Linearity Properties of the SOBER-t32 Key Loading [⋆] [⋆⋆]

Markus Dichtl and Marcus Schafheutle

Siemens AG, Corporate Technology, 81730 München, Germany, Email:
Markus.Dichtl@mchp.siemens.de, Marcus.Schafheutle@mchp.siemens.de

Abstract. In the course of the evaluation of the stream cipher SOBER-t32 submitted to NESSIE, a correlation between initial states has been found for related keys. With high probability some sums of bits of the initial state after key loading do not change their value when a bit of the key is inverted. This holds also for the loading of frame keys. It is shown that the required condition for the frame keys is met very naturally when using counters as frame keys. The linearity properties of the SOBER-t32 key loading are caused by non-optimal diffusion of the non-linear filter function of the cipher.

1 Introduction

SOBER-t32 is a synchronous additive stream cipher designed for key sizes up to 256 bits. SOBER-t32 was submitted to NESSIE by Philip Hawkes and Gregory Rose at Qualcomm Australia. NESSIE (New European Schemes for Signatures, Integrity, and Encryption) is a project within the IST program of the European Commission. Its main purpose is to put forward a portfolio of strong cryptographic primitives that has been obtained after an open call and been evaluated using a transparent and open process.

2 Description of SOBER-t32

The stream cipher is constructed from a linear feedback shift register (*LFSR*), a non-linear filter (*NLF*), and a form of irregular decimation, called stuttering. SOBER-t32 outputs the key stream as 32-bit blocks. The *LFSR* is of length 17 and operates over $GF(2^{32})$.

The *NLF* consists of *XOR* ($\oplus$), addition modulo 2^{32} ($\boxplus$), and a 32-to-32 bit transformation called f-function. The output of the non-linear filter at time t is described as

$$NLF(t) = ((f(s_t \boxplus s_{t+16}) \boxplus s_{t+1} \boxplus s_{t+6}) \oplus \text{const}) \boxplus s_{t+13}$$

where s_{t+k} is the content of the k'th shift register cell at time t, const is a session key dependent constant value, derived during the key loading phase.

The function f uses the 8 most significant bits of its input as an input to a lookup table S-box with 32 bits of output. The 24 least significant bits of the input are just XOR-ed to the output of the S-box in order to obtain the result of the function f.

The stuttering decimates the output of the *NLF* in an irregular fashion. For the stuttering of SOBER-t32 it can be shown that there is an average of $\frac{12}{25}$ key stream output per clock of the *LFSR*.

With the size of the *LFSR* and the size of the key dependent parameter const, it is obvious that the initial state size of SOBER-t32 is $2^{17 \cdot 32 + 32}$.

A detailed specification of SOBER-t32 can be found at the NESSIE Web site [NES].

[⋆] The work described in this paper has been supported by the Commission of the European Communities through the IST program under contract IST-1999-12324.

[⋆⋆] The information in this document is provided as is, and no warranty is given or implied that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

3 SOBER-t32 key loading

The key loading determines the initial state of the *LFSR* and the value of const from the key. It relies on the operations "`Include()`" and "`Diffuse()`".

`Include(X)` adds the 32-bit word X to the *LFSR* cell r_{15}. `Diffuse()` clocks the register, computes the output of the *NLF*, and XORs this output to the *LFSR* cell r_4.

For key loading, the key is divided into 32-bit words. The `Include()` operator is applied to each of these words, and each `Include()` operation is followed by a `Diffuse()` operation. As an immediate consequence one observes that the key words included last are diffused less than those included first. The `Include()` operation is also applied to the key length in bytes. Then the `Diffuse()` operation is applied 17 more times.

4 Diffusion properties of the *NLF*

A closer inspection of *NLF* shows that its diffusion properties are not ideal. Modifications in the most significant 8 bits of the input of f are very efficiently diffused over the output word, whereas for modifications in the 24 least significant bits of the input of f no diffusion by f occurs. The only way of diffusion for these bit positions is by carry propagation. However, carry chains tend to be short. Burks, Goldstine, and von Neumann found out in 1946 [BGvN46] that on average the longest carry chain in adding k-bit numbers is of length $\log_2(k)$. Hence carry badly propagates bit modifications to bit positions far away from the bit position of the modification.

Nevertheless long carry chains occur from time to time. This explains why the linearity properties of the SOBER-t32 key loading described in the next section do not hold always, but only with a very high probability. The low probability long carry chains provide enough diffusion to disturb the linear relationships occasionally.

A lot of diffusion occurs by the clocking of the *LFSR*. However, this linear operation in $GF(2^{32})$ is also linear in $GF(2)$.

The linear recurrence over $GF(2^{32})$ of the SOBER-t32 *LFSR* can be shown, see [Her85], to be equivalent to implementing 32 parallel bit-wise *LFSR*s, each of length $17 \cdot 32 = 544$. These linear recurrences are identical, represented by the primitive polynomial $p_{32}(x)$ over $GF(2)$:

$$\begin{aligned}
p_{32}(x) = 1 &+ x^{17} + x^{19} + x^{21} + x^{23} + x^{25} + x^{27} + x^{29} + x^{30} + x^{31} + x^{33} + x^{34} + x^{35} + x^{37} + x^{38} \\
&+ x^{39} + x^{41} + x^{42} + x^{46} + x^{47} + x^{49} + x^{50} + x^{51} + x^{53} + x^{54} + x^{55} + x^{56} + x^{57} + x^{58} \\
&+ x^{59} + x^{61} + x^{62} + x^{63} + x^{64} + x^{65} + x^{66} + x^{67} + x^{68} + x^{70} + x^{74} + x^{76} + x^{77} + x^{78} \\
&+ x^{79} + x^{84} + x^{85} + x^{87} + x^{89} + x^{90} + x^{91} + x^{92} + x^{95} + x^{97} + x^{98} + x^{100} + x^{101} + x^{102} \\
&+ x^{109} + x^{111} + x^{113} + x^{114} + x^{117} + x^{118} + x^{121} + x^{125} + x^{130} + x^{131} + x^{132} + x^{133} \\
&+ x^{137} + x^{138} + x^{140} + x^{142} + x^{143} + x^{145} + x^{146} + x^{147} + x^{148} + x^{149} + x^{151} + x^{153} \\
&+ x^{156} + x^{160} + x^{163} + x^{164} + x^{165} + x^{172} + x^{173} + x^{175} + x^{176} + x^{177} + x^{179} + x^{180} \\
&+ x^{184} + x^{185} + x^{186} + x^{190} + x^{191} + x^{193} + x^{198} + x^{200} + x^{201} + x^{202} + x^{206} + x^{207} \\
&+ x^{208} + x^{209} + x^{210} + x^{211} + x^{212} + x^{213} + x^{219} + x^{220} + x^{221} + x^{225} + x^{227} + x^{229} \\
&+ x^{231} + x^{232} + x^{233} + x^{235} + x^{236} + x^{238} + x^{239} + x^{240} + x^{241} + x^{242} + x^{244} + x^{245} \\
&+ x^{246} + x^{247} + x^{249} + x^{252} + x^{255} + x^{258} + x^{262} + x^{263} + x^{264} + x^{265} + x^{266} + x^{277} \\
&+ x^{279} + x^{281} + x^{284} + x^{285} + x^{288} + x^{289} + x^{290} + x^{291} + x^{292} + x^{294} + x^{296} + x^{300} \\
&+ x^{301} + x^{302} + x^{304} + x^{306} + x^{307} + x^{309} + x^{310} + x^{316} + x^{321} + x^{323} + x^{324} + x^{325} \\
&+ x^{327} + x^{334} + x^{335} + x^{336} + x^{337} + x^{340} + x^{341} + x^{342} + x^{344} + x^{345} + x^{346} + x^{347} \\
&+ x^{350} + x^{352} + x^{355} + x^{357} + x^{360} + x^{361} + x^{362} + x^{363} + x^{364} + x^{365} + x^{368} + x^{373} \\
&+ x^{377} + x^{379} + x^{381} + x^{382} + x^{383} + x^{385} + x^{388} + x^{389} + x^{390} + x^{391} + x^{392} + x^{394}
\end{aligned}$$

$$\begin{aligned}
&+ x^{398} + x^{403} + x^{404} + x^{405} + x^{406} + x^{407} + x^{413} + x^{416} + x^{420} + x^{421} + x^{422} + x^{425}\\
&+ x^{426} + x^{428} + x^{430} + x^{431} + x^{433} + x^{435} + x^{436} + x^{437} + x^{438} + x^{440} + x^{441} + x^{442}\\
&+ x^{445} + x^{446} + x^{447} + x^{448} + x^{449} + x^{450} + x^{453} + x^{458} + x^{461} + x^{463} + x^{465} + x^{466}\\
&+ x^{469} + x^{471} + x^{473} + x^{474} + x^{477} + x^{478} + x^{479} + x^{481} + x^{483} + x^{484} + x^{487} + x^{488}\\
&+ x^{489} + x^{490} + x^{493} + x^{494} + x^{496} + x^{499} + x^{500} + x^{503} + x^{505} + x^{506} + x^{508} + x^{511}\\
&+ x^{513} + x^{514} + x^{516} + x^{519} + x^{521} + x^{524} + x^{527} + x^{529} + x^{532} + x^{536} + x^{540} + x^{544}
\end{aligned}$$

5 Linearity properties of the SOBER-t32 key loading

The insufficient diffusion explained in the previous section is the reason for the existence of sums of bits from the initial state of the shift register which keep their value if some bit of the key is inverted. We denote the bits of the initial state of the shift register by $b_1, b_2, \ldots$ where b_1 is the least significant bit of the 17th *LFSR* cell, b_{32} the most significant bit of this word, b_{33} the least significant bit of the 16th *LFSR* cell, ... We computed the sum

$$\begin{aligned}
&b_{542} + b_{537} + b_{531} + b_{530} + b_{529} + b_{528} + b_{527} + b_{525} + b_{524} + b_{520} + b_{519} + b_{518} + b_{516} + b_{514} + b_{513} +\\
&b_{478} + b_{477} + b_{474} + b_{473} + b_{471} + b_{470} + b_{469} + b_{466} + b_{465} + b_{383} + b_{382} + b_{381} + b_{380} + b_{379} + b_{377} +\\
&b_{371} + b_{370} + b_{369} + b_{363} + b_{362} + b_{361} + b_{360} + b_{356} + b_{256} + b_{254} + b_{250} + b_{249} + b_{248} + b_{247} + b_{241} +\\
&b_{238} + b_{235} + b_{234} + b_{232} + b_{231} + b_{229} + b_{228} + b_{125} + b_{122} + b_{119} + b_{118} + b_{117} + b_{115} + b_{112} + b_{111} +\\
&b_{109} + b_{108} + b_{104} + b_{103} + b_{102} + b_{100} + b_{98} + b_{97} + b_{62} + b_{61} + b_{58} + b_{57} + b_{55} + b_{54} + b_{53} + b_{50} + b_{49}
\end{aligned}$$

in $GF(2)$ for 100000 keys chosen randomly. In 99957 cases the value of this sum remained the same when the least significant bit of the last key word was inverted.

We also found 16 other sums of this kind and 7 bit sums whose values change with high probability when the least significant bit of the last key word is toggled. In all cases the success probability determined from 100000 trials was at least 99.4 percent.

Of course, only linearly independent solutions were considered. By forming linear combinations, many more sums of this kind could be found, which do not provide additional information.

We also identified sums whose values remain invariant under the inversion of other key bits with high probability or change their value with high probability. In total, we found 249 such equations with a success probability of at least 98.6 percent. Again, these probabilities were determined by using 100000 random keys.

For each of the 11 least significant bits of the last key word such sums were found. Apparently from more significant bit positions, the carry chains reach the 8 most significant bit positions with sufficient probability to provide enough non-linear diffusion to destroy such linearity properties. For 8 of the 9 least significant bit positions of the second to last key word such sums exist as well.

For earlier key words, the number of applications of the `Diffuse()` operation seems to be sufficiently high in order to prevent the existence of such sums.

One way to strengthen SOBER-t32 against the linearity properties described is to increase the number of final `Diffuse()` steps for key loading. Our experiments showed 21 final steps instead of 17 to be sufficient.

6 Linearity properties of the SOBER-t32 rekeying

For some applications it is convenient to be able to generate more than one key stream from one key. To make the streams different, SOBER-t32 can process an initialization vector, called frame key, which can be assumed to be public. Ideally, the streams generated with the same key but different frame keys should be completely independent. We are not able to show correlations between the streams generated, but between the initial states derived from different frame keys but the same key. Since the loading of the frame key is very similar to the key loading, it does not come as a big surprise that sums of the kind described also exist for the frame key loading.

First the cipher key is loaded, then the frame key. The only difference is that for key loading the value of const in the *NLF* is zero. For frame key loading the const value determined in the key loading phase is used. (This values is also used for the actual generation of the stream.)

The following sum of bits of the initial *LFSR* state almost never changed its value when the second to least significant bit of the to last key frame word was toggled:

$$b_{511} + b_{508} + b_{507} + b_{506} + b_{505} + b_{501} + b_{494} + b_{492} + b_{491} + b_{488} + b_{487} + b_{486} + b_{448} + b_{447} + b_{445} + b_{444} + b_{441} + b_{440} + b_{437} + b_{436} + b_{434} + b_{433} + b_{432} + b_{431} + b_{428} + b_{427} + b_{425} + b_{422} + b_{417} + b_{351} + b_{348} + b_{346} + b_{345} + b_{344} + b_{341} + b_{339} + b_{336} + b_{335} + b_{334} + b_{333} + b_{331} + b_{330} + b_{325} + b_{224} + b_{222} + b_{221} + b_{218} + b_{217} + b_{215} + b_{213} + b_{211} + b_{209} + b_{205} + b_{203} + b_{202} + b_{201} + b_{199} + b_{198} + b_{195} + b_{193} + b_{96} + b_{93} + b_{91} + b_{90} + b_{88} + b_{84} + b_{82} + b_{81} + b_{80} + b_{79} + b_{78} + b_{73} + b_{72} + b_{71} + b_{65} + b_{32} + b_{31} + b_{29} + b_{28} + b_{25} + b_{24} + b_{21} + b_{20} + b_{18} + b_{17} + b_{16} + b_{15} + b_{12} + b_{11} + b_{9} + b_{6} + b_{1}$$

The value of the sum did not change in 99806 cases of 100000 where both key and frame key were chosen randomly. In total, 28 sums with probabilities above 99 percent were identified.

Our experiments suggest that the non-zero values of the variable const cause the diffusion for the key frame loading to be a little better than for the key loading, where const is zero.

Whereas it might look quite artificial that single key bits should be inverted as required for the linearity properties of the key loading described in the previous section, the inversion of individual bits of key frame bits occurs in real life. Most commonly, the key frame is just a binary counter. Counter states where the only bit difference is at a bit position of low significance occur very frequently. We have seen in the case of the key loading that only bit positions of low significance can be inverted with good probabilities for the linearity properties, and this holds also for the key frame loading. So when using the counter method for key frames, invariant sums of initial state bits occur frequently.

7 Relation to other attacks

The key loading of previous versions of SOBER has been the base for earlier attacks. In the original version of SOBER, the key loading was linear. This was exploited by the attack of Bleichenbacher and Patel [BP99].

In the description of the NESSIE submission of SOBER-t32 [NES] a paper by Bleichenbacher, Patel, and Meier [BPM] is quoted in which a correlation between initial states of SOBER-II (an attempt to fix the problems of SOBER) for different key frames but the same initial key material was found. The updated key and frame loading used in SOBER-t32 is claimed by the authors of the cipher to destroy this correlation. Above, we have shown another correlation that they did not succeed to destroy.

8 Applicability of the linearity properties to SOBER-t16

The linearity properties we identified for SOBER-t32, do not exist in SOBER-t16. SOBER-t16 is very similar to SOBER-t32, but based on 16-bit words. As we pointed out above, the non-linear diffusion of SOBER-t32 relies on carry propagation. This also holds for SOBER-t16, but shorter carry chains, which occur with higher probability, are sufficient for the non-linear diffusion within the 16-bit words. If the key loading of SOBER-t16 is reduced to 15 final `Diffuse()` operations instead of 17, the linearity properties appear.

9 Conclusion

We have found high probability correlations of sums of initial state bits of SOBER-t32 for related keys and also for related key frames. Such correlations are undesirable for a stream cipher, even when it is not clear how to exploit them for an attack. As we have identified the non-optimal diffusion of the *NLF* as the main source of the problem, we suggest not to rely on carry propagation as a means of diffusion in the next version of SOBER.

References

[BGvN46] A.W. Burkes, H.H. Goldstine, and J. von Neumann, *Preliminary discussion of the logical design of an electronic computing instrument*, Tech. report, Institute for Advanced Study Report, Princeton, NJ, 1946.

[BP99] D. Bleichenbacher and S. Patel, *SOBER cryptanalysis*, Proceedings of Fast Software Encryption '99, Lecture Notes in Computer Science, Springer Verlag, 1999, pp. 305–316.

[BPM] D. Bleichenbacher, S. Patel, and W. Meier, *Analysis of the SOBER stream cipher*, Tech. report, TIA contribution TR45.AHAG/99.08.30.12.

[Her85] T. Herlestam, *On functions of linear shift register sequences*, Proceedings of EUROCRYPT '85, Lecture Notes in Computer Science, Springer Verlag, 1985, pp. 119–129.

[NES] *NESSIE web site*, `http://www.cryptonessie.org`.