УДК 621.3.06

І.Д.ГОРБЕНКО, д-р. техн. наук, А.В.ПОТІЙ, канд. техн. наук, Ю.А.ІЗБЕНКО, канд. техн. наук

ТЕНДЕНЦИИ РАЗВИТИЯ СОВРЕМЕННЫХ СХЕМ ПОТОЧНОГО ШИФРОВАНИЯ

В настоящее время при разработке криптографических примитивов научные сообщества ведущих мировых стран придерживаются подхода, подразумевающего открытое исследование и обсуждение рассматриваемых на конкурсной основе схем (алгоритмов) криптопреобразований. Начало открытым исследованиям было положено в 1997 году в США в рамках проекта AES (Advanced Encryption Standard), организованного Национальным институтом стандартизации и технологий США для определения схемы-кандидата на стандарт блочного шифрования. После двухлетних исследований и обсуждения в рамках данного проекта в качестве лучшего был признан алгоритм Rijndael, принятый в настоящее время в качестве федерального стандарта США (FIPS 197) [1]. Необходимость принятия нового федерального стандарта была обусловлена низкой стойкостью используемых алгоритмов: как действующий стандарт (DES), так и его различные модификации (GDES, RDES) не обеспечивали должного уровня стойкости. Необходимость проведения открытых исследований продиктована желанием избежать ситуации, когда принимаемые и используемые методы криптопреобразований оказываются неспособными обеспечить требуемую стойкость. В качестве примера можно сослаться на криптоалгоритм A5, применяемый для шифрования телефонных разговоров в европейской системе мобильной цифровой связи GSM. Он по сути обладает нулевой стойкостью. Вскоре после оглашения деталей алгоритма в Internet было опубликовано сразу несколько методов вскрытия данного алгоритма [2, 3].

Проект AES как концептуальный подход к разработке качественных криптографических примитивов показал весьма высокую эффективность в широком смысле. За данным проектом последовали европейский проект NESSIE (New European Schemes for Signature, Integrity and Encryption) [12] и японский CRYPTREC [13], также основанные на идеологии открытых исследований, и ставящие своей целью проведение конкурса на разработку криптографических стандартов. Программы NESSIE и CRYPTREC значительно шире и предусматривают создание всех криптографических примитивов, необходимых для практических приложений, в том числе алгоритмов (примитивов) поточного шифрования.

Данная статья посвящена рассмотрению тенденций развития современных схем поточного шифрования. В качестве объекта исследований рассматриваются претенденты на европейский стандарт поточного шифрования. В последние несколько лет в открытой печати много внимания стали уделять вопросам практической реализации схем поточного шифрования. Актуальность применения методов поточного шифрования в коммерческих приложениях и открытых системах обусловлена, в первую очередь, существенным ростом объемов трафика передачи данных в сетях связи. Лишь поточные шифры обеспечивают приемлемую скорость шифрования на канальном, сетевом и транспортных уровнях. Конкурс NESSIE является первым проектом, в рамках которого широко обсуждаются именно схемы поточного шифрования. Следует отметить, что криптографические примитивы проекта NESSIE уже рассматриваются ISO относительно возможности включения схем поточного шифрования в новый международный стандарт ISO/IEC 18033.

Результаты анализа развития современных схем поточного шифрования будут полезны разработчикам схем поточного шифрования в частности, и криптографических примитивов в целом. На основе анализа материалов конкурса NESSIE, а также собственных результатов, формулируются рекомендации и предложения, а также высказываются взгляды на современные тенденции по созданию и применению схем поточного шифрования.

Анализ схем поточного шифрования, рассмотренных в проекте NESSIE

На рассмотрение конкурса NESSIE первоначально было представлено 6 схем поточного шифрования (табл.1). Схемы SNOW, SOBER-t16, SOBER-t32, LILI-128 являются схемами с классическим сочетанием линейного рекуррентного регистра (ЛРР) и нелинейной функции (НЛФ). BMGL и LEVIATHAN имеют нетрадиционные для схем поточного шифрования конструкции: первая схема по сути аппроксимирует схему блочного шифрования, вторая основана на использовании множества бинарных деревьев. При анализе стойкости данных схем использовались отличные от первых четырех схем как терминология, так и принципы доказательства стойкости. В проект NESSIE для рассмотрения принимались схемы с двумя уровнями стойкости: *высокий*, при котором длина ключа и внутренняя память составляют по меньшей мере 256 бит, и *нормальный*, при котором длина ключа и внутренняя память составляют по меньшей мере 128 бит. Как видно из табл.1, LILI-128 и SOBER-t16 заявлены на нормальный уровень стойкости, SOBER-t32 – на высокий, схемы SNOW, BMGL и LEVIATHAN заявлены на оба уровня стойкости. После первого отборочного тура из списка претендентов были исключены схемы LILI-128 и LEVIATHAN как не отвечающие заявленному уровню стойкости. Аналогично во втором отборочном туре были отбракованы SOBER-t16, SOBER-t32 и SNOW с 256 – битным ключом. Алгоритм BMGL в силу низких скоростных характеристик предложено использовать только как датчик псевдослучайной последовательности. В третьем отборочном туре с рассмотрения был снят как не отвечающий требованиям последний из оставшихся претендентов - SNOW со 128 – битным ключом. Однако острая практическая потребность в схемах поточного шифрования обусловила необходимость рассмотрения новых схем. В связи с этим на рассмотрение конкурса дополнительно были представлены схемы SOBER-128 и TURING. В настоящее время данные схемы являются объектом исследования на предмет возможности принятия их в качестве европейского стандарта поточного шифрования. Некоторые данные о них приведены в табл.1.

Таблица 1

Название схемы	Авторы	Тип схемы	Длины ключей, бит
Первоначально представленные схемы			
SNOW	Patrick Ekdahl and Thomas Johansson (Lund University, **Sweden**)	Синхронная, основанная на идее классического суммирующего генератора. Принадлежит к классу схем с равномерным движением регистра	128, 256
SOBER-t16, SOBER-t32	Philip Hawkes and Gregory Rose (Qualcomm International, **Australia**)	Синхронная, основанная на идее классического суммирующего генератора. Принадлежит к классу схем с равномерным движением регистра и неравномерным усечением	128 (SOBER-t16) 256 (SOBER-t32)
LILI-128	E. Dawson, W. Millan, L.Penna, L. Simpson (Queensland University of Technology, **Australia**) and J. Golic (University of Belgrade, **Yugoslavia**)	Синхронная, основанная на идее классического суммирующего генератора. Принадлежит к классу схем с неравномерным движением регистра	128
BMGL	Johan Håstad (Royal Institute of Technology, Stockholm, **Sweden**) and Mats Näslund (Ericsson Research, **Sweden**)	Генератор псевдослучайных чисел, реализованный на основе блочного шифра Rijndael	128, 256
LEVIATHAN	David McGrew (Cisco Systems, San Jose, **USA)**	Синхронная, основанная на идее использования множества структур двоичных деревьев	128, 256

Продолжение табл.1

Дополнительно представленные схемы			
SOBER-128	Philip Hawkes and Gregory Rose (Qualcomm International, **Australia**)	Синхронная, основанная на идее классического суммирующего генератора. Принадлежит к классу схем с равномерным движением регистра	128
TURING	Philip Hawkes and Gregory Rose (Qualcomm International, **Australia**)	Синхронная, основанная на идее классического суммирующего генератора. Принадлежит к классу схем с равномерным движением регистра	128 - 256

Рассмотрим схему **SNOW** [5]. Структурная схема шифра приведена на рис. 1. Она состоит из линейного рекуррентного регистра длиной 16 над $GF(2^{32})$. Первые 32 бита ЛРР задают состояния конечного автомата (КА). В свою очередь КА состоит (рис. 2) из блока подстановки S и двух 32-разрядных регистров $R1$, $R2$.

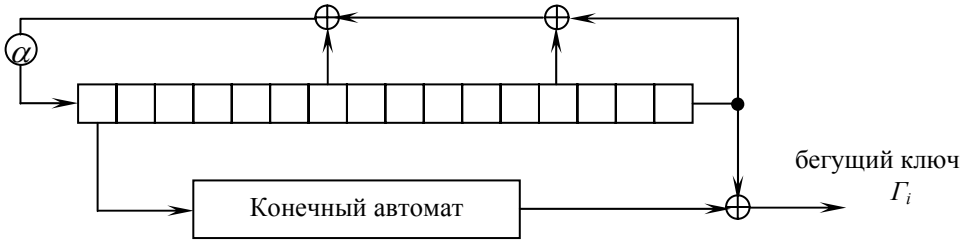

Рис. 1 - Схема SNOW

На выходе SNOW формируется гамма шифрующая Γ_i, которая в явном виде используется для поточного шифрования. Процесс генерации Γ_i осуществляется следующим образом. Сначала осуществляется установка (инициализация) ключа. В процессе установки ключа производится начальное заполнение сдвигового регистра и регистров $R1$, $R2$ конечного автомата. Затем вычисляются первые 32 бита гаммы шифрующей Γ_i путем побитового сложения выхода КА и содержимого последней ячейки сдвигового регистра. Далее осуществляется сдвиг содержимого регистра, и вычисляются следующие 32 бита Γ_i побитовым сложением выхода КА и содержимого последней ячейки сдвигового регистра, и т.д.

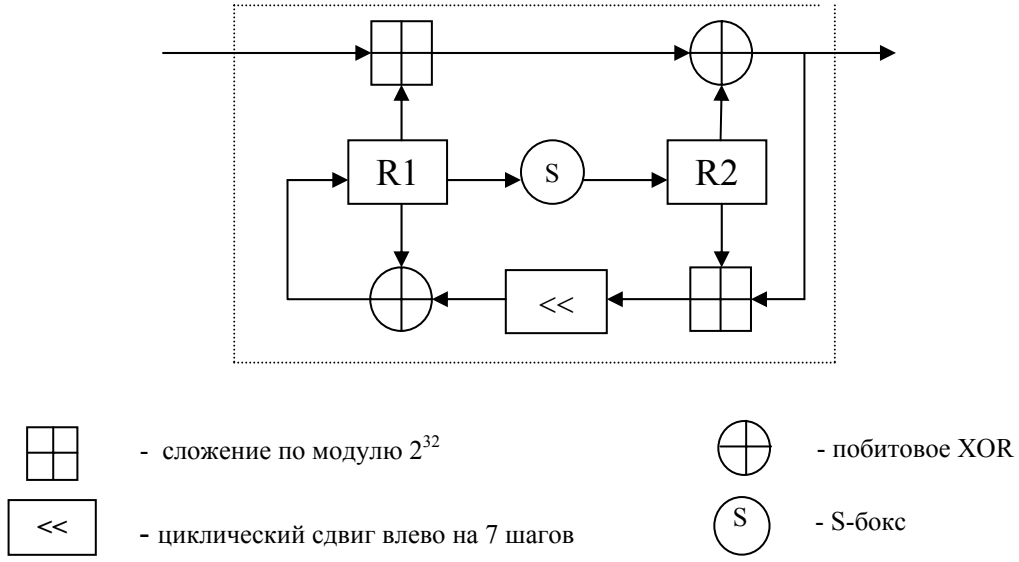

Рис. 2 - Конечный автомат

Как видно из рис. 1, ЛРР строится с использованием примитивного полинома $p(x) = x^{16}+x^{13}+x^7+\alpha$, который задает обратную связь над полем $GF(2^{32})$.

Элементы поля $GF(2^{32})$ формируются с использованием неприводимого полинома вида $\pi(x) = x^{32} + x^{29} + x^{20} + x^{15} + x^{10} + x + 1$ над полем $GF(2)$, причем $\pi(\alpha) = 0$. Ячейки сдвигового регистра обозначаются как $s(1), s(2), ..., s(16) \in GF(2^{32})$. Элементы представлены в поле $GF(2^{32})$ с использованием базиса $\{\alpha^{31}, \ldots, \alpha^2, \alpha, 1\}$. Если $y \in GF(2^{32})$, то y может быть представлен вектором $(y_{31}, y_{30}, \ldots, y_1, y_0)$, где $y = y_{31}\alpha^{31} + y_{30}\alpha^{30} + \ldots + y_1\alpha + y_0$.

После сдвига регистра, значение ячейки $s(1)$ поступает на вход КА. На i-ом шаге выход КА[i] вычисляется по правилу:

$$KA[i] = (s(1) \boxplus R1[i]) \oplus R2[i].$$

Для формирования бегущего ключа Γ_i выход КА складывается по модулю 2 с содержимым $s(16)$, т.е.

$$\Gamma_i = KA[i] \oplus s(16).$$

Зашифрование информации M_i осуществляется по правилу $C_i = M_i \oplus \Gamma_i$.

В КА на каждом i-ом цикле выработки бегущего ключа содержимое регистров $R1$ и $R2$ обновляется следующим образом:

$$R1[i] = ((KA[i\text{-}1] \boxplus R2[i\text{-}1]) <<) \oplus R1[i\text{-}1];$$
$$R2[i] = S(R1[i\text{-}1]). \qquad (1)$$

Здесь операция $\boxplus$ обозначает целочисленное сложение по модулю 2^{32}, операция $x<<$ - циклический сдвиг x на 7 тактов влево, и операция $\oplus$ - побитовое сложение по модулю 2 (XOR).

Подстановка, обозначаемая как $S(x)$, состоит из 4 идентичных биективных $8\to8$ S-блоков. Операция подстановки обеспечивает, как видно из (1), перестановку битов регистра $R1$. Блок подстановки работает следующим образом. Входная величина x разбивается на 4 байта, от старшего до младшего байтов. Каждый из входных байтов, обозначаемый как $\omega = (\omega_7,\omega_6,\ldots,\omega_0)$, преобразуется соответствующим нелинейным отображением 8 бит в 8 бит с операциями в поле $GF(2^8)$ по модулю полинома $\pi(x) = x^8 + x^5 + x^3 + x + 1$. В результате формируется выходной вектор $r = (r_7, r_6,\ldots, r_0)$. Нелинейное отображение выполняется в поле $GF(2^8)$ по правилу:

$$r = \omega^7 + \beta^2 + \beta + 1 \pmod{\pi(x)},$$

где β есть решение сравнения $\pi(\beta) = 0$.

После нелинейного отображения, примененного к каждому байту, биты в результирующем 32-битном слове переставляются, и байты конкатенируются. Правило перестановки имеет вид:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
3	10	20	24	0	14	17	29	7	13	18	25	5	12	23	27

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	8	21	26	4	9	19	31	2	11	16	28	6	15	22	30

Используя векторную систему обозначений, это можно записать как:

$$y = (y_{31}, y_{30}, y_{29}, \dots, y_1, y_0) \rightarrow (y_8, y_0, y_{24}, \dots, y_{15}, y_{27}).$$

Для SNOW определены два режима работы: стандартный и *IV*-режим.

В стандартном режиме SNOW работает как быстрый криптографический генератор псевдослучайных чисел, т.е. для каждого начального заполнения (секретного ключа k) SNOW вырабатывает последовательность псевдослучайных чисел.

В *IV*-режиме генератор инициализируется посредством двух переменных: секретного ключа k и известной переменной инициализации (*IV*). Это означает, что для заданного секретного ключа k генератор производит множество последовательностей псевдослучайных чисел для каждого значения *IV*. Так как генерируемые последовательности являются неразличимыми от истинно случайных последовательностей, то можно сказать, что *IV*-режим работает как *псевдослучайная функция увеличения длины* (от множества значений *IV* к множеству возможных последовательностей). Длина выходных последовательностей обычно больше, чем длина *IV*. В SNOW *IV* – 64 битное значение, которое представляется как два 32 битных слова (IV_2, IV_1). Таким образом, значение *IV* распределяется в диапазоне от 0 до 2^{64}-1, где IV_2 – старшее слово, а IV_1 – младшее слово.

IV-режим предлагается использовать в приложениях, требующих частой переустановки начальных значений, где ключ k – фиксированное значение, а значение *IV* изменяется. При этом шифр не теряет в производительности, так как инициализация ключа в *IV*-режиме использует меньше тактов SNOW, чем в стандартном режиме (32 вместо 64).

Установка ключа осуществляется следующим образом. Секретный исходный ключ k представляется как $k = (k(1), k(2), k(3), k(4))$ для 128-битного ключа и $k = (k(1), k(2), k(3), k(4), k(5), k(6), k(7), k(8))$ для 256-битного ключа. При 128-битном ключе он вводится в сдвиговый регистр следующим образом:

$s(1) = k(1) \oplus IV_1$, $s(2) = k(2)$, $s(3) = k(3)$, $s(4) = k(4) \oplus IV_2$,
$s(5) = k(1) \oplus 1$, $s(6) = k(2) \oplus 1$, $s(7) = k(3) \oplus 1$, $s(8) = k(4) \oplus 1$,

для второй половины

$s(9) = k(1)$, $s(10) = k(2)$, $s(11) = k(3)$, $s(12) = k(4)$,
$s(13) = k(1) \oplus 1$, $s(14) = k(2) \oplus 1$, $s(15) = k(3) \oplus 1$, $s(16) = k(4) \oplus 1$,

где «1» обозначает 32-битный единичный вектор.

При 256-битовом ключе сдвиговый регистр инициализируется по правилу:

$s(1) = k(1) \oplus IV_1$, $s(2) = k(2)$, $s(3) = k(3)$, $s(4) = k(4) \oplus IV_2$,
$s(5) = k(5)$, $s(6) = k(6)$, $s(7) = k(7)$, $s(8) = k(8)$,
$s(9) = k(1) \oplus 1$, …, $s(16) = k(8) \oplus 1$.

В стандартном режиме предполагается, что $IV_1 = IV_2 = 0$. После инициализации ЛРР регистры $R1$ и $R2$ устанавливаются в ноль.

Схема запускается v раз, без генерации выходной последовательности. В стандартном режиме $v = 64$, в *IV*-режиме $v = 32$. Вместо генерации Γ_i выход КА складывается по модулю два с содержимым ячеек $s(7)$, $s(13)$, $s(16)$. В результате на каждом цикле работы КА формируется новое значение первой ячейки регистра $s(1)$

$$s(1)[i+1] = \alpha \cdot (s(7) \oplus s(13) \oplus s(16) \oplus KA[i]).$$

После v циклов сдвиговый регистр, а также регистры $R1$, $R2$ принимают значения, вычисленные на последнем цикле. Первые 32 бита бегущего ключа являются результатом побитового сложения $s(16) \oplus KA[i]$. Максимально допустимая длина бегущего ключа, сформированного с использованием одного ключа, составляет 2^{50} 32-битных слов, т.е., $2^{50} \cdot 2^5 = 2^{55} \approx 3.6 \cdot 10^{16}$ битов. Эту величину можно считать близкой к расстоянию единственности SNOW. После этого схема должна быть перезапущена.

Вывод. Схема SNOW является современным машинно-ориентированным поточным шифром, обладающим высоким быстродействием. Конструкция шифра позволяет осуществить эффективную аппаратную реализацию. Заявленные уровни стойкости – нормальный и высокий. Конструкция шифра – комбинация ЛРР и конечного автомата – позволяет, с точки зрения разработчиков, противостоять известным криптоаналитическим атакам. Нововведением, отличающим данный шифр от предшественников, является введение *IV*-режима, позволяющего осуществлять частую реинициализацию шифра.

SOBER-t16 и SOBER-t32 [6, 7]. Данное семейство шифров работает с длиной слова w бит: для SOBER-t16 w=16, SOBER-t32 – w=32. Соответственно все операции выполняются над полем $GF(2^w)$: $GF(2^{16})$ для SOBER-t16 и $GF(2^{32})$ для SOBER-t32.

Обобщенная схема SOBER представлена на рис.3. Она состоит из ЛРР, нелинейной функции (нелинейного фильтра) и блока неравномерного усечения, называемого “задержкой” или “усечением”. В нелинейном фильтре используются операции сложения по модулю 2^w и нелинейные преобразования, обозначаемые как функция f_w. Константа *Konst* инициализируется при ключевой загрузке. Блок неравномерного усечения фильтрует выходной поток нелинейной функции по закону, вид которого зависит от значений входных данных. Структура функции f_w представлена на рис. 4.

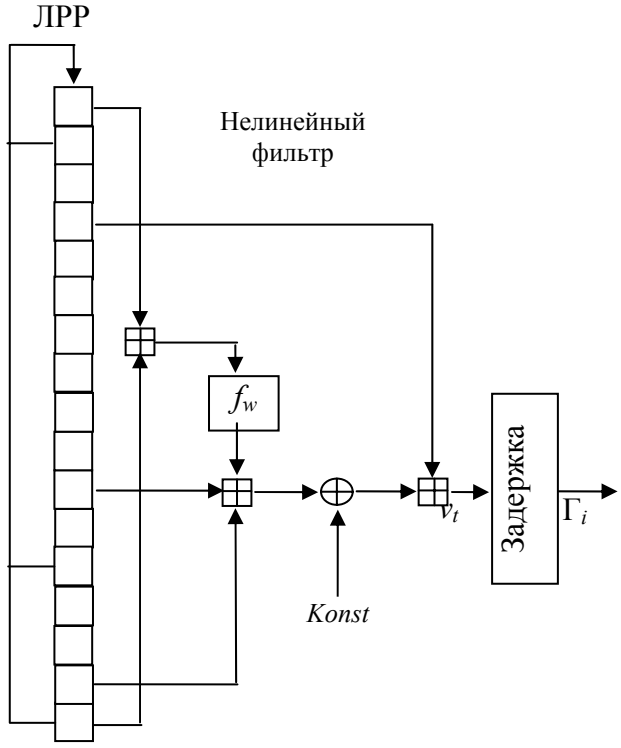

Рис. 3 - Схема Sober-t16 (Sober-t32)

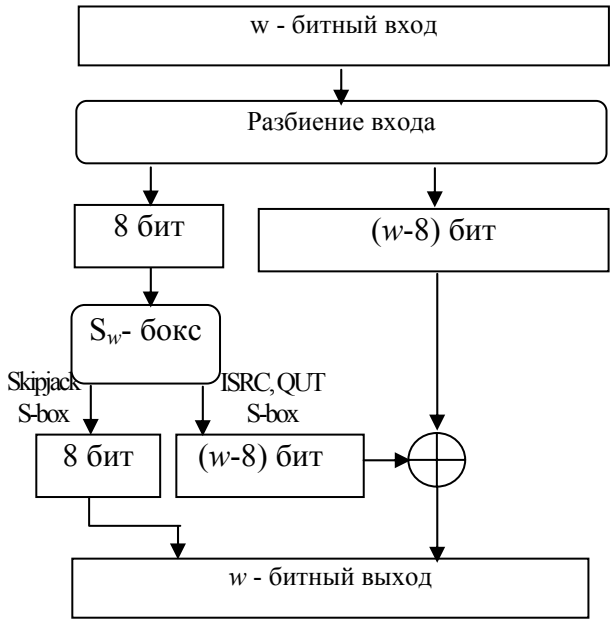

Рис. 4 - Структура функции f_w

Процесс генерации бегущего ключа Γ_i происходит следующим образом. Первоначально производится начальное заполнение рекуррентного регистра ключевыми данными. Затем

вычисляются первые 16 бит (32 бита) бегущего ключа путем последовательного применения к определенным ячейкам регистра нелинейной фильтрации и задержки. После этого осуществляется сдвиг содержимого регистра и аналогичным образом вычисляются следующие 16 бит (32 бита) Γ_i и т.д.

ЛРР строится с использованием примитивного полинома обратной связи над полем $GF(2^w)$:

$$s_{t+17} = \alpha \cdot s_{t+15} \oplus s_{t+4} \oplus \beta \cdot s_t \text{ для Sober-t16,}$$
$$s_{t+17} = s_{t+15} \oplus s_{t+4} \oplus \lambda \cdot s_t \quad \text{для Sober-t32,}$$

где $\alpha = 0$xE382, $\beta = 0$x67C3, $\lambda = 0$xC2DB2AA3, а s_t являются w битовыми ячейками сдвигового регистра. Все арифметические операции в поле $GF(2^w)$ выполняются по модулю соответствующих полиномов вида:

$$\pi(x) = x^{16} + x^{14} + x^{12} + x^7 + x^6 + x^4 + x^2 + x + 1, \quad \text{для Sober-t16, и}$$
$$\pi(x) = x^{32} + (x^{24} + x^{16} + x^8 + 1)(x^6 + x^5 + x^2 + 1), \quad \text{для Sober-t32,}$$

над $GF(2)$, и $\pi(\alpha) = 0$. Элементы представлены в поле $GF(2^w)$ с использованием базиса $\{\alpha^{w-1}, \ldots, \alpha^2, \alpha, 1\}$. Если $y \in GF(2^{32})$, то y представим как $(y_{31}, y_{30}, \ldots, y_1, y_0)$, где $y = y_{31}\alpha^{31} + y_{30}\alpha^{30} + \ldots + y_1\alpha + y_0$.

Сложение двух элементов в $GF(2^w)$ представляется как сложение соответствующих полиномов с приведением коэффициентов по модулю два. Умножение двух элементов в $GF(2^w)$ представляется как умножение соответствующих полиномов с приведением коэффициентов полиномов по модулю два. Результат умножения затем приводится по модулю неприводимого полинома $\pi(x)$. Операции вычисления s_{t+17} из s_{t+15}, s_{t+4} и s_t называются сдвигом ЛРР. Начальное состояние $\sigma_0 = (s_0, \ldots, s_{16})$ определяет генерируемый поток. Текущее состояние регистра после t сдвигов сохраняется в регистре $R = (r_0, \ldots, r_{16})$ как

$$R = (r_0, \ldots, r_{16}) = (s_t, \ldots, s_{t+16}) = \sigma_t.$$

При указанном сдвиге ЛРР состояние регистра изменяется с $R = \sigma_t$ на $R = \sigma_{t+1}$. Изменение происходит путем выполнения следующих действий:

- из s_{t+15}, s_{t+4} и s_t вычисляется s_{t+17};
- циклически сдвигаются значения $r_1, \ldots, r_{16}$ в позиции $r_0, \ldots, r_{15}$;
- устанавливается $r_{16} = s_{t+17}$.

После сдвига регистра значения s_t, s_{t+1}, s_{t+6}, s_{t+13}, s_{t+16} и *Konst* поступают на вход нелинейного фильтра. Функция нелинейной фильтрации включает в себя операции сложения по модулю 2^{32}, w-битный XOR и подстановку $8 \times w$. Выход нелинейного фильтра вычисляется следующим образом:

$$v_t = ((f(s_t \boxplus s_{t+16}) \boxplus s_{t+1} \boxplus s_{t+6}) \oplus Konst) \boxplus s_{t+13},$$

где $f(a)$ = S-бокс$(aH) \oplus (0 \| aL)$, "$\boxplus$" обозначает сложение по модулю 2^w, "$\oplus$" обозначает XOR, aH – восемь старших бит переменной a, aL — (w-8) младших бит переменной a, $\|$ - знак конкатенации. После t сдвигов регистра соответствующий выход НЛФ определяется из регистра R по правилу

$$v_t = ((f(r_0 \boxplus r_{16}) \boxplus r_1 \boxplus r_6) \oplus Konst) \boxplus r_{13}.$$

Затем выход нелинейной функции v_t поступает на вход блока неравномерного усечения (децимации), который прореживает выход НЛФ. Первое w-разрядное слово на выходе НЛФ является первой командой управления усечением (SCW). SCW разбита на $w/2$ блоков по два бита (дибит). Начиная с самого младшего дибита, блок неравномерного усечения считывает значение дибита и осуществляет одно из четырех действий (табл. 2) согласно значению дибита. Когда все дибиты считаны, ЛРР сдвигается, и выход НЛФ становится следующей SCW.

Таблица 2

Дибит	Действие блока неравномерного усечения
00	Сдвинуть ЛРР, на выход ничего не подается
01	Сдвинуть ЛРР, выход НЛФ побитно складывается (XOR) с 0x6996 (0x6996C53A), сдвинуть ЛРР снова (без генерации выхода)
10	Сдвинуть ЛРР (без генерации выхода), сдвинуть ЛРР снова и принять значение выхода НЛФ как выход схемы
11	Сдвинуть ЛРР, выход НЛФ побитно складывается (XOR) с 0x9669 (0x96693AC5)

Выход преобразователя “задержка” и является бегущим ключом схемы.

Процесс ключевой инициализации осуществляется следующим образом. Ключевая инициализация/реинициализация состоит из двух операций:

- Include(X); добавляет слово X к r_{15} по модулю 2^w.
- Diffuse(); сдвигает регистр и заменяет величину r_4 на результат побитового сложения (XOR) выхода НЛФ с r_4: ($r_4 \oplus v$).

Основная функция загрузки ключа выполняется операцией Loadkey(k[],keylen), где k[] – массив, содержащий keylen байт ключа, каждый элемент массива содержит один байт ключа. Операция Loadkey() использует значения k[] для преобразования текущего состояния регистра.

Алгоритм Loadkey(k[],keylen) состоит из следующих этапов:

1. Для Sober-t16: разбиение k[] на $kwl = \lceil \text{keylen}/2 \rceil$ слов и запись в массив kw[] kwl слов:
   keylen четное {kw[0], kw[1],…}={(k[0], k[1]), (k[2], k[3]),…}
   keylen нечетное {kw[0], kw[1],…}={(0, k[0]), (k[1], k[2]),…}
   Для Sober-t32: разбиение k[] на $kwl = \lceil \text{keylen}/4 \rceil$ слов и запись в массив kw[] kwl слов:
   keylen ≡ 0(mod 4):{kw[0], kw[1], …}={(k[0], k[1], k[2], k[3]), (k[4], k[5], k[6], k[7]), …}
   keylen ≡ 1(mod 4):{ kw[0], kw[1], …}={(0, k[0], k[1], k[2]), (k[3], k[4], k[5], k[6]), …}
   keylen ≡ 2(mod 4):{ kw[0], kw[1], …}={(0, 0, k[0], k[1]), (k[2], k[3], k[4], k[5]), …}
   keylen ≡ 3(mod 4):{ kw[0], kw[1],…}={(0, 0, 0, k[0]), (k[1], k[2], k[3], k[4]), …}
2. Для каждого i, $0 \le i \le kwl-1$, выполнить Include(kw[i]) и применить Diffuse().
3. Include(keylen).
4. Применить Diffuse() 17 раз.

17-кратное применение Diffuse() гарантирует, что каждый бит начального заполнения регистра будет являться результатом применения нелинейной функции к ключу.

Полный алгоритм загрузки секретным t-байтовым ключем K[0],..,K[t-1] имеет вид:

1. Семнадцать ячеек регистра инициализируются первыми 17 числами Фибоначчи через установку r_0=1 и r_1=1 и вычисление $r_i = r_{i-1} + r_{i-2}$, для $2 \le i \le 16$. Значение *Konst* устанавливается в 0.
2. Применяется Loadkey(k[],t).
3. Сдвигается регистр, значению *Konst* присваивается значение выхода НЛФ.

Период Sober-t16 и Sober-t32 составляет приблизительно $2^{17\cdot16+16}$ и $2^{17\cdot32+32}$ бит соответственно. После генерации указанного числа бит бегущего ключа необходимо ввести новые ключи.

Вывод. Схема SOBER является современным машинно-ориентированным поточным шифром, обладающим высоким быстродействием. Конструкция шифра позволяет осуществить эффективную аппаратную реализацию. Заявленные уровни стойкости – нормальный и высокий. Конструкция шифра – комбинация ЛРР, нелинейного фильтра и блока неравномерного усечения – позволяет, с точки зрения разработчиков, противостоять известным криптоаналитическим атакам. Отличительной особенностью шифра является оригинальная конструкция нелинейного фильтра и использование блока неравномерного усечения. Последний, с точки зрения разработчиков, повышает стойкость схемы к некоторым видам криптоатак. Однако его использование ведет, во-первых, к понижению быстродействия схемы, что само по себе подрывает идею применения его как высокоскоростного шифра, и, во-вторых, уменьшает длину периода – одного из основных показателей стойкости поточных шифров.

LILI-128 [8]**.** Данный шифр не прошел отборочный тур. Однако здесь он рассматривается как пример классической схемы поточного шифрования на основе управляющего регистра.

Компоненты схемы сгруппированы в две подсистемы: подсистему управления движением регистра и подсистему генерации данных. Схема LILI-128 представлена на рис. 5.

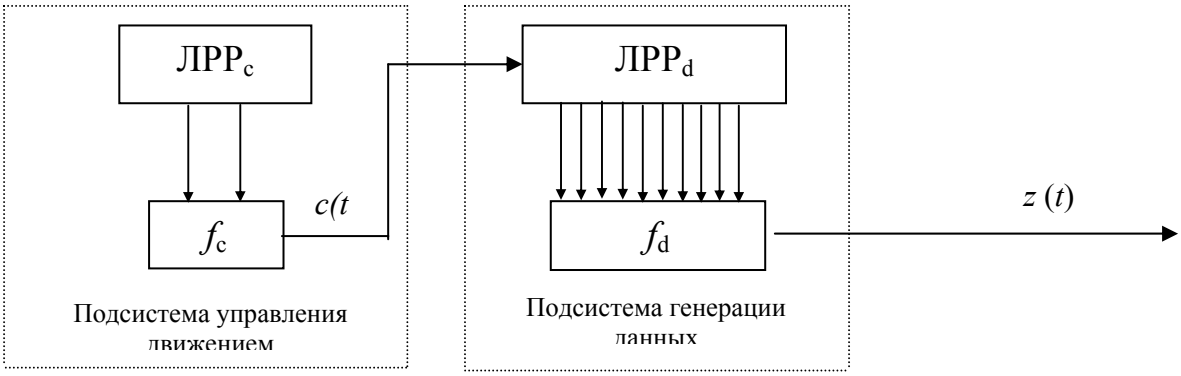

Рис. 5 – Схема LILI-128

Подсистема управления движением использует псевдослучайную двоичную последовательность, сгенерированную двоичным регистром управления ЛРР$_c$ длиной L_c=39 бит с равномерным движением, и функцию f_c, оперирующую содержимым $k = 2$ ячеек ЛРР$_c$ для генерации псевдослучайной целочисленной последовательности, $c = \{c(t)\}_{t=1}^{\infty}$. Полином обратной связи ЛРР$_c$ имеет вид:

$$h(x) = x^{39} + x^{35} + x^{33} + x^{31} + x^{17} + x^{15} + x^{14} + x^2 + 1.$$

Начальное нулевое заполнение ЛРР$_c$ исключается принудительным присваиванием любой из ячеек “1”. В момент времени t содержимое 12 и 20 ячеек ЛРР$_c$ является входными данными функции f_c, выходом которой является целочисленное $c(t)$, такое что $c(t) \in \{1, 2, 3, 4\}$. Функция f_c определена как

$$f_c\,(x_{12}, x_{20}\,) = 2(x_{12}) + x_{20} + 1.$$

Подсистема генерации данных использует целочисленную последовательность $c(t)$ для управления двоичным ЛРР$_d$ длиной $L_d = 89$ бит: каждый раз регистр сдвигается c раз. Полином обратной связи ЛРР$_d$ имеет вид:

$$h(x) = x^{89} + x^{83} + x^{80} + x^{55} + x^{53} + x^{42} + x^{39} + x + 1.$$

Начальное нулевое заполнение ЛРР$_d$ исключается принудительным присваиванием любой из ячеек “1”.

Значения десяти фиксированных ячеек (0, 1, 3, 7, 12, 20, 30, 44, 65, 80) ЛРР$_d$ является входными данными для специально выбранной булевой функции f_d. Вид булевой функции не приводится, известно лишь, что функция является сбалансированной, обладает корреляционным иммунитетом третьего порядка, алгебраическая степень функции равна 6, функция имеет нелинейную структуру и достигает нелинейности 480. Двоичный выход f_d - поток бит $z(t)$. После того, как $z(t)$ вычислен, два ЛРР сдвигаются и процесс повторяется для формирования потока $z = \{z(t)\}_{t=1}^{\infty}$.

Вывод. Данная схема представляет классический поточный шифр с управляющим регистром. Обладая общепринятой конструкцией, схема изначально имеет очевидные недостатки: во-первых, схема обладает низким быстродействием, поскольку все операции выполняются над полем $GF(2)$, и, во-вторых, обладает некоторыми уязвимыми местами: образующий полином ЛРР является прореженным, показатели стойкости нелинейной функции выбраны не наилучшим образом.

BMGL [9]. BMGL – конструкция синхронного генератора псевдослучайных чисел. Криптографическое ядро данного шифра – блочный шифр Rijndael. В спецификации разработчики показали, что атаки на шифр сводятся к атакам на Rijndael.

Предлагаемый алгоритм поточного шифрования основан на односторонней функции перестановки f. Если обозначить через $\{0,1\}^n$ – множество двоичных строк x, длиной $|x| = n$, а $B = \{b_r\}$ – семейство двоичных функций, то есть $b_r(x) \in \{0,1\}$, тогда конструкция BMGL описывается следующим образом.

Определение. Пусть n, m, L, и λ - целые числа, такие, что $L = \lambda m$ и пусть f – односторонняя функция, такая, что для любого n функция f является взаимно-однозначным соответствием $\{0,1\}^n \to \{0,1\}^n$. Тогда генератор BMGL$_{n,m,L}(f)$ разворачивает $n + nm$ бит в L бит следующим образом. Пусть входные данные интерпретируются как x_0 и $R \in \mu_m$, где μ_m – множество случайных двоичных матриц R размером $m \times n$. Допустим, что $x_i = f(x_{i-1})$, $i = 1,2,\ldots$, λ, тогда выход будет $\left\{B_R^m(x_i)\right\}_{i=1}^{\lambda}$, где $B_R^m(x)$ - функция скалярного произведения двоичного вектора x на случайную бинарную матрицу R.

Таким образом, здесь L – длина выходной последовательности, n – длина ключа / размер блока, а m – количество бит выходной последовательности, получаемых за одну итерацию.

Используя функцию блочного шифра $f = f_p(k)$, где p — открытый текст, конструкция BMGL$_{n,m,L}(f)$ может рассматриваться как новый режим работы Rijndael, в некотором смысле, более усовершенствованная версия режима обратной связи по выходу. Режим, который может быть назван режимом обратной связи по ключу, показан ниже на рис. 6.

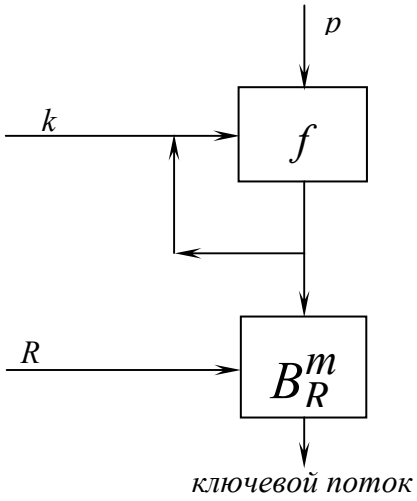

Рис. 6 – Режим обратной связи по ключу

Конструкция алгоритма требует, чтобы размер блока, n, шифра соответствовал длине ключа ($n = |k|$) и был равен 256 бит. При таком размере ключа m рекомендуется брать из интервала 30 – 50.

Для инициализации шифра необходимо $n + nm$ бит начального значения: n – для исходного значения x_0 (секретного ключа) и nm – для задания матрицы R. Предлагаются программные методы незначительного уменьшения начального значения, а также уменьшение на n бит посредством задания матрицы специальным способом.

Предполагается, что рассматриваемый поточный шифр медленнее, чем такие алгоритмы как RC4 или SEAL. Тем не менее, практика показала, что такая конструкция вполне допустима в тех случаях, когда надёжность требуемого поточного шифра (или псевдослучайного генератора) стоит на первом месте. Другим преимуществом BMGL как поточного шифра является то, что он может быть основан на любой односторонней функции в случае, если в алгоритме Rijndael будут найдены недостатки, а также на другом блочном шифре или даже криптографической хеш-функции.

Предложен обобщённый интерфейс шифра, разделяющий ключевой поток на сегменты, каждый из которых может быть сгенерирован в любом порядке.

В результате оптимизации алгоритм BMGL позволяет производить частую переустановку ключей.

Выводы. Данная схема, основанная на базе блочного шифра Rijndael, представляет собой интерес скорее как генератор псевдослучайных чисел, чем поточный шифр. Низкое быстродействие схемы не позволяет использовать ее в приложениях, требующих высокоскоростного обмена данными.

Дополнительно представленные схемы имеют следующее описание.

SOBER-128 [10]. Конструкция, в принципе, является схожей с конструкцией SOBER-t32 за исключением устройства нелинейной функции NLF, и обеспечивает *нормальный* уровень стойкости, что обусловлено использованием 128 битного ключа. Обобщенная схема 32-разрядного шифра SOBER-128 представлена на рис.7.

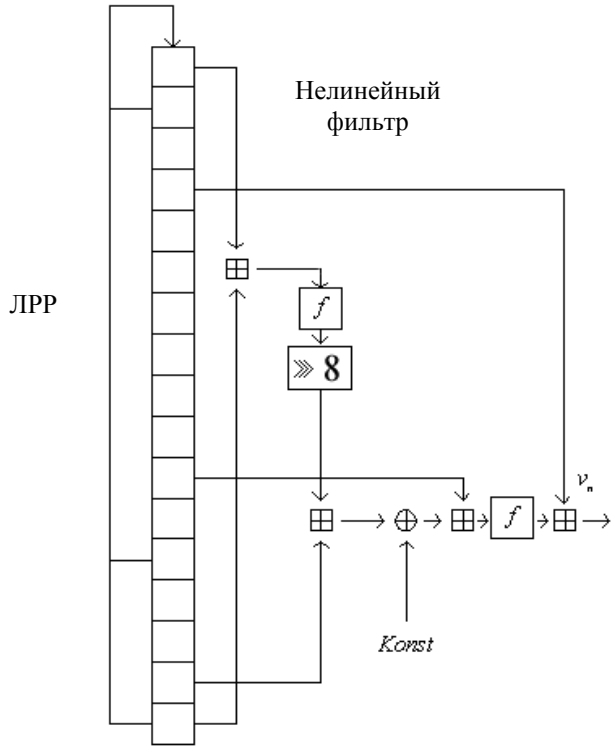

Рис. 7 - Схема SOBER-128

Структура функции f_w схемы аналогична структуре SOBER - t32. Изменения в конструкции SOBER-128 можно представить тремя основными категориями:

1. Изменения в функционировании схемы, повышающие ее эффективность и не ухудшающие при этом ее криптостойкость:
   - S-бокс реконструирован таким образом, что те же самые преобразования производятся с меньшей вычислительной сложностью;
   - изменена структура ЛРР для повышения эффективности обновления содержимого его ячеек и понижения размера таблицы умножения; новые преобразования при этом изоморфны старым;
   - в местах, где это возможно, данные генерируются 17-словными блоками, что позволяет производить значительную оптимизацию распараллеливания процессов;
   - схема поддерживает только те ключи и вектора инициализации, размер которых кратен 4, поскольку поддержка ключей и векторов инициализации нечетной длины значительно усложняет исходный код;
   - схема имеет байт-ориентированную структуру вследствие широкого использования пользователями вычислительных платформ на базе Intel-микросхем.
2. Изменения, повышающие криптостойкость схемы:
   - удален блок неравномерного усечения. Данный механизм несколько повышал стойкость схемы, однако его использование влекло за собой понижение быстродействия; кроме того, данный механизм делал схему уязвимой к side-channel атакам;
   - усилено нелинейное преобразование путем добавления фиксированного сдвига и повторного применения S-бокса;
   - в процессе ключевой инициализации "*Konst*" принимает ненулевое значение.

3.Добавлена функция формирования MAC - кодов (данная особенность в рамках этой статьи рассматриваться не будет).

В отличие от SOBER - t32, данная схема элементы поля $GF(2^{32})$ представляет в виде подполя $GF((2^8)^4)$. Это изоморфное стандартное представление, но не идентичное. Подполе $B = GF(2^8)$ байта представлено по модулю неприводимого полинома $z^8 + z^6 + z^3 + z^2 + 1$. Образующий полином и точки съема остались без изменений. Как видно из рис.7, нелинейная функция имеет следующий вид:

$$v_t = f(((f(s_t \boxplus s_{t+16}) >>> 8 \boxplus s_{t+1}) \oplus Konst) \boxplus s_{t+6}) \boxplus s_{t+13},$$

где ">>>" обозначает операцию циклического сдвига.

Процедура ключевой инициализации аналогична инициализации SOBER - *t*32, с той лишь разницей, что при использовании векторов инициализации используется *Loadkey* (*IV*[], *m*), где *m* – размерность в байтах вектора инициализации IV, *IV*[0], …, *IV*[*m*].

Вывод. SOBER – 128, улучшенный вариант SOBER - t32, является высокоскоростным поточным шифром и может быть эффективно реализован как программно, так и аппаратно. Внесенные в конструкцию схемы изменения позволили повысить стойкость схемы к криптоаналитическим атакам, уменьшить объемы используемой памяти, увеличить скорость преобразований, увеличить период генерируемой последовательности до максимально возможной, а также обеспечили возможность частой реинициализации схемы за счет введения векторов инициализации.

Turing [11]. Данный шифр работает с длиной слова 32 бита, соответственно все операции выполняются над полем $GF(2^{32})$. Шифр обеспечивает *нормальный* и *высокий* уровни стойкости, оперируя ключом длины вплоть до 256 бит и генерируя за каждый цикл функционирования гамму шифрующую длиной 160 бит.

Обобщенная схема Turing представлена на рис.8. Она состоит из ЛРР и нелинейной функции (нелинейного фильтра).

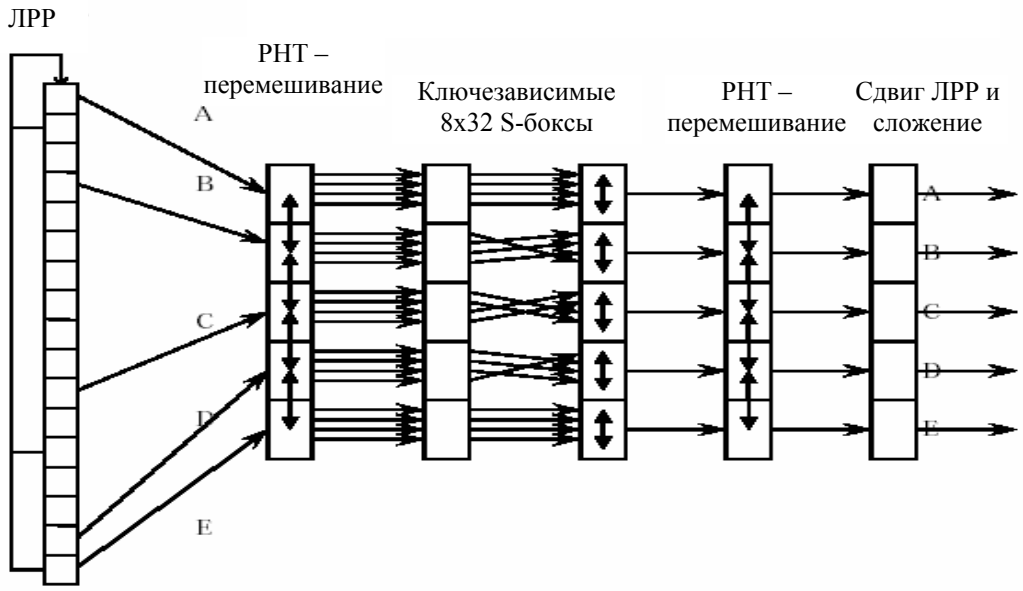

Рис. 8 - Схема Turing

Пять слов A, B, C, D, и E, взятых из соответствующих точек съема регистра, перемешиваются с использованием «псевдоадамарового» преобразования (РНТ-перемешивание), проходят через высоко нелинейный блок подстановки (S-бокс), затем снова

перемешиваются и комбинируются с четырьмя новыми словами, взятыми из ЛРР, для генерации 160 бит гаммы шифрующей.

ЛРР полностью аналогичен регистру, используемому в SOBER-128. Нелинейный фильтр состоит из:

- ЛРР сдвига и выбора 5 входных слов;
- перемешивания данных слов посредством псевдоадамарового преобразования;
- преобразования байт с ключезависимыми преобразованиями, перемешивания слов с использованием четырех 8→32 бита нелинейных S-боксов;
- повторного перемешивания полученных слов посредством псевдоадамарового преобразования;
- повторного сдвига ЛРР и сложения по mod 2^{32} с дополнительными словами.

S-боксы состоят из последовательно применяемых фиксированного 8→8 бит S-бокса и фиксированного 8→32 бит нелинейного Q-бокса, использующих в процессе функционирования данные, модифицированные переменными, полученными в процессе ключевой инициализации. Слова B, C и D сдвигаются влево на 8, 16 и 24 бита соответственно. Ниже приведен фрагмент кода на языке C, реализующего прохождение данных через S-бокс:

```
WORD S0(BYTE b)
{
    int           i;
    WORD          ws;

    ws = 0;
    for (i = 0; i < keylen; ++i) {
           b = Sbox[B(K[i], 0) ^ b];
           ws ^= ROTL(Qbox[b], i + 0); /* "+0" for MSB */
    }
    ws  = (ws & 0x00FFFFFFUL) | (b << 24);
    return ws;
}
```

где K[i] – ключевой материал, $0 \le i < N$, N – число слов в ключе.

Псевдоадамарово преобразование предназначено для перемешивания данных и реализовано следующим образом:

$$\begin{bmatrix} A \\ B \\ C \\ D \\ E \end{bmatrix} = \begin{bmatrix} 2 & 1 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 & 1 \\ 1 & 1 & 2 & 1 & 1 \\ 1 & 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \\ D \\ E \end{bmatrix}$$

Фрагмент кода на языке C, реализующего данное преобразование, будет иметь вид:

E = A + B + C + D + 2E;

A += E; B += E; C += E; D += E.

Выходными данными алгоритма является промежуточная гамма, полученная посредством конкатенации слов A, B, C, D, и E.

Процесс ключевой инициализации использует прохождение вводимого ключа через нелинейный фильтр, гарантируя, таким образом, статистическую независимость полученного и вводимого ключей. Процесс загрузки вектора инициализации сконструирован таким образом, чтобы при совместном вводе с ключом, имея различные длины, не формировать идентичные начальные состояния [11].

Вывод. TURING является высокоскоростным поточным шифром. Оригинальная конструкция шифра, включающая в себя блоки рассеивания (PHT-перемешивание) и высоко нелинейные преобразования (S-, Q - боксы), позволяет в явном виде реализовывать два

основополагающих принципа криптопреобразований – перемешивание и рассеивание. Схема является легко программно и аппаратно реализуемой. Введение векторов инициализации обеспечивает возможность частой реинициализации схемы.

Заключение. Тенденции развития современных схем поточного шифрования показывают, что на сегодняшний день имеют преимущество классические схемы построения данных шифров на основе регистров сдвига с нелинейной функцией, т.н. фильтр-генераторы, построенные над расширенными полями. Высокую скорость шифрпреобразований обеспечивают линейные регистры, построенные над полем $GF(2^q)$, где q - длина машинного слова, и нелинейные преобразования, представленные в виде блоков подстановки (S-боксов). Использование блоков подстановки, свойственных более блочным шифрам, на наш взгляд, связано с тем, что данные блоки являются "стандартизированными", хорошо изученными и зарекомендовавшими себя высоко нелинейными преобразованиями. Такие преобразования сводят вычислительные затраты к минимуму, поскольку любому входному вектору соответствует свой выходной вектор и вычислительная сложность сводится к осуществлению операции адресации в массиве и взятии операнда по адресу. Наиболее перспективные современные поточные шифры представлены в табл.3.

Таблица 3

	Уровень стойкости	Стойкость к аналитическим атакам	Основные компоненты	*IV*-режим
SNOW 2.0	нормальный (длина ключа 128 бит)	стоек	ЛРР, конечный автомат	предусмотрен
SOBER – 128	нормальный (длина ключа 128 бит)	стоек	ЛРР, нелинейный фильтр	предусмотрен
TURING	нормальный, высокий (длина ключа 128, 256 бит)	стоек	ЛРР, блоки рассеивания и замешивания	предусмотрен

Важным принципом проектирования остается обеспечение баланса между стойкостью схемы и ее технической эффективностью. Так, одним из доминирующих показателей эффективности схем ПШ становится быстродействие – при высокой криптографической стойкости современные схемы должны обеспечивать скорость шифрпреобразований порядка нескольких Гбит/с.

Актуальной задачей формирования системы безопасности информации государства, нормативно-правового обеспечения защиты информации является разработка национальных стандартов на механизмы криптографической защиты информации. Использование результатов и опыта международных конкурсов позволяет ускорить процесс разработки национальных стандартов.

Список литературы: ***1***. *FIPS PUB 197:2001. Advanced Encryption Standard (AES).* ***2***. *J.D.Golic. Cryptanalysis of Alleged A5 Stream Cipher. Advances in Cryptology: Proc. Eurocrypt'97, pp. 239-255, Springer-Verlag, 1997.* ***3***. *A.Biryikov, A.Shamir. Real Time Cryptanalysis of the Alleged A5/1 on a PC. 09.09.1999.* ***4***. *Andrew Rukhin, Juan Soto. A Statistical Test Suite for Random and Pseudorandom Number Generator for Cryptographic Application. NIST Special Publication 800-22, September 2001.* ***5***. *P. Ekdahl, T. Johansson. SNOW – a new stream cipher. http://www.cryptonessie.org.* ***6***. *P. Hawkes, G.G.Rose. Primitive Specification and Supporting Documentation for SOBER-t16 Submission to NESSIE. http://www.cryptonessie.org.* ***7***. *P. Hawkes, G.G. Rose. Primitive Specification and Supporting Documentation for SOBER-t32 Submission to NESSIE. http://www.cryptonessie.org.* ***8***. *L.Simpson, E.Dawson, J.Golić, W.Millan. The LILI-128 Keystream Generator. http://www.cryptonessie.org.* ***9***. *Johan Håstad, Mats Näslund. BMGL: Synchronous Key-stream Generator with Provable Security. http://www.cryptonessie.org.* ***10***. *P. Hawkes, G.G. Rose. Primitive Specification for SOBER-128. http://www.qualcomm.com.au.* ***11***. *P. Hawkes, G.G. Rose. Turing: a fast stream cipher. http://www.qualcomm.com.au..* ***12***. *NESSIE Call for Cryptographic Primitives, Version 2.2, 8th March 2000: http://cryptonessie.org* ***13***. *CRYPTREC. Technical report of Cryptography Research and Evaluation Committees. http://cryptrec.jp.*