Эссе по тематике потоковых шифров

Костиков Егор Вячеславович

Декабрь 2023

Содержание

1. Введение

В данной работе приводится описание основных потоковых шифров: описываются шифры RC4, Snow3G, LILI128, SEAL, MUGI, а также шифры семейства A5 (поточные шифры A5/1 и A5/2; блочный шифр A5/3 не рассматривается в данном эссе). Также приводится более подробное описание и анализ шифра Snow3G.

2. Теремины и определения

Симметричная криптосистема (симметричные шифр) — способ шифрования, в котором для шифрования и расшифрования применяется один и тот же криптографический ключ.

Потоковый шифр — симметричный шифр, в котором каждый символ открытого текста преобразуется в символ шифрованного текста в зависимости не только от используемого ключа, но и от его расположения в потоке открытого текста. Основан на использовании псевдослучайного генератора.

Псевдослучайный генератор — детерминированный алгоритм, который из данной действительно случайной последовательности бит длины n, называемой начальным состоянием, или seed, вырабатывает двоичную последовательность длины l, которая «похожа» на случайную и называется псевдослучайной последовательностью.

Ключевой поток (ключевая последовательность) — псевдослучайная последовательность, генерируемая в результате работы поточного шифра.

Регистр сдвига с линейной обратной связью — сдвиговый регистр битовых слов, у которого значение входного бита однозначно задается некоторой функцией, исходя из значений остальных битов регистра до сдвига. Степень многочлена является длиной сдвигового регистра.

S-блок — функция в коде программы или аппаратная система, принимающая на входе n бит, преобразующая их по определённому алгоритму и возвращающая на выходе m бит.

Далее в работе используются следующие обозначения:

- $X <<< n$ — сдвиг содержимого X на n бит влево;
- $X >>> n$ — сдвиг содержимого X на n бит вправо;
- $x \parallel y$ — конкатенация строк x и y;
- $=, \longleftarrow$ — операторы присваивания;
- c^i — строка $\underbrace{c...c}_{i}$;
- $x \oplus y$ — побитовое сложение x и y;
- $x \boxplus y$ — сложение x и y по модулю 2^{32}.

3. Описание основных потоковых шифров

3.1. Шифр RC4

Описание шифра RC4 приводится на основе статьи [1].

Шифр RC4 состоит из двух основых компоненетов, которые реализуют алгоритмы KSA (алгоритм выработки ключей) и PRGA (алгоритм генерации псевдослучайной последовательности). Шифр имеет параметр N (обычно равный 256), определяющий размер слова ключевого потока. На вход поступает ключ key, имеющий длину l байт (обычно от 5 до 32 байт). На основе ключа key следующим образом определяется массив K размера N: $K[i] = key[i \bmod l]$, где $0 \leqslant i \leqslant (N-1)$.

- Описание алгоритма KSA:

  $For\ i = 0\ to\ N-1\ do$ {
  $S[i] = i;$
  }
  $j = 0;$
  $For\ i = 0\ to\ N-1\ do$ {
  $j = (j + S[i] + K[i])\ mod\ N;$
  Поменять местами значения $S[i]$ и $S[j]$.
  }

- Описание алгоритма PRGA:

  $i = 0; j = 0;$
  Цикл генерации ключевого потока {
  $i = (i+1)\ mod\ N;$
  $j = (j + S[i])\ mod\ N;$
  Поменять местами значения $S[i]$ и $S[j]$;
  $t = (S[i] + S[j])\ mod\ N;$
  Выдать очередное слово ключевого потока: $z = S[t]$.
  }

3.2. Шифр Snow3G

3.2.1. Описание шифра

Приведем описание шифра Snow3G, следуя статье [2] спецификации данного шифра.

Генератор шифра состоит из регистра сдвига с линейной обратной связью LFSR и конечного автомата FSM, который, в свою очередь, состоит из трех 32-битных регистров, которые будем обозначать R_1, R_2 и R_3.

Регистр LFSR состоит из 16 ячеек памяти s_0, s_1, s_2, ..., s_{15}, в каждой из которых содержится 32 бита.

Введем и опишем используемые в работе шифра функции:

- $MULx$

  Функция принимает два 8-битных параметра V и c.

  Если старший бит V равен 1, то $MULx(V, c) = (V <<< 1) \oplus c$;

  В противном случае: $MULx(V, c) = V <<< 1$.

- $MULxPOW$

  Функция принимает два 8-битных параметра V и c, а также неотрицательное целое значение i.

  Если значение i равно 0, то $MULxPOW(V, i, c) = V$;

  В противном случае: $MULxPOW(V, i, c) = MULx(MULxPOW(V, i - 1, c), c)$.

- MUL_α

  Функция принимает один 8-битный параметр c.

  $MUL_\alpha(c) = MULxPOW(c, 23, 0xA9) || MULxPOW(c, 245, 0xA9) || MULxPOW(c, 48, 0xA9) || MULxPOW(c, 239, 0xA9)$.

- DIV_α

  Функция принимает один 8-битный параметрс c.

  $DIV_\alpha(c) = MULxPOW(c, 16, 0xA9) || MULxPOW(c, 39, 0xA9) || MULxPOW(c, 6, 0xA9) || MULxPOW(c, 64, 0xA9)$.

Кроме того, работа шифра основана на использовании следующих S-блоков:

- S_R и S_Q

  Данные S-блоки по 8-битному числу x ($0 \leqslant x = x_0 \parallel x_1 \leqslant 0xFF$) выдают 8-битное число $y = y_0 \parallel y_1$, которое определяется на основе приведенных ниже таблиц: номер выбираемой строки соответствует числу x_0, номер выбираемого столбца — x_1.

- S-Box S_1 Пусть 32-битное число w представимо в виде $w = w_0 \parallel w_1 \parallel w_2 \parallel w_3$. Тогда $S_1(w) = r_0 \parallel r_1 \parallel r_2 \parallel r_3$, где:
  - $r_0 = MULx(S_R(w_0), 0x1B) \oplus S_R(w_1) \oplus S_R(w_2) \oplus MULx(S_R(w_3), 0x1B) \oplus S_R(w_3)$;
  - $r_1 = MULx(S_R(w_0), 0x1B) \oplus S_R(w_0) \oplus MULx(S_R(w_1), 0x1B) \oplus S_R(w_2) \oplus S_R(w_3)$;
  - $r_2 = S_R(w_0) \oplus MULx(S_R(w_1), 0x1B) \oplus S_R(w_1) \oplus MULx(S_R(w_2), 0x1B) \oplus S_R(w_3)$;
  - $r_3 = S_R(w_0) \oplus S_R(w_1) \oplus MULx(S_R(w_2), 0x1B) \oplus S_R(w_2) \oplus MULx(S_R(w_3), 0x1B)$.
- S-Box S_2 Пусть 32-битное число w представимо в виде $w = w_0 \parallel w_1 \parallel w_2 \parallel w_3$. Тогда $S_2(w) = r_0 \parallel r_1 \parallel r_2 \parallel r_3$, где:
  - $r_0 = MULx(S_Q(w_0), 0x69) \oplus S_Q(w_1) \oplus S_Q(w_2) \oplus MULx(S_Q(w_3), 0x69) \oplus S_Q(w_3)$;
  - $r_1 = MULx(S_Q(w_0), 0x69) \oplus S_Q(w_0) \oplus MULx(S_Q(w_1), 0x69) \oplus S_Q(w_2) \oplus S_Q(w_3)$;
  - $r_2 = S_Q(w_0) \oplus MULx(S_Q(w_1), 0x69) \oplus S_Q(w_1) \oplus MULx(S_Q(w_2), 0x69) \oplus S_Q(w_3)$;
  - $r_3 = S_Q(w_0) \oplus S_Q(w_1) \oplus MULx(S_Q(w_2), 0x69) \oplus S_Q(w_2) \oplus MULx(S_Q(w_3), 0x69)$.

Рассмотрим два основных режима работы шифра Snow3G:

x_1 / x_0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Rijndael S-Box

x_1 / x_0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	25	24	73	67	D7	AE	5C	30	A4	EE	6E	CB	7D	B5	82	DB
1	E4	8E	48	49	4F	5D	6A	78	70	88	E8	5F	5E	84	65	E2
2	D8	E9	CC	ED	40	2F	11	28	57	D2	AC	E3	4A	15	1B	B9
3	B2	80	85	A6	2E	02	47	29	07	4B	0E	C1	51	AA	89	D4
4	CA	01	46	B3	EF	DD	44	7B	C2	7F	BE	C3	9F	20	4C	64
5	83	A2	68	42	13	B4	41	CD	BA	C6	BB	6D	4D	71	21	F4
6	8D	B0	E5	93	FE	8F	E6	CF	43	45	31	22	37	36	96	FA
7	BC	0F	08	52	1D	55	1A	C5	4E	23	69	7A	92	FF	5B	5A
8	EB	9A	1C	A9	D1	7E	0D	FC	50	8A	B6	62	F5	0A	F8	DC
9	03	3C	0C	39	F1	B8	F3	3D	F2	D5	97	66	81	32	A0	00
A	06	CE	F6	EA	B7	17	F7	8C	79	D6	A7	BF	8B	3F	1F	53
B	63	75	35	2C	60	FD	27	D3	94	A5	7C	A1	05	58	2D	BD
C	D9	C7	AF	6B	54	0B	E0	38	04	C8	9D	E7	14	B1	87	9C
D	DF	6F	F9	DA	2A	C4	59	16	74	91	AB	26	61	76	34	2B
E	AD	99	FB	72	EC	33	12	DE	98	3B	C0	9B	3E	18	10	3A
F	56	E1	77	C9	1E	9E	95	A3	90	19	A8	6C	09	D0	F0	86

S-Box S_Q

- Инициализация

  Шифр инициализируется с использованием 128-битного ключа $k = k_0 \parallel k_1 \parallel k_2 \parallel k_3$, состоящего из четырех 32-битных слов k_0, k_1, k_2 и k_3, а также с использованием 128-битного вектора инициализации $IV = IV_0 \parallel IV_1 \parallel IV_2 \parallel IV_3$, также состоящего из четырех 32-битных слов IV_0, IV_1, IV_2 и IV_3. При этом с использованием данных параметров происходит инициализация содержимого ячеек регистра LFSR по следующим правилам:

$s_0 = k_0 \oplus 1^{32}$	$s_1 = k_1 \oplus 1^{32}$	$s_2 = k_2 \oplus 1^{32}$	$s_3 = k_3 \oplus 1^{32}$
$s_4 = k_0$	$s_5 = k_1$	$s_6 = k_2$	$s_7 = k_3$
$s_8 = k_0 \oplus 1^{32}$	$s_9 = k_1 \oplus 1^{32} \oplus IV_3$	$s_{10} = k_2 \oplus 1^{32} \oplus IV_2$	$s_{11} = k_3 \oplus 1^{32}$
$s_{12} = k_0 \oplus IV_1$	$s_{13} = k_1$	$s_{14} = k_2$	$s_{15} = k_3 \oplus IV_0$

Регистры R_1, R_2 и R_3 конечного автомата FSM инициализируются нулями: $R_1 = R_2 = R_3 = 0^{32}$.

Будем считать, что $s_0 = s_{0,0} \| s_{0,1} \| s_{0,2} \| s_{0,3}$ и $s_{11} = s_{11,0} \| s_{11,1} \| s_{11,2} \| s_{11,3}$. Генерации ключевого потока предшествуют следующие действия:

$For\ t = 1\ to\ 32\ do$ {

$F = (s_{15} \boxplus R_1) \oplus R_2;$

$TEMP_1 = R_2 \boxplus (R_3 \oplus s_5);$

$R_3 = S_2(R_2); R_2 = S_1(R_1); R_1 = TEMP_1;$

$TEMP_2 = (s_{0,1} \| s_{0,2} \| s_{0,3} \| 0x00) \oplus MUL_\alpha(s_{0,0}) \oplus s_2 \oplus (0x00 \| s_{11,0} \| s_{11,1} \| s_{11,2}) \oplus DIV_\alpha(s_{11,3}) \oplus F;$

$s_0 = s_1;\ s_1 = s_2;\ s_2 = s_3;\ s_3 = s_4;\ s_4 = s_5;\ s_5 = s_6;\ s_6 = s_7;\ s_7 = s_8;\ s_8 = s_9;$

$s_9 = s_{10};\ s_{10} = s_{11};\ s_{11} = s_{12};\ s_{12} = s_{13};\ s_{13} = s_{14};\ s_{14} = s_{15};\ s_{15} = TEMP_2.$

}

Схема работы шифра при инициализации приведна ниже.

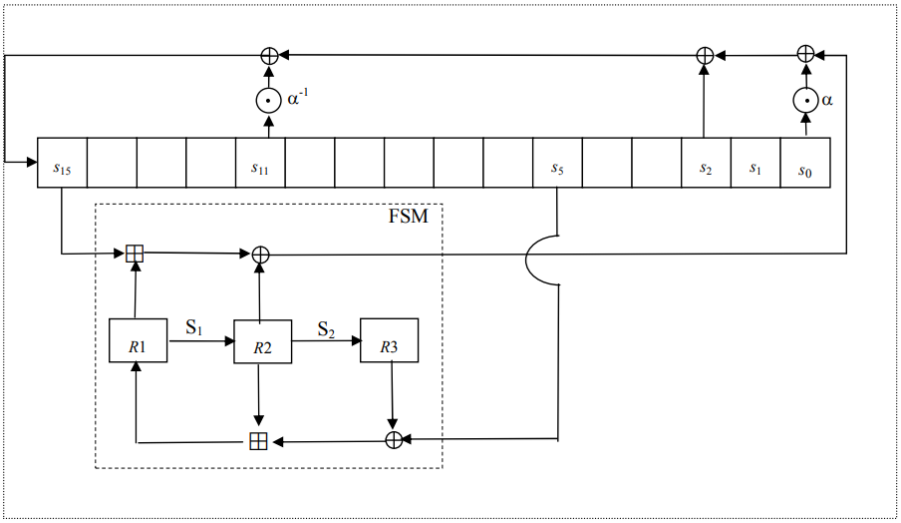


Схема выполнения Snow3G при инициализации

- Генерация ключевого потока

После инициализации для генерации n 32-битных слов ключевого потока будут выполнены следующие действия:

$For\ t = 1\ to\ n\ do$ {

$F = (s_{15} \boxplus R_1) \oplus R_2;$

$TEMP_1 = R_2 \boxplus (R_3 \oplus s_5);$

$R_3 = S_2(R_2); R_2 = S_1(R_1); R_1 = TEMP_1;$

Одно 32-битное слово ключевого потока z_t вычисляется следующим образом:

$z_t = F \oplus s_0;$

$TEMP_2 = (s_{0,1} \| s_{0,2} \| s_{0,3} \| 0x00) \oplus MUL_\alpha(s_{0,0}) \oplus s_2 \oplus (0x00 \| s_{11,0} \| s_{11,1} \| s_{11,2}) \oplus DIV_\alpha(s_{11,3}) \oplus F;$

$s_0 = s_1;\ s_1 = s_2;\ s_2 = s_3;\ s_3 = s_4;\ s_4 = s_5;\ s_5 = s_6;\ s_6 = s_7;\ s_7 = s_8;\ s_8 = s_9;$

$s_9 = s_{10}; s_{10} = s_{11}; s_{11} = s_{12}; s_{12} = s_{13}; s_{13} = s_{14}; s_{14} = s_{15}; s_{15} = TEMP_2.$

}

Схема работы шифра при генерации ключевого потока приведна ниже.

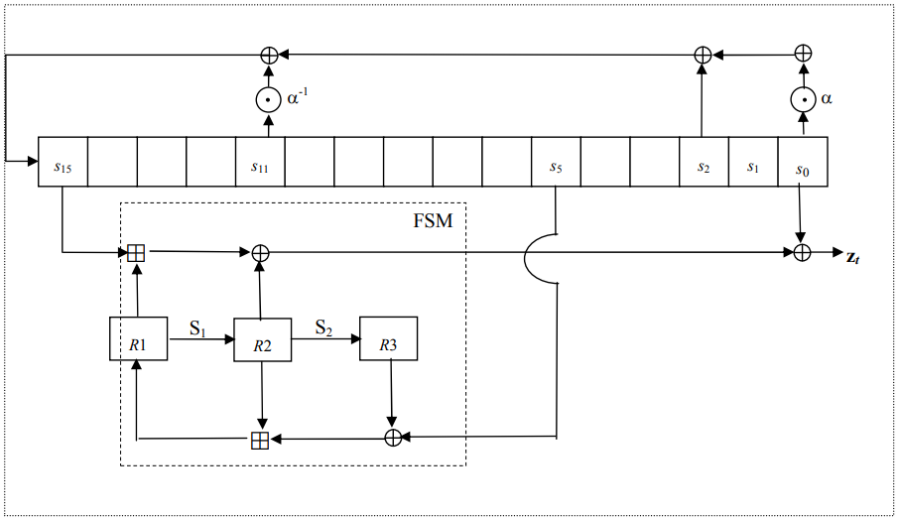


Схема выполнения Snow3G при генерации ключевого потока

3.2.2. Анализ шифра

В статье [3] устанавливаются оценки временной и пространственной сложностей компонентов шифра Snow3G. Справедливы следующие оценки:

- Временная и пространственная сложности функций $MULx$, $MULxPOW$, MUL_α, DIV_α равны **O(1)**;
- Временная и пространственная сложности S-блоков S_1, S_2 равны **O(1)**;
- Пусть n — число 32-битных слов ключевого потока, которые необходимо сгенерировать. Тогда временная сложность алгоритма Snow3G равна **O(n)**;
- Пространственная сложность алгоритма Snow3G равна **O(1)**.

Приведем описание некоторых атак на шифр Snow3G:

1) Описание данной атаки приводится на основе работы [4].

   Для ключа k случайным образом выбирается вектор инициализации $IV = IV_0 \| IV_1 \| IV_2 \| IV_3$, по которому строится множество из 256 наборов, в котором варьируется старший байт IV_0^0 32-битного слова IV_0: он принимает все значения от 0 до 255. Обозначим первое слово ключевого потока, сгенерированного на ключе k и векторе инициализации IV, для которого $IV_0^0 = i, 0 \leqslant i \leqslant 255$, через $z_{i,0}$, а соответствующее содержимое в j-ой ячейке LFSR через $s_{i,j}$. Тогда: $\bigoplus_{i=0}^{255} z_{i,0} = \bigoplus_{i=0}^{255} (s_{i,0} \oplus R_{i,2}) \oplus \bigoplus_{i=0}^{255} (s_{i,15} \boxplus R_{i,1}) = \bigoplus_{i=0}^{255} (s_{i,15} \boxplus R_{i,1})$. Показывается, что младший байт $\bigoplus_{i=0}^{255} z_{i,0}$ всегда равен нулю. Это свойство подтверждается авторами экспериментально: авторами работы было выбрано 2^6 векторов инициализации — в каждом случае младший байт рассматриваемой суммы оказался равен нулю. Данное свойство представляет собой простую различительную атаку сложности 2^8 по времени и по памяти для 13-раундового Snow3G.

2) Описание данной атаки приводится на основе работы [5].

   В данной статье производится сравнение линейных аппроксимаций шифра Snow3G, а также приводится описание атаки на основе использования линейной аппроксимации, направленной на восстановление секретного ключа. Для построения побитовой линейной аппроксимации конечного автомата используются следующие выражения:

- $\Phi \cdot (z_{t-1} \oplus s_{t-1}) = \Phi \cdot (s_{t+14} \boxplus R_1^{t-1}) \oplus \Phi \cdot R_2^{t-1}$;
- $\Gamma \cdot (z_t \oplus s_t) = \Gamma \cdot (s_{t+15} \boxplus R_1^t) \oplus \Gamma \cdot R_2^t$;
- $\Lambda \cdot (z_{t+1} \oplus s_{t+1}) = \Lambda \cdot (s_{t+16} \boxplus R_1^{t+1}) \oplus \Lambda \cdot R_2^{t+1}$, где Φ, Γ, Λ — некоторые 32-битные константы.

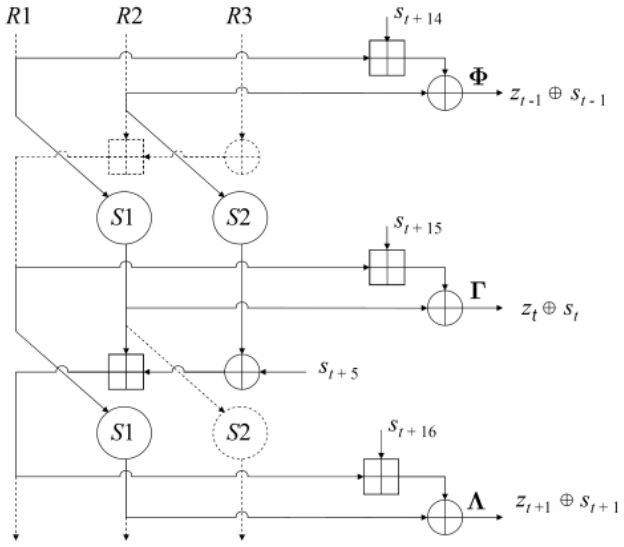


Схема указанной линейной аппроксимации FSM

В результате преобразований система равенств, указанная выше, может быть приведена к виду: $\Phi \cdot z_{t-1} \oplus \Gamma \cdot z_t \oplus \Lambda \cdot z_{t+1} = \Phi \cdot (s_{t-1} \oplus s_{t+14}) \oplus \Gamma \cdot (s_t \oplus s_{t+15}) \oplus \Lambda \cdot (s_{t+1} \oplus s_{t+5} \oplus s_{t+16}) \oplus e^{(t)}$, где значения $e^{(t)}$ определяют двоичные шумы, вносимые побитовыми линейными аппроксимациями. В работе показывается, что наилучшее качество аппроксимации достигается на следующих наборах значений (Λ, Φ, Γ): $\Lambda = 0x1014190f$, $\Phi = 0x00000020$, $\Gamma = 0x00000001$ и $\Lambda = 0x1014190f$, $\Phi = 0x00000030$, $\Gamma = 0x00000001$.

На основе указанного выше представления описывается алгоритм, реализующий атаку, направленную на восстановление секретного ключа:

Параметры алгоритма: N, m_2, $l(= 512)$, l'.

Вход алгоритма: слова $z_0, z_1, ..., z_{D-1}$, где $D = N/2 + 1$, ключевого потока, а также две тройки значений (Φ, Γ, Λ).

Генерально предложенный алгоритм состоит из следующих шагов:

а) Этап предварительной обработки с применением алгоритма k-дерева Вагнера (Wagner's k-tree algorithm);

б) Этап обработки — восстановление l' бит начального состояния LFSR с применением быстрого преобразования Уолша (FWT — Fast Walsh Transform);

в) Восстановление оставшихся $l - l'$ бит начального состояния LFSR;

г) Восстановление секретного ключа путем выполнения действий в порядке, обратном инициализации.

Сложности по времени и по памяти ограничены сверху значением $2^{174.16}$.

3) Описание данной атаки приводится на основе работы [6]. В данной работе описывается, в частности, различительная атака на Snow3G.

Отметим, что преобразования, задаваемые S-блоками S_1 и S_2 могут быть определны следующим образом: $S_1(W) = L_1 S_R(W)$, $S_2(W) = L_2 S_Q(W)$, где L_1, L_2 определяются на основе таблиц S-блоков и являются невырожденными.
$z^{t-1} = (S_R^{-1}(L_1^{-1} R_2^t) \boxplus s_{15}^{t-1}) \oplus S_Q^{-1}(L_2^{-1} R_3^t) \boxplus s_0^{t-1}$;

Рассматривается следующая схема аппроксимации (через $X[0]$ обозначен нулевой байт 32-битного слова X):

- $z^{t-1}[0] = n_0 \oplus (s_{15}^{t-1} \oplus s_0^{t-1})[0]$;

- $z^t[0] = n_1 \oplus (s_{15}^t \oplus s_0^t)[0]$;
- $L_1^{-1} z^{t+1}[0] = n_2 \oplus (L_1^{-1} s_5^t \oplus L_1^{-1} s_{15}^{t+1} \oplus L_1^{-1} s_0^{t+1})[0]$.

Убирая вклад LFSR, получаем: $S^t = ((s_{15}^{t-1} \oplus s_0^{t-1})[0], (s_{15}^t \oplus s_0^t)[0], (L_1^{-1} s_{15}^{t+1} \oplus L_1^{-1} s_0^{t+1})[0])$.

Авторами доказывается тот факт, что если существуют такие t_1, t_2, t_3, что $s_0^0 \oplus s_0^{t_1} \oplus s_0^{t_2} \oplus s_0^{t_3} = 0$, то $S^t \oplus S^{t+t_1} \oplus S^{t+t_2} \oplus S^{t+t_3} = (0, 0, 0)$.

В предположении, что такие t_1, t_2, t_3 найдены, можно составить следующие выборки:
$x^t = \sum_{i=0}^{3} (z^{t+t_i-1}[0], z^{t+t_i}[0], L_1^{-1} z^{t+t_i+1}[0])$, где $t_0 = 0$, которые используются для различения последовательности от истинно случайной. Для решения задачи нахождения необходимых t_1, t_2, t_3 используется представленный в статье [7] алгоритм.

Авторами показано, что требуется порядка 2^{163} символов ключевого потока, чтобы отличить выборки от истинно случайных.

Сложности по времени и по памяти равны примерно 2^{172}.

В работе [8] описываются результаты статистического тестирования шифра Snow3G с использованием набора статистических тестов NIST, в который входят следующие тесты: частотный побитовый тест, частотный блочный тест, тест на последовательность одинаковых битов, тест на самую длинную серию единиц в блоке, тест ранга двоичной матрицы, тест дискретного преобразования Фурье (спектральный), тест на совпадение неперекрывающихся шаблонов, тест на совпадение перекрывающихся шаблонов, универсальный статистический тест Маурера, тест на линейную сложность, тест на периодичность, тест приблизительной энтропии, тест кумулятивных сумм, тест случайного отклонения, другой тест случайного отклонения. Указанные тесты были применены для оценки случайности длинного ключевого потока (для каждого из 300 случайно выбранных ключей генерировалось 2^{20} бит ключевого потока, IV нулевой), для оценки случайности короткого ключевого потока (для каждого из 307200 случайно выбранных ключей генерировалось 2^{10} бит ключевого потока, IV нулевой), для оценки корреляции между различными значениями IV (генерируются 300 наборов данных из случайно выбранных ключей; каждый набор представляет собой объединение 256 последовательностей длиной 4096 бит каждая; первая последовательность генерируется с нулевым IV, каждая последующая — с увеличивающимся на 1 значением IV).

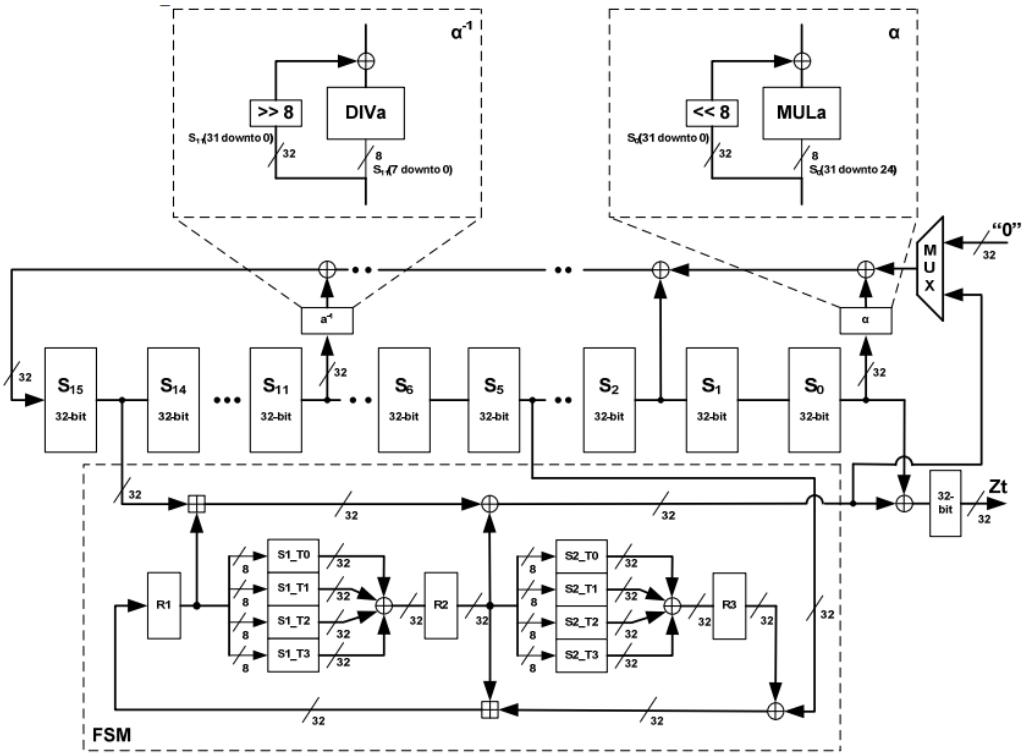
Шифром были успешно пройдены все тесты оценки случайности длинного ключевого потока и оценки корреляции между различными значениями IV — тесты не выявили отклонений от истинно случайных последовательностей. При этом для набора данных для оценки случайности короткого ключевого потока шифром не было пройдено 8 тестов: тест на последовательность одинаковых битов, тест на самую длинную серию единиц в блоке, тест ранга двоичной матрицы, тест дискретного преобразования Фурье, тест на совпадение неперекрывающихся шаблонов, тест на совпадение перекрывающихся шаблонов, тест приблизительной энтропии, тест на периодичность — на тестах допускается возможность отличить первые 1024 бита ключевого потока от истинно случайной последовательности.

В работе [9] исследуются различные подходы к программной реализации шифра Snow3G на двух основных платформах мобильных телефонов. Авторами работы были подготовлены реализации шифра для платформы iOS с использованием языка программирования Objective-C, а также для платформы Android с использованием языка программирования Java.

Traditional		HardCode		Circular Buffers		Sliding Windows		Loop Unrolling	
Time (ms)		Time (ms)		Time (ms)		Time (ms)		Time (ms)	
iOS	Android	iOS	Android	iOS	Android	iOS	Android	iOS	Android
491.1	1772.05	342.5	1513.97	834	1057.719	184.1	943.569	291.3	1697.843

В данной работе рассматриваются различные методы реализации сдвига LFSR для минимизации времени работы щифра. Для каждого метода производились замеры времени генерации 10^7 байт ключевой последовательности в целом, а также замеры отводимого при этом времени на сдвиги содержимого LFSR. Результаты тестирования приведены в таблице ниже. Лучшей по времени выполнения оказалась реализация LFSR с использованием метода скользящих окон (Sliding Windows). В данной реализации размер регистра LFSR увеличивается в 2 раза, его содержимое дублируется во второй половине. Используется один указатель, изначально указывающий на начало второй половины регистра; при достижении конца регистра он возвращается в исходное положение. Вычисляемое новое значение записывается в первой позиции окна и в позиции, на которую указывает указатель. После этого окно (указатель) сдвигается на одну позицию.

В статье [10] описывается эффективная аппаратная реализация шифра Snow3G с использованием ASIC.



Предлагаемая авторами [10] подробная архитектура аппаратной реализации Snow3G

Для вычисления значений S-блоков предлагается использовать специальные таблицы $S1\_T0, S1\_T1, S1\_T2, S1\_T3$ и $S2\_T0, S2\_T1, S2\_T2, S2\_T3$: для $w = w_0 \parallel w_1 \parallel w_2 \parallel w_3$ $S_1(w) = S1\_T0(w_3) \oplus S1\_T1(w_2) \oplus S1\_T2(w_1) \oplus S1\_T3(w_0)$, $S_2(w) = S2\_T0(w_3) \oplus S2\_T1(w_2) \oplus S2\_T2(w_1) \oplus S2\_T3(w_0)$. Через мультиплексор (MUX) происходит выбор режима работы Snow3G: инициализация или генерация ключевой последовательности. Операция сложения по модулю 2^{32} реализована с помощью сумматоров с ускоренным переносом. Функции MUL_α и DIV_α реализованы с помощью некоторых таблиц MUL_{alpha} и DIV_{alpha} соответственно.

Аппаратная реализация состояла из 25016 вентилей, критический путь составил 4.03 нс. На тестах было показано, что предложенная авторами реализация позволяет достичь максимальной частоты в 249 МГц, и максимального значения пропускной способности в 7968 Мбит/с, хоть и является менее компактной по сравнению с другими аппаратными реализациями других поточных шифров.

3.3. Шифр LILI128

Описание шифра LILI128 приводится в соответствии с описанием, представленным в работе [11] разработчиков данного шифра.

Генератор состоит из двух регистров сдвига с линейной обратной связью $LFSR_c$ и $LFSR_d$, состоящих из 39 и 89 битовых ячеек памяти соответственно. Кроме того, для генерации ключевого потока используются две функции: f_c и f_d, которые описываются далее. Функционально генератор может быть разделен на две подсистемы: подсистема управления тактами (Clock-Control Subsystem) и подсистема генерации данных (Data Generation Subsystem).

При инициализации генератора используется 128-битный ключ, первыми 39 битами которого инициализируется регистр $LFSR_c$, а оставшимися 89 — регистр $LFSR_d$.

Характеристики подсистемы управления тактами:

- Используется для генерации псевдослучайной последовательности $c = \{c(t)\}_{t=1}^{\infty}$. В момент времени t значением $c(t)$ определяется количество тактов, которое необходимо сделать регистру $LFSR_d$ подсистемы генерации данных перед генерацией очередного бита $z(t)$ ключевого потока;
- Полином обратной связи для регистра $LFSR_c$ имеет следующий вид: $x^{39}+x^{35}+x^{33}+x^{31}+x^{17}+x^{15}+x^{14}+x^2+x+1$;
- Функция $f_c : \mathbb{F}_2^2 \longrightarrow \{1, 2, 3, 4\}$ зависит от двух параметров и имеет следующий вид: $f(x_{12}, x_{20}) = 2x_{12} + x_{20} + 1$.

Характеристики подсистемы генерации данных:

- Используется для генерации псевдослучайной последовательности $z = \{z(t)\}_{t=1}^{\infty}$, составляющией ключевой поток;
- Полином обратной связи для регистра $LFSR_d$ имеет следующий вид: $x^{89} + x^{83} + x^{80} + x^{55} + x^{53} + x^{42} + x^{39} + x + 1$;
- Функция $f_d : \mathbb{F}_2^{10} \to \mathbb{F}_2$ задается своей таблицей истинности. Вектор значений функции длины 1024 задается следующим образом:

0,0,1,1,1,1,0,0,1,1,0,0,0,0,1,1,1,1,0,0,0,0,1,1,0,0,1,1,1,1,0,0,
0,0,1,1,1,1,0,0,1,1,0,0,0,0,1,1,1,1,0,0,0,0,1,1,0,0,1,1,1,1,0,0,
0,0,1,1,1,1,0,0,1,1,0,0,0,0,1,1,1,1,0,0,0,0,1,1,0,0,1,1,1,1,0,0,
1,1,0,0,0,0,1,1,0,0,1,1,1,1,0,0,0,0,1,1,1,1,0,0,1,1,0,0,0,0,1,1,
0,1,0,1,1,0,1,0,1,0,1,0,0,1,0,1,1,0,1,0,0,1,0,1,0,1,0,1,1,0,1,0,
0,1,0,1,1,0,1,0,1,0,1,0,0,1,0,1,1,0,1,0,0,1,0,1,0,1,0,1,1,0,1,0,
0,1,0,1,1,0,1,0,1,0,1,0,0,1,0,1,1,0,1,0,0,1,0,1,0,1,0,1,1,0,1,0,
1,0,1,0,0,1,0,1,0,1,1,0,1,0,0,1,0,1,1,0,1,0,1,0,1,0,0,1,0,1,
0,1,1,0,0,1,1,0,1,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,0,1,1,0,0,1,1,0,
0,1,1,0,0,1,1,0,1,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,0,1,1,0,0,1,1,0,
0,1,1,0,1,0,0,1,0,1,1,0,1,0,0,1,1,0,0,1,0,1,1,0,1,0,0,1,0,1,1,0,
0,1,1,0,1,0,0,1,1,0,0,1,0,1,1,0,0,1,1,0,1,0,0,1,1,0,0,1,0,1,1,0,
0,0,0,0,1,1,1,1,1,1,1,1,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,1,1,1,1,
1,1,1,1,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,1,1,1,1,1,1,1,1,0,0,0,0,
0,0,1,1,0,0,1,1,1,1,0,0,1,1,0,0,1,1,0,0,1,1,0,0,0,0,1,1,0,0,1,1,
1,1,0,0,1,1,0,0,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,1,1,1,0,0,1,1,0,0,
0,0,1,1,1,1,0,0,0,0,0,1,1,1,1,0,0,1,1,0,0,0,0,0,1,1,1,1,0,0,0,0,1,1,
1,1,0,0,0,0,0,1,1,1,1,0,0,0,0,1,1,0,0,1,1,1,1,1,0,0,0,0,1,1,1,1,0,0,
0,0,1,1,1,1,0,0,1,1,0,0,0,0,1,1,0,0,1,1,1,1,0,0,1,1,0,0,0,0,1,1,
1,1,0,0,0,0,1,1,1,1,0,0,1,1,0,0,0,0,1,1,0,0,1,1,1,1,0,0,1,1,0,0,
0,1,0,1,0,1,0,1,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,0,1,0,1,0,1,0,1,
1,0,1,0,1,0,1,0,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,1,0,1,0,1,0,1,0,
0,1,0,1,1,0,1,0,0,1,0,1,1,0,1,0,1,0,1,0,0,1,0,1,1,0,1,0,0,1,0,1,
1,0,1,0,0,1,0,1,1,0,1,0,0,1,0,1,0,1,1,0,1,0,0,1,0,1,1,0,1,0,1,0,
0,1,0,1,1,0,1,0,1,0,0,1,0,1,0,1,1,0,1,0,1,0,1,0,0,1,0,1,0,1,0,1,
1,0,1,0,0,1,0,1,0,1,1,0,1,0,1,0,1,0,0,1,0,1,0,1,1,0,1,0,1,0,1,0,
0,1,1,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,1,0,
1,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,0,1,1,0,0,1,1,0,0,1,1,0,0,1,1,0,
0,1,1,0,0,1,1,0,1,0,0,1,1,0,0,1,0,1,1,0,0,1,1,0,1,0,0,1,1,0,0,1,
1,0,0,1,1,0,0,1,0,1,1,0,0,1,1,0,1,0,0,1,1,0,0,1,0,1,1,0,0,1,1,0,
0,1,1,0,1,0,0,1,0,1,1,0,1,0,0,1,0,1,1,0,1,0,0,1,0,1,1,0,1,0,0,1,
1,0,0,1,0,1,1,0,1,0,0,1,0,1,1,0,1,0,0,1,0,1,1,0,1,0,0,1,0,1,1,0

Генерация очередного бита $z(t)$ ключевого потока происходит по следущей процедуре:

1) На вход функции f_d подается $n = 10$ битов регистра $LFSR_d$ — биты под номерами 0, 1, 3, 7, 12, 20, 30, 44, 65, 80, в результате чего функцией f_d возвращается бит $z(t)$ ключевого потока.
2) На вход функции f_c подается $k = 2$ бита регистра $LFSR_c$ — биты под номерами 12 и 20, в результате чего функцией f_c возвращается величина количиства сдвигов $c(t)$.
3) Происходит один сдвиг содержимого регистра $LFSR_c$ и $c(t)$ (от 1 до 4) сдвигов содержимого регистра $LFSR_d$.

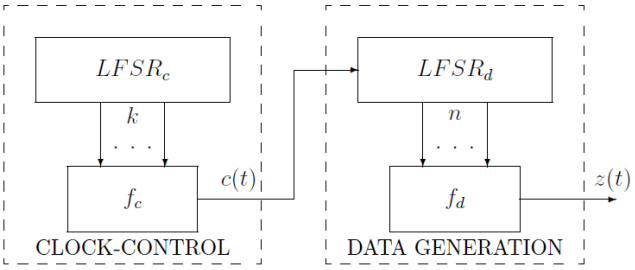


Схема генератора $LILI128$

3.4. Шифр SEAL

Приведем определение шифра SEAL согласно статье [12] авторов данного шифра.

В шифре используются 160-битный ключ a, 32-битная строка n и длину генерируемой последовательности L. Выполнение шифра может быть разделено на две стадии: подготовка необходимых таблиц и непосредственно генерация ключевой последовательности.

Стадия 1 — подготовка вспомогательных таблиц. При шифровании используются значения трех таблиц: R, S и T, содержимое которых определяется значением ключа a. Для заполнения данных таблиц предназначена функция Γ.

Функция Γ определяется через функцию G, которая для 160-битной строки a и 32-битного числа i возвращает 160-битное значение $G_a(i)$.

Для описания функции G необходимо определить при различных целочисленных значениях $0 \leqslant t \leqslant 79$ шестнадцатиричные константы K_t и булевы функции f_t следующим образом:

- Для $0 \leqslant t \leqslant 19$: $K_t = 0x5A827999$ и $f_t(B,C,D) = (B\&C) \vee (\bar{B}\&D)$;
- Для $20 \leqslant t \leqslant 39$: $K_t = 0x6ED9EBA1$ и $f_t(B,C,D) = B \oplus C \oplus D$;
- Для $40 \leqslant t \leqslant 59$: $K_t = 0x8F1BBCDC$ и $f_t(B,C,D) = (B\&C) \vee (B\&D) \vee (C\&D)$;
- Для $60 \leqslant t \leqslant 79$: $K_t = 0xCA62C1D6$ и $f_t(B,C,D) = B \oplus C \oplus D$.

Будем рассматривать входную строку a как последовательность из пяти 32-битных слов: $a = H_0 \,||\, H_1 \,||\, H_2 \,||\, H_3 \,||\, H_4$. Также определим 512-битное слово $M = i \,||\, 0^{480}$, которое разобьем на 16 последовательных 32-битных слов: $M_0 = i$, $M_1 = M_2 = \ldots = M_{15} = 0$. После этого выполняется следующий алгоритм, в результате которого вычисляется значение $G_a(i)$:

1) *For* $t = 16$ *to* 79 *do* {
   $W_t = (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) <<< 1$;
   }
2) Определим $A = H_0, B = H_1, C = H_2, D = H_3, E = H_4$;
3) *For* $t = 16$ *to* 79 *do* {
   $TEMP = (A <<< 5) + f_t(B,C,D) + E + W_t + K_t$;
   $E = D; D = C; C = B <<< 30; B = A; A = TEMP$;
   }
4) $H_0 = H_0 + A; H_1 = H_1 + B; H_2 = H_2 + C; H_3 = H_3 + D; H_4 = H_4 + E$;
5) $G_a(i) = H_0 \,||\, H_1 \,||\, H_2 \,||\, H_3 \,||\, H_4$.

Введем функцию Γ: $\Gamma_a(i) = H^i_{i \bmod 5}$, где $H_0^{5j} || H_1^{5j+1} || H_2^{5j+2} || H_3^{5j+3} || H_4^{5j+4} = G_a(j)$, $j = \lfloor i/5 \rfloor$. Значения, возвращаемые функцией Γ, используются для формирования таблиц R, S, T по следующим правилам:

- $T[i] = \Gamma_a(i)$, где $0 \leqslant i < 512$;
- $S[j] = \Gamma_a(0x1000 + j)$, где $0 \leqslant j < 256$;

- $R[k] = \Gamma_a(0x2000 + k)$, где $0 \leqslant k < 256$.

Кроме того, для генерации ключевого потока необходима функция инициализации $Initialize$. На вход данной функции подаются параметр n, являющийся входным параметром данного шифра, некоторое число l, четыре регистра A, B, C, D и четыре 32-битных слова n_1, n_2, n_3, n_4. Тело функции имеет следующий вид:

procedure Initialize$(n, \ell,\ A, B, C, D,\ n_1, n_2, n_3, n_4)$

$A \leftarrow n \oplus R[4\ell];$
$B \leftarrow (n \ggg 8) \oplus R[4\ell + 1];$
$C \leftarrow (n \ggg 16) \oplus R[4\ell + 2];$
$D \leftarrow (n \ggg 24) \oplus R[4\ell + 3];$

for $j \leftarrow 1$ **to** 2 **do**
$P \leftarrow A$ & 0x7fc; $B \leftarrow B + T[P/4];\ A \leftarrow A \ggg 9;$
$P \leftarrow B$ & 0x7fc; $C \leftarrow C + T[P/4];\ B \leftarrow B \ggg 9;$
$P \leftarrow C$ & 0x7fc; $D \leftarrow D + T[P/4];\ C \leftarrow C \ggg 9;$
$P \leftarrow D$ & 0x7fc; $A \leftarrow A + T[P/4];\ D \leftarrow D \ggg 9;$

$(n_1,\ n_2,\ n_3,\ n_4) \leftarrow (D,\ B,\ A,\ C);$

$P \leftarrow A$ & 0x7fc; $B \leftarrow B + T[P/4];\ A \leftarrow A \ggg 9;$
$P \leftarrow B$ & 0x7fc; $C \leftarrow C + T[P/4];\ B \leftarrow B \ggg 9;$
$P \leftarrow C$ & 0x7fc; $D \leftarrow D + T[P/4];\ C \leftarrow C \ggg 9;$
$P \leftarrow D$ & 0x7fc; $A \leftarrow A + T[P/4];\ D \leftarrow D \ggg 9;$

Функция $Initialize$

В результате работы функции $Initialize$, используя значения таблиц R и T, инициализируются значениями регистры A, B, C и D, а также переменные n_1, n_2, n_3 и n_4.

Стадия 2 — генерация ключевого потока. Для генерации непосредственно ключевого потока используется псевдослучайная функция $SEAL$. На вход данной функции поступают 160-битный ключ a, 32-битная строка n и значение длины генерируемой последовательности L. Значение L может быть настолько большим или настолько малым, насколько это необходимо для целевого приложения, но ожидаемая длина выходных данных составляет от нескольких байт до нескольких тысяч байт.

На каждом новом шаге для генерации ключевой последовательности функцией $SEAL$ вызывается функция $Initialize$ со значением l, равным номеру теущего шага. Генерация функцией $SEAL$ ключевого потока на очередной итерации происходит следующим образом:

1) На очередной итерации с использованием содержимого одного из регистров A, B, C, D (9 старших битов) выбирается значение таблицы T. Выбранное значение складывается с находящимся в регистре B, C, D, A значением соответственно;

2) Содержимое регистра, использовавшегося для получения значения талицы T, сдвигается циклически вправо на 9 позиций;

3) Содержимое регистра, с содержимым которого складывалось значение таблицы T, складывается с содержимым сдвинутого на шаге 2 регистра, либо не изменяется;

4) Значения регистров A, B, C, D складываются со значениями таблицы S, определяемыми номером текущей итерации. Полученное значение добавляется в ключевую последовательность;

5) Если длина сгенерированной к данному моменту ключевой последовательности превышает заданное значение L, то функция возвращает элементы 0, ..., $(L - 1)$ ключевой последовательности;

6) К содержимому регистров A, B, C, D прибавляются значения n_1, n_2, n_3, n_4.

Для очередного значения l может быть выполнено 64 итерации генерации элементов ключевой последовательности. Если за 64 итерации длина сгенерированной ключевой последовательности меньше L, то происходит переход к следующему шагу: увеличивается знаечение l.

function SEAL(a, n, L)

$y = \lambda$;

for $\ell \leftarrow 0$ **to** ∞ **do**

Initialize$(n, \ell,\ A, B, C, D,\ n_1, n_2, n_3, n_4)$;

for $i \leftarrow 1$ **to** 64 **do**

$P \leftarrow A$ & 0x7fc; $B \leftarrow B + T[P/4];\ A \leftarrow A \rangle\rangle\rangle\ 9;\ B \leftarrow B \oplus A$;
$Q \leftarrow B$ & 0x7fc; $C \leftarrow C \oplus T[Q/4];\ B \leftarrow B \rangle\rangle\rangle\ 9;\ C \leftarrow C + B$;
$P \leftarrow (P + C)$ & 0x7fc; $D \leftarrow D + T[P/4];\ C \leftarrow C \rangle\rangle\rangle\ 9;\ D \leftarrow D \oplus C$;
$Q \leftarrow (Q + D)$ & 0x7fc; $A \leftarrow A \oplus T[Q/4];\ D \leftarrow D \rangle\rangle\rangle\ 9;\ A \leftarrow A + D$;

$P \leftarrow (P + A)$ & 0x7fc; $B \leftarrow B \oplus T[P/4];\ A \leftarrow A \rangle\rangle\rangle\ 9$;
$Q \leftarrow (Q + B)$ & 0x7fc; $C \leftarrow C + T[Q/4];\ B \leftarrow B \rangle\rangle\rangle\ 9$;
$P \leftarrow (P + C)$ & 0x7fc; $D \leftarrow D \oplus T[P/4];\ C \leftarrow C \rangle\rangle\rangle\ 9$;
$Q \leftarrow (Q + D)$ & 0x7fc; $A \leftarrow A + T[Q/4];\ D \leftarrow D \rangle\rangle\rangle\ 9$;

$y \leftarrow y \ \parallel\ B + S[4i-4] \ \parallel\ C \oplus S[4i-3] \ \parallel\ D + S[4i-2] \ \parallel\ A \oplus S[4i-1]$;

if $|y| \geq L$ **then return** $y_0 y_1 \ldots y_{L-1}$;

if $odd(i)$ **then** $(A,\ B,\ C,\ D) \leftarrow (A + n_1,\ B + n_2,\ C \oplus n_1,\ D \oplus n_2)$
else $(A,\ B,\ C,\ D) \leftarrow (A + n_3,\ B + n_4,\ C \oplus n_3,\ D \oplus n_4)$;

Функция *SEAL*

3.5. Шифр MUGI

Далее приводится описание шифра MUGI, следуя статье [13] разработчиков данного шифра.

На вход шифру подается 128-битный секретный ключ $K = K_0 \parallel K_1$, старшие 64 бита которого обозначим через K_0, а оставшиеся — через K_1, и 128-битный публичный вектор инициализации $IV = IV_0 \parallel IV_1$, старшие 64 бита которого обозначим через IV_0, а оставшиеся — через IV_1.

Для функционирования MUGI определяется внутреннее состояние, включающее в себя:

- состояние a: три 64-битных регистра состояния (a_0, a_1, a_2);
- буфер b: шестнадцать 64-битных буферных регистра $(b_0, b_1, ..., b_{15})$.

В каждый момент времени t внутреннее состояние содержит некоторые значения: $a^{(t)} = (a_0^{(t)}, a_1^{(t)}, a_2^{(t)})$ и $b^{(t)} = (b_0^{(t)}, b_1^{(t)}, ..., b_{15}^{(t)})$. Для обновления состояния a предназначена функция ρ, для обновления буфера b — функция λ. Таким образом, обновление внутреннего состояния описывается следующим образом: $(a^{(t+1)}, b^{(t+1)}) = (\rho(a^{(t)}, b^{(t)}), \lambda(a^{(t)}, b^{(t)})) = \Upsilon(a^{(t)}, b^{(t)})$.

Выполняемые функцией ρ действия при вызове $\rho(a^{(t)}, b^{(t)})$:

- $a_0^{(t+1)} = a_1^{(t)}$;
- $a_1^{(t+1)} = a_2^{(t)} \oplus F(a_1^{(t)}, b_4^{(t)}) \oplus C_1$, где $C_1 = 0xBB67AE8584CAA73B$;
- $a_2^{(t+1)} = a_0^{(t)} \oplus F(a_1^{(t)}, b_{10}^{(t)} <<< 17) \oplus C_2$, где $C_2 = 0x3C6EF372FE94F82B$.

Выполняемые функцией λ действия при вызове $\lambda(a^{(t)}, b^{(t)})$:

- $b_j^{(t+1)} = b_{j-1}^{(t)}$, где $j \neq 0, 4, 10$;
- $b_0^{(t+1)} = b_{15}^{(t)} + a_0^{(t)}$;
- $b_4^{(t+1)} = b_3^{(t)} + b_7^{(t)}$;
- $b_{10}^{(t+1)} = b_9^{(t)} \oplus (b_{13}^{(t)} <<< 32)$.

Опишем выполняемые при инициализации действия (через t_0 обозначен момент начала инициализации):

Инициализация содержимого состояния a: $a_0^{(t_0)} = K_0$, $a_1^{(t_0)} = K_1$, $a_2^{(t_0)} = (K_0 <<< 7) \oplus (K_1 >>> 7) \oplus C_0$, где $C_0 = 0x6A09E667F3BCC908$.

После этого происходит инициализации содержимого буфера b с использованием значений регистра a_0: $b_{15-i} = (\rho^{(i+1)}(a^{(t_0)}, 0))_0$, где $0 \leqslant i \leqslant 15$.

Введем обозначения: $a(K) = \rho^{16}(a^{(t_0)}, 0)$, $a(K, IV)_0 = a(K)_0 \oplus IV_0$, $a(K, IV)_1 = a(K)_1 \oplus IV_1$, $a(K, IV)_2 = a(K)_2 \oplus (IV_0 <<< 7) \oplus (IV_1 >>> 7) \oplus C_0$. Затем ещё раз происходит обновление состояния a вызовом $\rho^{16}(a(K, IV), 0)$.

Завершающий этап инициализации: $a^{(1)} = \Upsilon^{16}(\rho^{16}(a(K, IV), 0), b(K))$, где через $b(K)$ обозначен инициализированный ранее секретным ключом K буфер b. После этого содержимое внутреннего состояния определяется в результате выполнения описанной ранее функции Υ.

Сгенерированным в момент времени t словом ключевого потока считается 64-битное значение регистра a_2: $z_t = a_2^{(t)}$.

3.6. Шифры семейства A5

Описания алгоритмов не были опубликованы консорциумом GSM. Описания шифров A5/1, A5/2 и A5/3 приводятся согласно работам [14], [15].

3.6.1. Шифр A5/1

Шифр A5/1 состоит из трех регистров сдвига с линейной обратной связью:

1) 19-битный регистр R_1, полином обратной связи которого имеет следующий вид: $x^{19} + x^5 + x^2 + x + 1$;
2) 22-битный регистр R_2, полином обратной связи которого имеет следующий вид: $x^{22} + x + 1$;
3) 23-битный регистр R_3, полином обратной связи которого имеет следующий вид: $x^{23} + x^{15} + x^2 + x + 1$.

Кроме того, имеется механизм управления тактированием: для управления ... в каждом регистре выбрано по одному специальному биту, называемому битом синхронизации: для регистра $R1$ — 8, для регистров R_2 и R_3 — 10. С использованием значений указанных битов вычисляется значение функции $f(x, y, z) = (x\&y) \vee (x\&z) \vee (y\&z)$, где x, y, z — значения битов синхронизации регистров R_1, R_2, R_3 соответственно. После вычисления значения $f(x, y, z)$ происходит сдвиг только тех регистров, бит синхронизации которых равен $f(x, y, z)$.

Инициализация. На вход функции инициализации передются секретный 64-битный ключ Key и 22-битный номер фрейма $Frame$ (в GSM системах в качестве единицы передаваемой информации используется фрейм, состоящий из 228 бит). Процедура инициализации состоит из следующих шагов:

1) $R_1 = 0^{19}, R_2 = 0^{22}, R_3 = 0^{23}$;
2) $For\ i\ =\ 0\ to\ 63\ do$ {

   Сложение по модулю 2 младших битов (битов на нулевой позиции) регистров R_1, R_2 и R_3 со значением $Key[i]$;

   Сдвиг содержимого всех трех регистров;

   }
3) $For\ i\ =\ 0\ to\ 21\ do$ {

   Сложение по модулю 2 младших битов (битов на нулевой позиции) регистров R_1, R_2 и R_3 со значением $Frame[i]$;

   Сдвиг содержимого всех трех регистров;

   }
4) $For\ i\ =\ 0\ to\ 99\ do$ {

   Сдвиги содержимого регистров R_1, R_2, R_3 с использованием механизма управления тактированием.

   }

Генерация ключевой последовательности. Элементом ключевой последовательности после выполнения очередного такта является сумма по модулю два выходных битов регистров. После выполнения функции инициализации выполняется 228 тактов работы генератора, в результате чего формируется ключевая последовательность. Затем номер фрейма увеличивается на единицу, повторяются процессы инициализации и генерации ключевого потока.

3.6.2. Шифр A5/2

Шифр A5/2 также состоит из описанных ранее регистров R_1, R_2, R_3 и дополнительного 17-битного регистра сдвига с линейной обратной связью R_4, полином обратной связи которого имеет вид: $x^{17} + x^5 + 1$. Также присутствует механизм тактирования, главной частью которого является регистр R_4: для сдвига регистров вычисляется функция $f(x, y, z) = (x\&y) \vee (x\&z) \vee (y\&z)$ от 3, 7 и 10 битов регистра R_4, после чего:

- регистр R_1 сдвигается, если значение 10 бита R_4 равно $f(x, y, z)$;
- регистр R_2 сдвигается, если значение 3 бита R_4 равно $f(x, y, z)$;
- регистр R_3 сдвигается, если значение 7 бита R_4 равно $f(x, y, z)$.

Инициализация. На вход функции инициализации передются секретный 64-битный ключ Key и 22-битный номер фрейма $Frame$ (в GSM системах в качестве единицы передаваемой информации используется фрейм, состоящий из 228 бит). Процедура инициализации состоит из следующих шагов:

1) $R_1 = 0^{19}, R_2 = 0^{22}, R_3 = 0^{23}, R_4 = 0^{17}$;
2) $For\ i\ =\ 0\ to\ 63\ do$ {

   Сложение по модулю 2 младших битов (битов на нулевой позиции) регистров R_1, R_2, R_3 и R_4 со значением $Key[i]$;

   Сдвиг содержимого всех четырех регистров;

   }
3) $For\ i\ =\ 0\ to\ 21\ do$ {

   Сложение по модулю 2 младших битов (битов на нулевой позиции) регистров R_1, R_2, R_3 и R_4 со значением $Frame[i]$;

   Сдвиг содержимого всех четырех регистров;

   }
4) 3, 7 и 10 биты регистра R_4 устанавливаются в 1;
5) $For\ i\ =\ 0\ to\ 99\ do$ {

   Сдвиги содержимого регистров R_1, R_2, R_3 с использованием механизма управления тактированием.

   }

Генерация ключевой последовательности. Элементом ключевой последовательности после выполнения очередного такта является сумма по модулю два выходных битов регистров и значений функций $f(x, y, z)$ от следующих битов регистров:

- для регистра R_1: биты 12, 14 и 15;
- для регистра R_2: биты 9, 13 и 16;
- для регистра R_3: биты 13, 16 и 18.

После выполнения функции инициализации выполняется 228 тактов работы генератора, в результате чего формируется ключевая последовательность. Затем номер фрейма увеличивается на единицу, повторяются процессы инициализации и генерации ключевого потока.

3.6.3. Шифр A5/3

Шифр A5/3 (или же KASUMI), в отличие от шифров A5/1 и A5/2, является блочным шифром.

4. Заключение

В работе были описаны основные поточные шифры: RC4, Snow3G, LILI128, SEAL, MUGI и шифры семейства A5. Шифр Snow 3G — это поточный шифр, который был предложен в 2006 году в качестве стандарта шифрования для мобильных сетей третьего поколения (3G). Имеет ряд известных теоретических атак, нацеленных как на различение сгенерированной шифром ключевой последовательности от истинно случайной, так и на полное восстановление ключа. Кроме того, шифр не проходит статистические тесты оценки случайности короткого ключевого потока, что дополнительно допускает возможность успешной атаки различения.

Список литературы

1. *Maitra S.*, *Paul G.* Analysis of RC4 and Proposal of Additional Layers for Better Security Margin // Progress in Cryptology - INDOCRYPT 2008. — 2008. — С. 27—39.
2. Specification of the 3GPP Confidentiality and Integrity Algorithms UEA2 & UIA2. Document 2: SNOW 3G Specification. — 2006. — Сент.
3. *Orhanou G.*, *Hajji S.*, *Bentaleb Y.* SNOW 3G Stream Cipher Operation and Complexity Study // Contemporary Engineering Sciences. — 2010. — Т. 3. — С. 97—111.
4. *Biryukov A.*, *Priemuth-Schmid D.*, *Zhang B.* Multiset Collision Attacks on Reduced-Round SNOW 3G and SNOW $3G^{\oplus}$ // Applied Cryptography and Network Security. — 2010.
5. *Gong X.*, *Zhang B.* Comparing Large-unit and Bitwise Linear Approximations of SNOW 2.0 and SNOW 3G and Related Attacks // IACR Transactions on Symmetric Cryptology. — 2021. — Июнь.
6. *Yang J.*, *Johansson J.*, *Maximov A.* Vectorized linear approximations for attacks on SNOW 3G // IACR Transactions on Symmetric Cryptology. — 2019. — Т. 2019. — С. 249—271.
7. *Löndahl C.*, *Johansson T.* Improved algorithms for finding lowweight polynomial multiples in F2[x] and some cryptographic applications // Designs, codes and cryptography. — 2014.
8. *Bohm P.* Statistical Evaluation of Stream Cipher SNOW 3G // ENGINEERING FACULTY SCIENTIFIC CONFERENCE – 13 th edition – with international participation. — 2008. — Нояб. — С. 363—366.
9. Software Implementation of the SNOW 3G Generator on iOS and Android Platforms / J. Molina-Gil [и др.] // Logic Journal of the IGPL. — 2015.
10. *Kitsos P.*, *Selimis G.*, *Koufopavlou O.* High Performance ASIC Implementation of the SNOW 3G Stream Cipher // International Conference on Very Large Scale Integration (VLSI SOC). — 2008. — Окт.
11. The LILI-128 Keystream Generator / E. Dawson [и др.]. — 2000. — Дек.
12. *Rogaway P.*, *Coppersmith D.* A Software-Optimized Encryption Algorithm // J. Cryptology. — 1998. — Сент. — Т. 11. — С. 273—287.
13. A New Keystream Generator MUGI / D. Watanabe [и др.]. — 2002. — Февр.
14. *Biham E.*, *Dunkelman O.* Cryptanalysis of the A5/1 GSM Stream Cipher // Progress in Cryptology —INDOCRYPT 2000. — 2000. — С. 43—51.
15. *Barkan E.*, *Biham E.*, *Keller N.* Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communication // Advances in Cryptology - CRYPTO 2003. — 2003. — С. 600—616.