ roundkey[2];
    s[3] = m[w] ^ in_state[3] ^ const[15] ^ roundkey[3];

    funcARX0 (&s[1], &y[0], &const[12]);
    funcARX1 (&s[2], &y[1], &const[13]);
    funcARX2 (&s[3], &y[2], &const[14]);
    funcARX3 (&s[0], &y[3], &const[15]);
```

```
    c[w] = y[0]^y[1]^y[2]^y[3];

    g[0] = c[w] ^ y[0];
    g[1] = c[w] ^ y[1];
    g[2] = c[w] ^ y[2];
    g[3] = c[w] ^ y[3];

    out_state[1] ^= g[0];
    out_state[2] ^= g[1];
    out_state[3] ^= g[2];
    out_state[0] ^= g[3];
  }

  in_state[0] = out_state[0];
  in_state[1] = out_state[1];
  in_state[2] = out_state[2];
  in_state[3] = out_state[3];

  m[0] = c[0];  // load input words to next round
  m[1] = c[1];
  m[2] = c[2];
  m[3] = c[3];

 } // next round

 c_block[0] = c[0];  // output words
 c_block[1] = c[1];
 c_block[2] = c[2];
 c_block[3] = c[3];
}
```

Приложение Б. (продолжение)

```
uint32_t rol32(uint32_t word, uint32_t shift)
{
    return (word << shift) | (word >> ((-shift) & 31));
}

uint64_t rol64(uint64_t word, uint64_t shift)
{
    return (word << shift) | (word >> ((-shift) & 63));
}

void funcARX0 (uint32_t x[2], uint32_t y[2], uint32_t constants[2]) {
    uint32_t a, b, at, bt;

    a = x[0];
    b = x[1];

    at = rol32(a,  8)  ^ b ;
    bt = rol32(b, 16) ^ a ;

    a = (a + at) ;
    b = (b + bt) ;

    a =  rol32(a, 16) ^ constants[0];
    b =  rol32(b,  8)  ^ constants[1];

    at = rol32(a,  8)  ^ b ;
    bt = rol32(b, 16) ^ a ;

    a = (a + at) ;
    b = (b + bt) ;

    y[0] = a;
    y[1] = b;
}
```

```
void funcARX1 (uint32_t x[2], uint32_t y[2], uint32_t constants[2]) {

    uint32_t a, b, at, bt;

    a = x[0];
    b = x[1];

    at = rol32(a, 8)   ^ b ;
    bt = rol32(b, 16) ^ a ;

    a = (a + at) ;
    b = (b + bt) ;

    a =  rol32(a,  8)  ^ constants[0];
    b =  rol32(b, 16) ^ constants[1];

    at = rol32(a, 16) ^ b ;
    bt = rol32(b,  8)  ^ a ;

    a = (a + at) ;
    b = (b + bt) ;

    y[0] = rol32(a, 8);
    y[1] = rol32(b, 8);
}
```

Приложение В.
результаты экспериментальных исследований

Гистограмма распределения усреднённых значений в таблице DDT, 100 подстановок (был использован частичный расчёт таблицы)

pdsboxARX_24x24			Random sbox 24x24		
[minval = 0.16, maxval = 2.56, step = 0.12, counted values = 469762020]			[minval = 0.3, maxval = 1.94, step = 0.082, counted values = 469762020]		
num bin	hist_x	hist_y	num bin	hist_x	hist_y
0	0.160000	2	0	0.300000	34
1	0.280000	560	1	0.382000	1449
2	0.400000	36006	2	0.464000	32193
3	0.520000	1167654	3	0.546000	397440
4	0.640000	7027328	4	0.628000	2847586
5	0.760000	39649383	5	0.710000	12671131
6	0.880000	107018082	6	0.792000	36791751
7	1.000000	168913627	7	0.874000	72612464
8	1.120000	95997949	8	0.956000	100691648
9	1.240000	37146470	9	1.038000	100875924
10	1.360000	9727624	10	1.120000	88173452
11	1.480000	2250120	11	1.202000	34756780
12	1.600000	539387	12	1.284000	14055495
13	1.720000	210943	13	1.366000	4461131
14	1.840000	58472	14	1.448000	1123076
15	1.960000	14516	15	1.530000	227248
16	2.080000	3167	16	1.612000	37494
17	2.200000	599	17	1.694000	5070
18	2.320000	116	18	1.776000	603
19	2.440000	13	19	1.858000	45
20	2.560000	2	20	1.940000	6

Приложение В.
(продолжение)
результаты экспериментальных исследований

Гистограмма распределения усреднённых значений в таблице LAT, 100 подстановок (был использован частичный расчёт таблицы)

pdsboxARX_24x24			Random sbox 24x24		
[minval = -2.93255e-05, maxval = 4.98295e-05, step = 3.95775e-06, counted values = 198]			[minval = -3.01301e-05, maxval = 3.92795e-05, step = 3.47048e-06, counted values = 198]		
num bin	hist_x	hist_y	num bin	hist_x	hist_y
0	-0.000029	2	0	-0.000030	2
1	-0.000025	3	1	-0.000027	1
2	-0.000021	3	2	-0.000023	4
3	-0.000017	11	3	-0.000020	7
4	-0.000013	9	4	-0.000016	6
5	-0.000010	22	5	-0.000013	15
6	-0.000006	25	6	-0.000009	17
7	-0.000002	24	7	-0.000006	23
8	0.000002	29	8	-0.000002	24
9	0.000006	21	9	0.000001	21
10	0.000010	14	10	0.000005	23
11	0.000014	10	11	0.000008	11
12	0.000018	14	12	0.000012	13
13	0.000022	1	13	0.000015	14
14	0.000026	5	14	0.000018	6
15	0.000030	3	15	0.000022	2
16	0.000034	0	16	0.000025	6
17	0.000038	0	17	0.000029	1
18	0.000042	1	18	0.000032	0
19	0.000046	0	19	0.000036	1
20	0.000050	1	20	0.000039	1

Приложение В.
(продолжение)
результаты экспериментальных исследований

Значения алгебраической степени (algebraic degree) для ARX-функций 24x24:

funcARX0: [9, 16, 18, 20, 21, 23, 24, 23, 23, 24, 23, 23, 9, 16, 17, 19, 22, 23, 23, 23, 24, 24, 24, 23]
funcARX1: [23, 24, 24, 24, 9, 16, 18, 20, 22, 23, 23, 24, 23, 23, 24, 23, 9, 16, 18, 20, 23, 24, 24, 23]
funcARX2: [23, 23, 23, 24, 23, 23, 9, 16, 18, 20, 22, 23, 23, 24, 24, 23, 24, 24, 9, 16, 18, 20, 21, 23]
funcARX3: [22, 23, 24, 23, 24, 23, 23, 23, 9, 16, 18, 20, 23, 23, 23, 23, 23, 24, 24, 23, 9, 16, 18, 20]

минимальные значения: **λs = 9**

Значения алгебраической степени (algebraic degree) для эквивалентных подстановок PDSboxARX_24x24:

eq sbox: [23, 24, 23, 23, 24, 23, 24, 23, 23, 24, 23, 24, 23, 24, 23, 23, 24, 23, 24, 23, 24, 23, 23, 23]

минимальное значение: **λs = 23**

Приложение Г.

Результаты тестирования производительности pCollapserARX256-32x2 на одном ядре процессора Intel Core i7-11700K (Max 5.0 GHz**), использовались реализации алгоритмов программы OpenSSL 1.1.1f**

Алгоритм	**Тип реализации**	**Скорость на одном ядре, Мбайт/с (размер блока, байт)**						**Макс. производи-тельность, такт/байт**
		16	64	256	1024	8192	16384	
Кузнечик-ecb	OpenSSL 1.1.1f	161	182	198	205	206	206	24,3
Магма-cnt	OpenSSL 1.1.1f	90	95	96	97	97	97	51,5
sm4-ecb	OpenSSL 1.1.1f	177	187	189	189	189	189	26,5
sha3-256	OpenSSL 1.1.1f	42	169	411	498	563	572	8,7
sm3	OpenSSL 1.1.1f	52	127	238	312	362	365	13,7
ChaCha20	OpenSSL 1.1.1f	626	1057	3992	7522	8317	8411	**0,6**
pCollapserARX, 256 бит	тривиальная, последовательная, icx, язык Си	154*	**308**	–«»–	–«»–	–«»–	–«»–	16,2
	последовательная, usuba std, clang, язык Си	386*	**772**	–«»–	–«»–	–«»–	–«»–	6,5
	последовательная, AVX2, компилятор icx, язык Си	613*	**1226**	–«»–	–«»–	–«»–	–«»–	4,1
	параллельная (8 экземпляров), компилятор usuba, AVX512, язык Си	173*	693*	**2770**	–«»–	–«»–	–«»–	1,8

* – получено делением от основного значения, кратно размеру блока