РОССИЙСКАЯ ФЕДЕРАЦИЯ

(19) **RU** (11) **2 710 669**(13) **C1**

(51) МПК
H04L 9/06 (2006.01)
G06F 15/80 (2006.01)

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ

(52) СПК
H04L 9/0618 (2019.08); G06F 15/8007 (2019.08)

(21)(22) Заявка: **2018138851, 06.11.2018**

(24) Дата начала отсчета срока действия патента:
06.11.2018

Дата регистрации:
30.12.2019

Приоритет(ы):
(22) Дата подачи заявки: **06.11.2018**

(45) Опубликовано: **30.12.2019** Бюл. № 1

Адрес для переписки:
127287, Москва, Старый Петровско-Разумовский пр-д, 1/23, стр. 1, Открытое акционерное общество "Информационные технологии и коммуникационные системы"

(72) Автор(ы):
Рыбкин Андрей Сергеевич (RU)

(73) Патентообладатель(и):
Открытое акционерное общество "Информационные технологии и коммуникационные системы" (RU)

(56) Список документов, цитированных в отчете о поиске: **RU 2390949 C2, 27.05.2010. US 9614666 B2, 04.04.2017. US 7512779 B2, 31.03.2009. US 7599489 B1, 06.10.2009. US 2011/0138192 A1, 09.06.2011.**

(54) Способ шифрования данных

(57) Реферат:

Изобретение относится к вычислительной технике. Технический результат заключается в повышении производительности процесса шифрования. Способ шифрования s сообщений $m_1, m_2, \ldots, m_s$, представленных в двоичном виде и имеющих длину, равную 128 бит каждый, где $s=t \cdot n$, причем t, n - натуральные числа, реализуемый посредством вычислительной системы, имеющей процессор с SIMD-архитектурой, заключающийся в том, что вычисляют u=0; (A) вычисляют параллельно с использованием SIMD-инструкций процессора значения $c_{ut+1}, c_{ut+2}, c_{ut+3}, \ldots, c_{ut+t} \in V_{128}$; вычисляют преобразования $RSHIFT_4$ вида $V_8 \rightarrow V_8$, преобразования CMPR вида $V_8 \times V_8 \rightarrow V_8$ и преобразования BLEND вида $V_8 \times V_8 \times V_8 \rightarrow V_8$; преобразования T_2, T_3, T_4 вида $V_4 \rightarrow V_4$ и преобразования α_0, α_1, T_1, T_5, T_6 вида $V_4 \rightarrow V_8$ вычисляют путем загрузки данных из вспомогательных таблиц, содержащих векторы значений этих преобразований; преобразования $MULT_{148,0}$, $MULT_{148,1}$, $MULT_{195,0}$, $MULT_{195,1}$ вида $V_4 \rightarrow V_8$ вычисляют путем загрузки данных из вспомогательных таблиц, содержащих векторы значений этих преобразований; вычисляют u=u+1; если u<n, то переходят к этапу (A); получают зашифрованные сообщения c_i, i=1, 2, …, s. 1 табл.

RU 2710669 C1

RU 2710669 C1

RUSSIAN FEDERATION

FEDERAL SERVICE
FOR INTELLECTUAL PROPERTY

(19) **RU** (11) **2 710 669**(13) **C1**

(51) Int. Cl.
H04L 9/06 (2006.01)
G06F 15/80 (2006.01)

(12) ABSTRACT OF INVENTION

(52) CPC
H04L 9/0618 (2019.08); G06F 15/8007 (2019.08)

(21)(22) Application: **2018138851, 06.11.2018**

(24) Effective date for property rights:
06.11.2018

Registration date:
30.12.2019

Priority:
(22) Date of filing: **06.11.2018**

(45) Date of publication: **30.12.2019** Bull. № 1

Mail address:
127287, Moskva, Staryj Petrovsko-Razumovskij pr-d, 1/23, str. 1, Otkrytoe aktsionernoe obshchestvo "Informatsionnye tekhnologii i kommunikatsionnye sistemy"

(72) Inventor(s):
Rybkin Andrej Sergeevich (RU)

(73) Proprietor(s):
Otkrytoe aktsionernoe obshchestvo "Informatsionnye tekhnologii i kommunikatsionnye sistemy" (RU)

RU 2710669 C1

(54) DATA ENCRYPTION METHOD

(57) Abstract:

FIELD: computer equipment.

SUBSTANCE: invention relates to the computer equipment. Encryption method of s messages m_1, m_2, ..., m_s presented in binary form and having length equal to 128 bits each, where s=t·n, wherein t, n are natural numbers, realized by a computer system having a processor with a SIMD architecture, consisting in that u=0; (A) is calculated in parallel using the processor SIMD instructions values c_{ut+1}, c_{ut+2}, c_{ut+3}, ..., $c_{ut+t} \in V_{128}$; performing $RSHIFT_4$ conversions of $V_8 \rightarrow V_8$ type, CMPR conversion of $V_8 \times V_8 \rightarrow V_8$ type and conversion BLEND of $V_8 \times V_8 \times V_8 \rightarrow V_8$ type; transformations T_2, T_3, T_4 of $V_4 \rightarrow V_4$ type and conversion α_0, α_1, T_1, T_5, T_6 of $V_4 \rightarrow V_8$ type is calculated by loading data from auxiliary tables containing vectors of values of these transformations; conversions $MULT_{148.0}$, $MULT_{148.1}$, $MULT_{195.0}$, $MULT_{195.1}$ of $V_4 \rightarrow V_8$ type is calculated by loading data from auxiliary tables containing vectors of values of these transformations; calculating u=u+1; if u<n, then proceeding to step (A); obtaining encrypted messages c_i, i=1, 2, …, s.

EFFECT: high efficiency of the encryption process.

1 cl, 1 tbl

RU 2710669 C1

Область техники, к которой относится изобретение

Предполагаемое изобретение относится к способам шифрования данных, в частности к блочному шифрованию данных с применением вычислительных платформ с SIMD-архитектурой.

Уровень техники

Шифрование является традиционным способом обеспечения конфиденциальности данных при их передаче и хранении. К одним из наиболее распространенных методов шифрования относятся алгоритмы, основанные на применении блочных шифров. Такие шифры оперируют фрагментами данных фиксированной длины - блоками, и сочетают в себе стойкость и высокую скорость работы.

Тенденция на увеличение объемов хранимой информации и скорости передаваемых данных требует от используемых блочных шифров высокой производительности. Эффективным методом увеличения быстродействия алгоритма шифрования является использование параллельных вычислений. Одним из способов организации параллельных вычислений в случае программной реализации алгоритма является использование SIMD-технологий, в основе которых лежит применение одной инструкции процессора для одновременной обработки нескольких фрагментов данных, предварительно размещенных на одном регистре.

SIMD-технологии получили широкое распространение и поддерживаются на большинстве современных вычислительных платформ, в том числе на процессорах общего назначения Intel и AMD. В настоящее время существует несколько типовых наборов SIMD-инструкций, каждый из которых предназначен для работы с регистрами определенной длины.

Применительно к программной реализации алгоритмов блочного шифрования, SIMD-технологии, как правило, используются для эффективной обработки сразу нескольких входных блоков данных. С помощью SIMD-технологий алгоритм шифрования, предназначенный для обработки одного блока, выполняют одновременно для нескольких блоков. Эффективность такого подхода напрямую зависит от возможности параллельного выполнения использующихся в алгоритме шифрования преобразований и операций, которая, в свою очередь, определяется наличием в вычислительной платформе соответствующих им SIMD-инструкций. В случае возможности распараллеливания каждой из операций такой подход позволяет выполнять обработку нескольких блоков данных за время, необходимое для обработки одного блока данных, то есть производительность алгоритма растет пропорциональному количеству одновременно обрабатываемых блоков. Поскольку число одновременно обрабатываемых блоков определяется длиной используемых регистров, производительность в этом случае растет пропорционально увеличению длины используемых регистров.

Национальный стандарт Российской Федерации ГОСТ Р 34.12-2015 специфицирует два блочных шифра: «Кузнечик» и «Магма» [1]. Для алгоритма шифрования «Магма» известен способ выполнения, допускающий эффективное применение SIMD-технологий в случае его программной реализации [2]. Под эффективностью здесь и далее будем подразумевать достаточно высокую скорость выполнения относительно других способов и пропорциональный рост производительности программных реализаций на основе способа при увеличении длины используемых регистров.

Рассмотрим существующие способы выполнения алгоритма зашифрования блочного шифра «Кузнечик». При описании преобразований шифра «Кузнечик» будем придерживаться обозначений, принятых в [1], опуская при этом, для простоты,

вспомогательные преобразования, введенные для установления соответствия между двоичными строками, числами и элементами поля.

Известен способ выполнения алгоритма зашифрования блочного шифра «Кузнечик» в соответствии с его описанием в [1]. Недостатками способа являются низкая производительность алгоритма в случае его программного исполнения без использования SIMD-технологий, а также невозможность эффективного применения существующих SIMD-инструкций при реализации способа.

Известен способ выполнения алгоритма зашифрования блочного шифра «Кузнечик» на основе использования таблиц большого размера [3]. Все вычисления при использовании такого способа сводятся к операциям загрузки данных из больших таблиц и побитовому сложению двоичных строк. Способ обладает достаточно высокой скоростью работы в случае его программного исполнения даже без применения SIMD-технологий, а в случае их применения производительность способа может быть дополнительно увеличена.

Недостатком способа является неэффективность применения существующих SIMD-инструкций для параллельного выполнения операций загрузки данных из больших таблиц, что приводит к тому, что производительность реализаций на основе способа растет непропорционально увеличению длины используемых регистров.

Другим недостатком является потенциальная уязвимость программных реализаций на основе способа к атакам по времени обращения к кэш-памяти вычислительной платформы [4], что приводит к недопустимости применения этих реализаций в случае возможности осуществления такого рода атак.

Известен способ выполнения нелинейного преобразования алгоритма «Кузнечик», являющегося частью алгоритма зашифрования, основанный на декомпозиции этого преобразования [5]. Согласно данному способу, преобразование π: $V_8 \to V_8$, использующееся в блочном шифре «Кузнечик» для определения нелинейного преобразования [1], вычисляют следующим образом:

$$\pi(a) = \begin{cases} \omega(\nu_0(MSB_4(\alpha(a))) \| \sigma(0)), & \text{если } LSB_4(\alpha(a)) = 0, \\ \omega(a' \| \sigma(\phi(a') \circ LSB_4(\alpha(a)))), & \text{если } LSB_4(\alpha(a)) \neq 0, \end{cases} \quad (1)$$

$$a' = \nu_1(I(LSB_4(\alpha(a))) \circ MSB_4(\alpha(a))),$$

где

$$LSB_4 : V_8 \to V_4, \quad LSB_4(x_7 \| x_6 \| \ldots \| x_0) = x_3 \| x_2 \| x_1 \| x_0,$$

$$MSB_4 : V_8 \to V_4, \quad MSB_4(x_7 \| x_6 \| \ldots \| x_0) = x_7 \| x_6 \| x_5 \| x_4,$$

$$x_i \in V_1, \quad i = 0, 1, \ldots, 7,$$

«$\circ$» - операция умножения в поле $GF(2^4)[X]/(X^4 \oplus X^3 \oplus 1)$, а линейные преобразования α, ω вида $V_8 \to V_8$ и нелинейные преобразования ν_0, ν_1, σ, φ, I вида $V_4 \to V_4$ определены в [5].

Данный способ расширяет возможности применения SIMD-технологий в случае программной реализации преобразования π. В частности, он позволяет свести ряд вычислений к преобразованиям вида $V_4 \to V_4$, имеющим эффективную реализацию с помощью SIMD-технологий.

Недостатком способа является наличие преобразований вида $V_8 \to V_8$ и операций

умножения в поле $GF(2^4)[X]/(X^4 \oplus X^3 \oplus 1)$, имеющих относительно высокую сложность реализации в случае применения SIMD-технологий.

Наиболее близким к предлагаемому способу является способ выполнения алгоритма зашифрования блочного шифра «Кузнечик» с помощью слайсинг техники, представленный в [3]. Способ основан на применении преобразований, эквивалентных имеющимся в шифре, но обладающих при этом более эффективной реализацией с использованием SIMD-технологий.

Согласно данному способу, алгоритм зашифрования блочного шифра «Кузнечик» выполняют в соответствии с соотношениями, приведенными в [1]:

$$E_{K_1,\dots,K_{10}} : V_{128} \to V_{128},$$
$$E_{K_1,\dots,K_{10}}(a) = X[K_{10}]LSX[K_9]\dots LSX[K_2]LSX[K_1](a), \quad (2)$$

где

$$X[k]: V_{128} \to V_{128}, \quad X[k](a) = k \oplus a, \quad k, a \in V_{128},$$
$$S: V_{128} \to V_{128}, \quad S(a) = S(a_{15} \| \dots \| a_0) = \pi(a_{15}) \| \dots \| \pi(a_0),$$
$$a_i \in V_8, \quad i = 0,1,\dots,15,$$
$$L: V_{128} \to V_{128}, \quad L(a) = R^{16}(a), \quad (3)$$
$$R: V_{128} \to V_{128}, \quad R(a) = R(a_{15} \| \dots \| a_0) = l(a_{15},\dots,a_0) \| a_{15} \| \dots \| a_1,$$
$$a_i \in V_8, \quad i = 0,1,\dots,15,$$

$K_1, \dots, K_{10} \in V_{128}$ - итерационные ключи.

Вычисление преобразования $l:(V_8)^{16} \to V_8$ основывают на алгоритме, приведенном в [1]:

$$l(a_{15}, a_{14}, \dots, a_0) = 148 \cdot a_{15} \oplus 32 \cdot a_{14} \oplus 133 \cdot a_{13} \oplus 16 \cdot a_{12} \oplus$$
$$\oplus 194 \cdot a_{11} \oplus 192 \cdot a_{10} \oplus 1 \cdot a_9 \oplus 251 \cdot a_8 \oplus 1 \cdot a_7 \oplus 192 \cdot a_6 \oplus \quad (4)$$
$$\oplus 194 \cdot a_5 \oplus 16 \cdot a_4 \oplus 133 \cdot a_3 \oplus 32 \cdot a_2 \oplus 148 \cdot a_1 \oplus 1 \cdot a_0$$

где «·» - операция умножения в поле $GF(2^8)[X]/(X^8 \oplus X^7 \oplus X^6 \oplus X \oplus 1)$.

При этом для возможности эффективной реализации преобразования l с использованием SIMD-технологий в данный алгоритм вносят ряд изменений.

Для каждой константы $const \in \{1, 148, 32, 133, 16, 194, 192, 251\}$ вводят преобразования вида $V_4 \to V_8$:

$$MULT_{const,0}(x) = const \cdot x,$$
$$MULT_{const,1}(x) = const \cdot LSHIFT_4(x), \quad (5)$$

где

$$LSHIFT_4 : V_4 \to V_8,$$
$$LSHIFT_4(x_3 \| x_2 \| x_1 \| x_0) = x_3 \| x_2 \| x_1 \| x_0 \| 0 \| 0 \| 0 \| 0, \quad (6)$$
$$x_i \in V_1, \quad i = 0,1,2,3$$

Операции умножения в поле, имеющиеся в (4), выполняют на основе преобразований

из (5) и соотношений:

$$\forall const \in \{1,148,32,133,16,194,192,251\}:$$
$$const \cdot x = MULT_{const,0}(LSB_4(x)) \oplus MULT_{const,1}(MSB_4(x)) \quad (7)$$

Эффективность использования преобразований из (5) обуславливается тем, что любое преобразование вида $V_4 \to V_8$ может быть реализовано с помощью таблицы, состоящей из 2^4 элементов, длиной 8 бит каждый. Вычисление преобразования в этом случае сводится к загрузке выходного значения из таблицы по входному значению, используемому в качестве индекса для выбора соответствующего элемента таблицы. Преимуществом применения таблиц такого размера является возможность эффективного распараллеливания операции загрузки из этих таблиц при помощи существующих SIMD-инструкций. Данное замечание справедливо и для таблиц, состоящих из меньшего числа элементов и/или из элементов меньшей длины.

Вычисление преобразования π: $V_8 \to V_8$ основывают на алгоритме, предложенном в [5] и описанном в (1). Для повышения эффективности реализации преобразования π с использованием SIMD-технологий вводят следующие преобразования вида $V_4 \to V_8$:

$$\alpha_0(x_3 \| x_2 \| x_1 \| x_0) = \alpha(0\|0\|0\|0\|x_3\|x_2\|x_1\|x_0),$$
$$\alpha_1(x_3 \| x_2 \| x_1 \| x_0) = \alpha(x_3\|x_2\|x_1\|x_0\|0\|0\|0\|0),$$
$$\omega_0(x_3 \| x_2 \| x_1 \| x_0) = \omega(0\|0\|0\|0\|x_3\|x_2\|x_1\|x_0), \quad (8)$$
$$\omega_1(x_3 \| x_2 \| x_1 \| x_0) = \omega(x_3\|x_2\|x_1\|x_0\|0\|0\|0\|0),$$
$$x_i \in V_1, \quad i = 0,1,2,3$$

Преобразования α и ω выполняют на основе преобразований (8) и соотношений, справедливых в силу линейности α и ω:

$$\alpha(x) = \alpha_0(LSB_4(x)) \oplus \alpha_1(MSB_4(x)),$$
$$\omega(x) = \omega_0(LSB_4(x)) \oplus \omega_1(MSB_4(x)). \quad (9)$$

Поскольку преобразования α_0, α_1, ω_0, ω_1, ν_0, ν_1, σ, φ, I имеют вид $V_4 \to V_4$, $V_4 \to V_8$, их вычисление может быть эффективно распараллелено при помощи существующих SIMD-инструкций по аналогии с преобразованиями (5).

Описанный способ принимается за прототип. Преимуществом способа является возможность эффективного применения существующих SIMD-инструкций для используемых операций, в том числе для загрузки данных из используемых таблиц. Это приводит к пропорциональному росту скорости программных реализаций на основе способа при увеличении длины используемых регистров.

Еще одним преимуществом способа является возможность получения на его основе программных реализаций, стойких к атакам по времени обращения к кэш-памяти вычислительной платформы, в силу задействования только малых по размеру вспомогательных таблиц.

Недостатком способа является его относительно невысокая производительность, обусловленная, в том числе, выполнением преобразования l без учета взаимосвязи между константными элементами поля, содержащимися в (4), а также необходимостью выполнения трудоемкой операции умножения в поле $GF(2^4)[X]/(X^4 \oplus X^3 \oplus 1)$ при

вычислении преобразования π.

Раскрытие изобретения

Техническим результатом является повышение производительности процесса зашифрования.

При этом предлагаемый способ сохраняет возможность эффективного применения SIMD-технологий при его реализации, выражающуюся в пропорциональном росте производительности в случае увеличения длины используемых регистров, а также сохраняет возможность получения реализаций на основе способа, стойких к атакам по времени обращения к кэш-памяти вычислительной платформы.

Предлагаемый способ шифрования s сообщений $m_1, m_2, \ldots, m_s$, представленных в двоичном виде и имеющих длину равную 128 бит каждый, где $s=t\cdot n$, причем t, n - натуральные числа, реализуемый посредством вычислительной системы, имеющей процессор с SIMD-архитектурой, заключается в том, что

- вычисляют

u=0;

- (A) вычисляют параллельно с использованием SIMD-инструкций процессора значения $c_{ut+1}, c_{ut+2}, c_{ut+3}, \ldots, c_{ut+t} \in V_{128}$:

$$c_{ut+1} = X[K_{10}]LSX[K_9]\ldots LSX[K_2]LSX[K_1](m_{ut+1}),$$

$$c_{ut+2} = X[K_{10}]LSX[K_9]\ldots LSX[K_2]LSX[K_1](m_{ut+2}),$$

$$c_{ut+3} = X[K_{10}]LSX[K_9]\ldots LSX[K_2]LSX[K_1](m_{ut+3}),$$

$$\ldots$$

$$c_{ut+t} = X[K_{10}]LSX[K_9]\ldots LSX[K_2]LSX[K_1](m_{ut+t}),$$

где

$X[k]: V_{128} \to V_{128}$, $X[k](x)=k \oplus x$, причем $k, x \in V_{128}$;

$S: V_{128} \to V_{128}$, $S(x) = S(x_{15} \| \ldots \| x_0) = \pi(x_{15}) \| \ldots \| \pi(x_0)$, причем $x_i \in V_8$, i=0, 1,…, 15;

$L: V_{128} \to V_{128}$, $L(x)=R^{16}(x)$, причем $x \in V_{128}$;

$R: V_{128} \to V_{128}$, $R(x) = R(x_{15} \| \ldots \| x_0) = l(x_{15}, \ldots, x_0) \| x_{15} \| \ldots \| x_1$, причем $x_i \in V_8$, i=0, 1,…, 15;

$K_1, \ldots, K_{10} \in V_{128}$ - итерационные ключи;

V_p - множество всех двоичных строк длины p;

p - неотрицательное целое число;

причем для получения значения $\pi(\alpha) \in V_8$, для произвольного $a \in V_8$, вычисляют

$$r_1 = \alpha_0(mask \wedge a) \oplus \alpha_1(mask \wedge RSHIFT_4(a)),$$

$$r_2 = mask \wedge r_1,$$

$$r_3 = mask \wedge RSHIFT_4(r_1),$$

$r_4=T_2(r_2)+_{256}T_3(r_3)$,

$$r_5 = (mask \wedge (r_4 +_{256} (mask \wedge RSHIFT_4(r_4)))) \wedge (\neg CMPR(r_3, 0)),$$

$r_6=T_4(r_5)+_{256}T_3(r_2)$,

$$\pi(a) = BLEND(T_5(r_5) \oplus T_6(mask \wedge (r_6 +_{256} (mask \wedge \wedge RSHIFT_4(r_6)))), T_1(r_3) \oplus 220, CMPR(r_2, 0)),$$

где «$+_{256}$» - операция сложения в кольце вычетов по модулю 256;

«$\wedge$» - операция побитового "И";

«$x \wedge (\neg y)$» - операция побитового "И" с отрицанием одного из аргументов;

«$\oplus$» - операция побитового сложения;

$$mask = 0 \| 0 \| 0 \| 0 \| 1 \| 1 \| 1 \| 1 \in V_8;$$

преобразование $RSHIFT_4$ вида $V_8 \to V_8$ преобразование CMPR вида $V_8 \times V_8 \to V_8$ и преобразование BLEND вида $V_8 \times V_8 \times V_8 \to V_8$ вычисляют в соответствии с соотношениями

$$RSHIFT_4(x_7 \| x_6 \| x_5 \| x_4 \| x_3 \| x_2 \| x_1 \| x_0) = y_3 \| y_2 \| y_1 \| y_0 \| x_7 \| x_6 \| x_5 \| x_4,$$

где значения $y_i \in V_1$, i=0, 1, 2, 3,

значения $x_i \in V_1$, i=0, 1,…, 7,

$$CMPR(x, y) = \begin{cases} 255, & \text{если } x = y, \\ 0, & \text{если } x \neq y; \end{cases}$$

$$BLEND(x, y, z) = \begin{cases} x, & \text{если } MSB_1(z) = 0, \\ y, & \text{если } MSB_1(z) = 1, \end{cases}$$

где $MSB_1: V_8 \to V_1$, $MSB_1(x_7 \| x_6 \| \ldots \| x_0) = x_7$, $x_i \in V_1$, $i = 0, 1, \ldots, 7$;

преобразования T_2, T_3, T_4 вида $V_4 \to V_4$ и преобразования α_0, α_1, T_1, T_5, T_6 вида $V_4 \to V_8$ вычисляют путем загрузки данных из вспомогательных таблиц, содержащих векторы значений этих преобразований:

α_0=(0, 112, 58, 74, 20, 100, 46, 94, 154, 234, 160, 208, 142, 254, 180, 196),

α_1=(0, 2, 17, 19, 116, 118, 101, 103, 24, 26, 9, 11, 108, 110, 125, 127),

T_1=(32, 20, 48, 34, 36, 2, 54, 50, 0, 4, 38, 16, 18, 6, 22, 52),

T_2=(0, 15, 14, 3, 13, 6, 2, 8, 12, 11, 5, 10, 1, 4, 7, 9),

T_3=(0, 15, 1, 12, 2, 9, 13, 7, 3, 4, 10, 5, 14, 11, 8, 6),

T_4=(14, 8, 5, 14, 2, 13, 9, 1, 3, 5, 3, 5, 12, 13, 12, 15),

T_5=(52, 22, 0, 4, 20, 54, 50, 16, 48, 38, 34, 32, 2, 18, 6, 36),

T_6=(220, 0, 152, 147, 153, 146, 215, 78, 214, 11, 69, 1, 68, 10, 79, 221),

где векторы значений приведены в виде

$f=(f(0), f(1), \ldots, f(15))$,

где f - произвольное преобразование, множеством входных аргументов которого является V_4;

для получения значения $l(a_{15}, a_{14}, \ldots, a_0) \in V_8$, для произвольных

a_{15}, a_{14}, …, $a_0 \in V_8$, вычисляют

$t_1=a_3\oplus a_{13}$,

$t_{2=}a_8+_{256}a_8$,

$t_3=a_1\oplus a_{15}\oplus t_1\oplus BLEND(t_2, t_2\oplus 195, a_8)$,

$t_4=a_2\oplus a_{14}$,

$t_5=t_4+_{256}t_4$,

$t_6=BLEND(t_5, t_5\oplus 195, t_4)\oplus t_1\oplus a_4\oplus a_{12}\oplus a_8$,

$t_7=a_6\oplus a_{10}$,

$t_8=t_7+_{256}t_7$,

$t_9=a_5\oplus a_{11}\oplus t_7$,

$$l(a_{15}, a_{14}, ..., a_0) = a_0 \oplus a_7 \oplus a_9 \oplus t_1 \oplus t_9 \oplus MULT_{148,1}(mask \wedge$$
$$\wedge RSHIFT_4(t_3)) \oplus MULT_{148,0}(mask \wedge t_3) \oplus$$
$$\oplus MULT_{195,1}(mask \wedge RSHIFT_4(t_9)) \oplus$$
$$\oplus MULT_{195,0}((mask \wedge t_9) \oplus (mask \wedge RSHIFT_4(t_6))) \oplus$$
$$\oplus BLEND(t_8, t_8 \oplus 195, t_7) \oplus LSHIFT_4(mask \wedge t_6),$$

где преобразование $LSHIFT_4$ вида $V_4\rightarrow V_8$ вычисляют в соответствии с соотношением

$LSHIFT_4$: $V_4\rightarrow V_8$,

$$LSHIFT_4(x_3 \| x_2 \| x_1 \| x_0) = x_3 \| x_2 \| x_1 \| x_0 \| 0 \| 0 \| 0 \| 0,$$

$x_i\in V_1$, i=0, 1, …, 3;

преобразования $MULT_{148,0}$, $MULT_{148,1}$, $MULT_{195,0}$, $MULT_{195,1}$ вида $V_4\rightarrow V_8$

вычисляют путем загрузки данных из вспомогательных таблиц, содержащих векторы значений этих преобразований:

$MULT_{148,0}$=(0, 148, 235, 127, 21, 129, 254, 106, 42, 190, 193, 85, 63, 171, 212, 64),

$MULT_{148,1}$=(0, 84, 168, 252, 147, 199, 59, 111, 229, 177, 77, 25, 118, 34, 222, 138),

$MULT_{195,0}$=(0, 195, 69, 134, 138, 73, 207, 12, 215, 20, 146, 81, 93, 158, 24, 219),

$MULT_{195,1}$=(0, 109, 218, 183, 119, 26, 173, 192, 238, 131, 52, 89, 153, 244, 67, 46),

где векторы значений приведены в виде

$f=(f(0), f(1), …, f(15))$,

где f - произвольное преобразование, множеством входных аргументов которого является V_4;

- вычисляют

u=u+1;

- если u<n, то переходят к этапу (A);

получают зашифрованные сообщения c_i, i=1, 2, …, s.

Результат достигается за счет сведения базовых преобразований алгоритма зашифрования блочного шифра «Кузнечик» к эквивалентным преобразованиям, имеющим более эффективную реализацию с помощью существующих SIMD-инструкций, а также за счет учета особенностей базовых преобразований с целью минимизации количества требуемых для их выполнения операций.

Согласно предлагаемому способу, алгоритм зашифрования блочного шифра «Кузнечик» выполняют в соответствии с (2). Преобразования X, S, L вычисляют в соответствии с (3).

Рассмотрим способ выполнения преобразования l. Заметим, что в силу свойства дистрибутивности операции умножения относительно сложения в поле $GF(2^8)[X]/(X^8 \oplus X^7 \oplus X^6 X \oplus 1)$, а также в силу равенств, справедливых $\forall x \in V_8$:

$133 \cdot x = (1 \oplus 16 \oplus 148) \cdot x = 1 \cdot x \oplus 16 \cdot x \oplus 148 \cdot x,$

$32 \cdot x = (16 \cdot 2) \cdot x = 16 \cdot (2 \cdot x),$

$251 \cdot x = (148 \cdot 2 \oplus 16) \cdot x = 148 \cdot (2 \cdot x) \oplus 16 \cdot x,$

$192 \cdot x = (194 \oplus 2) \cdot x = 194 \cdot x \oplus 2 \cdot x,$

$194 \cdot x = (195 \oplus 1) \cdot x = 195 \cdot x \oplus 1 \cdot x,$

$16 \cdot x = 195 \cdot MSB_4(x) \oplus LSHIFT_4(LSB_4(x)),$

преобразование l для любых $a_{15}, a_{14}, \ldots, a_0 \in V_8$ может быть вычислено следующим образом:

$$l(a_{15}, a_{14}, ..., a_0) = 148 \cdot a_{15} \oplus 32 \cdot a_{14} \oplus 133 \cdot a_{13} \oplus 16 \cdot a_{12} \oplus 194 \cdot a_{11} \oplus$$
$$\oplus 192 \cdot a_{10} \oplus 1 \cdot a_9 \oplus 251 \cdot a_8 \oplus 1 \cdot a_7 \oplus 192 \cdot a_6 \oplus 194 \cdot a_5 \oplus 16 \cdot a_4 \oplus$$
$$\oplus 133 \cdot a_3 \oplus 32 \cdot a_2 \oplus 148 \cdot a_1 \oplus 1 \cdot a_0 =$$
$$= 1 \cdot (a_0 \oplus a_7 \oplus a_9) \oplus 148 \cdot (a_1 \oplus a_{15}) \oplus 32 \cdot (a_2 \oplus a_{14}) \oplus 133 \cdot (a_3 \oplus$$
$$\oplus a_{13}) \oplus 16 \cdot (a_4 \oplus a_{12}) \oplus 194 \cdot (a_5 \oplus a_{11}) \oplus 192 \cdot (a_6 \oplus a_{10}) \oplus 251 \cdot a_8 =$$
$$= 1 \cdot (a_0 \oplus a_7 \oplus a_9) \oplus 148 \cdot (a_1 \oplus a_{15}) \oplus (16 \cdot 2) \cdot (a_2 \oplus a_{14}) \oplus (1 \oplus 16 \oplus$$
$$\oplus 148) \cdot (a_3 \oplus a_{13}) \oplus 16 \cdot (a_4 \oplus a_{12}) \oplus (195 \oplus 1) \cdot (a_5 \oplus a_{11}) \oplus (195 \oplus 1 \oplus$$
$$\oplus 2) \cdot (a_6 \oplus a_{10}) \oplus (148 \cdot 2 \oplus 16) \cdot a_8 =$$
$$= 1 \cdot (a_0 \oplus a_7 \oplus a_9 \oplus a_3 \oplus a_{13} \oplus a_5 \oplus a_{11} \oplus a_6 \oplus a_{10}) \oplus 148 \cdot (a_1 \oplus a_{15} \oplus$$
$$\oplus a_3 \oplus a_{13} \oplus 2 \cdot a_8) \oplus 16 \cdot (2 \cdot (a_2 \oplus a_{14}) \oplus a_3 \oplus a_{13} \oplus a_4 \oplus a_{12} \oplus a_8) \oplus$$
$$\oplus 195 \cdot (a_5 \oplus a_{11} \oplus a_6 \oplus a_{10}) \oplus 2 \cdot (a_6 \oplus a_{10}) =$$
$$= a_0 \oplus a_7 \oplus a_9 \oplus a_3 \oplus a_{13} \oplus a_5 \oplus a_{11} \oplus a_6 \oplus a_{10} \oplus 148 \cdot (a_1 \oplus a_{15} \oplus a_3 \oplus$$
$$\oplus a_{13} \oplus 2 \cdot a_8) \oplus 195 \cdot (a_5 \oplus a_{11} \oplus a_6 \oplus a_{10} \oplus MSB_4(2 \cdot (a_2 \oplus a_{14}) \oplus a_3 \oplus$$
$$\oplus a_{13} \oplus a_4 \oplus a_{12} \oplus a_8)) \oplus 2 \cdot (a_6 \oplus a_{10}) \oplus LSHIFT_4(LSB_4(2 \cdot (a_2 \oplus$$
$$\oplus a_{14}) \oplus a_3 \oplus a_{13} \oplus a_4 \oplus a_{12} \oplus a_8)).$$

Для выполнения операций умножения в поле введем следующие преобразования:

$$BLEND : V_8 \times V_8 \times V_8 \to V_8, \quad BLEND(x, y, z) = \begin{cases} x, & \text{если } MSB_1(z) = 0, \\ y, & \text{если } MSB_1(z) = 1, \end{cases}$$
$$MULT_{148,0} : V_4 \to V_8, \quad MULT_{148,0}(x) = 148 \cdot x,$$
$$MULT_{148,1} : V_4 \to V_8, \quad MULT_{148,1}(x) = 148 \cdot LSHIFT_4(x), \qquad (10)$$
$$MULT_{195,0} : V_4 \to V_8, \quad MULT_{195,0}(x) = 195 \cdot x,$$
$$MULT_{195,1} : V_4 \to V_8, \quad MULT_{195,1}(x) = 195 \cdot LSHIFT_4(x),$$

где

$MSB_1:V_8 \to V_1$, $MSB_1(x_7 \| x_6 \| ... \| x_0) = x_7$, $x_i \in V_1$, i=0, 1, ..., 7

С учетом (10) и равенства

$2 \cdot 128 = 195$,

где $2, 128, 195 \in GF(2^8)[X]/(X^8 \oplus X^7 \oplus X^6 \oplus X \oplus 1)$,

получаем, что $\forall x \in V_8$:

$2 \cdot x = BLEND(x +_{256} x, (x +_{256} x) \oplus 195, x)$,

$148 \cdot x = MULT_{148,0}(LSB_4(x)) \oplus MULT_{148,1}(MSB_4(x))$,

$195 \cdot x = MULT_{195,0}(LSB_4(x)) \oplus MULT_{195,1}(MSB_4(x))$,

где «$+_{256}$» - операция сложения в кольце вычетов по модулю 256.

Для выполнения преобразования $LSHIFT_4$ воспользуемся (6). Для выполнения преобразований MSB_4, LSB_4 воспользуемся соотношениями

$$LSB_4(x) = mask \wedge x, \quad MSB_4(x) = mask \wedge RSHIFT_4(x), \tag{11}$$

где «$\wedge$» - операция побитового "И",

$mask = 0 \| 0 \| 0 \| 0 \| 1 \| 1 \| 1 \| 1 \in V_8$,

$RSHIFT_4:V_8 \to V_8$,

$$RSHIFT_4(x_7 \| x_6 \| x_5 \| x_4 \| x_3 \| x_2 \| x_1 \| x_0) = y_3 \| y_2 \| y_1 \| y_0 \| x_7 \| x_6 \| x_5 \| x_4$$

причем $y_j \in V_1$, i=0, 1, 2, 3 - не зависящие от $x_i \in V_1$, i=0, 1, …, 7 значения.

В результате способ вычисления преобразования l принимает следующий вид:

- на вход преобразованию l поступают значения $a_{15}, a_{14}, \ldots, a_0 \in V_8$;
- вычисляют значения

$$
\begin{aligned}
&t_1 = a_3 \oplus a_{13},\\
&t_2 = a_8 +_{256} a_8,\\
&t_3 = a_1 \oplus a_{15} \oplus t_1 \oplus BLEND(t_2, t_2 \oplus 195, a_8),\\
&t_4 = a_2 \oplus a_{14},\\
&t_5 = t_4 +_{256} t_4,\\
&t_6 = BLEND(t_5, t_5 \oplus 195, t_4) \oplus t_1 \oplus a_4 \oplus a_{12} \oplus a_8,\\
&t_7 = a_6 \oplus a_{10},\\
&t_8 = t_7 +_{256} t_7,\\
&t_9 = a_5 \oplus a_{11} \oplus t_7,\\
&l(a_{15}, a_{14}, ..., a_0) = a_0 \oplus a_7 \oplus a_9 \oplus t_1 \oplus t_9 \oplus MULT_{148,1}(mask \wedge\\
&\qquad \wedge RSHIFT_4(t_3)) \oplus MULT_{148,0}(mask \wedge t_3) \oplus\\
&\qquad \oplus MULT_{195,1}(mask \wedge RSHIFT_4(t_9)) \oplus\\
&\qquad \oplus MULT_{195,0}((mask \wedge t_9) \oplus (mask \wedge RSHIFT_4(t_6))) \oplus\\
&\qquad \oplus BLEND(t_8, t_8 \oplus 195, t_7) \oplus LSHIFT_4(mask \wedge t_6).
\end{aligned}
\tag{12}
$$

Предложенный способ позволяет вычислять преобразование l при помощи пяти преобразований вида $V_4 \to V_8$: $MULT_{148,0}$, $MULT_{148,1}$, $MULT_{195,0}$, $MULT_{195,1}$, $LSHIFT_4$; преобразования $RSHIFT_4$ вида $V_8 \to V_8$; преобразования BLEND вида $V_8 \times V_8 \times V_8 \to V_8$; а также операций сложения в кольце вычетов по модулю 256 («$+_{256}$»), побитового "И" («$\wedge$») и побитового сложения («$\oplus$»).

Преобразования $MULT_{148,0}$, $MULT_{148,1}$, $MULT_{195,0}$, $MULT_{195,1}$ могут быть вычислены с помощью вспомогательных таблиц, содержащих векторы значений этих преобразований:

$MULT_{148,0}$=(0, 148, 235, 127, 21, 129, 254, 106, 42, 190, 193, 85, 63, 171, 212, 64),

$MULT_{148,1}$=(0, 84, 168, 252, 147, 199, 59, 111, 229, 177, 77, 25, 118, 34, 222, 138),

$MULT_{195,0}$=(0, 195, 69, 134, 138, 73, 207, 12, 215, 20, 146, 81, 93, 158, 24, 219),

$MULT_{195,1}$=(0, 109, 218, 183, 119, 26, 173, 192, 238, 131, 52, 89, 153, 244, 67, 46),

где векторы значений получены из (10) и приведены в виде

$f=(f(0), f(1), \ldots, f(15))$,

где f - произвольное преобразование, множеством входных аргументов которого является V_4.

Рассмотрим способ выполнения преобразования π. Согласно [5], преобразование π может быть вычислено в соответствии с (1).

Для выполнения операции умножения в поле $GF(2^4)[X]/(X^4 \oplus X^3 \oplus 1)$ перейдем от векторного представления элементов поля к степенному представлению. Для этого введем преобразования:

$$EXP: V_4 \to V_4, \quad EXP(x) = \begin{cases} 0, & \text{если } x = 0, \\ X^x, & \text{если } x \neq 0, \end{cases}$$

$$LOG: V_4 \to V_4, \quad LOG(x) = \begin{cases} 0, & \text{если } x = 0, \\ EXP^{-1}(x), & \text{если } x \neq 0, \end{cases}$$

$$MOD: V_8 \to V_4, \quad MOD(x) = \begin{cases} 15, & \text{если } x \bmod 15 = 0, \\ x \bmod 15, & \text{если } x \bmod 15 \neq 0, \end{cases}$$

где X=2 - элемент поля $GF(2^4)[X]/(X^4 \oplus X^3 \oplus 1)$, являющийся образующим элементом мультипликативной группы этого поля.

Тогда $\forall x,$ $y \in V_4$, $x \neq 0$, $y \neq 0$:

$$x \circ y = EXP(MOD(LOG(x) +_{256} LOG(y))) \qquad (13)$$

С учетом (8), (9) и (13), а также соотношений

$$\forall x \in V_4 : \varphi(x) \neq 0,$$

$$I(x) = 0 \Leftrightarrow x = 0,$$

преобразование π может быть вычислено как

$$\pi(a) = \begin{cases} \omega_1(\nu_0(MSB_4(\alpha(a)))) \oplus \omega_0(\sigma(0)), & \text{если } LSB_4(\alpha(a)) = 0, \\ \omega_1(a') \oplus \omega_0(\sigma(EXP(MOD(LOG(\phi(a')) +_{256} \\ \qquad +_{256} LOG(LSB_4(\alpha(a))))))), & \text{если } LSB_4(\alpha(a)) \neq 0, \end{cases}$$

$$a' = \begin{cases} \nu_1(EXP(0)), & \text{если } MSB_4(\alpha(a)) = 0, \\ \nu_1(EXP(MOD(LOG(I(LSB_4(\alpha(a)))) +_{256} \\ \qquad +_{256} LOG(MSB_4(\alpha(a)))))), & \text{если } MSB_4(\alpha(a)) \neq 0. \end{cases}$$

Перейдем от величины a' к величине $a''=EXP^{-1}(\nu_1^{-1}(a'))$. С учетом того, что $\omega_0(\sigma(0))=220 \in V_8$,

получаем, что

$$\pi(a) = \begin{cases} \omega_1(\nu_0(MSB_4(\alpha(a)))) \oplus 220, & \text{если } LSB_4(\alpha(a)) = 0, \\ \omega_1(\nu_1(EXP(a''))) \oplus \\ \quad \oplus \omega_0(\sigma(EXP(MOD(LOG(\phi(\nu_1(EXP(a'')))) +_{256} \\ \qquad +_{256} LOG(LSB_4(\alpha(a)))))), & \text{если } LSB_4(\alpha(a)) \neq 0, \end{cases}$$

$$a'' = \begin{cases} 0, & \text{если } MSB_4(\alpha(a)) = 0, \\ MOD(LOG(I(LSB_4(\alpha(a)))) +_{256} \\ \qquad +_{256} LOG(MSB_4(\alpha(a)))), & \text{если } MSB_4(\alpha(a)) \neq 0. \end{cases}$$

Введем преобразования

$$
\begin{aligned}
&T_1 : V_4 \to V_8, \quad T_1(x) = \omega_1(v_0(x)),\\
&T_2 : V_4 \to V_4, \quad T_2(x) = LOG(I(x)),\\
&T_3 : V_4 \to V_4, \quad T_3(x) = LOG(x),\\
&T_4 : V_4 \to V_4, \quad T_4(x) = LOG(\phi(v_1(EXP(x)))),\\
&T_5 : V_4 \to V_8, \quad T_5(x) = \omega_1(v_1(EXP(x))),\\
&T_6 : V_4 \to V_8, \quad T_6(x) = \omega_0(\sigma(EXP(x)).
\end{aligned}
\tag{14}
$$

Тогда

$$
\pi(a) = \begin{cases} T_1(MSB_4(\alpha(a))) \oplus 220, & \text{если } LSB_4(\alpha(a)) = 0,\\ T_5(a") \oplus T_6(MOD(T_4(a") +_{256} & \\ \quad +_{256} T_3(LSB_4(\alpha(a))))), & \text{если } LSB_4(\alpha(a)) \neq 0, \end{cases}
$$

$$
a" = \begin{cases} 0, & \text{если } MSB_4(\alpha(a)) = 0,\\ MOD(T_2(LSB_4(\alpha(a))) +_{256} & \\ \quad +_{256} T_3(MSB_4(\alpha(a)))), & \text{если } MSB_4(\alpha(a)) \neq 0. \end{cases}
$$

Для выполнения преобразования MOD заметим, что

$\forall x$, y∈V_4, x≠0, y≠0: LOG(x)+$_{256}$LOG(y)∈{2, 3,…,30},

$\forall x$=2, 3, …, 30∈V_8: MOD(x)=LSB_4(x+$_{256}$$MSB_4$(x))

Для выполнения преобразований α, LSB_4, MSB_4 воспользуемся (9) и (11).

Для вычисления кусочно-заданных функций воспользуемся соотношениями:

$$
u = \begin{cases} x, & \text{если } z = 0,\\ y, & \text{если } z \neq 0, \end{cases} \Leftrightarrow u = BLEND(y, x, CMPR(z, 0)),
$$

$$
u = \begin{cases} 0, & \text{если } z = 0,\\ y, & \text{если } z \neq 0, \end{cases} \Leftrightarrow u = y \wedge (\neg CMPR(z, 0)),
$$

где «¬» - операция побитового отрицания,

$$
CMPR : V_8 \times V_8 \to V_8, \quad CMPR(x, y) = \begin{cases} 255, & \text{если } x = y,\\ 0, & \text{если } x \neq y \end{cases}
$$

В результате, способ вычисления преобразования π принимает следующий вид:

- на вход преобразованию π поступает значение a∈V_8;
- вычисляют значения

$$
\begin{aligned}
r_1 &= \alpha_0(mask \wedge a) \oplus \alpha_1(mask \wedge RSHIFT_4(a)),\\
r_2 &= mask \wedge r_1,\\
r_3 &= mask \wedge RSHIFT_4(r_1),\\
r_4 &= T_2(r_2) +_{256} T_3(r_3),\\
r_5 &= (mask \wedge (r_4 +_{256} (mask \wedge RSHIFT_4(r_4)))) \wedge (\neg CMPR(r_3, 0)),\\
r_6 &= T_4(r_5) +_{256} T_3(r_2),\\
\pi(a) &= BLEND(T_5(r_5) \oplus T_6(mask \wedge (r_6 +_{256} (mask \wedge\\
&\quad \wedge RSHIFT_4(r_6)))), T_1(r_3) \oplus 220, CMPR(r_2, 0))
\end{aligned}
\qquad (15)
$$

Предложенный способ позволяет вычислять преобразование π при помощи трех преобразований вида $V_4 \to V_4$: T_2, T_3, T_4; пяти преобразований вида $V_4 \to V_8$: α_0, α_1, T_1, T_5, T_6; преобразования $RSHIFT_4$ вида $V_8 \to V_8$; преобразования CMPR вида $V_8 \times V_8 \to V_8$; преобразования BLEND вида $V_8 \times V_8 \times V_8 \to V_8$; а также операций сложения в кольце вычетов по модулю 256 («$+_{256}$»), побитового "И" («$\wedge$»), побитового "И" с отрицанием одного из аргументов («$x \wedge (\neg y)$») и побитового сложения («$\oplus$»).

Преобразования α_0, α_1, T_1, T_2, T_3, T_4, T_5, T_6 могут быть вычислены с помощью вспомогательных таблиц, содержащих векторы значений этих преобразований:

α_0=(0, 112, 58, 74, 20, 100, 46, 94, 154, 234, 160, 208, 142, 254, 180, 196),

α_1=(0, 2, 17, 19, 116, 118, 101, 103, 24, 26, 9, 11, 108, 110, 125, 127),

T_1=(32, 20, 48, 34, 36, 2, 54, 50, 0, 4, 38, 16, 18, 6, 22, 52),

T_2=(0, 15, 14, 3, 13, 6, 2, 8, 12, 11, 5, 10, 1, 4, 7, 9),

T_3=(0, 15, 1, 12, 2, 9, 13, 7, 3, 4, 10, 5, 14, 11, 8, 6),

T_4=(14, 8, 5, 14, 2, 13, 9, 1, 3, 5, 3, 5, 12, 13, 12, 15),

T_5=(52, 22, 0, 4, 20, 54, 50, 16, 48, 38, 34, 32, 2, 18, 6, 36),

T_6=(220, 0, 152, 147, 153, 146, 215, 78, 214, 11, 69, 1, 68, 10, 79, 221),

где векторы значений получены из формул (8), (14) и приведены в виде $f=(f(0), f(1), \ldots, f(15))$,

где f - произвольное преобразование, множеством входных аргументов которого является V_4.

Итоговое количество преобразований и операций, необходимых для вычисления преобразований l и π предлагаемым способом, а также способом, выбранным в качестве прототипа, приведено в табл. 1. Для наглядности, однотипные преобразования, предполагающие одинаковый способ выполнения при реализации, например, использование вспомогательных таблиц, объединены в одной строке, при этом численные значения в соответствующей строке означают суммарное количество применений преобразований данного типа в конкретном способе.

Для возможности сравнения предлагаемого способа со способом, выбранным в качестве прототипа, используется следующий вариант выполнения умножения в поле $GF(2^4)[X]/(X^4 \oplus X^3 \oplus 1)$, допускающий применение SIMD-инструкций для каждой из имеющихся операций:

- на вход операции умножения поступают значения $x, y \in V_4$;

- вычисляют значения

$$p_1 = (x \wedge (\neg CMPR(y \wedge mask_0, 0))) \oplus (LSHIFT_1(x) \wedge (\neg CMPR(y \wedge \wedge mask_1, 0))) \oplus (LSHIFT_2(x) \wedge (\neg CMPR(y \wedge mask_2, 0))) \oplus \oplus (LSHIFT_3(x) \wedge (\neg CMPR(y \wedge mask_3, 0))),$$

$$x \circ y = (mask \wedge p_1) \oplus MULT_9(mask \wedge RSHIFT_4(p_1)),$$

где

$$mask_0 = 0\|0\|0\|0\|0\|0\|0\|1 \in V_8,$$

$$mask_1 = 0\|0\|0\|0\|0\|0\|1\|0 \in V_8,$$

$$mask_2 = 0\|0\|0\|0\|0\|1\|0\|0 \in V_8,$$

$$mask_3 = 0\|0\|0\|0\|1\|0\|0\|0 \in V_8,$$

$LSHIFT_1$, $LSHIFT_2$, $LSHIFT_3$:$V_4 \to V_8$,

$$LSHIFT_1(x_3\|x_2\|x_1\|x_0) = 0\|0\|0\|x_3\|x_2\|x_1\|x_0\|0,$$

$$LSHIFT_2(x_3\|x_2\|x_1\|x_0) = 0\|0\|x_3\|x_2\|x_1\|x_0\|0\|0,$$

$$LSHIFT_3(x_3\|x_2\|x_1\|x_0) = 0\|x_3\|x_2\|x_1\|x_0\|0\|0\|0,$$

$x_i \in V_1$, i=0, 1, 2, 3,

$$MULT_9 : V_4 \to V_4, \quad MULT_9(x) = 9 \circ x.$$

Тогда алгоритм вычисления преобразования π согласно способу, выбранному в качестве прототипа, принимает следующий вид:

- на вход преобразованию π поступает значение $a \in V_8$;

- вычисляют значения

$$s_1 = \alpha_0(mask \wedge a) \oplus \alpha_1(mask \wedge RSHIFT_4(a)),$$

$$s_2 = mask \wedge s_1,$$

$$s_3 = mask \wedge RSHIFT_4(s_1),$$

$$s_4 = BLEND(\nu_1(I(s_2) \circ s_3), \nu_0(s_3), CMPR(s_2, 0)),$$

$$\pi(a) = \omega_0(\sigma(s_2 \circ \varphi(s_4))) \oplus \omega_1(s_4),$$

причем умножение в поле $GF(2^4)[X]/(X^4 \oplus X^3 \oplus 1)$ выполняют описанным выше способом.

Преобразование l в прототипе выполняется в соответствии с (4), (5), (7).

Таким образом, практически по каждому типу преобразований и операций, использующемуся в прототипе, удается добиться существенного сокращения количества преобразований и операций этого типа в случае применения предлагаемого способа. При этом в предлагаемом способе появляется необходимость выполнения незначительного числа новых типов преобразований и операций, однако, малочисленность и эффективность этих операций позволяет утверждать, что негативный эффект от их введения значительно меньше позитивного эффекта от сокращения

количества имеющихся в прототипе преобразований и операций.

Таблица 1. Количество преобразований и операций, необходимых для вычисления преобразований l и π в предлагаемом способе и прототипе.

Операция или преобразование	Преобразование l		Преобразование π	
	Предлагаемый способ	Прототип	Предлагаемый способ	Прототип
$MULT_{195,0}$, $MULT_{195,1}$, $MULT_9$, $MULT_{const,0}$, $MULT_{const,1}$, $const \in \{1, 148, 32, 133, 16, 194, 192, 251\}$, α_0, α_1, ω_0, ω_1, I, ν_0, ν_1, φ, σ, T_1, T_2, T_3, T_4, T_5, T_6	4	32	9	11
$BLEND$	3	0	1	1
$RSHIFT_4$	3	16	4	4
$LSHIFT_1$, $LSHIFT_2$, $LSHIFT_3$, $LSHIFT_4$	1	0	0	6
$CMPR$	0	0	2	9
$+_{256}$	3	0	4	0
$\wedge$	6	32	8	16
$x \wedge (\neg y)$	0	0	1	8
$\oplus$	26	31	3	10

В связи с тем, что предлагаемый способ и способ, выбранный в качестве прототипа, отличаются только в части вычисления преобразования l и π, а производительность выполнения этих преобразований выше в предлагаемом способе, общая производительность предлагаемого способа превосходит общую производительность прототипа. Следует отметить, что данное соотношение производительности способов справедливо как в случае обработки одного блока данных, так и в случае параллельной обработки нескольких блоков данных с помощью SIMD-инструкций.

При этом предлагаемый способ сохраняет важные свойства прототипа. Сохраняется пропорциональный рост скорости программных реализаций на основе способа при увеличении длины используемых регистров, поскольку все использующиеся в предлагаемом способе преобразования и операции эффективно распараллеливаются с помощью существующих SIMD-инструкций. Также сохраняется возможность получения на основе способа реализаций, стойких к атакам по времени обращения к кэш-памяти вычислительной платформы, поскольку в предлагаемом способе используются только малые по размеру вспомогательные таблицы и, как следствие, появляется возможность задействования только инструкций с фиксированным временем выполнения.

Осуществление изобретения

Для осуществления предложенного способа рассмотрим вариант его программной

реализации на вычислительной платформе с SIMD-архитектурой.

В качестве вычислительной платформы с SIMD-архитектурой используем процессор общего назначения Intel Core i7-4700. Для повышения эффективности параллельных вычислений задействуем регистры максимальной для данной модели процессора длины - 256 бит. Для работы с этими регистрами используем SIMD-инструкций из наборов AVX и AVX2.

Рассмотрим пример одновременного зашифрования 32 блоков данных, длиной 128 бит каждый. Будем считать, что итерационные ключи $K_1, \ldots, K_{10} \in V_{128}$ уже вычислены и алгоритм зашифрования получает их на вход вместе с шифруемыми данными $a_1, a_2, \ldots, a_{32} \in V_{128}$.

Для осуществления предложенного способа выполняют следующие предварительные процедуры:

- для каждого преобразования $MULT_{148,0}$, $MULT_{148,1}$, $MULT_{195,0}$, $MULT_{195,1}$, α_0, α_1, T_1, T_2, T_3, T_4, T_5, T_6 формируют переменную длиной 256 бит, необходимую для табличного вычисления соответствующего преобразования, и
- побайтно заполняют младшие 128 бит каждой переменной элементами вектора значений соответствующего преобразования, причем в младшем (первом) байте размещают первый элемент вектора значений, во втором байте - второй элемент вектора значений и т.д.;
- заполняют старшие 128 бит переменной аналогично младшим 128 битам переменной;
- для константы mask формируют переменную длиной 256 бит и побайтно заполняют ее значениями константы mask;
- для итерационного ключа формируют переменную rk к длиной 256 бит;
- для данных формируют 16 переменных длиной 256 бит каждая: $data_1, data_2, \ldots, data_{16}$, и
- побайтно заполняют переменную $data_i$, i=1, 2, …, 16, размещая в ней i-е байты каждого из 32 блоков $a_1, a_2, \ldots, a_{32} \in V_{128}$: в младшем (первом) байте размещают i-й байт блока a_1, во втором байте размещают i-й байт блока a_2 и т.д.

Для одновременного зашифрования блоков a_i, i=1, 2, …, 32, в соответствии с формулой:

$E_{K1, \ldots, K10} : V_{128} \to V_{128}$,

$E_{K1,\ldots, K10}(a_i) = X[K_{10}]LSX[K_9]\ldots LSX[K_2]LSX[K_1(a_i)]$,

последовательно осуществляют изменение значений переменных $data_1, data_2, \ldots, data_{16}$, соответствующее выполнению преобразований: $X[K_j]$, j=1, 2, …, 10, L и S.

Для выполнения преобразования $X[K_j]$, j=1, 2, …, 10: для каждого i=1, 2, …, 16

- побайтно заполняют переменную rk значениями i-го байта ключа K_j;
- вычисляют $data_i := data_i \oplus rk$.

Для выполнения линейного преобразования L: для каждого i=0, 1, …, 15 последовательно вычисляют

$$data_{i+1} := l'(data_{((i+15) \bmod 16)+1}, data_{((i+14) \bmod 16)+1}, \ldots, data_{(i \bmod 16)+1}),$$

где l' представляет собой параллельное применение преобразования l к j-м байтам аргументов l' для получения j-го байта результата преобразования l', j=1, 2, …, 32.

Преобразование l выполняют в соответствии (12), а параллельное применение l осуществляют с использованием следующих SIMD-инструкций:

- "vpshufb" для реализации преобразований $MULT_{148,0}$, $MULT_{148,1}$, $MULT_{195,0}$, $MULT_{195,1}$;
- "vpblendvb" для реализации преобразования BLEND;
- "vpsrlw" для реализации преобразования $RSHIFT_4$;
- "vpsllw" для реализации преобразования $LSHIFT_4$;
- "vpaddb" для реализации операции «$+_{256}$»;
- "vpand" для реализации операции «$\wedge$»;
- "vpxor" для реализации операции «$\oplus$».

При реализации преобразований $MULT_{148,0}$, $MULT_{148,1}$, $MULT_{195,0}$, $MULT_{195,1}$, и задействовании константы mask используют введенные для них переменные.

Для осуществления нелинейного преобразования S: для каждого i=1, 2, …, 16 вычисляют

$data_i:=\pi'(data_i)$,

где π' представляет собой параллельное применение преобразования π к j-му байту аргумента π' для получения j-го байта результата π', j=1, 2, …, 32.

Преобразование π выполняют в соответствии с (15), а параллельное применение π осуществляют с использованием следующих SIMD-инструкций:

- "vpshufb" для реализации преобразований α_0, α_1, T_1, T_2, T_3, T_4, T_5, T_6;
- "vpblendvb" для реализации преобразования BLEND;
- "vpsrlw" для реализации преобразования $RSHIFT_4$;
- "vpcmpeqb" для реализации преобразования CMPR;
- "vpaddb" для реализации операции «$+_{256}$»;
- "vpand" для реализации операции «$\wedge$»;
- "vpandn" для реализации операции «$x \wedge (\neg y)$»;
- "vpxor" для реализации операции «$\oplus$».

При реализации преобразований α_0, α_1, T_1, T_2, T_3, T_4, T_5, T_6 и задействовании константы mask используют введенные для них переменные.

Для получения зашифрованных блоков $b_1, b_2, \ldots, b_{32} \in V_{128}$ из итоговых значений переменных $data_1, data_2, \ldots, data_{16}$, выполняют перекомпоновку байт, обратную той, что применялась при начальном заполнении переменных $data_1, data_2, \ldots, data_{16}$, для чего побайтно заполняют блок b_i, i=1, 2, …, 32, размещая в нем i-е байты каждой из 16 переменных $data_1, data_2, \ldots, data_{16}$, причем в младшем (первом) байте размещают i-й байт переменной $data_1$, во втором байте размещают i-й байт переменной $data_2$ и т.д.

В случае необходимости применения алгоритма зашифрования к большему числу блоков данных выполняют разбиение всех входных блоков на группы, состоящие не более чем из 32 блоков, после чего обрабатывают каждую группу в отдельности. При этом, если обрабатываемая группа содержит менее 32 блоков, ее предварительно дополняют произвольными блоками в количестве, необходимом для получения 32 блоков в группе, а результат зашифрования добавленных блоков игнорируют.

Скорость описанной программной реализации алгоритма зашифрования блочного шифра «Кузнечик», основанной на предлагаемом способе, составляет 360 Мбайт/с при

выполнении вычислений в одном потоке на одном ядре процессора Intel Core i7-4700. Программная реализация алгоритма зашифрования блочного шифра «Кузнечик», основанная на способе, выбранном в качестве прототипа, согласно [3], имеет скорость равную 300 Мбайт/с на аналогичной вычислительной платформе. Таким образом, прирост производительности предлагаемого способа относительно прототипа составляет порядка 20%.

Источники информации, принятые во внимание при составлении заявки

1. ГОСТ Р 34.12-2015. Информационная технология, Криптографическая защита информации, Блочные шифры. Москва, Стандартинформ, 2016.

2. Евразийский патент №021803, приоритет от 25.04.2012 г.

3. Rybkin A. S. On software implementation of Kuznyechik on Intel CPUs. Математические вопросы криптографии, 9:2 (2018), с. 117-127.

4. Bernstein D. J. Cache-timing attacks on AES. 2005 (статья по адресу: https://cr.yp.to/antiforgery/cachetiming-20050414.pdf).

5. Biryukov, A., Perrin, L., Udovenko, A. Reverse-engineering the S-box of Streebog, Kuznyechik and STRIBOBrl. In Annual International Conference on the Theory and Applications of Cryptographic Techniques (2016), 372-402, Springer, Berlin, Heidelberg.

(57) Формула изобретения

Способ шифрования s сообщений $m_1, m_2, \ldots, m_s$, представленных в двоичном виде и имеющих длину, равную 128 бит каждый, где $s=t\cdot n$, причем t, n - натуральные числа, реализуемый посредством вычислительной системы, имеющей процессор с SIMD-архитектурой, заключающийся в том, что

вычисляют

u=0;

(A) вычисляют параллельно с использованием SIMD-инструкций процессора значения $c_{ut+1}, c_{ut+2}, c_{ut+3}, \ldots, c_{ut+t} \in V_{128}$:

$$c_{ut+1} = X[K_{10}]LSX[K_9]\ldots LSX[K_2]LSX[K_1](m_{ut+1}),$$

$$c_{ut+2} = X[K_{10}]LSX[K_9]\ldots LSX[K_2]LSX[K_1](m_{ut+2}),$$

$$c_{ut+3} = X[K_{10}]LSX[K_9]\ldots LSX[K_2]LSX[K_1](m_{ut+3}),$$

$\ldots$

$$c_{ut+t} = X[K_{10}]LSX[K_9]\ldots LSX[K_2]LSX[K_1](m_{ut+t}),$$

где

$X[k]: V_{128} \to V_{128}$, $X[k](x)=k \oplus x$, причем $k, x \in V_{128}$;

$S: V_{128} \to V_{128}$, $S(x) = S(x_{15}\|\ldots\|x_0) = \pi(x_{15})\|\ldots\|\pi(x_0)$, причем $x_i \in V_8$, i=0, 1, …, 15;

$L: V_{128} \to V_{128}$, $L(x)=R^{16}(x)$, причем $x \in V_{128}$;

$R: V_{128} \to V_{128}$, $R(x) = R(x_{15}\|\ldots\|x_0) = l(x_{15},\ldots,x_0)\|x_{15}\|\ldots\|x_1$, причем $x_i \in V_8$, i=0, 1, …, 15;

$K_1, \ldots, K_{10} \in V_{128}$ - итерационные ключи;

V_p - множество всех двоичных строк длины p;

p - неотрицательное целое число;

причем для получения значения $\pi(a)\in V_8$, для произвольного $a\in V_8$, вычисляют

$$r_1 = \alpha_0(mask \wedge a) \oplus \alpha_1(mask \wedge RSHIFT_4(a)),$$

$$r_2 = mask \wedge r_1,$$

$$r_3 = mask \wedge RSHIFT_4(r_1),$$

$r_4=T_2(r_2)+_{256}T_3(r_3)$,

$$r_5 = (mask \wedge (r_4 +_{256} (mask \wedge RSHIFT_4(r_4)))) \wedge (\neg CMPR(r_3, 0)),$$

$r_6=T_4(r_5)+_{256}T_3(r_2)$,

$$\pi(a) = BLEND(T_5(r_5) \oplus T_6(mask \wedge (r_6 +_{256} (mask \wedge \wedge RSHIFT_4(r_6)))), T_1(r_3) \oplus 220, CMPR(r_2, 0)),$$

где «$+_{256}$» - операция сложения в кольце вычетов по модулю 256;

«$\wedge$» - операция побитового "И";

«$x \wedge (\neg y)$» - операция побитового "И" с отрицанием одного из аргументов;

«$\oplus$» - операция побитового сложения;

$$mask = 0\|0\|0\|0\|1\|1\|1\|1 \in V_8;$$

преобразование $RSHIFT_4$ вида $V_8 \to V_8$, преобразование CMPR вида $V_8 \times V_8 \to V_8$ и преобразование BLEND вида $V_8 \times V_8 \times V_8 \to V_8$ вычисляют в соответствии с соотношениями

$$RSHIFT_4(x_7\|x_6\|x_5\|x_4\|x_3\|x_2\|x_1\|x_0) = y_3\|y_2\|y_1\|y_0\|x_7\|x_6\|x_5\|x_4,$$

где $y_i\in V_1$, i=0, 1, 2, 3, - не зависящие от $x_i\in V_1$, i=0, 1,…, 7, значения;

$$CMPR(x,y) = \begin{cases} 255, & \text{если } x = y, \\ 0, & \text{если } x \neq y; \end{cases}$$

$$BLEND(x,y,z) = \begin{cases} x, & \text{если } MSB_1(z) = 0, \\ y, & \text{если } MSB_1(z) = 1, \end{cases}$$

где $MSB_1: V_8 \to V_1$, $MSB_1(x_7\|x_6\|\ldots\|x_0) = x_7$, $x_i\in V_1$, i=0, 1, …, 7;

преобразования T_2, T_3, T_4 вида $V_4 \to V_4$ и преобразования α_0, α_1, T_1, T_5, T_6 вида $V_4 \to V_8$ вычисляют путем загрузки данных из вспомогательных таблиц, содержащих векторы значений этих преобразований:

α_0=(0, 112, 58, 74, 20, 100, 46, 94, 154, 234, 160, 208, 142, 254, 180, 196),

α_1=(0, 2, 17, 19, 116, 118, 101, 103, 24, 26, 9, 11, 108, 110, 125, 127),

T_1=(32, 20, 48, 34, 36, 2, 54, 50, 0, 4, 38, 16, 18, 6, 22, 52),

T_2=(0, 15, 14, 3, 13, 6, 2, 8, 12, 11, 5, 10, 1, 4, 7, 9),

T_3=(0, 15, 1, 12, 2, 9, 13, 7, 3, 4, 10, 5, 14, 11, 8, 6),

T_4=(14, 8, 5, 14, 2, 13, 9, 1, 3, 5, 3, 5, 12, 13, 12, 15),

T_5=(52, 22, 0, 4, 20, 54, 50, 16, 48, 38, 34, 32, 2, 18, 6, 36),

T_6=(220, 0, 152, 147, 153, 146, 215, 78, 214, 11, 69, 1, 68, 10, 79, 221),

где векторы значений приведены в виде

$f=(f(0), f(1), \ldots, f(15))$,

где f - произвольное преобразование, множеством входных аргументов которого является V_4;

для получения значения $l(a_{15}, a_{14}, ..., a_0) \in V_8$, для произвольных $a_{15}, a_{14}, \ldots, a_0 \in V_8$, вычисляют

$t_1=a_3 \oplus a_{13}$,

$t_2=a_8 +_{256} a_8$,

$t_3=a_1 \oplus a_{15} \oplus t_1 \oplus BLEND(t_2, t_2 \oplus 195, a_8)$,

$t_4=a_2 \oplus a_{14}$,

$t_5=t_4 +_{256} t_4$,

$t_6=BLEND(t_5, t_5 \oplus 195, t_4) \oplus t_1 \oplus a_4 \oplus a_{12} \oplus a_8$,

$t_7=a_6 \oplus a_{10}$,

$t_8=t_7 +_{256} t_7$,

$t_9=a_5 \oplus a_{11} \oplus t_7$,

$$l(a_{15}, a_{14}, ..., a_0) = a_0 \oplus a_7 \oplus a_9 \oplus t_1 \oplus t_9 \oplus MULT_{148,1}(mask \wedge \wedge RSHIFT_4(t_3)) \oplus MULT_{148,0}(mask \wedge t_3) \oplus \oplus MULT_{195,1}(mask \wedge RSHIFT_4(t_9)) \oplus \oplus MULT_{195,0}((mask \wedge t_9) \oplus (mask \wedge RSHIFT_4(t_6))) \oplus \oplus BLEND(t_8, t_8 \oplus 195, t_7) \oplus LSHIFT_4(mask \wedge t_6),$$

где преобразование $LSHIFT_4$ вида $V_4 \to V_8$ вычисляют в соответствии с соотношением

$LSHIFT_4: V_4 \to V_8$, $LSHIFT_4(x_3 \| x_2 \| x_1 \| x_0) = x_3 \| x_2 \| x_1 \| x_0 \| 0 \| 0 \| 0 \| 0$,

$x_i \in V_1$, $i=0, 1, \ldots, 3$;

преобразования $MULT_{148,0}$, $MULT_{148,1}$, $MULT_{195,0}$, $MULT_{195,1}$ вида $V_4 \to V_8$ вычисляют путем загрузки данных из вспомогательных таблиц, содержащих векторы значений этих преобразований:

$MULT_{148,0}=(0, 148, 235, 127, 21, 129, 254, 106, 42, 190, 193, 85, 63, 171, 212, 64)$,

$MULT_{148,1}=(0, 84, 168, 252, 147, 199, 59, 111, 229, 177, 77, 25, 118, 34, 222, 138)$,

$MULT_{195,0}=(0, 195, 69, 134, 138, 73, 207, 12, 215, 20, 146, 81, 93, 158, 24, 219)$,

$MULT_{195,1}=(0, 109, 218, 183, 119, 26, 173, 192, 238, 131, 52, 89, 153, 244, 67, 46)$,

где векторы значений приведены в виде

$f=(f(0), f(1), \ldots, f(15))$,

где f - произвольное преобразование, множеством входных аргументов которого является V_4;

вычисляют

$u=u+1$;

если $u<n$, то переходят к этапу (A);

получают зашифрованные сообщения c_i, $i=1, 2, \ldots, s$.