Guess-and-Determine Attacks on SNOW

Philip Hawkes and Gregory G. Rose

Qualcomm Australia, Level 3, 230 Victoria Rd, Gladesville, NSW 2111, Australia
{phawkes, ggr}@qualcomm.com

Abstract. This paper describes guess-and-determine attacks on the stream cipher SNOW. The first attack has a data complexity of $O(2^{64})$ and a process complexity of $O(2^{256})$. The second attack has process complexity of $O(2^{224})$, and a data complexity of $O(2^{95})$.

1 Introduction

SNOW is synchronous stream cipher proposed by Patrik Ekdahl and Thomas Johansson [3]. The cipher accepts both 128-bit and 256-bit keys. SNOW has been selected as a contender in the second phase of the NESSIE crypto-algorithm evaluation project (see www.cryptonessie.org).

This paper presents *guess-and-determine attacks* (GD attacks) on SNOW. GD attacks have proven effective in analyzing word-oriented stream ciphers. For example, the best known attacks on the SOBER family of stream ciphers (SOBER-t16 [5] and SOBER-t32 [6] are also contenders in the NESSIE project), are GD attacks [1,2,4]. These attacks exploit the relationships between internal values (such as the recurrence relationship in a shift register), and the relationship used to construct the key-stream values from the internal values. A GD attack *guesses* some internal values and then exploits the relationships to *determine* other internal values; hence the name. The cipher is "broken" when a complete internal state has been determined from the guessed values. The basic attack on SNOW has a data complexity of $O(2^{64})$ and a process complexity of $O(2^{256})$. The other attack is based on this first attack; reducing the process complexity at the cost of increased data complexity. The second attack has process complexity of $O(2^{224})$, and a data complexity of $O(2^{95})$.

The paper is arranged as follows. Section 2 describes the SNOW algorithm. Section 3 summarizes the basic attack and describes a "tricks" used in the attack. Section 3.2 describes the steps involved in the third phase of the attack. Section 3.3 discusses the complexity. The second attack is described in Section 4.

2 SNOW

SNOW is based on a *Linear Feedback Shift Register* (LFSR) with a recurrence defined over the Galois field of order 2^{32}, denoted $GF(2^{32})$. The state of the LFSR at time t is denoted $(s_{t+15}, ..., s_t)$, where the values s_{t+i} are 32-bit words.

K. Nyberg and H. Heys (Eds.): SAC 2002, LNCS 2595, pp. 37–46, 2003.

The next state of the LFSR is $(s_{t+16}, ..., s_{t+1})$, where s_{t+16} is defined using a recurrence over $GF(2^{32})$:

$$s_{t+16} = \alpha(s_t \oplus s_{t+3} \oplus s_{t+9}), \tag{1}$$

where $\oplus$ denotes the field addition (equivalent to the bit-wise exclusive-OR operation), α is a specified field element and multiplication is performed in the field. The values of α and the basis for the field are chosen so that for $X = (x_{31}, .., x_0) \in GF(2^{32})$,

$$\alpha X = (x_{30}, .., x_0, 0) \oplus (x_{31} \cdot \text{0x80421009}).$$

Values from the LFSR are combined with values in a *Finite State Machine* (FSM) to generate the key-stream. The FSM contains two 32-bit values; these values (at time t) are denoted by $(R1_t, R2_t)$. The 32-bit output of the FSM at time t is

$$f_t = (s_{t+15} \boxplus R1_t) \oplus R2_t,$$

where $\boxplus$ denotes addition modulo 2^{32}. The 32-bit output of the cipher at time t is computed as

$$z_t = f_t \oplus s_t.$$

The next state of the FSM is computed as

$$\begin{aligned} R1_{t+1} &= R1_t \oplus ROT(f_t \boxplus R2_t, 7) \\ R2_{t+1} &= S(R1_t), \end{aligned}$$

where $ROT(A, B)$ denotes the cyclic rotation of A by B bits towards the most-significant bit (MSB), and $S()$ is defined by four invertible 8-bit S-boxes and a bit permutation. In this document, S is treated as a single, invertible 32-bit S-box.

3 The Basic Attack

It is assumed that the attacker has observed a portion of key-stream $\{z_t\}$, $t = 1..N$, where N is large enough to give a high probability of the attack succeeding. The GD attack has four "phases":

Phase One The attacker make the following assumptions regarding the internal values of the FSM at time t:

Assumption 1 $R2_t = S(R1_t \oplus (2^{32} - 1))$,
Assumption 2 $R2_{t+14} = S(R1_{t+14} \oplus (2^{32} - 1))$.

Phase Two The attacker *guesses* the values of s_t, s_{t+1}, s_{t+2}, s_{t+3}, $R1_t$ and $R1_{t+14}$, (32-bits each: a total of 192 bits).

Phase Three The attacker *determines* the LFSR state $(s_{t+15}, \ldots, s_t)$ from the values guessed in Phase Two, based on the assumptions in Phase One (the details of Phase Three are discussed in Section 3.2).

Phase Four The attacker tests the correctness of the LFSR state $(s_{t+15}, \ldots, s_t)$ and FSM state $(R1_t, R2_t)$ by producing a key-stream using these states and comparing this with the observed key-stream. If the streams agree, then the states are correct. If the streams do not agree, and the attacker can try further guesses (Phase Two) for that values of t (Phase One) then return to Phase Two. If all guesses have been tried for that value of t, then the assumptions in Phase One must be incorrect, so the attacker increases t and returns to Phase One.

The probability that both assumptions in Phase One are correct is 2^{-64}. Thus, around 2^{64} values of t will be tried before finding internal states where $R2_t = S(R1_t \oplus (2^{32}-1))$ and $R2_{t+14} = S(R1_{t+14} \oplus (2^{32}-1))$. If the assumptions are incorrect, then none of the guesses in Phase two will result in the correct LFSR state and FSM state in Phase Three, so none of the guesses will produce the correct key-stream in Phase Four.

The bulk of the detail of the attack is in Phase Three. The remainder of Section 3 describes a "trick" exploited in Phase Three.

3.1 A Trick

The "assumed" relationship between $R2_t$ and $R1_t$ (in Phase One) is specially chosen to allow s_{t+14} to be determined from $R1_t$. Note that due to the assumed form of $R2_t = (R1_t \oplus (2^{32}-1))$):

$$\begin{aligned} R1_{t-1} &= invS(R2_t) \\ &= invS(\ S(R1_t \oplus (2^{32}-1))\ ) \\ &= R1_t \oplus (2^{32}-1), \\ \Rightarrow R1_{t-1} \oplus R1_t &= (2^{32}-1), \end{aligned}$$

where $invS$ is the inverse of the S mapping. Furthermore,

$$R1_{t-1} \ = \ R1_t \oplus (2^{32}-1) \ = \ -R1_t - 1 \,(\mathrm{mod}\ 2^{32}),$$

for all values of $R1_t$. The FSM internal state is updated by computing $R1_t$ from $R1_{t-1}$, f_{t-1} and $R2_{t-1}$ as

$$\begin{aligned} R1_t &= R1_{t-1} \oplus ROT(f_{t-1} \boxplus R2_{t-1}, 7), \\ \Rightarrow \qquad (R2_{t-1} \boxplus f_{t-1}) &= ROT(R1_{t-1} \oplus R1_t, -7) \\ &= ROT((2^{32}-1), -7) \\ &= (2^{32}-1). \end{aligned}$$

Note that since $(R2_{t-1} \boxplus f_{t-1}) = (2^{32}-1)$, this implies that

$$(R2_{t-1} \oplus f_{t-1}) = (2^{32}-1).$$

Now that the attacker knows that $(R2_{t-1} \oplus f_{t-1})$, the attacker can "reverse" the FSM to compute s_{t+14}:

$$\begin{aligned} s_{t+14} &= (R2_{t-1} \oplus f_{t_1}) - R1_{t-1} \pmod{2^{32}} \\ &= (2^{32} - 1) - (-R1_t - 1) \pmod{2^{32}} \\ &= R1_t. \end{aligned}$$

The assumed relationship between $R2_t$ and $R1_t$ is especially chosen to allow s_{t+14} to be determined from $R1_t$. Similarly, the assumed relationship between $R2_{t+14}$ and $R1_{t+14}$ is especially chosen to allow s_{t+28} to be determined from $R1_{t+14}$:

$$s_{t+28} = R1_{t+14}.$$

The assumptions are specially designed to allow s_{t+14} and s_{t+28} to be determined "for free". This is discussed in more detail in Section 3.3.

3.2 Details of Phase Three

This section concerns the details or Phase Three: determining a full LFSR state from the values guessed in Phase Two and based on the assumptions in Phase One. Phase Three is divided into 29 "steps". In the following description, a value s_{t+i} in the LFSR is often represented using the value "i": for example, 0 represents s_t and 6 represents s_{t+6}. Also, the following notation is used:

"$F \rightarrow$" denotes exploiting the relationship between s_{t+i+15}, $R1_{t+i}$, $R2_{t+i}$ and f_{t+i};
"$G \rightarrow$" denotes exploiting the relationship between $R1_{t+i+1}$, f_{t+i}, $R1_{t+i}$ and $R2_{t+i}$;
"$S \rightarrow$" denotes exploiting the relationship $R2_{t+i+1} = S(R1_{t+i})$.

After guessing the values in Phase Two, the attacker has guessed all the bits in the values for 0, 1, 2, 3 (that is, s_t, s_{t+1}, s_{t+2}, s_{t+3}), and $R1_t$, $R2_t$ $R1_{t+14}$, $R2_{t+14}$. For a given guess, all the bits of the following values can be immediately determined by exploiting the relationships in the FSM:

Step 0.a $s_t \oplus z_t = f_t$.
Step 0.b $f_t, R1_t, R2_t$ $F \rightarrow s_{t+15}$.
Step 0.c $f_t, R1_t, R2_t$ $G \rightarrow R1_{t+1}$.
Step 0.d $R1_t$ $S \rightarrow R2_{t+1}$.

These four steps can be repeated for $i = 1, 2, 3$, to determine (among other values) s_{t+16}, s_{t+17} and s_{t+18}.

Step 1/2/3.a $s_{t+i} \oplus z_{t+i} = f_{t+i}$.
Step 1/2/3.b $f_{t+i}, R1_{t+i}, R2_{t+i}$ $F \rightarrow s_{t+i+15}$.
Step 1/2/3.c $f_{t+i}, R1_{t+i}, R2_{t+i}$ $G \rightarrow R1_{t+i+1}$.
Step 1/2/3.d $R1_{t+i}$ $S \rightarrow R2_{t+i+1}$.

The attacker can also utilize the "trick" described in Section 3.1.

Step 4 $s_{t+14} = R1_t$.
Step 5 $s_{t+28} = R1_{t+14}$.

The attacker has now guessed or determined all the bits in the values:

$$\text{0-3, 14-18, 28}, R1_{t+i}, R2_{t+i},\ i \in \{\text{0-4, 14}\}.$$

The four basic steps are now repeated for $i = 14, \ldots, 18$.

Step 6/../10.a $s_{t+i} \oplus z_{t+i} = f_{t+i}$.
Step 6/../10.b $f_{t+i}, R1_{t+i}, R2_{t+i}\ F \rightarrow s_{t+i+15}$.
Step 6/../10.c $f_{t+i}, R1_{t+i}, R2_{t+i}\ G \rightarrow R1_{t+i+1}$.
Step 6/../10.d $R1_{t+i}\ S \rightarrow R2_{t+i+1}$.

(Step 6 has $i = 14$, Step 7 has $i = 15$, and so forth). The attacker has now guessed or determined all the bits of the values:

$$\text{0-3, 14-18, 28-33}, R1_{t+i}, R2_{t+i},\ i \in \{\text{0-4,14-19}\}.$$

The attacker can exploit linear relationships between state words s_t, that result from the linear recurrence. The linear relationships exploited in this attack correspond to the linear recurrence (1), denoted by "$L \rightarrow$", and the linear relationship corresponding to the "square" of the linear recurrence:

$$s_{t+32} = \alpha^2(s_t \oplus s_{t+6} \oplus s_{t+18}),$$

denoted by "$L2 \rightarrow$". The attacker determines more internal values using the following steps:

Step 11 $14, 17, 30$ $L \rightarrow 23$.
Step 12 $0, 3, 16$ $L \rightarrow 9$.
Step 13 $0, 18, 32$ $L2 \rightarrow 6$.
Step 14 $6, 9, 15$ $L \rightarrow 22$.
Step 15 $15, 18, 31$ $L \rightarrow 24$.
Step 16 $16, 22, 29$ $L \rightarrow 13$.
Step 17 $1, 17$ $L2 \rightarrow (s_{t+4} \oplus s_{t+10})$.
Step 18 $(s_{t+4} \oplus s_{t+10}), 22$ $L2 \rightarrow 36$.
Step 19 $23, 29, 36$ $L \rightarrow 20$.
Step 20 $17, 20, 33$ $L \rightarrow 26$.
Step 21 $2, 14, 28$ $L2 \rightarrow -4$.
Step 22 $17, 31$ $L2 \rightarrow (s_{t-1} \oplus s_{t+5})$.
Step 23 $-4, (s_{t-1} \oplus s_{t+5})$ $L \rightarrow 12$.
Step 24 $3, 6, 12$ $L \rightarrow 19$.
Step 25 $12, 15, 28$ $L \rightarrow 21$.
Step 26 $9, 12, 18$ $L \rightarrow 25$.
Step 27 $s_{t+19} \oplus z_{t+19} = f_{t+19}$.
Step 28 $f_{t+19}, R1_{t+19}, R2_{t+19}$ $F \rightarrow s_{t+34}$.
Step 29 $18, 21, 34$ $L \rightarrow 27$.

The attacker has now determined a full LFSR state $(s_{t+27}, \ldots s_{t+12})$. The LFSR is "rewound" to determine $(s_{t+15}, \ldots, s_t)$.

3.3 Complexity

Recall the assumptions that $R2_t = S(R1_t \oplus (2^{32}-1))$ and $R2_{t+14} = S(R1_{t+14} \oplus (2^{32}-1))$. The probability that both of these assumptions are correct is 2^{-64}. The attacker will have to try around 2^{64} values of t before finding internal states where $R2_t = S(R1_t \oplus (2^{32}-1))$ and $R2_{t+14} = S(R1_{t+14} \oplus (2^{32}-1))$. That is, the attacker will need $O(2^{64})$ outputs. For each value of t, the attack requires guessing 192 bits (all 32 bits of s_t, s_{t+1}, s_{t+2}, s_{t+3}, $R1_t$, $R2_{t+14}$). This corresponds to 2^{192} guesses for each value of t. Thus, the data complexity of the attack is 2^{64} and the process complexity is $2^{64} \cdot 2^{192} = 2^{256}$.

Note that the effect of forcing $R2_t$ to be related to $R1_t$ (rather than guessing $R2_t$ independently of $R1_t$) is to increase the data complexity by a factor of 2^{32}, while not changing the process complexity. However, by choosing the relationship between $R1_t$ and $R2_t$, the attacker is able to determine more information about s_{t+14}. If attacker had simply guessed $R2_t$, then the attacker could determine

$$\begin{aligned}(R_{t-1} \boxplus f_{t-1}) &= ROT((R1_{t-1} \oplus R1_t), -7) \\ &= ROT(invS(R2_t) \oplus R1_t, -7).\end{aligned}$$

However, the attacker may be unable to determine much information about $(R2_{t-1} \oplus f_{t-1})$ (required to obtain s_{t+14}). For example, if $(R_{t-1} \boxplus f_{t-1}) = 0$, then two possible "solutions" for $(R1_{t-1}, f_{t-1})$ are $(R1_{t-1}, f_{t-1}) = (0,0)$ and $(R1_{t-1}, f_{t-1}) = (2^{32}-1, 1)$. If $(R1_{t-1}, f_{t-1}) = (0,0)$, then $(R_{t-1} \oplus f_{t-1}) = 0$, while if $(R1_{t-1}, f_{t-1}) = (2^{32}-1, 1)$, then $(R_{t-1} \oplus f_{t-1}) = (2^{32}-2)$. This uncertainty in the value of $(R_{t-1} \oplus f_{t-1})$ means that the attacker determines less information about s_{t+14}. This is why the attacker assumes the value of $R2_t$, rather than guessing it. The same argument applies to $R2_{t+14}$.

3.4 Observation

This attack is aided by the choice of inputs to the recurrence relation:

$$s_{t+16} = \alpha(s_t \oplus s_{t+3} \oplus s_{t+9}).$$

There is a gap of 3 words between the inputs s_t and s_{t+3}, and gap of $6 = 2 \times 3$ words between the inputs s_{t+3} and s_{t+9}. This difference of 6 words is unfortunate because it is the same gap as between the first two inputs to the square of the linear recurrence:

$$s_{t+32} = \alpha^2(s_t \oplus s_{t+6} \oplus s_{t+18}).$$

Thus the value of $(s_{t+i} \oplus s_{t+6+i})$ can be considered as a single input to either equation. For example, in Step 17, the values of s_{t+1} and s_{t+17} are used to determine the value of $(s_{t+4} \oplus s_{t+10})$ by exploiting the linear recurrence. In Step 18, the value of $(s_{t+4} \oplus s_{t+10})$ is then combined with the value of s_{t+22} to obtain s_{t+36}, by exploiting the square of the linear recurrence. The value of s_{t+36} is determined despite the attacker being unable to determine the values

of s_{t+4} and s_{t+10}. If the linear recurrence did not have this property, then it is likely that fewer state words could be derived from the guessed words, and the attacker would be unable to derive a full state from the guessed words. This may force the attacker to guess more internal values before being able to derive a full state, thus increasing the complexity. That is, it is likely that guess-and-determine attacks could be forced to have a larger complexity by the use of a different linear recurrence.

4 Reducing the Process Complexity

The process complexity of the attack can be reduced if the attacker can avoid guessing all of s_{t+3}. We have explored various ways of achieving this: as of yet we have been unable to decrease the complexity significantly without increasing the data complexity.

It is assumed that the attacker has observed a portion of key-stream $\{z_t\}$, $t = 1..N$, where N is large enough to give a high probability of the attack succeeding. Like the first attack, this GD attack has four "phases":

Phase One The attacker make the following assumptions regarding the internal values of the FSM at time t:

$$\begin{aligned}&\textbf{Assumption 1} && R2_t = S(R1_t \oplus (2^{32}-1)),\\ &\textbf{Assumption 2} && R2_{t+14} = S(R1_{t+14} \oplus (2^{32}-1)),\\ &\textbf{Assumption 3} && R1_{t+3} \in \{0, 2^{31}\}.\end{aligned}$$

Phase Two The attacker *guesses* the values of s_t, s_{t+1}, $R1_t$, $R1_{t+14}$, (32-bits each: a total of 128 bits), and $R1_{t+3} \in \{0, 2^{31}\}$) (1 bit).

Phase Three The attacker *determines* the LFSR state $(s_{t+15}, \ldots, s_t)$ from the values guessed in Phase Two, based on the assumptions in Phase One (the details of Phase Three are discussed below).

Phase Four The attacker tests the correctness of the LFSR state $(s_{t+15}, \ldots, s_t)$ and FSM state $(R1_t, R2_t)$ by producing a key-stream using these states and comparing this with the observed key-stream.

The probability that both assumptions in Phase One are correct is 2^{-95}. Thus, around 2^{95} values of t will be tried before finding internal states where $R2_t = S(R1_t \oplus (2^{32}-1))$, $R2_{t+14} = S(R1_{t+14} \oplus (2^{32}-1))$, and $R1_{t+3} \in \{0, 2^{31}\}$. If the assumptions are incorrect, then none of the guesses in Phase two will result in the correct LFSR state and FSM state in Phase Three, so none of the guesses will produce the correct key-stream in Phase Four.

After guessing the values in Phase Two, the attacker has values for 0, 1, and $R1_t$, $R2_t$, $R1_{t+3}$, $R1_{t+14}$, $R2_{t+14}$ (using the assumptions in Phase One). For a given guess, all the bits of the following values can be immediately determined for $i = 0, 1$;

$$\begin{aligned}&\textbf{Step i.a} && s_{t+i} \oplus z_{t+i} && = && f_{t+i}.\\ &\textbf{Step i.b} && f_{t+i}, R1_{t+i}, R2_{t+i} && F \rightarrow && s_{t+i+15}.\\ &\textbf{Step i.c} && f_{t+i}, R1_{t+i}, R2_{t+i} && G \rightarrow && R1_{t+i+1}.\\ &\textbf{Step i.d} && R1_{t+i} && S \rightarrow && R2_{t+i+1}.\end{aligned}$$

Recall that $R1_{t+3}$ has been guessed to be either 0 or 2^{31} = 0x80000000. The attacker uses the function G to compute f_{t+2}.

$$\textbf{Step 2.a}\ R1_{t+3}, R1_{t+2}, R2_{t+2}\ G \rightarrow f_{t+2}.$$

Now that the attacker has computed f_{t+2}, they can compute s_{t+2}, s_{t+17} and $R2_{t+3}$.

$$\begin{array}{ll} \textbf{Step 2.b}\ f_{t+2} \oplus z_{t+2} & = \ s_{t+2}. \\ \textbf{Step 2.c}\ f_{t+2}, R1_{t+2}, R2_{t+2}\ F & \rightarrow s_{t+17}. \\ \textbf{Step 2.d}\ R1_{t+2} & S \rightarrow R2_{t+3}. \end{array}$$

The attacker can utilize the "trick" described in Section 3.1.

$$\begin{array}{l} \textbf{Step 3}\ s_{t+14} = R1_t. \\ \textbf{Step 4}\ s_{t+28} = R1_{t+14}. \end{array}$$

The attacker has now guessed or determined all the bits in the values:

$$\text{0-2, 14-17, 28}, R1_{t+i}, R2_{t+i},\ i \in \{\text{0-3, 14}\}.$$

The four basic steps are now repeated for $i = 14, \ldots, 17$.

$$\begin{array}{ll} \textbf{Step 5/../8.a}\ s_{t+i} \oplus z_{t+i} & = \ f_{t+i}. \\ \textbf{Step 5/../8.b}\ f_{t+i}, R1_{t+i}, R2_{t+i}\ F & \rightarrow s_{t+i+15}. \\ \textbf{Step 5/../8.c}\ f_{t+i}, R1_{t+i}, R2_{t+i}\ G & \rightarrow R1_{t+i+1}. \\ \textbf{Step 5/../8.d}\ R1_{t+i} & S \rightarrow R2_{t+i+1}. \end{array}$$

(Step 5 has $i = 14$, Step 6 has $i = 15$, and so forth). The attacker has now guessed or determined all the bits of the values:

$$\text{0-2, 14-17, 28-32}, R1_{t+i}, R2_{t+i},\ i \in \{\text{0-3,14-18}\}.$$

Exploiting the linear recurrence and the linear relationships corresponding to the square of the connection polynomial, allows the attacker to determine more internal values:

$$\begin{array}{ll} \textbf{Step 9}\ \ 14, 17, 30 & L \rightarrow\ 23. \\ \textbf{Step 10}\ 2, 14, 28 & L2 \rightarrow -4. \\ \textbf{Step 11}\ 17, 31 & L2 \rightarrow (s_{t-1} \oplus s_{t+5}). \\ \textbf{Step 12}\ -4, (s_{t-1} \oplus s_{t+5}) & L \rightarrow\ 12. \\ \textbf{Step 13}\ 12, 15, 28 & L \rightarrow\ 21. \\ \textbf{Step 14}\ 0, 16 & L \rightarrow\ (s_{t+3} \oplus s_{t+9}). \\ \textbf{Step 15}\ (s_{t+3} \oplus s_{t+9}), 21 & L2 \rightarrow 35. \end{array}$$

Note that if $R1_{t+3} \in \{0, 2^{31}\}$, then

$$\begin{aligned} f_{t+3} &= (s_{t+18} \boxplus R1_{t+3}) \oplus R2_{t+3} \\ &= (s_{t+18} \oplus R1_{t+3}) \oplus R2_{t+3}, \\ \Rightarrow \qquad z_{t+3} &= s_{t+3} \oplus f_{t+3} \\ &= s_{t+3} \oplus (s_{t+18} \oplus R1_{t+3} \oplus R2_{t+3}), \\ \Rightarrow \qquad (s_{t+3} \oplus s_{t+18}) &= z_{t+3} \oplus R1_{t+3} \oplus R2_{t+3}. \end{aligned}$$

The attacker can use the value of $(s_{t+3} \oplus s_{t+18})$ to compute other values of s_{t+i} by exploiting linear relationships. For example, that attacker can combine the two equations

$$\begin{aligned} & s_{t+19} = \alpha \cdot (s_{t+3} \oplus s_{t+6} \oplus s_{t+12}), \\ & s_{t+6} = s_t \oplus s_{t+18} \oplus \alpha^{-2} s_{t+32}, \\ \Rightarrow \quad & s_{t+19} = \alpha \cdot ((s_{t+3} \oplus s_{t+18}) \oplus s_t \oplus s_{t+12} \oplus \alpha^{-2} s_{t+32}). \end{aligned}$$

Since the attacker knows $(s_{t+3} \oplus s_{t+18})$, 0, 12 and 32, the attacker can perform the: following step

Step 16 $s_{t+19} = \alpha \cdot ((s_{t+3} \oplus s_{t+18}) \oplus s_t \oplus s_{t+12} \oplus \alpha^{-2} s_{t+32})$.

The attacker has now guessed or determined all the bits of the values:

$$\text{0-2, 12,14-17,19,21,23, 28-32,35}, R1_{t+i}, R2_{t+i},\ i \in \{\text{0-3,14-18}\}.$$

Now that the attacker knows s_{t+19}, there are many other values that the attacker can compute.

Step 17 $16, 19, 32$	$L \rightarrow 25$.
Step 18 $19, 28, 35$	$L \rightarrow 22$.
Step 19 $16, 22, 29$	$L \rightarrow 13$.
Step 20 $s_{t+13} \oplus z_{t+13}$	$= f_{t+13}$.
Step 21 $R2_{t+14}$	$S \rightarrow R1_{t+13}$.
Step 22 $s_{t+28}, R1_{t+13}, f_{t+13}$	$F \rightarrow R2_{t+13}$.
Step 23 $R2_{t+13}$	$S \rightarrow R1_{t+12}$.
Step 24 $s_{t+12} \oplus z_{t+12}$	$= f_{t+12}$.
Step 25 $R1_{t+12}, R1_{t+13}, f_{t+12}$	$G \rightarrow R2_{t+12}$.
Step 26 $R1_{t+12}, R2_{t+12}, f_{t+12}$	$F \rightarrow 27$.

As an update, the attacker has now guessed or determined the values:

$$\text{0-2, 12-17,19,21-23, 25, 27-32,35}, R1_{t+i}, R2_{t+i},\ i \in \{\text{0-3,12-18}\}.$$

The final steps are:

Step 27 $1, 13, 27$	$L2 \rightarrow -5$.
Step 28 $16, 30$	$L2 \rightarrow (s_{t-2} \oplus s_{t+4})$.
Step 29 $-5, (s_{t-2} \oplus s_{t+4})$	$L \rightarrow 11$.
Step 30 $11, 14, 27$	$L \rightarrow 20$.
Step 31 $s_{t+11} \oplus z_{t+11}$	$= f_{t+11}$.
Step 32 $R2_{t+12}$	$S \rightarrow R1_{t+11}$.
Step 33 $R1_{t+11}, R1_{t+12}, f_{t+11}$	$G \rightarrow R2_{t+11}$.
Step 34 $R1_{t+11}, R2_{t+11}, f_{t+11}$	$F \rightarrow 26$.
Step 35 $17, 20, 26$	$L \rightarrow 33$.
Step 36 $s_{t+33}, R1_{t+18}, R2_{t+18}$	$F \rightarrow f_{t+18}$.
Step 37 $z_{t+18} \oplus f_{t+18}$	$= s_{t+18}$.
Step 38 $15, 18, 31$	$L \rightarrow 24$.

The attacker has now determined a full LFSR state $(s_{t+26}, \ldots s_{t+11})$. The LFSR is "rewound" to determine $(s_{t+15}, \ldots, s_t)$.

The probability that the three Phase One assumptions are correct is 2^{-95}. The attacker will have to try around 2^{95} values of t before finding internal states where the assumptions are correct; that is, the attacker will need $O(2^{95})$ outputs. For each value of t, the attack requires guessing 129 bits (the MSB of $R1_{t+3}$ and all 32 bits of s_t, s_{t+1}, $R1_t$, $R2_{t+14}$). This corresponds to 2^{129} guesses for each value of t. Thus, the data complexity of the attack is 2^{95} and the process complexity is $2^{95} \cdot 2^{129} = 2^{224}$.

It is possible to reduce the data complexity of the second attack by allowing other values of $R1_{t+3}$. However, if $R1_{t+3}$ is no longer either 0 or 2^{31}, then there will be more than one possible value for $(s_{t+3} \oplus s_{t+18})$. The attacker can test the possible combinations of $(R1_{t+3}, (s_{t+3} \oplus s_{t+18}))$, starting with the most probable combinations and leading down to the less probable combinations. This results in an increase in the process complexity. We have not conducted a detailed analysis of the expected complexity using this approach.

5 Conclusion

This paper demonstrates two guess-and-determine attacks on SNOW. The first attack has a process complexity of $O(2^{256})$ and a data complexity of $O(2^{64})$. The second attack has a process complexity of $O(2^{224})$ and a data complexity of $O(2^{95})$. It is likely that guess-and-determine attacks could be forced to have a larger complexity if the linear recurrence were changed, as discussed in Section 3.4.

References

1. S. Blackburn, S. Murphy, F. Piper, and P. Wild. A SOBERing remark. Technical report, Information Security Group, Royal Holloway University of London, Egham, Surrey TW20 0EX, U.K., 1998.
2. D. Bleichenbacher and S Patel. SOBER cryptanalysis. *Fast Software Encryption, FSE'99 Lecture Notes in Computer Science, vol. 1636, L. Knudsen ed., Springer-Verlag*, pages 305–316, 1999.
3. P. Ekdahl and T. Johansson. SNOW - a new stream cipher, 2000. This paper is found in the NESSIE webpages: http://www.cosic.esat.kuleuven.ac.be/nessie/workshop/submissions/snow.zip.
4. P. Hawkes and G. Rose. Exploiting multiples of the connection polynomial in word-oriented stream ciphers. *Advances in Cryptology, ASIACRYPT2000, Lecture Notes in Computer Science, vol. 1976, T. Okamoto ed., Springer-Verlag*, pages 302–316, 2000.
5. P. Hawkes and G. Rose. Primitive specification and supporting documentation for SOBER-t16 submission to NESSIE, 2000. See: http://www.cosic.esat.kuleuven.ac.be/nessie/workshop/submissions/sober-t16.zip.
6. P. Hawkes and G. Rose. Primitive specification and supporting documentation for SOBER-t32 submission to NESSIE, 2000. See: http://www.cosic.esat.kuleuven.ac.be/nessie/workshop/submissions/sober-t32.zip.