On the Sosemanuk related key-IV sets

Aleksandar Kircanski and Amr M. Youssef

Concordia Institute for Information Systems Engineering
Concordia University
Montreal, Quebec, H3G 1M8, CANADA.
{a_kircan,youssef}@encs.concordia.ca

Abstract. Sosemanuk is a software-based stream cipher that has passed all three stages of the ECRYPT stream cipher project and is currently a member of the eSTREAM software portfolio. In the recent works on cryptanalysis of Sosemanuk, its relatively small inner state size of 384 bits was identified to be one of the reasons that the attacks were possible. In this paper, we show that another consequence of the small inner state size of Sosemanuk is the existence of several classes of (K, IV), (K', IV') pairs that yield correlated keystreams. In particular, we provide a distinguisher which requires less than 2 kilobytes of data and an inner state recovery algorithm that works for two sets of key-IV pairs of expected size $\approx 2^{128}$ each. In addition, a distinguisher requiring 2^{52} keystream words is provided for another set of pairs of Sosemanuk instances. The expected number of such key-IV pairs is 2^{192}. Although the security of Sosemanuk is not practically threatened, the found features add to understanding of the security of the cipher and also provide the basis for an elegant attack in the fault analysis model.

1 Introduction

Sosemanuk [5] is a fast software-oriented stream cipher that has passed all the three phases of the ECRYPT eSTREAM competition and is currently a member of the eSTREAM Profile 1 (software portfolio). It uses a 128-bit initialization vector and allows keys of either 128-bit or 256-bits, whereas the claimed security is always 128-bits. The design of Sosemanuk is based on the SNOW 2.0 stream cipher [10] and utilizes elements of the Serpent block cipher [6]. Sosemanuk aims to fix weaknesses of the SNOW 2.0 design and achieves better performance, notably in the cipher's initialization phase.

The preliminary analysis [5], conducted during the Sosemanuk design process, includes the assessment of the cipher with respect to different cryptanalytic attacks such as correlation attacks, distinguishing attacks and algebraic attacks. Public analysis that followed can be divided into two threads: guess-and-determine attacks and linear attacks. In [21], Tsunoo *et al.* reported a guess-and-determine attack that requires about 2^{224} operations. The computational complexity of the latter attack was reduced to 2^{176} operations by Feng *et al.* [12]. The two recent works on linear cryptanalysis of Sosemanuk include [16, 9]. The time complexity and storage requirement of both attacks is about 2^{148}. In particular, Cho *et al.* [9] relax the amount of required keystream needed by the attack by Lee *et al.* [16] by a factor of about 2^{10} and require about $2^{135.7}$ keystream bits for the inner state recovery by linear cryptanalysis to succeed. It should be noted that the size of the inner state in Sosemanuk was stated as a principal reason that the

attacks in [21, 16] were possible. In [19], an attack in the fault analysis model requiring about 6144 faults to recover the secret inner state was provided. Finally, the slide attacks on stream ciphers were introduced in [8], where Grain stream cipher was shown to be susceptible to this type of analysis.

When compared to other recent software oriented stream ciphers, such as HC-128 [23], Rabbit [7] and SNOW 2.0 [10], SOSEMANUK has the following two particularities:

(i) A comparatively small inner state of 384 bits
(ii) Delay mechanism, by which words are sent to the output every 4 steps

Property (i) stems from the SOSEMANUK design goal which aims to reduce the processor cache pressure in order to allow the execution of the cipher primarily utilizing processor cache registers, thus achieving higher speed [5]. As a consequence of having a relatively small state, a measure of defense against guess-and-determine attacks had to be introduced. Instead of releasing 32-bit words to the output as they are generated in each step, the words are accumulated and at every 4 steps, i.e., at the end of one SOSEMANUK iteration, the 4 accumulated 32-bit words are permuted, passed through a non-linear layer and finally released as a 128-bit keystream word, which resulted in increased complexity of guess-and-determine attacks. In other words, the SOSEMANUK iteration is *self-similar*, in the sense that it contains 4 identical steps.

In this paper, we examine the impact of properties (i) and (ii) on the security of SOSEMANUK. In particular, of interest are related inner states of SOSEMANUK that are preserved over time and that, as shown in this work, yield correlated keystreams. A trivial example of such a relation is full equality between the inner states. The existence of key-IV pairs that lead to related inner states follows due to property (i), more precisely, due to the ratio between the key-IV size and inner state size, given a reasonable assumption on the random behavior of 6-round cascades of the Serpent block cipher and a simple birthday paradox argument. As for particularity (ii), the self-similarity of the iteration induced by the delay mechanism gives rise to slid inner states. Again, such slid states are achievable due to the ratio between the key-IV size and the inner state size.

Underlying inner state relation	Expected # of related key-IV pairs	Distinguisher complexity	Inner state recovery
Full equality	2^{128}	*trivial*	-
LFSR equality, (Sect. 4)	2^{192}	2^{52} words	-
Slid LFSR, Slid FSM, $d = 1$ (Sect. 5)	2^{128}	$< 2^8$ words	2^{32} time, $2^{23.8}$ space
Slid LFSR, Slid FSM, $d = 2$ (Sect. 5)	2^{128}	$< 2^8$ words	$2^{47.7}$ time, $2^{23.8}$ space
Slid LFSR	2^{192}	-	-

Table 1. The expected number of related key-IV pairs, distinguisher and inner state recovery complexities.

The results in the paper are summarized in Table 1. In particular, a distinguisher requiring 2^{51} keystream words generated by two SOSEMANUK instances with equal LFSRs is constructed. The expected size of the set of key-IV pairs leading to such related inner states is about 2^{192} pairs. Next, it is shown that there exist two distinct types of slid SOSEMANUK inner states, the difference being in the state slide distance $d \in \{1, 2\}$. Distinguishers requiring less than 2 kilobytes of data are constructed for both slid states variants. Moreover, an inner state recovery algorithm that requires a work of less than 2^{32} and $2^{47.7}$ SOSEMANUK iterations is given, for the cases $d = 1$ and $d = 2$, respectively. The two sets of key-IV pairs leading to variants of such slid inner states are expected to have about 2^{128} pairs each. Finally, the relation by which only the corresponding two LFSRs are slided is pointed out. It is presented in the last row of Table 1 and the question whether it is possible to efficiently distinguish and mount inner state recovery attacks against such instance pairs is brought up for future research.

From the perspective of fault analysis, the slide properties of SOSEMANUK provided in this paper show that the security of SOSEMANUK is highly sensitive to instruction skipping type of faults. Inserting such faults has been shown to be practically possible, for example by using supply voltage variation [3, 4, 22, 20]. If the injected fault in the cryptographic device causes certain instructions to be skipped, slid SOSEMANUK keystreams may be obtained and then, the practical-complexity inner state recovery algorithms provided in this paper apply.

The rest of the paper is organized as follows. In Section 2, the specification of SOSEMANUK is provided. Inner state relations preserved by the iteration steps are identified and analyzed in Section 3. A distinguisher for SOSEMANUK instances with identical LFSRs is given in Section 4. The inner state recovery algorithm for the case of slide SOSEMANUK pairs is provided in Section 5. Finally, our conclusion is provided in Section 6.

2 The Sosemanuk specification and notation conventions

The following notation will be utilized throughout the rest of the paper:

- x^i, $x^{j,\ldots,i}$: i-th bit of word x and bits from i to j of word x, respectively
- $\boxplus, \times$: addition and multiplication modulo 2^{32}, respectively
- $\oplus$ bit-wise XOR
- $\lll$: left rotation defined on 32 bit values
- $w_{t,i}$: a 4-bit value defined by $w_{t,i} = z_{t,\lfloor \frac{i}{8} \rfloor}^{i \bmod 8}\ z_{t,\lfloor \frac{i}{8} \rfloor+4}^{i \bmod 8}\ z_{t,\lfloor \frac{i}{8} \rfloor+8}^{i \bmod 8}\ z_{t,\lfloor \frac{i}{8} \rfloor+12}^{i \bmod 8}$, for $t \geq 0$, $0 \leq i \leq 31$ where $z_{t,i}$ represents the i-th byte in the big-endian notation of the keystream word z_t. In other words, $w_{t,i}$ extracts the 4 bits from z_t that correspond to the application of one Serpent S-box.
- IS_t: SOSEMANUK inner state after t steps have been executed (four steps represent one full cipher iteration)

While the claimed security level of SOSEMANUK is 128 bits, it supports a variable key length of 128 or 256 bits, whereas the length of the initialization value can be only 128 bits. As depicted in Fig. 1, the secret inner state of SOSEMANUK consists of 12 32-bit words and utilizes three main components to generate the keystream output: a linear feedback shift register (LFSR), a finite state machine (FSM) and the function $Serpent1$

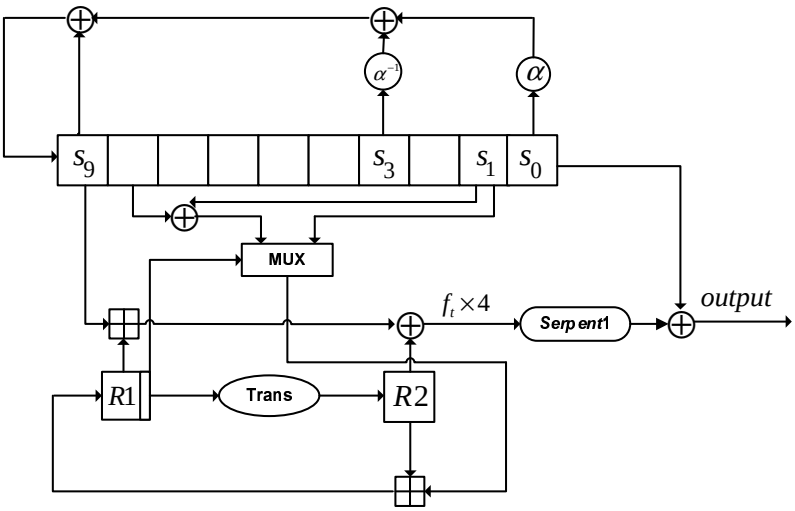

Fig. 1. Overview of the SOSEMANUK stream cipher

that represents the S-box layer of a Serpent round, specified using Serpent S-box S_2. The inner state at time t is denoted by $(s_t, \ldots, s_{t+9}, R1_t, R2_t)$. For notation convenience, the FSM counters in the specifications below are moved by 1 when compared to the original cipher description.

For the purpose of specifying one SOSEMANUK iteration, let α be a root of the primitive polynomial $P(X) = X^4 + \beta^{23}X^3 + \beta^{245}X^2 + \beta^{48}X + \beta^{239}$ over GF(2^8) and let β be a root of the primitive polynomial $Q(X) = X^8 + X^7 + X^5 + X^3 + 1$ over GF(2). Also, let

$$mux(c, x, y) = \begin{cases} x \text{ if } c = 0 \\ y \text{ if } c = 1 \end{cases}$$

Finally, let $Trans(x) = (M \times x) \lll 7$ and $M = 0x54655307$. A SOSEMANUK inner state update step consists of repeating the following transformations for 4 times:

$$R1_{t+1} = (R2_t \boxplus mux(lsb(R1_t), s_{t+1}, s_{t+1} \oplus s_{t+8}) \quad (1)$$

$$R2_{t+1} = Trans(R1_t) \quad (2)$$

$$s_{t+10} = s_{t+9} \oplus \alpha^{-1} s_{t+3} \oplus \alpha s_t \quad (3)$$

The multiplications by α and α^{-1} in (3) are in GF(2^{32}). Once every four steps, a 128-bit keystream word is sent to the output. In each step, between applying (2) and (3), a 32-bit f_t value is computed using

$$f_t = (s_{t+9} \boxplus R1_{t+1}) \oplus R2_{t+1} \quad (4)$$

The 128-bit keystream word is computed using

$$z_t = Serpent1(f_t f_{t+1} f_{t+2} f_{t+3}) \oplus s_t s_{t+1} s_{t+2} s_{t+3} \quad (5)$$

As for the initialization procedure of SOSEMANUK, it consists of expanding the key K by the Serpent key schedule and encrypting the IV using the first 24 Serpent rounds. Denote the 128-bit i-th round output by $(Y_3^i, Y_2^i, Y_1^i, Y_0^i)$. The SOSEMANUK inner state at $t = 0$ is then defined by

$$\begin{aligned}(s_6, s_7, s_8, s_9) &= (Y_3^{12}, Y_2^{12}, Y_1^{12}, Y_0^{12})\\ (s_5, R2_0, s_4, R1_0) &= (Y_3^{18}, Y_2^{18}, Y_1^{18}, Y_0^{18})\\ (s_0, s_1, s_2, s_3) &= (Y_3^{24}, Y_2^{24}, Y_1^{24}, Y_0^{24})\end{aligned}$$

where the 12-th and 18-th round outputs are taken right after the linear layer and the 24-th round output is taken after the addition with the 25-th subkey.

3 The existence and the size of the related key-IV sets

In this section, we argue the existence of related key-IV sets and determine their respective expected sizes. First, we identify relations between the two inner states that are preserved by the SOSEMANUK iteration step. A trivial example of such iteration-preserving relation is the full equality between the two inner states.

For the purpose of identifying more such relations, observe that one SOSEMANUK iteration can be regarded as 4 *steps*, each consists of one FSM update followed by one LFSR update, i.e., transformations due to a single application of (1), (2) and (3). The value f_t computed in step t is preserved and at the end of the iteration, i.e., every four steps, a 128-bit keystream output value is computed using (5) and sent to the output. Note that, as for the update part, the SOSEMANUK iteration is self-similar. In other words, the distance between two inner states expressed in steps is invariant with time, which makes the following definition meaningful.

Definition 1 *We say that the two inner states IS_t and IS_{t+d} (and also their corresponding cipher instances) are on distance d. Furthermore, if d is not a multiple of 4, we call such two inner states* slid *states.*

An analogous definition can be stated for distance between the two LFSRs. Now, four different iteration-preserving relations between the two inner states of SOSEMANUK can be identified:

(R1) Equality between inner states (inner states on distance $d = 0$)
(R2) Equality between LFSRs (LFSRs on distance $d = 0$)
(R3) Slid inner states (inner states on distance $1 \leq d \leq 3$)
(R4) Slid LFSRs (LFSRs on distance $1 \leq d \leq 3$)

Relations (R2) and (R4) leave the corresponding FSM content unrelated and are preserved over time since the LFSR is independent of the FSM. Relations (R1) and (R3) are trivially preserved. Note that if $d \notin \{0, 1, 2\}$, the two keystreams can be accordingly aligned, after which algorithms for (R1)-(R4) below become applicable. Since properly aligning the keystream adds more complexity to the distinguishing task, in this paper, we limit the related key-IV pair scope to only those that refer to the starting states in both instances.

To argue the existence of key-IV pairs that lead to pairs of states (R1)-(R4), let $E_K(Y^{12})$ and $E'_K(Y^{18})$ denote a composition of Serpent rounds $12, \ldots, 18$ and $18, \ldots, 24$, respectively, where E'_K also includes the addition with the 25-th round subkey. Then, it is convenient to represent the set of starting inner states with fixed value $Y^{12} = (s_6, s_7, s_8, s_9)$ by

$$\mathcal{I}(Y^{12}) = \left\{(Y^{12}, E_K(Y^{12}), E'_K(E_K(Y^{12}))) \mid K \in \{0,1\}^{128}\right\}$$

Clearly, the whole set of possible starting inner states is a union of $\mathcal{I}(Y^{12})$ sets for each $Y^{12} \in \{0,1\}^{128}$.

Consider, for example, the relation (R3) for $d = 1$. The Sosemanuk key-IV size is 256 bits and the inner state size is 384 bits. Assume for now that E_K and E'_K behave as random functions when regarded as functions of K for fixed inputs. Such an assumption is commonly adopted in the context of time-memory tradeoff attacks [14], see also [18]. A necessary condition for the inner state IS_0 to precede state IS'_0 is that $s_{10} = s'_9$, $s_9 = s'_8$, $s_8 = s'_7$, $s_7 = s'_6$. Consider the set $\bigcup_{\alpha \in \{0,1\}^{32}} \mathcal{I}(\alpha, Y_2^{12}, Y_1^{12}, Y_0^{12})$ for a fixed Y_2^{12}, Y_1^{12}, Y_0^{12}. The set contains at most $2^{128} \times 2^{32}$ elements. After introducing a constraint that fixes s_{10} as well, the size decreases to about 2^{128} values since, due to birthday paradox, there will be only a negligible number of collisions among its elements. The only place where its potential key-IV pairs can be found is the set $\mathcal{I}(Y_2^{12}, Y_1^{12}, Y_0^{12}, s_{10})$ which contains about 2^{128} elements. The birthday paradox can be applied to the two sets using the approximation $p(m, n) \approx 1 - e^{-n^2/m}$, where m is the number of elements in the two sets and n is the number of draws from the sets [17]. By noting that $p(2^{256}, 2^{128}) \approx 0.63212$, the probability that there exists a key-IV pair that yield two slid states on $d = 1$ will be given by $1 - 0.63212^{2^{128}} \approx 1$. Analogous reasoning holds for the case when $d = 2$. Again, clearly, in case (R4), a higher existence probability is achieved both for cases $d = 1$ and $d = 2$. A simpler probabilistic argument shows that in the case of relations (R1) and (R2), related key-IV pairs almost certainly exist.

In case the Serpent 6-round cascades randomness assumption above is abandoned, the existence of Sosemanuk related key-IV pairs is still almost certain. Assume for example that $(E_K(Y^{12}), E'_K(E_K(Y^{12})))$, regarded as a function of K, is injective in a degree significantly higher than it is the case for a randomly chosen function. In that case, the probabilistic arguments establishing relation (R1) between IS_0 and IS'_0 do not hold. However, consider the same arguments for IS_0 and IS'_n for a sufficiently large $n > 0$. Since the inner state has been randomized sufficiently by n Sosemanuk iterations, the probabilistic arguments above can be reiterated.

The expected number of related key-IV pairs in sets specified by (R1), (R2), (R3) and (R4) is 2^{128}, 2^{192}, 2^{128} and 2^{192}, respectively. The details of the counting are given in Appendix A.

4 Sosemanuk instances with the identical LFSRs

In this section, we provide a distinguisher for a pair of keystreams originating from cipher instances with same LFSRs, i.e., related according to relation (R2). Since in Sosemanuk the LFSR does not depend on the FSM registers, the equality of the LFSRs is preserved as the two instances iterate. At the first glance, it may appear that the FSM registers of the two cipher instances will become equal after about 2^{64} steps and that therefore it is

trivial to distinguish the two keystreams given such long keystreams. However, as shown below, this is not the case.

Lemma 1 *The inner states of two* SOSEMANUK *instances with the same LFSRs and different FSMs will never become equal.*

Proof: It suffices to show that one SOSEMANUK step preserves the FSM difference. Whenever $R1_t \neq R1'_t$, it follows that $R2_{t+1} \neq R2'_{t+1}$, since $Trans$ is injective. If $R1_t = R1'_t$, but $R2_t \neq R2'_t$, then $R1_{t+1} \neq R1'_{t+1}$, since the mux control bits in step t are equal. □

4.1 Distinguishing a pair of keystreams from random data

Lemma 1 states that as the two instances iterate, the corresponding FSM registers will not become identical. However, as shown below, once a zero-difference is established on certain bits of the two FSMs, the equality between the bits propagates in the next steps with biased probability, yielding almost equal keystream words. The distinguisher provided below is based on this property.

The equality between the keystream words is captured by event $O_t(l)$. Namely, consider the two inner states with identical LFSRs at some time t by $(s_t, \ldots, s_{t+9}, R1_t, R2_t)$ and $(s_t, \ldots, s_{t+9}, R1'_t, R2'_t)$, where $t \equiv 0 \bmod 4$. Let the event $O_t(l)$ take place if $w_{t,0} = w'_{t,0}, \ldots, w_{t,l-1} = w'_{t,l-1}$, i.e., if the bits in the two keystream words at time t that correspond to l least significant S-boxes are equal. Below, we show that if $l > 7$, i.e., if l is larger than the rotation constant in the $Trans$ function, the probability of event $O_t(l)$ is significantly higher than the corresponding probability for randomly generated data.

In particular, the reason that the probability of $O_t(l)$ is biased when $l > 7$ can be illustrated as follows. For some fixed l and $i \in \{1, 2\}$, define

$$\Delta(Ri_t^{l-1..0}) = \begin{cases} 0 & \text{if } Ri_t^{l-1..0} = Ri_t^{'l-1..0}, \\ 1 & \text{otherwise.} \end{cases}$$

Then, if t_0 is a starting step of some SOSEMANUK iteration, the event $O_{t_0}(l)$ will take place if $(\Delta(R1_t^{l-1..0}), \Delta(R2_t^{l-1..0})) = (0, 0)$ for $t \in \{t_0, t_0+1, t_0+2, t_0+3\}$. Now, the mentioned bias for $l > 7$ can be observed as follows. Consider the probability $P[(0,0) \to (0,0)]$, i.e., the probability that the equality between the corresponding bits in the two FSMs will be preserved. Since $(\Delta(R1_t^{l-1..0}), \Delta(R2_t^{l-1..0})) = (0, 0)$, the mux control bits will be equal in the two instances of the cipher and therefore $R1_{t+1}^{l-1..0} = R1_{t+1}^{'l-1..0}$ will hold. On the other hand, since $Trans(x) = (M \times x \bmod 2^{32}) \lll 7$, we have that $P[R1_{t+1}^{l+7..7} = R1_{t+1}^{'l+7..7}] = 1$ and $P[R1_{t+1}^{7..0} = R1_{t+1}^{'7..0}] = 2^{-7}$. Therefore

$$P[(0,0) \to (0,0)] = \begin{cases} 2^{-l} & \text{if } l \leq 7 \\ 2^{-7} & \text{if } l > 7 \end{cases}$$

Thus, increasing l to greater than 7 leaves the probability above constant, which shows that equality among the least significant bits of $R1$ and $R2$ between the two instances of the cipher propagates with good probability. To have a distinguisher, l needs to be increased up to a point where the probability of $O_t(l)$ for randomly generated data is significantly smaller than for SOSEMANUK outputs. Below, the advantage of the distinguisher for $l = 14$ is shown to be sufficiently high.

Since $(\Delta(R1_t^{l-1..0}), \Delta(R2_t^{l-1..0})) = (0,0)$ for $l = 14$ imposes two 14-bit conditions on the two inner states, the proportion of steps in which the $(0,0)$ event takes place is 2^{-28}. Consequently, it can be shown that, in 2^{53} steps, the event $(0,0)$ will occur more than $2^{24.9995}$ times with probability 0.98. Out of $2^{24.9995}$ $(0,0)$ states, only $2^{22.9995}$ will represent the starting step of a SOSEMANUK iteration. Each such starting step may develop into $O_t(14)$ event with probability $P[(0,0) \rightarrow (0,0)]^3 = 2^{-7\times 3} = 2^{-21}$. Out of $2^{22.9995}$ such steps, one will develop into $O_t(14)$ with probability $1-(1-2^{-21})^{2^{22.9995}} = 0.98$. Therefore, the probability that $O_t(14)$ will occur within 2^{53} steps, or 2^{51} SOSEMANUK iterations of the two cipher instances is higher than $0.98 \times 0.98 = 0.96$. On the other hand, the probability of that $O_t(14)$ will occur in a pair of randomly generated sequences of 2^{51} 128-bit words is $1-(1-2^{4\times 14})^{2^{51}} = 0.03$, which gives rise to the following distinguisher:

- If the event $O_t(14)$ took place for some t in the two keystreams, then return SOSEMANUK. Otherwise, return RANDOM.

The overall amount of data required for the distinguisher is $2^{51} \times 2 = 2^{52}$ keystream words. The probability of false positives is 0.03 and the probability of false negatives is $1 - 0.96 = 0.04$. Tuning the number of observed keystream words as well as the value of l yields different success rates with respect to false positives, false negatives and the required data amount.

5 Sosemanuk instances with slid inner states

In this section we present a practical inner state recovery algorithm, that requires a small amount of keystream generated by two instances following relation (R3) with $d = 1$ or $d = 2$. Such keystreams are easily distinguished from random keystreams given only about 32 and 64 keystream words generated by both instances for $d = 1$ and $d = 2$, respectively. The details of the distinguisher in question are provided in Appendix B.

5.1 Equations due to slid Sosemanuk instances

Consider the bits that interact with one particular S-box, up to the point where they reach the output as 4 bits dispersed in the keystream word z_t. The S-boxes are applied to the input values of the form $f_{t+3}^i f_{t+2}^i f_{t+1}^i f_t^i$, for $0 \leq i \leq 31$ and the corresponding S-box output, after addition to the LFSR bits, represents the 4 bit output $w_{t,i} = S(f_{t+3}^i f_{t+2}^i f_{t+1}^i f_t^i) \oplus s_{t+3}^i s_{t+2}^i s_{t+1}^i s_t^i$. The particular position of the $w_{t,i}$ bits in the 128-bit keystream word is made explicit in the notation part of Section 2.

Slid state generated keystreams introduce equations that leak the inner state material. In this section, and without loss of generality, these equations are written for $t = 0$. Expand $w_{0,i}$, $w_{4,i}$, $w_{8,i}$ and $w_{12,i}$ for some $0 \leq i \leq 31$ using the expression $S(f_{t+3}^i f_{t+2}^i f_{t+1}^i f_t^i) \oplus s_{t+3}^i s_{t+2}^i s_{t+1}^i s_t^i = w_{t,i}$. Now, in case the 4 starting keystream words due to inner state IS_1 are also known, the system can be extended as follows

$$S(f_3^i f_2^i f_1^i f_0^i) \oplus s_3^i s_2^i s_1^i s_0^i = w_{0,i}, \qquad S(f_4^i f_3^i f_2^i f_1^i) \oplus s_4^i s_3^i s_2^i s_1^i = w_{1,i} \tag{6}$$

$$S(f_7^i f_6^i f_5^i f_4^i) \oplus s_7^i s_6^i s_5^i s_4^i = w_{4,i}, \qquad S(f_8^i f_7^i f_6^i f_5^i) \oplus s_8^i s_7^i s_6^i s_5^i = w_{5,i} \tag{7}$$

$$S(f_{11}^i f_{10}^i f_9^i f_8^i) \oplus s_{11}^i s_{10}^i s_9^i s_8^i = w_{8,i}, \qquad S(f_{12}^i f_{11}^i f_{10}^i f_9^i) \oplus s_{12}^i s_{11}^i s_{10}^i s_9^i = w_{9,i} \tag{8}$$

$$S(f_{15}^i f_{14}^i f_{13}^i f_{12}^i) \oplus s_{15}^i s_{14}^i s_{13}^i s_{12}^i = w_{12,i}, \; S(f_{16}^i f_{15}^i f_{14}^i f_{13}^i) \oplus s_{16}^i s_{15}^i s_{14}^i s_{13}^i = w_{13,i} \tag{9}$$

If instead of IS_1, the keystream due to inner state IS_2 is known, the equations analogous to (6)-(9) for the case when the slide is by 2 steps are:

$$S(f_3^i f_2^i f_1^i f_0^i) \oplus s_3^i s_2^i s_1^i s_0^i = w_{0,i}, \qquad S(f_5^i f_4^i f_3^i f_2^i) \oplus s_5^i s_4^i s_3^i s_2^i = w_{2,i} \tag{10}$$

$$S(f_7^i f_6^i f_5^i f_4^i) \oplus s_7^i s_6^i s_5^i s_4^i = w_{4,i}, \qquad S(f_9^i f_8^i f_7^i f_6^i) \oplus s_9^i s_8^i s_7^i s_6^i = w_{6,i} \tag{11}$$

$$S(f_{11}^i f_{10}^i f_9^i f_8^i) \oplus s_{11}^i s_{10}^i s_9^i s_8^i = w_{8,i}, \qquad S(f_{13}^i f_{12}^i f_{11}^i f_{10}^i) \oplus s_{13}^i s_{12}^i s_{11}^i s_{10}^i = w_{10,i} \tag{12}$$

$$S(f_{15}^i f_{14}^i f_{13}^i f_{12}^i) \oplus s_{15}^i s_{14}^i s_{13}^i s_{12}^i = w_{12,i}, \quad S(f_{17}^i f_{16}^i f_{15}^i f_{14}^i) \oplus s_{17}^i s_{16}^i s_{15}^i s_{14}^i = w_{14,i} \tag{13}$$

The problem is how to recover the inner state given the right-hand values of (6)-(9) or (10)-(13), for each bit position $i = 0, \ldots, 31$.

5.2 Recovering the inner state

The inner state is recovered by first recovering the LFSR and then the FSM. To recover the LFSR, first, equations (6)-(9) or (10)-(13) are solved, yielding a constraint on the extended LFSR (i.e., $s_0, \ldots, s_{16}$.) Note that the equations have been deliberately chosen to cover variables of the extended LFSR, rather than the LFSR. That way, the dependence between the LFSR registers expressed by (3) can be used to further restrict the LFSR space. Namely, having a restriction against $s_0, \ldots, s_{16}$ in case of system (6)-(9), instead of against only $s_0, \ldots, s_9$, allows pruning the wrong $s_0, \ldots, s_9$ candidates by using (3) for $t = 0, \ldots, 6$.

To find the expected number of solutions to the system (6)-(9), the exact distribution of the number of solutions was calculated. The problem of calculating the distribution was divided into two parts. The distribution was found first for (6)-(7) and then for (8)-(9), where care has been taken about bits s_8 and f_8 which participate in both subsystems. Once the joint distribution that represents the distribution of the number of solutions of the full system (6)-(9) was calculated, the expected number of solutions for the system was found to be $2^{3.357}$. The same strategy has been applied for the case $d = 2$, and the expected number of solutions to (10)-(13) is found to be equal to $2^{4.5278}$. Since the 32 systems corresponding to different $0 \leq i \leq 31$ are independent, the overall number of candidates for $(s_0, \ldots, s_{16})$ is expected to be $2^{3.357 \times 32} \approx 2^{107.42}$ and $2^{32 \times 4.528} \approx 2^{144.896}$ for $d = 1$ and $d = 2$, respectively. As for mutual dependencies between $(s_0, \ldots, s_{16})$, relation (3) for $t = 0, \ldots, 6$ imposes 7 32-bit constraints on the extended LFSR. Therefore, the two restrictions ensure that the final number of candidates for the extended LFSR is restricted only to 1 (correct) value with high probability both in cases $d = 1$ and $d = 2$, since $7 \times 32 = 224 >> 107.42$ and $7 \times 32 = 224 >> 144.896$, respectively.

Recovering the LFSR To recover the LFSR in case $d = 1$, the attacker obtains the first 4 keystream words of both instances and solves 32 systems of the form (6)-(9), by simply trying all the possibilities for the $f_0^i, \ldots, f_{16}^i$ and whenever possible, deducing the corresponding $s_0^i, \ldots, s_{16}^i$. The resulting extended LFSR restriction is specified "i-th-bit-wise", that is to say, the solution of the 32 systems can be regarded as sets $S_0, \ldots, S_{31}$ where each set S_i contains about $2^{3.357}$ 17-bit values as candidates for value $s_0^i \ldots, s_{16}^i$. To prune the false candidates, a naive approach of going through all candidates of the 32 sets and discarding in case the α relations are not satisfied would require an impractical complexity of $2^{108.16}$ operations. A more efficient approach is to utilize the properties of

the α and α^{-1} multiplication in $GF(2^{32})$ in order to derive an early contradiction after choosing candidates from $S_{l_1}, \ldots, S_{l_m}$ for m significantly smaller than 32.

Multiplication by α and α^{-1} in $\mathrm{GF}(2^{32})$ can be represented as [5]:

$$\alpha(x) = (x \ll 8) \oplus T_\alpha(x \gg 24), \quad \alpha^{-1}(x) = (x \gg 8) \oplus T_\alpha^{-1}(x\&255) \tag{14}$$

where T_α and T_α^{-1} represent 8×32 S-boxes. Let $s_{t,b}$, $0 \leq b \leq 3$ denote the b-th least significant byte of the LFSR word s_t. Isolating the i-th bit of each value participating in relation (3) and using (14), the following bit relation is obtained

$$s_{t+10}^i = s_{t+9}^i \oplus s_{t+3}^{i+8} \oplus T_{\alpha^{-1}}^i(s_{t+3,0}) \oplus s_t^{i-8} \oplus T_\alpha^i(s_{t,3}) \tag{15}$$

with the convention that the bits out of the 32-bit range are equal to 0. Choosing a candidate from the set S_i for some $0 \leq i \leq 31$ produces one such equation for each $0 \leq t \leq 6$, i.e., 7 equations of form (15). In each equation, due to the candidate choice, s_{t+10}^i and s_{t+9}^i are known bits. No two equations among the 7 equations have common variables. More generally, choosing a candidate from each of the m different S_i sets for some $0 \leq m \leq 31$, generates 7 independent systems, each having m equations of the form (15). Since increasing the number of sets S_i from which the candidates are chosen enlarges the search space, the strategy is to select a minimal family of S_i sets so that each of the systems is solvable in $s_{t,3}$ and $s_{t+3,0}$, the two input bytes for T_α and T_α^{-1} S-boxes and moreover, so that the system becomes overdefined. Then, in case the solution to any of the systems does not exist, the candidates choice from the S_i sets can be discarded.

The strategy above is illustrated as follows. Let $i = 0$ and let a candidate be chosen from S_0. Note that (15) for $i = 0$ contains the unknown variable s_t^8. Since the goal is to make the systems solvable for T_α and T_α^{-1} S-box inputs, in order to fix s_8, a candidate from S_8 is chosen. Similarly, other enforced choices are from sets S_{16} and S_{24}. For a fixed $0 \leq t \leq 6$, the system is of the form

$$\begin{aligned}
s_{t+10}^0 &= s_{t+9}^0 \oplus s_{t+3}^8 \oplus T^0(s_{t+3,0}s_{t,3}) \\
s_{t+10}^8 &= s_{t+9}^8 \oplus s_{t+3}^{16} \oplus s_t^0 \oplus T^8(s_{t+3,0}s_{t,3}) \\
s_{t+10}^{16} &= s_{t+9}^{16} \oplus s_{t+3}^{24} \oplus s_t^8 \oplus T^{16}(s_{t+3,0}s_{t,3}) \\
s_{t+10}^{24} &= s_{t+9}^{24} \oplus s_t^{16} \oplus T^{24}(s_{t+3,0}s_{t,3})
\end{aligned} \tag{16}$$

where $T(s_{t+3,0}s_{t,3}) = T_\alpha^{-1}(s_{t+3,0}) \oplus T_\alpha(s_{t,3})$ is a 16×32 one-to-one S-box (as we verified) convenient to define for the consideration below. Given the system (16), the T S-box bits can be determined. Moreover, the choice of candidates from S_0 and S_{24} imposes a constraint on the T input by fixing the least significant bit of $s_{t,0}$ and $s_{t+3,0}$, respectively. The same holds for any i, $0 \leq i \leq 7$. Thus, the system constraints the T input-output value by by fixing its 4 output and 2 input bits.

Similarly, for $0 \leq i < 8$, choosing elements from S_{i+8k}, $k = 0, 1, 2, 3$, yields a system analogous to (16), for each $0 \leq t \leq 6$. Thus, the T input-output is constrained by fixing 2 input and 4 output bits, for each $0 \leq t \leq 6$. Now, extending the guess to sets $S_{i_0+8k}, \ldots, S_{i_n+8k}$, for $k = 0, 1, 2, 3$ and some $0 \leq i_0, \ldots, i_n \leq 7$ will overdefine the T input with good probability if $n \geq 3$. In particular, for $n = 3$, each system constraints the T input value by fixing 6 input bits and 12 output bits. Since there exists only 2^{16} possible input-output pairs for T, at most 2^{16} out of 2^{18} such constraints will

be satisfiable. As it is reasonable to assume that the constraint values are distributed uniformly, the candidate choice will pass with probability of at most 2^{-2} for one fixed $0 \leq t \leq 6$. Then, the pruning is done by verifying whether for the given choice from S_{i_0+8k}, S_{i_1+8k} and S_{i_2+8k}, for $k = 0, 1, 2, 3$ all of the 7 systems are satisfiable, which happens with probability less than $2^{-2\times 7} = 2^{-14}$. Since the initial number of candidates is expected to be $2^{12\times 3.357} = 2^{40.284}$, less than $2^{40.56-14} = 2^{26.56}$ candidates are expected to pass the test. In case $n = 4$, using analogous arguments, each of the 7 16-equation systems is satisfiable with probability of at most 2^{-8} and thus applying this criterion provides reduction of at least $2^{-8\times 7} = 2^{-56}$. The initial expected number of candidates in this case is $2^{3.357\times 16} = 2^{53.712}$ and thus no false candidates are expected to pass the test. It can thus be observed that for $n = 3$, the criterion is weaker but there is a smaller number of candidates, whereas for $n = 4$, the criterion is stronger, but the initial number of candidates is relatively high. Note that above values $i_0, \ldots, i_3$ can be chosen arbitrarily. However, to decrease the complexity of the search, it is useful to start from i_j values such that $|S_{i_j}|$ are small.

The discussion above indicates a breadth-first style search procedure to efficiently prune incorrect candidates. Namely, given the solution to (6)-(9), in the form of sets $S_0, \ldots, S_{31}$, the LFSR recovery proceeds as follows. Let $h(i) = \prod_{0\leq k\leq 3} |S_{i+8k}|$. Determine $i_0, \ldots, i_7$ so that $h(i_0)$, $h(i_1), \ldots, h(i_7)$ represents a sequence sorted in increasing order. Choose a candidate from each of the S_{i_0+8k}, S_{i_1+8k} and S_{i_2+8k}, $0 \leq k \leq 3$. Generate 7 systems, 12 equations of form (15) each, using the guessed candidates. If for any of the 7 systems, the constrained T input-output value does not exist, discard the candidates choice. Otherwise, for the candidates that passed the criterion, continue the search by guessing the next four S_{i_j+8k} sets simultaneously and looking for contradiction in a breadth-first manner, i.e., not by recursing into the search tree, but rather looking for contradiction after every fourplet guess. Note that solving the systems appearing during the attack can be done efficiently by utilizing precomputed tables. For some $0 \leq i_0, i_1, i_2 \leq 7$, consider verifying whether a guess from sets S_{i_0+8k}, S_{i_1+8k} and S_{i_2+8k}, $0 \leq k \leq 3$ yields a contradiction or not. Each such guess yields an 18-bit constraint on the input-output value of T, where the set of fixed bit positions uniquely depends on the set $\{i_0, i_1, i_2\}$. A yes/no information on the existence of such T input-output value can be preserved in precomputed tables $T_{\{i_0,i_1,i_2\}}$, for each set $\{i_0, i_1, i_2\} \in \{0, 1, \ldots, 7\}$. The storage demand amounts to $\binom{8}{3} \times 2^{18} \approx 2^{23.8}$ bits and the same number of T lookup operations is required to fill in the tables.

Estimated complexities and experimental results In the previous section, a procedure for recovering the LFSR was provided. After the LFSR is recovered, the FSM can be recovered by guessing one FSM register, then recovering the other one and verifying the guess against the keystream. The computational effort of such a procedure is less than 2^{32} Sosemanuk iterations. According to the previous subsection, the computational effort of the LFSR recovery amounts to the search over candidates from sets S_{i_0+8k}, S_{i_1+8k} and S_{i_2+8k}, for $0 \leq k \leq 3$, where $0 \leq i_0, i_1, i_2 \leq 7$ are the three points of the function $h(i) = \prod_{0\leq k\leq 3} |S_{i+8k}|$ with the smallest values. To estimate the expected value of such three smallest values, for both $d = 1$ and $d = 2$, the exact distribution of $h(i)$ was calculated exhaustively, using the exact distribution of $|S_{i+8k}|$, found at the beginning of this section. The attacker has a chance to choose the three smallest points

of h, out of 8 independent $h(i)$ for $0 \leq i < 8$. The 1^{st}, 2^{nd} and 3^{rd} order statistics of an 8 element sample has been found to be $2^{9.517}$, $2^{10.602}$ and $2^{11.372}$, respectively. Therefore, the overall attack requirements for $d = 1$ are

- Data of 4 words, produced by 2 slid Sosemanuk instances
- The computation of $2^{9.517+10.602+11.372} = 2^{31.491}$ table lookups for the LFSR recovery and the work equivalent to 2^{32} iterations for the FSM recovery
- Storage requirement of $2^{23.8}$ bits

For $d = 2$, the data and storage complexity remain the same. Since the expected 1^{st}, 2^{nd} and 3^{rd} order statistics of $h(i)$, $0 \leq i \leq 7$, are $2^{15.081}$, $2^{15.996}$ and $2^{16.627}$, respectively, the overall expected number of candidates to go through is $2^{15.081+15.996+16.627} = 2^{47.704}$. Note that the computational cost can be lowered at the cost of increasing the available data. Namely, if more than 4 keystream words are available, the attack can be performed using the 4 consecutive keystream words of both instances for time t where $h(i_0)$, $h(i_1)$, $h(i_2)$ are minimal.

The attack was implemented for $d = 1$ on a 2.4 GHz AMD processor, using the Java Sosemanuk implementation provided with the cipher specification. The search procedure behaved as predicted above and the two slid Sosemanuk inner states have been uniquely recovered in less than a day.

Slide Sosemanuk instances in the context of fault attacks The properties exposed in this section yield a fault attack, in which the attacker introduces instruction skipping faults [15], e.g., by varying the supply voltage of the device that performs encryption at carefully chosen times. In such a setting, transient faults start occurring in a relatively large number [4] and the attacker waits until the slid keystreams are detected, using the distinguisher from Section 5. At that point, the inner state recovery procedure provided above applies and the secret inner state of the cipher can be efficiently recovered.

6 Conclusion and future work

This work provides the first steps in the analysis of the related keys of Sosemanuk. We showed that Sosemanuk's related key pairs exist and that observing pairs of keystreams generated under such related key-IV pairs can lead to the recovery of the inner state. In particular, an efficient inner state recovery algorithm and also a distinguisher have been provided for two particular classes of expected size of 2^{128} key-IV pairs. A distinguisher requiring 2^{52} keystream words was presented for a larger class of related key-IV pairs, of size 2^{192}. While the results from this paper do not directly threaten the security of Sosemanuk, the non-random properties of Sosemanuk should be brought to light and the users of the cipher should be aware of it. Finally, this work also shows that Sosemanuk is highly sensitive to a specific kind of a fault analysis attack in which the execution flow is disturbed.

As for future work, the natural next step is to find an efficient procedure for constructing particular $((K, IV), (K', IV'))$ pairs that yield the presented related Sosemanuk instances.

References

1. Ahamadi, H., Eghidos, T. and Khazaei, S: Improved Guess and Determine Attack on Sosemanuk, Tehran, 2006, www.ecrypt.eu.org/stream/sosemanukp3.html
2. Baignères, T., Junod, P., Vaudenay, S.: How Far Can We Go Beyond Linear Cryptanalysis, ASIACRYPT 2004, LNCS-3392, pg. 432-450, Springer-Verlag, 2004
3. Bar-El, H., Choukri, H., Naccache, D., Tunstall, M. and Whelan, C.: The Sorcerer's Apprentice Guide to Fault Attacks, Proceedings of the IEEE, vol. 94, no. 2, pg. 370-382, 2006
4. Barenghi, A., Bertoni, G., Breveglieri, L., Pellicioli, M. and Pelosi, G: Low Voltage Fault Attacks to AES and RSA on General Purpose Processors, ePrint IACR Report, 130/2010
5. Berbain, C., Billet, O., Canteaut, A., Courtois, N., Gilbert, H., Goubin, L., Gouget, A., Granboulan, L., Lauradoux, C., Minier, M., Pornin, T., Sibert, H.: Sosemanuk, a Fast Software-Oriented Stream Cipher. The eSTREAM Finalists, LNCS-4986, pg. 98-118, Springer, 2008
6. Biham, E., Anderson, R. and Knudsen, L.: Serpent: A New Block Cipher Proposal, FSE 1998, LNCS-1372, pg. 222-238, Springer, 1998
7. Boesgaard, M., Vesterager, M., Pedersen, T., Christiansen, J., and Scavenius, O.: Rabbit: A new high-performance Stream Cipher, Proc. of FSE 2003, Springer-Verlag, LNCS-2887, pp. 307-329, 2003
8. Cannire, C.D., Küçük, O. and Preneel, B.: Analysis of Grain's Initialization Algorithm, AFRICACRYPT 2008, LNCS-5023, pg. 276-289, Springer, 2008
9. Cho, J.Y., Hermelin, M.: Improved Linear Cryptanalysis of Sosemanuk. In Proceedings of Information, Security and Cryptology, ICISC 2009. LNCS, vol. 5984, pp.101-117, Springer, 2010
10. Ekdahl, P., Johansson, T.: A New Version of the Stream Cipher SNOW. In: Nyberg, K., Heys, H. (eds) Selected Areas in Cryptography, SAC 2002, LNCS, vol. 2295, pp.47-61, Springer Verlag, 2002
11. eSTREAM, the ECRYPT Stream Cipher Project. Available at http://www.ecrypt.eu.org/stream/
12. Feng, X., Liu, J., Zhou, Z., Wu., C. and Feng, D.: Byte-Based Guess and Determine Attack on SOSEMANUK, ASIACRYPT 2010, LNCS-6477, pp. 146-157, Springer, 2010
13. Grinstead, C., M. and Snell. L., J.: Introduction to Probability, American Mathematical Society, 2nd edition, 1998
14. Hellman, M.: A Cryptanalytic Time-Memory Trade-Off, IEEE Transactions on Information Theory, IT-26, pg. 401-406, 1980
15. Kim, C.H., Quisquater, J.-J.: Fault Attacks for CRT Based RSA: New Attacks, New Results, and New Countermeasures. WISTP 2007. LNCS, vol. 4462, pp. 215-228. Springer, Heidelberg, 2007
16. Lee, J.-K., Lee, D.-H, Park S.: Cryptanalsyis of Sosemanuk and SNOW 2.0 Using Linear Masks. ASIACRYPT 2008, LNCS, vol. 5350, pp. 524-538, Springer, 2008
17. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography CRC Press (Boca Raton), 1997
18. Quisquater, J.-J. and Delescaille, J.-P., How Easy is Collision Search? Application to DES, Advances in Cryptology, Proceedings of Eurocrypt 1989, LNCS-434, pg. 429-434, Springer, 1990
19. Salehani, Y.E., Kircanski, A. and Youssef, A.M.: Differential Fault Analysis of Sosemanuk, AFRICACRYPT 2011, LNCS-6737, pg. 316-331, Springer, 2011
20. Schmidt, J.-M. and Herbst, C.: A Practical Fault Attack on Square and Multiply, In Fault Diagnosis and Tolerance in Cryptography, 3rd International Workshop, FDTC 2008, IEEE-CS Press, 2008

21. Tsunoo, Y., Saito, T., Shigeri, M., Suzaki, T., Ahmadi, H., Eghlidos, T. and Khazaei, S.: Evaluation of SOSEMANUK With Regard to Guess-and-Determine attacks, 2006, http://www.ecrypt.eu.org/stream/soemanukp3.html
22. Vertanen, O.: Java Type Confusion and Fault Attacks, Fault Diagnosis and Tolerance in Cryptography 2006, LNCS-4236, pg. 237-251, Springer, 2006
23. Wu, H.: The Stream Cipher HC-128, New Stream Cipher Designs, LNCS-4986, pp. 39-47, Springer, Heidelberg, 2008

A Estimating the expected number of related key-IV pairs

Having shown the almost certain existence of related key-IV pairs in Section 3, in this appendix, we estimate their corresponding number. This is done for each relation (R1)-(R4), under the assumption that $(E_K(Y^{12}), E'_K(E_K(Y^{12})))$ behaves randomly when regarded as functions of the key. Note that if m elements are drawn with replacement from a set of n elements, the expected number of drawn elements that will repeat is $E_1(m,n) = \binom{m}{2}\frac{1}{n}$, since the probability that two elements will collide is $\frac{1}{n}$ and there exist $\binom{m}{2}$ elements pairs. Similarly, if two sets of m elements are drawn from an n-element set, the expected number of collisions is $E_2(m,n) = \frac{m^2}{n}$.

As for case (R1), for some $Y^{12} \in \{0,1\}^{128}$, the expected number of collisions in $\mathcal{I}(Y^{12})$ is $E_1(2^{128}, 2^{256}) \approx 1$. Since there is 2^{128} possibilities for Y^{12}, we have that the final expected number of key-IV pairs that yield identical inner states is $2^{128} \times E_1(2^{128}, 2^{256}) \approx 2^{128}$. In case (R2), since only the LFSRs are required to be identical, we have that the expected number of key-IV pairs is $2^{128} \times E_1(2^{128}, 2^{192}) \approx 2^{192}$.

As for slid pairs, described by case (R3), let $Y_2^{12}, Y_1^{12}, Y_0^{12}$ be fixed. Consider the subset of the set $\bigcup_{\alpha \in \{0,1\}^{32}} \mathcal{I}(\alpha, Y_2^{12}, Y_1^{12}, Y_0^{12})$ that fixes the corresponding s^{10} value. As described in the previous subsection, such a set is expected to have 2^{128} elements. The corresponding slid pairs can only be in $\mathcal{I}(Y_2^{12}, Y_1^{12}, Y_0^{12}, s_{10})$. Since the two sets have 2^{128} elements, and there is 2^{128} such set pairs, the expected number of states that have a corresponding slid state is $2^{128} \times E_2(2^{128}, 2^{256}) \approx 2^{128}$. Even though when $\alpha = Y_2^{12} = Y_1^{12} = Y_0^{12}$ there may be an overlap between the corresponding sets, it is easy to show that the final expected number of slid pairs does not change significantly in this case. Applying analogous reasoning, it follows that the number of key-IV pairs corresponding to case (R4) is $2^{128} \times E_2(2^{128}, 2^{192}) \approx 2^{192}$.

B Distinguishing slid Sosemanuk instances

In this appendix, we show that the output of two SOSEMANUK instances on distance $d = 1$ or $d = 2$ can easily be distinguished from random stream. Let $z_0, z_4, \ldots, z_{4n}$ and $z_1, z_5, \ldots, z_{4n+1}$, be two SOSEMANUK keystream outputs generated by slid inner state on distance $d = 1$. First, we show that it is possible to efficiently compute the probability that the two slid SOSEMANUK instances generate a given sequence $z = (z_0, z_1, z_4, z_5, \ldots, z_{4n}, z_{4n+1})$.

Assume that sequence z is due to two slid SOSEMANUK instances and extract the S-box outputs corresponding to different bit positions $0 \leq i \leq 31$. Write down each of the 32 subsequences in the form

$$w_{0,i}, w_{1,i}, w_{4,i}, w_{5,i}, w_{8,i}, w_{9,i}, \ldots, w_{4n,i}, w_{4n+1,i} \tag{17}$$

The first 8 elements in the sequence (17) are generated according to (6)-(9) and the rest are generated by analogous equations. Consider the dependence between the elements of the sequence (17) by focusing for example on the value $w_{4,i}$ defined by the left-hand equation of (7). Observe that values participating in its generation $f_4, \ldots, f_7$ and $s_4, \ldots, s_7$ participate only in the generation of $w_{1,i}$ and $w_{5,i}$. The same observation generalizes to any sequence element in (17). We adopt the assumption that f_t and s_t, $t \geq 0$, are uniformly distributed. Although the assumption does not fully ressemble the real case, it serves to simplify the model and moreover, does not create any relevant deviation from the real case, as verified by the experimental results below. Now each value in (17) depends only on its predecessor. Thus, the behavior of the subsequence is completely determined by transition probabilities, which are different for even and odd steps of the sequence (17). As for the transitions of the form $w_{4t,i}$ to $w_{4t+1,i}$, the transition matrix $M_{1,1}$ is provided. The elements of $M_{1,1}$ that represent transition probabilities have been calculated by going through all of the left-hand values of a pair of equations (6) and recording the number of transitions for each $(w_{0,i}, w_{1,i})$:

$$M_{1,1} = \frac{1}{64} \begin{pmatrix}
4&4&3&5&3&5&6&2&4&4&3&5&3&5&6&2\\
4&4&3&5&3&5&6&2&4&4&3&5&3&5&6&2\\
4&4&5&3&5&3&2&6&4&4&5&3&5&3&2&6\\
4&4&5&3&5&3&2&6&4&4&5&3&5&3&2&6\\
3&5&4&4&6&2&3&5&3&5&4&4&6&2&3&5\\
3&5&4&4&6&2&3&5&3&5&4&4&6&2&3&5\\
5&3&4&4&2&6&5&3&5&3&4&4&2&6&5&3\\
5&3&4&4&2&6&5&3&5&3&4&4&2&6&5&3\\
3&5&6&2&4&4&3&5&3&5&6&2&4&4&3&5\\
3&5&6&2&4&4&3&5&3&5&6&2&4&4&3&5\\
5&3&2&6&4&4&5&3&5&3&2&6&4&4&5&3\\
5&3&2&6&4&4&5&3&5&3&2&6&4&4&5&3\\
6&2&3&5&3&5&4&4&6&2&3&5&3&5&4&4\\
6&2&3&5&3&5&4&4&6&2&3&5&3&5&4&4\\
2&6&5&3&5&3&4&4&2&6&5&3&5&3&4&4\\
2&6&5&3&5&3&4&4&2&6&5&3&5&3&4&4
\end{pmatrix}$$

The factor $\frac{1}{64}$ comes from the fact that fixing $w_{0,i}$ in (6) leaves 2^4 possibilities for f_0, f_1, f_2, f_3, s_0, s_1, s_2 and s_3 and thus $2^{4+2} = 64$ possibilities for the left-hand side of the equation for $w_{1,i}$. From matrix $M_{1,1}$ it is clear that the transition probabilities are heavily biased.

In the case of transitions from $w_{4t+1,i}$ to $w_{4(t+1),i}$, the bias does not exist and all the values in the 16×16 matrix $M_{1,2}$ are equal to $\frac{1}{16}$. Now the probability $\Pr_{D_0}[z]$, where D_0 is the distribution that describes the output of two slid SOSEMANUK instances, can be calculated efficiently. Namely, given a reasonable independence assumption, this probability is equal to a product of probabilities of 32 subsequences of form (17) extracted from z. Each of the subsequence probabilities is calculated by using the Markovian assumption and the transition matrices $M_{1,1}$, $M_{1,2}$. On the other hand, the probability $\Pr_{D_1}[z]$ of z occurring in a random stream is equal to $1/2^{(2n+2)\times 128}$.

Finally, as for the matrix $M_{2,1}$ that corresponds to the case $d = 2$, this 16×16 matrix has been found to be populated only by $\frac{13}{256}$ and $\frac{19}{256}$ values. Denote the elements of the matrix in question by $M_{2,1} = [a_{i,j}]$. Then, for $i \in \{1, 2, 3, 4, 13, 14, 15, 16\}$,

$a_{i,j} = \frac{13}{256}$ if $j \in \{1, 4, 5, 8, 9, 12, 13, 16\}$. For $i \in \{5, 6, 7, 8, 9, 10, 11, 12\}$, $a_{i,j} = \frac{13}{256}$ if $j \in \{2, 3, 6, 7, 10, 11, 14, 15\}$. Other $a_{i,j}$ elements are equal to $\frac{19}{256}$.

To optimally decide whether the sample z comes from distribution D_0 or D_1 [2], the acceptance region is defined by

$$\mathcal{A} = \{z : \mathrm{LR}(z) \geq 1\}, \text{ where } \mathrm{LR}(z) = \frac{\Pr_{\mathrm{D}_0}[z]}{\Pr_{\mathrm{D}_1}[z]}$$

If $z \in \mathcal{A}$, the distinguisher returns D_0 and otherwise D_1. Thus, to distinguish outputs of two SOSEMANUK slid instances from random data, it suffices to compute the probability of the sample under distributions D_0 and D_1 and compare the two probabilities.

The distinguisher above has been implemented and the following experiments have been performed to estimate the number of keystream words necessary for the overall probability of error P_e to be sufficiently low. Each experiment consisted of generating 32 keystream words by each of the two random instances of SOSEMANUK which are on distance 1 and then applying the distinguisher, whereas the probability multiplicative structure has been replaced by the usual logarithmic additive representation. The experiment was repeated for 1024 times and no false negatives or false positives have been reported. It can be concluded that the input size of 32 keystream words is sufficient to distinguish SOSEMANUK with high probability. The analogous experiment was executed for the case $d = 2$ using 64 keystream words from both instances. Neither false positives nor false negatives occurred in 1024 experiments repetitions, which shows that 64 keystream words are sufficient to make P_e sufficiently small when $d = 2$.