УДК 681.3.06

АНАЛИЗ СХЕМ ПОТОЧНОГО ШИФРОВАНИЯ, ПРЕДСТАВЛЕННЫХ НА ЕВРОПЕЙСКИЙ КОНКУРС NESSIE

Иван Горбенко, Александр Потий, Юрий Избенко*, Светлана Орлова
Харьковский Национальный университет радиоэлектроники
**Харьковский военный университет*

Анотація: **Розглядаються схеми потокового шифрування, подані в якості кандидатів на європейський стандарт (проект NESSIE). Наводяться результати криптоаналізу схем, показники складності реалізації, швидкодії, а також результати статистичного тестування схем с використанням пакету NIST STS.**
Summary: **In this work the stream encryption schemes, which were offered as the candidates for the European standard (the NESSIE project), are discussed. The results of cryptoanalysis of these schemes, characteristics of complexity of implementation and efficiency, and also outcomes of statistical testing of the schemes with usage of a package NIST STS are presented.**
Ключевые слова: **Криптографические стандарты, поточные шифры, проект NESSIE, криптографическая стойкость, криптоанализ.**

Введение

В настоящее время в Европе в рамках проекта NESSIE (New European Schemes for Signature, Integrity, and Encryption) проходит конкурс на разработку европейских криптографических стандартов. Как известно, в 2001 году в США по результатам конкурса, организованного Национальным институтом стандартизации и технологий США, был принят новый стандарт блочного шифрования AES-Rijndael. Данный алгоритм был признан после двухлетних исследований и открытого обсуждения в рамках проекта AES (Advanced Encryption Standard) в качестве лучшего и в настоящее время принят в качестве федерального стандарта США (FIPS 197) [1].

Проект AES как концептуальный подход к разработке качественных криптографических примитивов показал весьма высокую эффективность в широком смысле. Именно поэтому европейское сообщество пошло по этому пути для разработки новых европейских стандартов.

Программа NESSIE значительно шире и предусматривает создание следующих криптографических примитивов:

- алгоритм блочного шифрования;
- алгоритм поточного шифрования;
- алгоритм несимметричного шифрования;
- алгоритм формирования MAC-кодов и хэш-функций;
- цифровая подпись;
- идентификационная схема.

В таблице 1 представлен перечень претендентов на Европейские стандарты.

Таким образом, идя путем открытого обсуждения криптографических примитивов, в Европе одним проектом пытаются стандартизировать все основные механизмы безопасности. Изучение предложенных схем, анализ результатов их испытаний и собственные исследования, по нашему мнению, представляют большой интерес для украинского криптографического сообщества.

Данная статья посвящена рассмотрению и анализу претендентов 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 очень широко обсуждаются именно схемы поточного шифрования. Следует отметить, что криптографические примитивы проекта NESSIE уже рассматриваются ISO относительно возможности включения, например, схем поточного шифрования в новый международный стандарт ISO/IEC.

В Украине сегодня практически завершена разработка стандарта на цифровую подпись. На наш взгляд, сегодня сложились все условия для начала работ по разработке национальных стандартов блочного и поточного симметричного шифрования или выбора из рассматриваемых в проекте NESSIE наилучшего. Результаты анализа схем поточного шифрования, представленные в статье, будут полезны разработчикам современных схем поточного шифрования в частности, и криптографических примитивов в целом.

В первой части статьи приводится обзор схем поточного шифрования наиболее вероятных кандидатов на стандарт: SNOW, Sober-t16, Sober-t32, BMGL, Lili-128. Вторая часть статьи посвящена обсуждению теоретических предпосылок оценки стойкости данных схем. В третьей части приводятся результаты криптоанализа схем, полученных в ходе выполнения проекта, в четвертой части рассматриваются алгоритмы ввода ключей. Авторы выполнили значительный объем работ по программной реализации и исследованию на их основе схем поточного шифрования. В пятой части приводятся результаты исследования сложности реализации и производительности (быстродействия) схем. С использованием программных реализаций схем авторы выполнили их статистическое

тестирование с применением пакета NIST STS [2]. В шестой части статьи приводятся результаты статистического тестирования схем поточного шифрования. Отметим, что столь полные данные получены впервые. В рамках проекта NESSIE данные схемы не подвергались такому объемному тестированию и сравнению.

На основе анализа материалов конкурса, а также собственных результатов, формулируются рекомендации и предложения, а также высказываются взгляды на современные тенденции по созданию и применению схем поточного шифрования.

Таблица 1

Номинации	Претенденты	Разработчики
Блочные шифры	**IDEA**	MediaCrypt AG, **Швейцария;**
	Khazad	Scopus Tecnologia S.A., Brazil and K.U.Leuven, **Бельгия**;
	MISTY1	Mitsubishi Electric Corp., **Япония**;
	SAFER++64, SAFER++128	Cylink Corp., **США**, ETH Zürich, **Швейцария**, National Academy of Sciences, **Армения**;
	Camellia	Nippon Telegraph and Telephone Corp., **Япония** and Mitsubishi Electric, **Япония**;
	RC6	RSA Laboratories Europe, **Швеция** and RSA Laboratories, **США**;
	SHACAL	Gemplus, **Франция**.
Поточные шифры	**SOBER-t16, SOBER-t32**	Qualcomm International, **Автсралия**;
	SNOW	Lund Univ., **Швеция**;
	BMGL	Royal Institute of Technology, Stockholm and Ericsson Research, **Швеция**;
	LILI-128	Queensland University of Technology, **Великобритания**;
	LEVIATHAN	Cisco Systems, San Jose, **США**.
Несимметричные алгоритмы шифрования	**ACE Encrypt**	IBM Zurich Research Laboratory, **Швейцария**;
	EPOC-2	Nippon Telegraph and Telephone Corp., **Япония**;
	PSEC-2	Nippon Telegraph and Telephone Corp., **Япония**;
	ECIES	Certicom Corp., USA and Certicom Corp., **Канада**;
	RSA-OAEP	RSA Laboratories Europe, **Швеция** and RSA Laboratories, **США**.
Алгоритмы формирования MAC-кодов и хеш-функции	**Two-Track-MAC**	K.U.Leuven, **Бельгия** and debis AG, **Германия**;
	UMAC	Intel Corp., **США**, Univ. of Nevada at Reno, **США**, IBM Research Laboratory, **США**, Technion, **Израиль**, and Univ. of California at Davis, **США**;
	Whirlpool	Scopus Tecnologia S.A., **Бразилия** and K.U.Leuven, **Бельгия**.
Цифровые подписи	**ECDSA**	Certicom Corp., США and Certicom Corp., **Канада**;
	ESIGN	Nippon Telegraph and Telephone Corp., **Япония**;
	RSA-PSS	RSA Laboratories Europe, Швеция and RSA Laboratories, **США**;
	SFLASH	BULL CP8, **Франция**;
	QUARTZ	BULL CP8, **Франция**.
Идентификационные схемы	**GPS**	Ecole Normale Supérieure, Paris, BULL CP8, France Télécom and La Poste, **Франция**

I Описание схем

SNOW [3]. Шифр SNOW является синхронным поточным шифром, который основан на идее классического суммирующего генератора. Он принадлежит к классу схем с равномерным движением регистра. Авторы – Патрик Экдахл и Томас Йоханссон из Ландского университета, Департамент информационных технологий, Швеция. Возможные длины ключей – 128 и 256 бит.

Структурная схема генератора (шифрообразующего устройства) приведена на рисунке 1. Она состоит из линейного рекуррентного регистра (ЛРР) длиной 16 над $GF(2^{32})$, который задаёт состояния конечного автомата (КА). КА состоит (рис. 2) из блока подстановки S и двух 32-разрядных регистров $R1$, $R2$. В нем для формирования выходного потока и последующего обновления состояния регистров $R1$ и $R2$ используются операции сложения по модулю 2^{32} и по модулю 2, а также циклические сдвиги.

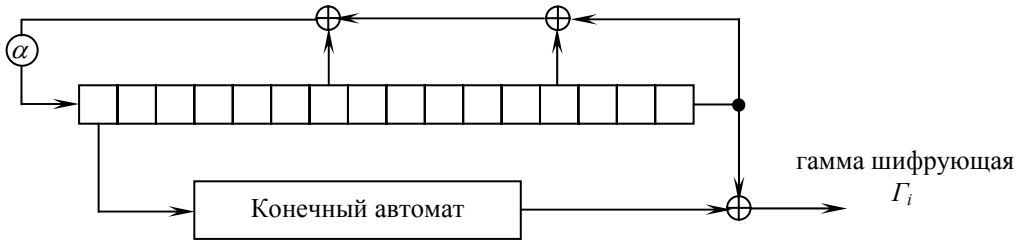

Рисунок 1 - Шифрообразующее устройство SNOW

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

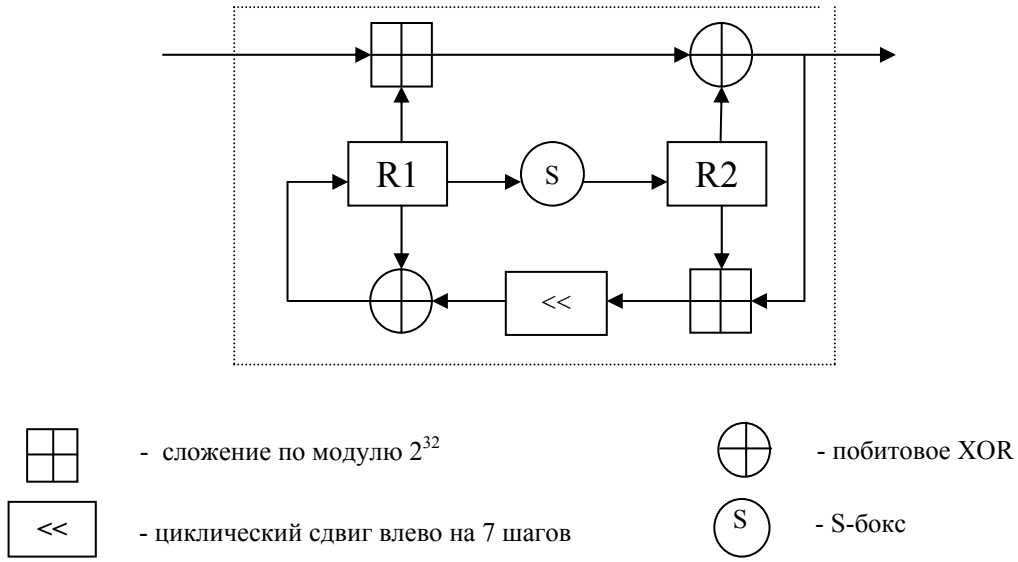

Рисунок 2 - Конечный автомат

Как видно из рисунка 1, ЛРР строится с использованием примитивного полинома $p(x)$, который задает обратную связь над полем $GF(2^{32})$, причем

$$p(x) = x^{16}+x^{13}+x^{7}+\alpha.$$

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 гаммы шифрующей 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 гаммы шифрующей содержимое регистров $R1$ и $R2$ обновляется следующим образом:

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

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

Подстановка, обозначаемая как $S(x)$, состоит из 4 идентичных биективных $8\rightarrow8$ 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. Перестановку можно описать следующим образом:

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

т.е., 31-я битовая позиция отображается в 3-ю, 30-й бит отображается в 10 и т.д. Используя векторную систему обозначений, это можно записать так:

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

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

$$s(1) = k(1), s(2) = k(2), \dots, s(4) = k(4), s(5) = k(1) \oplus 1, \dots, s(8) = k(4) \oplus 1,$$

аналогично для второй половины

$$s(9) = k(1), s(10) = k(2), \dots, s(12) = k(4), s(13) = k(1) \oplus 1, \dots, s(16) = k(4) \oplus 1,$$

где 1 обозначает 32-битный единичный вектор.

При 256-битовом ключе сдвиговый регистр инициализируется по правилу:

$$s(1) = k(1), s(2) = k(2), \dots, s(8) = k(8), s(9) = k(1) \oplus 1, \dots, s(16) = k(8) \oplus 1.$$

После инициализации ЛРР, регистры $R1$ и $R2$ устанавливаются в ноль. Генератор запускается 64 раза без генерации выходной последовательности; 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 64 циклов сдвиговый регистр, а также регистры $R1$, $R2$ принимают значения, вычисленные на последнем цикле. Первые 32 бита гаммы шифрующей являются результатом побитового сложения $s(16) \oplus KA[i]$, начиная с 65 цикла. Максимально допустимая длина гаммы шифрующей, сформированной 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 генератор должен быть перезапущен.

SOBER-t16 и SOBER-t32 [4, 5]. Синхронные поточные шифры семейства SOBER являются машинно-ориентированными шифрами, основанными на идее классического суммирующего генератора, и принадлежат к классу схем с равномерным движением регистра и неравномерным усечением. Авторы шифров – Филип Хаукес и Грегори Розе (Австралия). В Sober-t16 используется 128-битный ключ, а в Sober-t32 - 256-битный ключ. 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 генератора (шифрообразующего устройства) представлена на рис.3. ШУ состоит из ЛРР, нелинейной функции (НЛФ) (нелинейный фильтр) 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 рисунке 4.

Процесс генерации гаммы шифрующей Γ_i происходит следующим образом. Первоначально производится начальное заполнение рекуррентного регистра ключевыми данными. Затем вычисляются первые 16 бит (32 бита) гаммы шифрующей 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 $GF(2^w)$:

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

где $\alpha = 0xE382$, $\beta = 0x67C3$, $\lambda = 0xC2DB2AA3$, 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 $y = y_{31}\alpha^{31} + y_{30}\alpha^{30} + \ldots + y_1\alpha + y_0$.

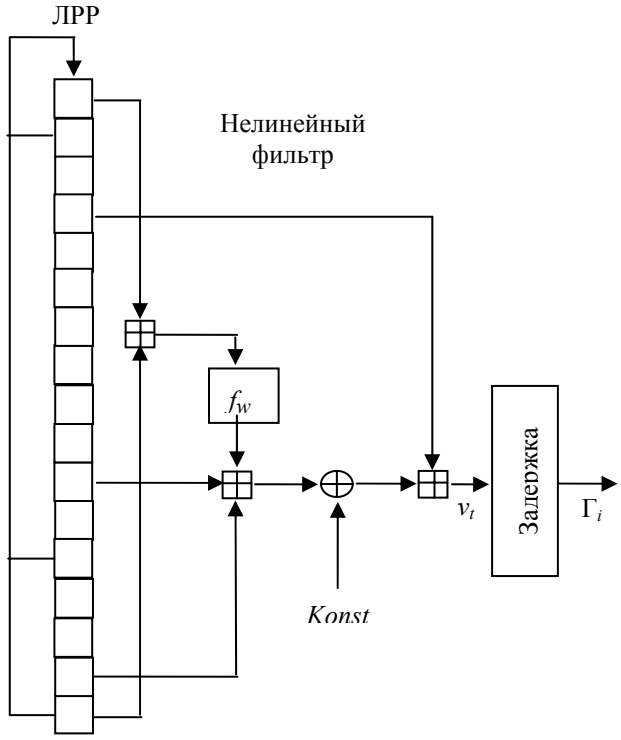

Рисунок 3 - Генератор Sober-t16(Sober-t32)

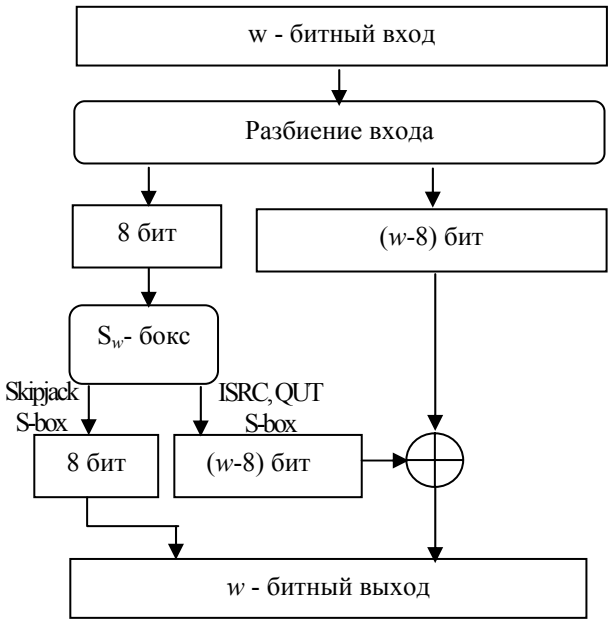

Рисунок 4 - Структура функции f_w

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 (таблица 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 генератора
11	Сдвинуть ЛРР, выход НЛФ побитно складывается (XOR) с 0x9669 (0x96693AC5)

Выход преобразователя “задержка” и является гаммой шифрующей генератора.

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 $2^{17 \cdot 16+16}$ и $2^{17 \cdot 32+32}$ бит соответственно. После генерации указанного числа бит гаммы шифрующей необходимо ввести новые ключи.

LILI-128 [6]**.** Синхронный поточный шифр LILI-128 является машинно-ориентированным шифром и принадлежит к классу схем с неравномерным движением. Авторы – Л.Симпсон, Е.Доусон, Й.Голич и У.Миллан. Шифр использует 128 битный ключ. Данный шифр не прошел отборочный тур. Однако здесь он рассматривается как пример классической схемы поточного шифрования на основе управляющего регистра.

Компоненты ШУ сгруппированы в две подсистемы: подсистему управления движением и подсистему генерации данных. Схема ШУ представлена на рисунке 5.

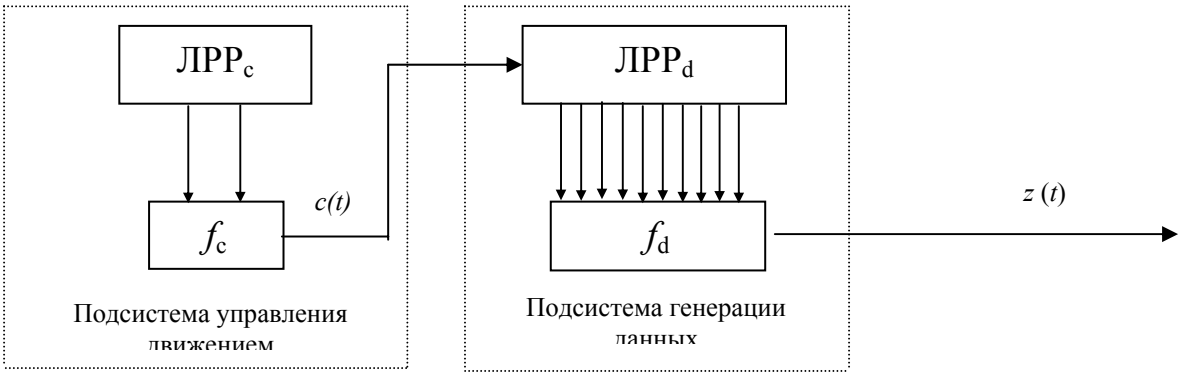

Рисунок 5 – Шифрообразующее устройство LILI-128

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 $z = \{z(t)\}_{t=1}^{\infty}$.

BMGL [7]. BMGL – конструкция синхронного генератора ключевого потока с доказуемой стойкостью. 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 $\{0,1\}^n \rightarrow \{0,1\}^n$. 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 $x_i = f(x_{i-1})$, $i = 1,2,\ldots, \lambda$, 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 рисунке 6.

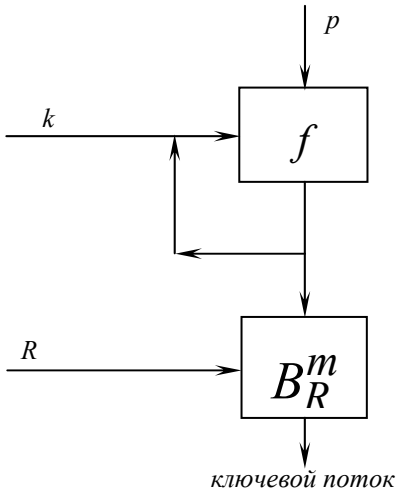

Рисунок 6 – Режим обратной связи по ключу

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.

II Теоретические предпосылки стойкости

Рассмотрим теоретические предпосылки стойкости схем поточного шифрования.

SNOW. Шифр состоит из двух конструктивных блоков — регистра сдвига и конечного автомата. Каждый из блоков предназначен для реализации в шифре необходимых криптографических характеристик.

Линейный рекуррентный регистр образован на примитивном полиноме, что гарантирует генерацию последовательности с максимальным периодом и хорошими статистическими свойствами; поле $GF(2^{32})$ формируется на основе неприводимого полинома. Такая организация регистра эквивалентна системе из 32 параллельных регистров над $GF(2)$, каждый длиной 16·32=512 бит. Поэтому результирующая линейная рекурсия может быть сформирована с использованием полинома $p(x)$ над $GF(2)$ вида

$$p(x) = 1 + x^7 + x^{13} + x^{16} + x^{70} + x^{82} + x^{88} + x^{105} + x^{111} + x^{114} + x^{117} + x^{118} + x^{126} + x^{130} + x^{132} + x^{135} + x^{136} + x^{138} + x^{140} + x^{142} + x^{147} + x^{153} + x^{154} + x^{156} + x^{159} + x^{160} + x^{164} + x^{168} + x^{174} + x^{176} + x^{177} + x^{180} + x^{189} + x^{195} + x^{198} + x^{203} + x^{207} + x^{209} + x^{212} + x^{213} + x^{216} + x^{219} + x^{224} + x^{227} + x^{228} + x^{233} + x^{234} + x^{237} + x^{239} + x^{240} + x^{245} + x^{248} + x^{251} + x^{257} + x^{272} + x^{284} + x^{287} + x^{293} + x^{296} + x^{308} + x^{311} + x^{317} + x^{323} + x^{329} + x^{332} + x^{335} + x^{341} + x^{344} + x^{371} + x^{377} + x^{380} + x^{407} + x^{413} + x^{431} + x^{437} + x^{440} + x^{443} + x^{449} + x^{452} + x^{455} + x^{461} + x^{464} + x^{512}.$$

Период такой линейной рекурсии равен $2^{16\cdot 32} - 1 = 2^{512} - 1 \approx 1.3\cdot 10^{154}$

Конечный автомат по сути своей является аналогом нелинейной функции фильтрации и предназначена для устранения линейной зависимости выхода регистра от входа, затрудняя, таким образом, возможность применения к выходной последовательности атаки Берлекемпа-Месси.

Регистры $R1$ и $R2$ предназначены для оперирования с переменными, которые являются результатом рассеивания и перестановок компонентами КА. Циклический сдвиг переменных и преобразование их в блоке подстановки S, помимо рассеивания и перестановки, устраняет возможность проведения после операции модулярного сложения атаки на наименьший значащий бит на выходе КА.

Анализ блока подстановки S показал [8], что применение линейного криптоанализа дает уравнение с максимальной эффективностью $|p-\frac{1}{2}| = 0.125$, т.е., уравнение

$$a \circ x = b \circ s\text{-}box(x), \text{ где } a = 79_{(16)},\ b = ff_{(16)}$$

с вероятностью $p = 0.625$, и уравнение

$$a \circ x = b \circ s\text{-}box(x), \text{ где } a = f0_{(16)},\ b = df_{(16)}$$

с вероятностью $p = 0.375$.

Дифференциальный анализ дает только малое отклонение распределения входных разностей от выходных разностей (так, при входной разности $\Delta_{\text{входн}}=ff_{(16)}$ получим выходную разность $\Delta_{\text{вых}}=e3_{(16)}$ с вероятностью $p = 0.02344$). В порядке определения корреляции между выходными битами блока и битами регистра рассмотрена линейная аппроксимация 32-в-32 бита S-блока $= (s_1,\ldots,s_{32})$ и показано, что наилучшая аффинная аппроксимация булевой функции s_i удерживается с вероятностью 0.625, каждая функция s_i имеет корреляционный иммунитет порядка 0, а лучшая корреляция одной из функций s_i удерживается с вероятностью 0.5625.

Считается [8], что очень тяжело определить корреляцию между входом и выходом КА с вероятностью значительно выше, чем $\frac{1}{2}$.

SOBER-t16, SOBER-t32. Данные шифры состоят из трех конструктивных блоков – регистра сдвига, нелинейной функции и задержки.

Линейный рекуррентный регистр образован на примитивном полиноме, что гарантирует генерацию последовательности с максимальным периодом и хорошими статистическими свойствами; элементы поля $GF(2^{16})$ ($GF(2^{32})$) вычисляются с использованием соответствующих неприводимых полиномов. Такая схема регистра эквивалентна схеме w, $w \in \{16, 32\}$, параллельных регистров над $GF(2)$, каждый длиной $17 \cdot w$. Поэтому длина регистров для Sober-t16 равна $17 \cdot 16 = 272$ битов каждый, а для Sober-t32 – $17 \cdot 32 = 544$ битов каждый. Такой линейный регистр строится для Sober-t16 в соответствии с полиномом $p_2(x)$ над $GF(2)$ вида

$$p_2(x) = 1 + x^2 + x^4 + x^6 + x^8 + x^{15} + x^{16} + x^{17} + x^{18} + x^{19} + x^{20} + x^{21} + x^{24} + x^{27} + x^{28} + x^{30} + x^{32} + x^{33} + x^{39} + x^{41} + x^{42} + x^{43} + x^{46} + x^{48} + x^{51} + x^{55} + x^{56} + x^{57} + x^{58} + x^{61} + x^{62} + x^{63} + x^{64} + x^{68} + x^{72} + x^{73} + x^{75} + x^{78} + x^{81} + x^{82} + x^{83} + x^{84} + x^{85} + x^{87} + x^{88} + x^{89} + x^{90} + x^{91} + x^{92} + x^{94} + x^{95} + x^{97} + x^{99} + x^{100} + x^{103} + x^{111} + x^{112} + x^{115} + x^{118} + x^{119} + x^{124} + x^{126} + x^{127} + x^{129} + x^{130} + x^{131} + x^{132} + x^{133} + x^{135} + x^{136} + x^{137} + x^{138} + x^{139} + x^{140} + x^{141} + x^{143} + x^{148} + x^{149} + x^{153} + x^{154} + x^{156} + x^{159} + x^{161} + x^{165} + x^{166} + x^{167} + x^{168} + x^{169} + x^{172} + x^{174} + x^{177} + x^{180} + x^{185} + x^{187} + x^{188} + x^{189} + x^{190} + x^{191} + x^{194} + x^{195} + x^{197} + x^{198} + x^{199} + x^{201} + x^{202} + x^{203} + x^{204} + x^{205} + x^{207} + x^{209} + x^{213} + x^{214} + x^{215} + x^{218} + x^{220} + x^{223} + x^{225} + x^{226} + x^{229} + x^{230} + x^{231} + x^{233} + x^{234} + x^{237} + x^{238} + x^{240} + x^{242} + x^{244} + x^{245} + x^{247} + x^{253} + x^{255} + x^{257} + x^{260} + x^{264} + x^{268} + x^{272}.$$

При этом обеспечивается формирование последовательности с периодом повторения $2^{17 \cdot 16} - 1 = 2^{272} - 1 \approx 7.6 \cdot 10^{81}$. Для Sober-t32 полином имеет вид

$$p(x) = 1 + x^{17} + x^{19} + x^{21} + x^{23} + x^{25} + x^{27} + x^{29} + x^{30} + x^{31} + x^{33} + x^{34} + x^{35} + x^{37} + x^{38} + x^{39} + x^{41} + x^{42} + x^{46} + x^{47} + x^{49} + x^{50} + x^{51} + x^{53} + x^{54} + x^{55} + x^{56} + x^{57} + x^{58} + x^{59} + x^{61} + x^{62} + x^{63} + x^{64} + x^{65} + x^{66} + x^{67} + x^{68} + x^{70} + x^{74} + x^{76} + x^{77} + x^{78} + x^{79} + x^{84} + x^{85} + x^{87} + x^{89} + x^{90} + x^{91} + x^{92} + x^{95} + x^{97} + x^{98} + x^{100} + x^{101} + x^{102} + x^{109} + x^{111} + x^{113} + x^{114} + x^{117} + x^{118} + x^{121} + x^{125} + x^{130} + x^{131} + x^{132} + x^{133} + x^{137} + x^{138} + x^{140} + x^{142} + x^{143} + x^{145} + x^{146} + x^{147} + x^{148} + x^{149} + x^{151} + x^{153} + x^{156} + x^{160} + x^{163} + x^{164} + x^{165} + x^{172} + x^{173} + x^{175} + x^{176} + x^{177} + x^{179} + x^{180} + x^{184} + x^{185} + x^{186} + x^{190} + x^{191} + x^{193} + x^{198} + x^{200} + x^{201} + x^{202} + x^{206} + x^{207} + x^{208} + x^{209} + x^{210} + x^{211} + x^{212} + x^{213} + x^{219} + x^{220} + x^{221} + x^{225} + x^{227} + x^{229} + x^{231} + x^{232} + x^{233} + x^{235} + x^{236} + x^{238} + x^{239} + x^{240} + x^{241} + x^{242} + x^{244} + x^{245} + x^{246} + x^{247} + x^{249} + x^{252} + x^{255} + x^{258} + x^{262} + x^{263} + x^{264} + x^{265} + x^{266} + x^{277} + x^{279} + x^{281} + x^{284} + x^{285} + x^{288} + x^{289} + x^{290} + x^{291} + x^{292} + x^{294} + x^{296} + x^{300} + x^{301} + x^{302} + x^{304} + x^{306} + x^{307} + x^{309} + x^{310} + x^{316} + x^{321} + x^{323} + x^{324} + x^{325} + x^{327} + x^{334} + x^{335} + x^{336} + x^{337} + x^{340} + x^{341} + x^{342} + x^{344} + x^{345} + x^{346} + x^{347} + x^{350} + x^{352} + x^{355} + x^{357} + x^{360} + x^{361} + x^{362} + x^{363} + x^{364} + x^{365} + x^{368} + x^{373} + x^{377} + x^{379} + x^{381} + x^{382} + x^{383} + x^{385} + x^{388} + x^{389} + x^{390} + x^{391} + x^{392} + x^{394} + x^{398} + x^{403} + x^{404} + x^{405} + x^{406} + x^{407} + x^{413} + x^{416} + x^{420} + x^{421} + x^{422} + x^{425} + x^{426} + x^{428} + x^{430} + x^{431} + x^{433} + x^{435} + x^{436} + x^{437} + x^{438} + x^{440} + x^{441} + x^{442} + x^{445} + x^{446} + x^{447} + x^{448} + x^{449} + x^{450} + x^{453} + x^{458} + x^{461} + x^{463} + x^{465} + x^{466} + x^{469} + x^{471} + x^{473} + x^{474} + x^{477} + x^{478} + x^{479} + x^{481} + x^{483} + x^{484} + x^{487} + x^{488} + x^{489} + x^{490} + x^{493} + x^{494} + x^{496} + x^{499} + x^{500} + x^{503} + x^{505} + x^{506} + x^{508} + x^{511} + x^{513} + x^{514} + x^{516} + x^{519} + x^{521} + x^{524} + x^{527} + x^{529} + x^{532} + x^{536} + x^{540} + x^{544}.$$

Период формируемой последовательности будет равен $2^{17 \cdot 32} - 1 = 2^{544} - 1 \approx 5.6 \cdot 10^{163}$.

Данные полиномы являются непрореженными, что исключает возможность проведения быстрой корреляционной атаки. Выбор точек съема значений $T_{\text{ЛРР}} = \{0,4,15,17\}$ определен в соответствии с

требованиями [10], поэтому они удовлетворяют полному множеству положительных разностей $\Delta T_{\text{ЛРР}} = \{2,4,11,13,15,17\}$, что обеспечивает стойкость схемы.

НЛФ предназначена для устранения статистической зависимости между выходом регистра сдвига и выходом НЛФ. Применение нелинейной функции обеспечивает защиту от корреляционной атаки и GD-атаки ("предположение и определение"). Выбор точек съема значений $T_{\text{НЛФ}} = \{0,1,6,13,16\}$ обоснован в [10]. Они удовлетворяют полному множеству положительных разностей $\Delta T_{\text{НЛФ}} = \{1,3,5,6,7,10,12,13,15,16\}$, что обеспечивает стойкость схемы к корреляционной и инверсионной атакам.

Блоки подстановки S являются комбинацией разработанных ранее Skipjack *S*-блока и ISRC *S*-блока. Для *S*-блока Sober-t16, преобразовывающего 8 бит в 16 (8×16), для всех 8 входных бит x справедливо выражение:

$$S\text{-блок}\ (x) = skipjack(x) \parallel isrc1(x),$$

и для *S*-блока Sober-t32, преобразовывающего 8 бит в 24 (8×24), для всех 8 входных бит x справедливо выражение:

$$S\text{-блок}\ (x) = skipjack(x) \parallel isrc2(x),$$

где $skipjack(x)$ обозначает 8×8 *S*-блок, $isrc1(x)$ обозначает 8×8 ISRC S-блок и $isrc2(x)$ обозначает 8×24 ISRC *S*-блок.

Для Skipjack S-блока [9] имеем уравнения с максимальной эффективностью $|p-\frac{1}{2}| = 0.109$, т.е., уравнение

$$a \circ x = b \circ s\text{-}box(x), \text{ где } a = 4d_{(16)},\ b = a8_{(16)}$$

с вероятностью $p = 0.391$, и уравнение

$$a \circ x = b \circ s\text{-}box(x), \text{ где } a = 7d_{(16)},\ b = a8_{(16)}$$

с вероятностью $p = 0.609$.

Для них дифференциальный анализ дает только малое отклонение распределения входных разностей от выходных разностей (так, при входной разности $\Delta_{\text{входн}} = 90_{(16)}$ получим выходную разность $\Delta_{\text{вых}} = b1_{(16)}$ с вероятностью $p = 0.0468$).

Для 8×8 ISRC S-блока [9] имеем уравнения с максимальной эффективностью $|p-\frac{1}{2}| = 0.195$, т.е., уравнение

$$a \circ x = b \circ s\text{-}box(x), \text{ где } a = 21_{(16)},\ b = bf_{(16)}$$

с вероятностью $p = 0.367$, и уравнение

$$a \circ x = b \circ s\text{-}box(x), \text{ где } a = bf_{(16)},\ b = 26_{(16)}$$

с вероятностью $p = 0.641$. В этом случае дифференциальный анализ дает только малое отклонение распределения входных разностей от выходных разностей (так, при входной разности $\Delta_{\text{входн}} = 82_{(16)}$ получим выходную разность $\Delta_{\text{вых}} = ac_{(16)}$ с вероятностью $p = 0.038$).

Для 8×24 ISRC S-блока [9] имеем уравнения с максимальной эффективностью $|p-\frac{1}{2}| = 0.195$, т.е., уравнение

$$a \circ x = b \circ s\text{-}box(x), \text{ где } a = 67_{(16)},\ b = a82d9d_{(16)}$$

с вероятностью $p = 0.313$, и уравнение

$$a \circ x = b \circ s\text{-}box(x), \text{ где } a = cd_{(16)},\ b = 9c5267_{(16)}$$

с вероятностью $p = 0.695$.

Блок децимации введён с целью повышения надежности схемы в целом, поскольку на взгляд разработчиков комбинация регистра сдвига и НЛФ не обеспечивает достаточной защиты от корреляционных атак. К положительным сторонам введения децимации следует отнести и то, что она делает невозможным применение атак, которые бы могли иметь успех при использовании только комбинации регистра сдвига и нелинейной функции. К отрицательным сторонам – снижение быстродействия схемы и уменьшение периода генерируемой последовательности практически в 2 раза – период составляет $\frac{6}{13}\cdot(2^{272}-1)$ и $\frac{12}{25}\cdot(2^{544}-1)$ для Sober-t16 и Sober-t32 соответственно, однако на наш взгляд при данной размерности гаммы последний недостаток является несущественным.

Поскольку данная схема основана на одном регистре, это исключает применение атаки декомпозиции.

LILI-128. Данный шифр состоит из двух схем: схемы управления движения и схемы генерации данных с регистрами 39 и 89 бит соответственно и нелинейными функциями.

Оба регистра основаны на примитивном полиноме, что гарантирует генерацию последовательности с максимальным периодом и хорошими статистическими свойствами. Ячейки, значения которых являются входными данными для нелинейной функции, удовлетворяют полному множеству положительных разностей. Используется сбалансированная корреляционно-иммунная функция третьего порядка, имеющая алгебраическую степень 6 и достигающая нелинейности 480.

Введение схемы управления обеспечивает линейную сложность генерируемой гаммы с нижней границей $\binom{89}{6}(2^{39} - 1) \cong 2^{68}$, что влечет за собой необходимость обладания 2^{69} битами гаммы для применения атаки Берлекемпа-Месси. Необходимость обладания таким объемом гаммы делает данную атаку невозможной.

BMGL. Разработчики данного алгоритма показали [7], что его надёжность обеспечивается в предположении сложности инвертирования итерируемой функции f, причём f – криптографическая функция, как, например, применяемая в блочном шифровании или хеш-функция. Полагается, что минимальное время успешного инвертирования итерируемой функции после i-го повторения примерно равно $2^n/i$. Из этого следует, что меньшее значение m обеспечивает более высокий уровень безопасности.

III Криптоанализ схем

При проведении криптоанализа схем поточного шифрования подразумевается, что атака может быть успешно реализована, если ее вычислительная сложность меньше, чем вычислительная сложность полного перебора всего пространства ключей шифра. Анализ показывает, что к рассматриваемым схемам поточного шифрования наибольшую вероятность применения имеют следующие атаки [12]:

1. Атака грубой силы или исчерпывающий поиск ключа (Exhaustive key search). Теоретически, данный вид атаки должен быть наиболее эффективным по сравнению с остальными атаками. При реализации этой атаки осуществляется полный перебор всех возможных ключей. Считается, что у стойких шифров сложность любой аналитической атаки должна быть не меньше сложности атаки “грубая сила”.

2. *Атака типа “компромисс время-память”* (Time-memory trade-off attack). Целью данной атаки является восстановление начального состояния сдвигового регистра (как правило ключей) по фрагменту шифрующей последовательности при условии, что криптоаналитику известны схема ШУ и некоторый фрагмент гаммы шифрующей. Анализ показывает, что сложность атаки зависит от длины перехваченной гаммы шифрующей и размера ключа (внутреннего состояния) ШУ. Данный вид атаки применим, если пространство состояний достаточно мало. В частности, такая атака была успешно применена к криптоалгоритму A5. Атака может быть реализована в два этапа:

- подготовительный этап, в котором составляется большой словарь, включающий все возможные пары (состояние ШУ, выходная последовательность Γ_i) длины n;
- основной этап, в котором делается предположение, что в ШУ введены определенные ключи и синхромаркеры, т.е., предположение об определенном заполнении всех ячеек памяти. На основе этих данных генерируется выходная последовательность (гамма шифрующая). Затем гамма Γ_i анализируется на предмет определения состояния (ключа и других данных), а по сути определения внутреннего состояния.

В качестве показателей сложности атаки можно рассматривать объем требуемой памяти M, размер памяти (длина регистра) N и количество бит рассматриваемой гаммы шифрующей D. Временную сложность можно рассчитать по формуле $T = D = N / M$, а время подготовительного этапа как $P = M = N/D$.

3. *Атака типа “предполагать и определять”* (Guess-and-determine attack). Основная идея такой атаки состоит в допущении, что криптоаналитику известны гамма шифрующая, количество сдвигов регистра между выходами схемы, полином обратной связи регистра степени r, функция фильтрации f и последовательность точек съема γ_i, $i = 1, \ldots, n$. Восстановление начального заполнения регистра происходит путем вывода некоторых фрагментов заполнения регистра на основе предположения о содержимом некоторых или всех фрагментов схемы (содержимое ячеек регистра, элементы нелинейной функции). В общем виде атака может выполняться в следующей последовательности:

- **Предполагается** (задается) содержимое некоторых ячеек регистра.
- **Определяется (**осуществляется**)** полное заполнение регистра путем применения линейной рекурсии регистра на основе принятых допущений.
- Генерируется выходная последовательность. Если сформированная выходная последовательность эквивалентна гамме шифрующей, предположение принимается верным, в противном случае процесс повторяется вновь.

Анализ показал, что сложность атаки пропорциональна количеству предположений. Так, если предполагается, что такой регистр содержит x ячеек, то число вариантов, которые необходимо рассмотреть, равно $(2^w)^x$, где w – разрядность ячеек регистра.

4. *Корреляционные атаки* (Correlations attacks). Атаки данного класса считаются наиболее “сильными” по отношению к поточным шифрам. Их основная идея заключается в нахождении корреляции между гаммой шифрующей и различными линейными комбинациями ключа. Обычно в качестве объекта исследования атаки

рассматривают нелинейную функцию, вносящую нелинейность в выходную последовательность регистра сдвига. При этом корреляционная атака должна строится с учетом особенности анализируемой нелинейной функции.

5. *Инверсионные атаки* (Inversion attacks). Целью данных атак является восстановление по фрагменту шифрующей последовательности начального состояния сдвигового регистра при условии, что известны полином обратной связи степени r, функция фильтрации f и последовательность точек съема γ_i, $i = 1, \ldots, n$. При выполнении атаки рассматривается M ячеек памяти, где $M = \gamma_n - \gamma_1$, и $M \le r$-1. Атаки базируются на том, что шифр обратим (инвертируем), если известно начальное состояние ячеек памяти. Действительно, пусть выходная последовательность $\{y(t)\}_{t=0}^{\infty}$ генератора задается как

$$y(t) = f(x(t - \gamma_1), \ldots, x(t- \gamma_n)), \qquad t \ge 0, \tag{2}$$

нелинейная функция $f(z_1, z_2, \ldots, z_n)$ сбалансирована для каждого значения $(z_2, \ldots, z_n)$, то есть,

$$f(z_1, z_2, \ldots, z_n) = z_1 + g(z_2, \ldots, z_n). \tag{3}$$

Тогда соотношение (2) представимо как

$$x(t) = y(t) + g(x(t - \gamma_2), \ldots, x(t- \gamma_n)), t \ge 0. \tag{4}$$

Инверсионная атака может осуществляться в следующей последовательности:

- Выбираются не проверенные ранее M бит $\{x(t)\}_{t=-M}^{-1}$ неизвестного начального состояния памяти.
- С использованием (3) по известному отрезку гаммы $\{y(t)\}_{t=0}^{r-M-1}$ генерируется отрезок $\{x(t)\}_{t=0}^{r-M-1}$ входной последовательности.
- Используя ЛРР, по первым r битам $\{x(t)\}_{t=0}^{r-M-1}$ генерируется последовательность $\{x(t)\}_{t=r-M}^{N-1}$.
- Используя $y(t)$, по $\{x(t)\}_{t=r-2M}^{N-1}$ вычисляется $\{\hat{y}(t)\}_{t=r-M}^{N-1}$ и сравнивается с наблюдаемым $\{y(t)\}_{t=r-M}^{N-1}$. Если они совпадают, принятое начальное состояние считается истинным, в противном случае процесс повторяется.

Особенностью этой атаки является то, что ее вычислительная сложность имеет порядок 2^M и не зависит от длины ЛРР – r. Для защиты от этой атаки рекомендуется выбирать M достаточно большим, предпочтительно близким к максимальному значению r-1, а также правильно выбирать номера точек съема [10].

6. *Атаки на различимость* (Distinguishing attacks). Атаки данного класса могут быть успешными, если гамма шифрующая отличима от истинно случайной последовательности.

Рассмотрим результаты применения рассмотренных типов атак к схемам поточного шифрования.

SNOW. Разработчики шифра рассматривают следующие типы атак.

Атака грубой силы или исчерпывающий поиск ключа. Возможно наиболее эффективная и простая атака. Сущность ее заключается в полном переборе всевозможных ключей длинной 128 и 256 бит. Одним из возможных показателей оценки стойкости поточного шифра для такой атаки является безопасное время

$$t_б = \frac{N}{\gamma k} P_y,$$

где N – количество ключей, которые не обходимо перебрать, γ – производительность криптоаналитической системы, $k = 3.1 \cdot 10^7$ с/год, P_y – вероятность успешного выполнения криптоанализа.

Атака типа “компромисс время-память”. Вследствие большого пространства состояний ШУ (512 бит регистра сдвига и двух 32 разрядных регистров $R1$ и $R2$) данная атака считается неприменимой.

Атака типа “предполагать и определять”. Согласно мнению разработчиков, атака данного типа имеет сложность 2^{352}. В то же время в [16] показано, что при использовании 256-битного ключа возможна атака данного класса со сложностью меньшей, чем полный перебор ключа.

Корреляционные атаки. При оценке защищенности от этой атаки разработчики приняли верхнюю границу корреляции между входом и выходом КА равной $0.5 \cdot 2^{-20}$ – именно такое граничное значение корреляции не может привести ко взлому быстрее, чем исчерпывающий поиск ключа. При использовании 256 битного ключа вычислительная сложность такой атаки составляет 2^{400}.

Инверсионные атаки. Вследствие выбора большого размера памяти M и правильного подбора точек съема γ вычислительная сложность составляет 2^{512}.

Атаки на различимость. Статистических слабостей данного шифра не обнаружено.

SOBER-t16, SOBER-t32. При оценке стойкости ШУ SOBER разработчики рассматривают следующие виды атак.

Атака грубой силы или исчерпывающий поиск ключа. Наиболее эффективная атака. В ней реализуется полный перебор всевозможных комбинаций – 128 и 256 бит соответственно.

Атака типа “компромисс время-память”. Вследствие большого пространства состояний – 272 и 544 бит регистра сдвига соответственно, и 16-ти или 32-х разрядной константы данная атака считается неприменимой.

Атака типа “предполагать и определять”. Атака данного типа имеет сложность не менее 2^{160} и 2^{320} соответственно. Поэтому атаки данного типа считаются неприменимыми.

Корреляционные атаки. Атаки данного класса считаются неприменимыми к данному шифру вследствие введения в конструкцию ШУ блока задержки. В целом же стойкость к данному виду атаки обуславливается выбором конечного полинома с высоким весом (136 и 272 ненулевых коэффициента соответственно) и степенью, удовлетворяющего условиям стойкости [10]. Кроме того, используемый блок подстановки *S* обеспечивает высокую нелинейность.

Инверсионные атаки. Вследствие выбора большого размера памяти *M* и правильного подбора точек съема *γ*, вычислительная сложность атаки составляет 2^{272} и 2^{544} соответственно.

Атаки на различимость. Согласно [17], атака данного класса возможна на Sober-t16. Наши исследования показали, что лишь в одном тесте тестирование прошли менее 96% последовательностей Sober-t16. Статистических слабостей Sober-t32 нами не обнаружено, однако данный шифр был отвергнут NESSIE в силу того, что атака данного класса применима к шифру, из конструкции которого была выведен блок задержки. По утверждениям разработчиков, схема на основе только ЛРР и НЛФ обладает достаточной стойкостью.

LILI-128. При анализе стойкости такого ШУ рассматриваются следующие виды атак.

Атака декомпозиции. Данная атака имеет сложность порядка 2^{112}, что является лучшим показателем, чем полный перебор.

Атака типа “компромисс время-память. Данный вид атаки применим вследствие достаточно малого пространства состояний. Временная сложность составляет 2^{100} и требуется 2^{28} бит памяти.

Атака Стива Бэббиджа. В работе [13] показано, что существует атака со сложностью 2^{124}, что меньше, чем полный перебор ключа. Там же показана возможность применения атаки на процедуру установки ключа со сложностью меньшей, чем полный перебор ключа.

Атака встраиванием. (Embedding attack). В неопубликованном отчёте Вильям Чамберс заявил о существовании атаки со сложностью 2^{96}.

Атака Йонсона и Йохансона. Используя предположение о начальном заполнении управляющего регистра и быструю корреляционную атаку на основе знаний о нелинейной функции подсистемы генерации данных, авторы [14] показали, что существует атака со сложностью 2^{71}.

Атаки на различимость. Представленная авторами алгоритма таблица истинности используемой булевой функции является сбалансированной, что дает основания считать о неприменимости данного типа атак.

В таблице 3 представлены обобщенные результаты определения стойкости поточных схем шифрования, которые принимали участие в первом этапе отбора кандидатов на стандарт.

Таблица 3

	Атака “время-память”	Атака “предполагать и определять”	Корреляц. атаки	Инверсионные атаки	Атака на различи-мость	Полный перебор ключей
Snow128	2^{512+64}	2^{352}	2^{400}	2^{512}	не прим.	2^{128}
Snow256	2^{512+64}	$<2^{256}$	2^{400}	2^{512}	не прим.	2^{256}
Sober-t32	2^{544+32}	2^{320}	$>>2^{256}$	2^{544}	прим.	2^{256}
Sober-t16	2^{272+16}	2^{160}	$>>2^{128}$	2^{272}	прим.	2^{128}
Lili-128	2^{100}	2^{124}	2^{71}	2^{128}	не прим.	2^{128}

Из анализа данных, приведенных в таблице 3, следует, что для алгоритмов SNOW и SOBER не существует аналитических атак, сложность которых была бы меньше сложности атаки с полным перебором ключей. Поэтому поточные схемы Snow128, Snow256, Sober-t16 и Sober-t32 к настоящему времени можно считать обеспечивающими реальную криптостойкость.

IV Ввод ключей

В данном разделе кратко рассматриваются вопросы ввода ключей.

SNOW. Алгоритм ввода ключей исключает нулевое начальное заполнение ШУ-векторов. Алгоритм перезагрузки разработчиками не представлен.

SOBER-t16, SOBER-t32. Установка ключей осуществляется таким образом, что каждый бит начального состояния является нелинейной функцией каждого бита сеансового ключа. Это позволяет противостоять атакам, основанным на установке (загрузке) ключей, например, атаке на связанные ключи. К достоинствам схемы следует отнести предусмотренную разработчиками возможность реинициализации (изменения начального состояния) регистра.

LILI-128. Генератор инициализируется простой загрузкой 128 бит ключа в регистры. Ключи, приведшие к загрузке любого регистра нулями, считаются недействительными. Алгоритм перезагрузки ключей разработчиками не представлен.

V Аппаратная и программная реализации ШУ

В данной статье рассматриваются некоторые практические вопросы относительно реализации схем поточного шифрования. Для сравнительного анализа используются только схемы, которые прошли первый этап оценки в рамках проекта Nessie. К таким схемам относятся Snow и Sober.

Сегодня, помимо высокой криптостойкости, к схемам поточного шифрования выдвигаются требования по обеспечению высокой производительности (большой скорости шифрования), гибкости и эффективности реализации.

В таблице 4 приведены показатели, характеризующие затраты памяти при аппаратном исполнении схем (в битах).

Таблица 4

	Snow128		Snow256		Sober-t32		Sober-t16	
ОЗУ	ключ регистр R1, R2	128 512 64	ключ регистр R1, R2	256 512 64	ключ регистр	256 544	ключ регистр	128 272
ПЗУ	S-блок	4096*8	S-блок	4096*8	S-блок 3 const 4 табл.умн	1024*8 12*8 4096*8	S-блок 3 const 4 табл.умн	512*8 6*8 2048*8

Для удобства сравнения рассматриваемых схем с точки зрения быстродействия в таблицах 5, 6 представлены значения количества выполняемых операций при установке ключей и генерации гаммы шифрующей, соответственно.

Таблица 5. Количество выполняемых операций при загрузке ключей

Операция	Snow128	Snow256	Sober-t16	Sober-t32	BMGL
Табличные преобразования	256	256	371	371	560
Сдвиги	268	280	70	78	0
Циклические сдвиги	64	64	0	0	0
And, Or, Xor	736	768	189	253	0
Add	158	174	212	242	544

Таблица 6. Количество выполняемых операций при генерации гаммы шифрующей

Операция	Snow128	Snow256	Sober-t16	Sober-t32	BMGL
Табличные преобразования	64	64	138	288	160
Сдвиги	112	112	39	111	80
Циклические сдвиги	16	16	0	0	0
And, Or, Xor	224	224	156	262	224
Add	77	77	63	119	20
Выходн. бит / цикл ЛРР	32	32	16	32	32

На рисунке 7 представлены результаты оценки быстродействия рассматриваемых схем при генерации 12.5 Мбайт гаммы шифрующей (без BMGL). На рисунке 8 представлены результаты оценки быстродействия регистров сдвига и построенных на их основе схем.

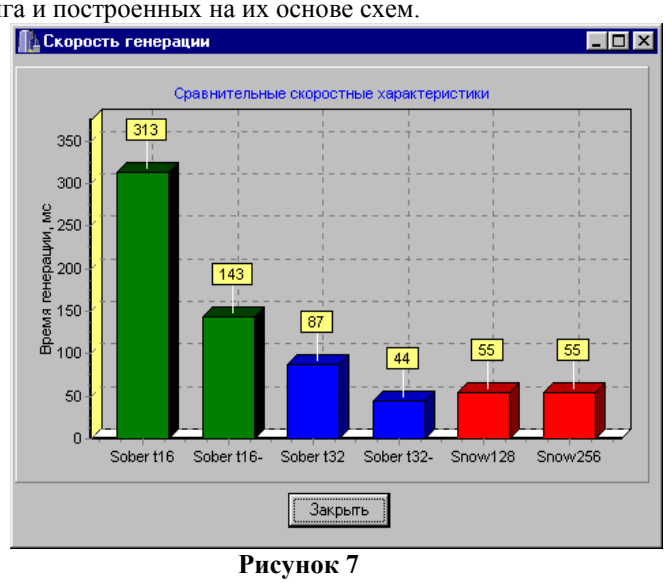

Рисунок 7

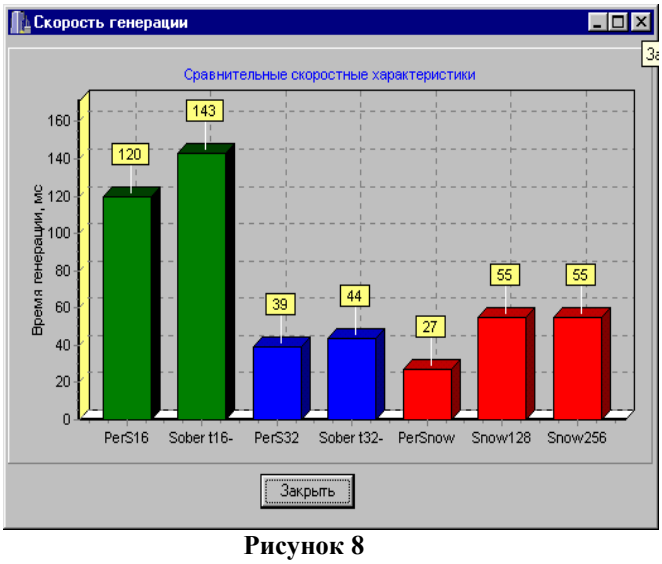

Рисунок 8

Исходные коды программ, реализующих схемы поточных шифров, написаны на языке С и не оптимизированы. Тестирование производилось на ПК Celeron 566, 64 Мб ОЗУ, ОС Windows 98. При рассмотрении быстродействия схем для Sober-t16 и Sober-t32 были реализованы два варианта конструкций – полная, обозначенная как “Sober-t16”, “Sober-t32”, и усеченная, с изъятием из схемы усечения, обозначенная как “Sobert16-”, “Sobert32-”.

Анализируя полученные результаты оценки быстродействия схем, можно отметить, что Sober-t32 в усеченном варианте превосходит SNOW по быстродействию на 20%. Полная же схема Sober-t32 уступает в быстродействии SNOW на 37%. При удалении из конструкции Sober-t32 усечения, быстродействие схемы возрастает практически в два раза. Сравнивая быстродействие регистров и построенных на их основе схем можно сделать вывод, что выполнение нелинейной функции SNOW занимает практически половину всего времени, на протяжении которого генерируется гамма шифрующая; на выполнение нелинейной функции SOBER используется около 10% всего времени.

В таблице 7 представлены данные о скоростных характеристиках рассматриваемых схем на используемой платформе.

Таблица 7

ШУ	Время генерации 1Мбайта гаммы шифрующей, с	Генерация гаммы шифрующей за 1 с, Мбайт
Snow 128	0.0044	227.27
Snow 256	0.0044	227.27
Sober-t32	0.00696	143.68
Sobert32-	0.00352	284.09
Sober-t16	0.02504	39.95
Sobert16-	0.01144	87.41

Анализ данных скорости генерации гаммы шифрующей для алгоритмов SNOW и SOBER показывает, что она удовлетворяет современным требованиям и может составлять единицы Гбит/с.

VI Статистический анализ

Тестирование ШУ производилось в соответствии с методикой, изложенной в [10]. В качестве статистического пакета тестирования использовался пакет NIST STS [13].

В соответствие с данной методикой решение о прохождении ШУ статистического тестирования принимается в случае, если выполняются правила:

- Правило 1. ШУ прошло тестирование по всем q тестам, ($q=\overline{1,189}$), и если значение коэффициента r_j находится внутри доверительного интервала [0.96, 1.00];
- Правило 2. ШУ прошло тестирование по всем q тестам, ($q=\overline{1,189}$), и если для всех тестов по критерию χ^2-Пирсона выполняется условие $P(\chi^2) > 0{,}0001$.

При выполнении тестирования выбраны следующие параметры:

- длина тестируемой последовательности $n = 10^6$ бит;
- количество тестируемых последовательностей $m = 100$. Таким образом, объем тестируемой выборки составил $N = 10^6 \times 100 = 10^8$ бит;
- уровень значимости $\alpha = 0.01$;
- количество тестов $q = 189$.

В таблице 8 приведены данные по прохождению ШУ тестов в соответствии с Правилом 1 (без BMGL).

Таблица 8

ШУ	Количество тестов, у которых тестирование прошли более 99% последовательностей	Количество тестов, у которых тестирование прошли более 96% последовательностей
Snow 128	135 (71,43%)	189 (100%)
Snow 256	141(74,60%)	189 (100%)
Sober-t32	119 (62,96%)	189 (100%)
Sober-t16	132 (69,84%)	188 (99,47%)

ШУ SNOW, Sober-t32 прошли все тесты. Генератор Sober-t16 не прошел один тест на проверку случайных отклонений. Наилучшие результаты показали Snow 128 и Snow 256, имеющие лучшие, в отличие от Sober-t32 и Sober-t16, значения прохождения тестов для обеих категорий испытаний.

В таблице 9 представлены сводные результаты по прохождению ШУ тестов по Правилу 2 (без BMGL).

Для ШУ Snow 128 низкие значения вероятностей P = 0.007, P = 0.003, P=0.009 получены по тесту на не перекрывающиеся шаблоны (r_{92} = 98%, r_{105} = 100%, r_{107} = 100% соответственно).

Для ШУ Sober-t32 низкие значения вероятностей получены:

- по тесту на не перекрывающиеся шаблоны (r_{34} = 99%) — P = 0.002
- по тесту на случайные отклонения (r_{181} = 100%) – P = 0.006

Для ШУ Sober-t16 низкое значение вероятности P = 0.007 получено по тесту на не перекрывающиеся шаблоны (r_{65} = 99%)

Таблица 9

ШУ	Количество тестов, у которых значение вероятности $P \le 0.01$	Количество тестов, у которых значение вероятности $P \le 0.001$
Snow 128	3	0
Snow 256	0	0
Sober-t32	2	0
Sober-t16	1	0

Таким образом, по Правилу 2 в целом тестирование прошли все ШУ. Предпочтительнее остальных выглядит ШУ Snow 256, показавшее хорошие результаты и при более жестких критериях ($P \le 0.01$ и $P \le 0.001$).

Анализ результатов статистического тестирования позволяет сделать вывод о том, что всем регистрам данных, которые лежат в основе схем ШУ, в той или иной степени, присуща некоторая статистическая слабость: ни один из них не сформировал последовательности, которая прошла бы все тесты. Однако анализ результатов тестирования ШУ показал, что они обладают привлекательными статистическими свойствами, поскольку все из них, за исключением ШУ Sober-t16, сформировали последовательности, прошедшие все тесты (Sober-t16 не прошел один тест). На основании этих результатов можно сделать вывод, что нелинейные функции рассматриваемых ШУ, помимо внесения нелинейности в выходные последовательности регистров, обеспечивают хорошие статистические свойства формируемых последовательностей.

В целом по итогам статистического тестирования ШУ можно расположить в следующем порядке предпочтения: Snow 256, Snow 128, Sober-t32 и Sober-t16.

Заключение

Известно, что после первого тура NESSIE в силу очевидных недостатков были отбракованы ШУ Leviathan и Lili-128. Анализ оставшихся поточных шифров позволяет говорить о том, что с точки зрения криптостойкости наиболее предпочтительным выглядит Snow 128, поскольку никакие виды криптоатак со сложностью меньшей, чем исчерпывающий поиск ключа, для него найдены не были. Однако отметим, что отвергнутый NESSIE поточный шифр Sober-t32 также устойчив ко всем видам криптоанализа при условии использовании в его конструкции блока задержки. С точки зрения быстродействия наилучшие результаты при установке ключа показали ШУ Sober-t32 и Sober-t16, а при генерации гаммы шифрующей все 32-разрядные схемы показывают приблизительно одинаковые результаты. Лишь Sober-t16 уступает остальным схемам по быстродействию более чем в два раза.

Таким образом, на основе вышеизложенного можно утверждать, что если в качестве основного критерия отбора поточного шифра выбрать его криптографическую стойкость, свое предпочтение среди 32-разрядных схем надо отдать Sober-t32 в силу того, что:

- в отличие от SNOW, он обладает бóльшим периодом гаммы шифрующей;
- использование 256-битного ключа обеспечивает “высокий” уровень стойкости, в то время как использование 128-битного ключа SNOW – “базовый” (по классификации AES и NESSIE);
- в отличие от SNOW, предусмотрена реинициализация регистра;
- в отличие от SNOW, образующий полином Sober-t32 является более высоковесовым – содержит 272 ненулевых коэффициента из 544 возможных, что делает Sober-t32 более устойчивым к криптоанализу, в то время как образующий полином SNOW содержит 82 ненулевых коэффициента из 512 возможных;
- в силу более традиционного подхода к построению нелинейной функции (выбор точек съема) анализ стойкости Sober-t32 представляется более легким: так, согласно [10], намного более важным, чем выбор функции фильтрации, является именно выбор последовательности точек съема для используемой функции; разработчики SOBER учли необходимые требования, предъявляемые к выбору ячеек съема, в SNOW же разработчики используют в качестве исходных данных, поступающих на вход функции (КА), только одну ячейку.

В случае, если основным критерием отбора поточного шифра выбрать его скоростные характеристики, свое предпочтение среди 32-разрядных схем надо отдать Snow 128 в силу того, что данный шифр обладает, по сравнению с Sober-t32, существенно большим быстродействием и более предпочтительными статистическими свойствами.

Уже в ходе подготовки данной статьи разработчиками SNOW были внесены изменения, которые улучшили его криптографические свойства. С учетом внесенных изменений можно говорить, что схемы SNOW и SOBER по своим криптографическим и другим свойствам практически не уступают друг другу. Анализ улучшенной схемы SNOW будет объектом будущих исследований.

В целом же, анализируя тенденции развития разработок в области поточных шифров, можно сформулировать следующие общие правила построения современных схем поточного шифрования:

1. Использование классической схемы синхронного поточного шифра на основе линейного рекуррентного регистра с последующим применением к выходу функции нелинейной фильтрации.

2. С целью увеличения быстродействия, обеспечения гибкости программной и аппаратной реализации, линейный рекуррентный регистр строится над полем $GF(2^{16})$ или $GF(2^{32})$. Таким образом, современные поточные шифры строятся на основе обобщенного регистра сдвига. В результате можно ввести и обосновать новый класс схем поточного шифрования – ***высокоскоростные машинно-ориентированные схемы поточного шифрования***.

3. Построение функций нелинейной фильтрации не на основе криптографически стойких булевых функций, а с использованием элементов цикловых функций блочных алгоритмов шифрования, а именно *S*-блоков и арифметических операций в полях. Так, в функции нелинейной фильтрации SNOW используются оригинальные таблицы подстановок, а в схемах SOBER функции нелинейной фильтрации строятся на основе *S*-блоков, применяемых в алгоритме Skipjack. Схема BMGL вообще построена на основе алгоритма блочного шифрования Rijndael. Использование цикловых функций блочных шифров, на наш взгляд, связано с тем, что данные функции являются "стандартизированными", хорошо изученными и зарекомендовавшими себя высоконелинейными 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.

4. К алгоритмам поточного шифрования выдвигаются новые требования к длине ключа. Длина ключа должна быть равна 128 или 256 битам. Предложенные схемы поддерживают все длины ключей. За счет перехода на обобщенные регистры сдвига существенно увеличился период гаммы шифрования. Применяемые в схемах SNOW и SOBER регистры эквивалентны регистрам над $GF(2)$ с длиной от 272 до 544 бит. Период гаммы обеспечивается не менее 2^{512} бит для SNOW и от 2^{272} до 2^{544} для Sober-t16 и Sober-t32, соответственно.

В настоящее время специалистами ХНУРЭ и ХВУ ведутся работы по исследованию возможности применения в качестве нелинейных функций булевых функций с операциями над $GF(2^{32})$ ($GF(2^{16})$). На данном этапе можно утверждать, что при равных статистических свойствах последовательностей, генерируемых булевыми функциями и блоками подстановок, схемы ШУ обладают приблизительно одинаковым быстродействием. Вопросы криптографической стойкости булевых функций с операциями над $GF(2^{32})$ ($GF(2^{16})$) находится в стадии исследований.

Литература: *1. FIPS PUB 197:2001. Advanced Encryption Standard (AES). 2. Andrew Rukhin, Juan Soto. A Statistical Test Suite for Random and Pseudorandom Number Generator for Cryptographic Application. NIST Special Publication 800-22, September 2001. 3. P. Ekdahl, T. Johansson. SNOW – a new stream cipher. http://www.cryptonessie.org. 4. P. Hawkes, G.G.Rose. Primitive Specification and Supporting Documentation for SOBER-t32 Submission to NESSIE. http://www.cryptonessie.org. 5. P. Hawkes, G.G. Rose. Primitive Specification and Supporting Documentation for SOBER-t16 Submission to NESSIE. http://www.cryptonessie.org. 6. L.Simpson, E.Dawson, J.Golić, W.Millan. The LILI-128 Keystream Generator. http://www.cryptonessie.org. 7. Johan Håstad, Mats Näslund. BMGL: Synchronous Key-stream Generator with Provable Security. http://www.cryptonessie.org. 8. Marcus Schafheutle. A First Report on the Stream Cipher SNOW. http://www.cryptonessie.org. 9. Marcus Schafheutle. A First Report on the Stream Cipher SOBER-t16 and SOBER-t32. http://www. cryptonessie.org. 9. J.Dj.Golic. On the security of nonlinear filter generators. Proceedings of Fast Software Encryption'96, Lecture Notes in Computer Science, vol. 1039, Springer-Verlag, 1996, pp.173-188. 10 .Juliette White. Initial report on the LILI-128 stream cipher. http://www.cryptonessie.org. 11. Secure evaluation of NESSIE First stage. NES/DOC/RHU/WP3/D13/2, 2001. 12. S. Babbage. Cryptanalysis of the LILI-128 stream cipher. Technical report, NESSIE report, 2001. 13. F. Jönsson, T. Johansson. A fast correlation attack on LILI-128. Technical report, Lund university report, 2001. 14. Потий А.В, Орлова С.Ю, Гриненко Т.А. Статистическое тестирование генераторов случайных и псевдослучайных чисел с использованием набора статистических тестов NIST STS // Правове, нормативне та метрологічне забезпечення захисту інформації в Україні. Науково-технічний збірник. Вип.2, 2001. – С. 206- 213. 15. anonymous. Guess-and-determine attacks on SNOW. In submitted to Crypto '02, 2002. 16. P. Ekdahl and T. Johansson. Distinguishing attacks on SOBER-t16 and t32. In Proceedings of Fast Software Encryption '02, 2002.*