World Applied Sciences Journal 11 (2): 190-195, 2010
ISSN 1818-4952

Traceable Bit Streams in SNOW 2.0 using Guess-and-Determine Attack

[1]Syed IrfanUllah, [2]Tarannum Naz and [3]Sikandar Hayat Khiyal

[1]Department of Computer Science, Faculty of Applied Sciences,
International Islamic University, H-10 Islamabad, Pakistan
[1,2]Fatima Jinnah Women University, The Mall, Old Presidency Rawalpindi, Pakistan

Abstract: Word Oriented Stream Ciphers is an efficient class of stream ciphers in which basic operation is performed on a block of bits called word. Word Oriented Stream Ciphers become very popular because they generate block of several bits instead of one bit per clock. SNOW family is a typical example of word oriented stream ciphers based on Linear Feedback Shift Register (LFSR). In this paper we discuss SNOW family against Guess-and-Determine (GD) Attack. Original SNOW 2.0 is an improved version of SNOW 1.0 claimed to be more secure and efficient in performance. The model claims that it is secure against Guess-and-Determine attack but our analysis show that it contains a series of patterns of bits that is traceable. We analyze the key stream generated by the SNOW 2.0 key stream generator against Guess-and-Determine attack to find out the probability and variance of the traceable bit patterns in key. Our analysis shows that logic based stream ciphers have these traceable patterns those cannot be avoided using logic based approaches. The algorithm is evaluated in three phases. The first phase discusses theoretical background of algorithm, the second phase discusses methodology adopted for Guess-and-Determine attack and the third phase discusses statistical analysis of attack.

Key words: Stream Cipher · Linear FeedBack Shift Register · SNOW 2.0 · Modified SNOW 2.0 · Guess-and-Determine Attack

INTRODUCTION

In March 2000, the NESSIE project started its first announcement on cryptographic primitives from different fields of cryptography. Forty cryptographic primitives submitted to NESSIE project, there were five stream ciphers and SNOW is one of them. The very first version called SNOW 1.0 was submitted to NESSIE project in 2000. Some weaknesses were found on SNOW 1.0 [1] then a second version named SNOW 2.0 has been introduced. SNOW 2.0 is an improved version of SNOW 1.0, proposed by Patrick Ekdahl and Thomas Johansson in 2002. The new version is schematically a small modification of the original construction. Although SNOW 2.0 was appear to be more secure, but some weaknesses have been found in it [2], which results in proposal of Modified Version of SNOW 2.0 [1]. As Guess-and-Determine attack is proven to be effective against word-oriented stream ciphers, so we apply statistical approaches to analyze the traceability of similar bit-patterns.

Guess-and-Determine (GD) attacks are one of the general attacks which have been effective on some stream ciphers. As it comes from the name, in Guess-and-Determine attacks, we attempt to obtain the states of all cells of the whole cipher system by guessing the contents of some of them initially and comparing the resulting key sequence with the running key sequence. If these two sequences are the same, we consider the initial guess as a proper one, otherwise we should try another guess; thus, the complexity of these attacks has the same order as the complexity of exhaustive search of the basis of the guessed elements space. In spite of a long time devotion to GD attacks' improvements on word-oriented stream ciphers, they have often been implemented heuristically. For instance, heuristic GD attacks on SNOW1.0 can be studied in [2-4].

In [5] some new criteria classes were proposed to find a sub-optimum basis for implementing a GD attack against the underlying stream cipher systematically. According to

Corresponding Author: Syed IrfanUllah, Department of Computer Science, Faculty of Applied Sciences,
International Islamic University, H-10 Islamabad, Pakistan.
E-mail: syedirfan_phd@yahoo.com.

these criteria classes a new algorithm, called *Chess Algorithm,* was introduced. Based on the novel idea [5], the Advanced GD attacks were introduced [6], which lead to the improvement of many previously heuristic GD attacks on some stream ciphers. For example, the computational complexities of the attacks on SNOW1.0 and SNOW2.0 were reduced from $O(2^{224})$ and $O(2^{288})$ to $O(2^{205})$ and $O(2^{267})$ respectively. It is worth mentioning that there are also better attacks on these ciphers, e.g., a distinguishing attack on SNOW2.0 with computational complexity of $O(2^{202})$ has recently been proposed in [7] which is the best published one so far [8].

SNOW 1.0 was captured by GD attack in a way that, Finite State Machine (FSM) has only one input function s(1) to the FSM. It enables an attacker to invert the operations in the FSM to derive more unknowns from only a few guesses [1]. SNOW 2.0 was captured by correlation attack.

Description of SNOW 2.0: The word size of SNOW 2.0 remains same (32 bits) as of SNOW 1.0 and length of LFSR is again 16, but the feedback polynomial is different. The Finite State Machine (FSM) has two input words instead of one, taken from the LFSR and the running key is generated by XOR between FSM output and the last entry of the LFSR, as in SNOW 1.0

The operations of cipher are as follows, first of all key initialization is performed. This operation provides starting states to LFSR and to give initial values to the internal FSM registers R1 and R2. There is a small difference in the operation of the cipher. In the first version, after the key initialization, the first symbol was read out before the cipher was clocked but in second version it is read out after the cipher is clocked once [9].

In SNOW 2.0, two different elements involved in the feedback loop, α and α^{-1}. SNOW 2.0 takes two parameters as input value, a secret key of either 128 or 256 bits and a publicly known 128-bit initialization value IV. The IV value is considered as a four word input IV = (IV_3, IV_2, IV_1, IV_0) where IV_0 is the least significant one. The possible range for IV is thus 0……2^{128}-1. This means that for a given key K, SNOW 2.0 implements a pseudorandom length increasing function from the set of IV values to the set of possible output sequences.

Key Initialization Process: SNOW 2.0 takes two parameters as input values; a secret key of either 128 or 256 bits and a publicly known 128 bit initialization variable IV. The IV value is considered as a four word input IV = (IV_3, IV_2, IV_1, IV_0), where IV_0 is the least significant word. The possible range for IV is thus 0…2^{128} – 1. This means that for a given secret key K, SNOW 2.0 implements a pseudo-random length-increasing function from the set of IV values to the set of possible output sequences.

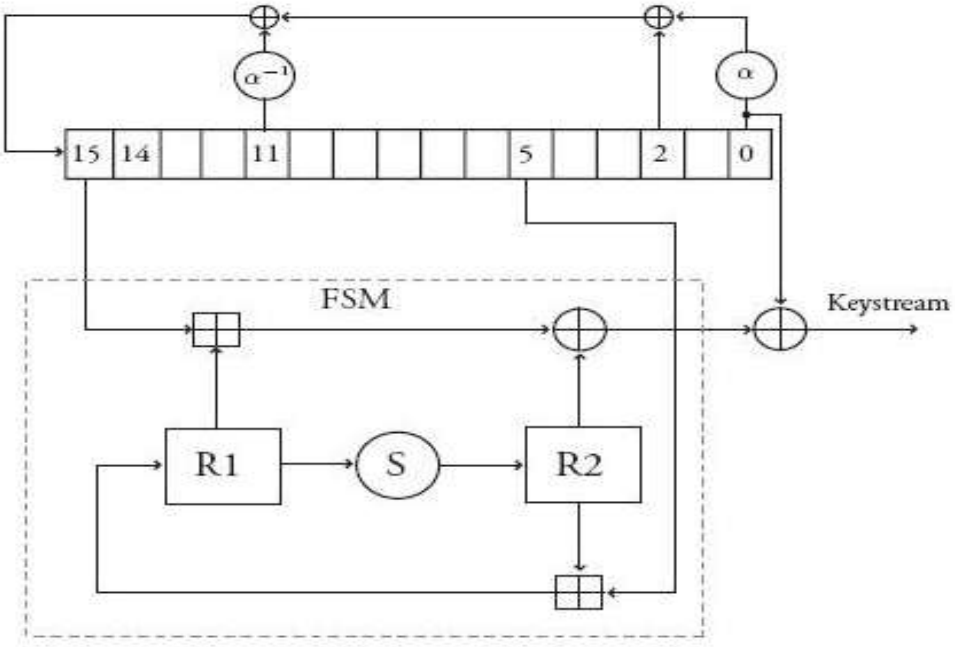

Fig. 1: A schematic model of SNOW 2.0

The model of SNOW 2.0 is shown in Figure 1, where we denote addition in F_2^{32} by the symbol $\oplus$, addition modulo 2^{32} by the symbol $\boxplus$, multiplication with α by α and S-Box operation is denoted by S.

The key initialization is done as follows. Denote the registers in the LFSR by (S_{15}, S_{14}, ……., S_0) from left to right in Figure 1.

Thus S_{15} corresponds to the element holding S_{t+15} during normal operation of the cipher. Let the secret key be denoted by K= (K_3, K_2, K_1, K_0) in the 128 bit case and K = (K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0) where each K_i is a word and K_0 is the least significant word [9]. First, the shift register is initialized with K and IV as follows,

s15 = k3⊕IV0, s14 = k2, s13 = k1, s12 = k0 ⊕ IV1, s11 = k3 ⊕ 1, s10 = k2⊕IV2, s9 = k1⊕1⊕IV3, s8 = k0⊕ 1,
and for the second half of the register,
s7 = k3, s6 = k2, s5 = k1, s4 = k0, s3 = k3 ⊕ 1,
s2 = k2 ⊕ 1, s1 = k1 ⊕ 1, s0 = k0 ⊕ 1,

where 1 denotes the all one vector (32 bits).

After the LFSR has been initialized, R1 and R2 are both set to Zero. Now the cipher is clocked 32 times without producing any output symbol. Instead, the output of the FSM is incorporated in the feedback loop as shown in Figure 2. Thus during the 32 clocks in the key initialization, the next element to be inserted into the LFSR is given by $S_{16}=\alpha^{-1} S_{t+11} \oplus S_{t+2} \oplus \alpha S_t \oplus F_t$ [9].

After the 32 clockings the cipher shifted back to normal operation as shown in Figure 1 and is clocked once before the first keystream symbol is produced [9].

Design of Dictionary Through Guess-and-determine Attack: After the designing of SNOW 2.0, the next step is to design an attack for SNOW 2.0. Reliability of SNOW 2.0 has to be checked against Guess-and-Determine (GD) Attack, due to that reason Guess-and-Determine (GD) Attack has been designed. It is designed in a way that

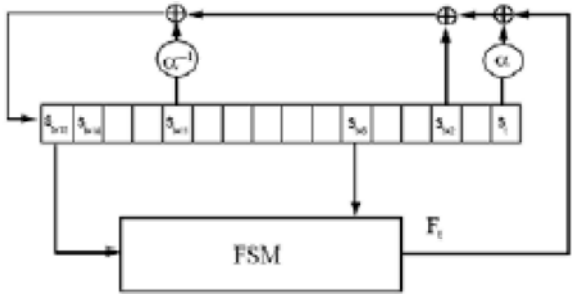

Fig. 2: Structure of SNOW 2.0 during Key Initialization

guess has been made on secret key and initialization values and determines running keystream on basis of these guesses. Guess-and-Determine (GD) Attacks can be considered as general attacks on stream ciphers. In the proposed system the Guess-and-Determine (GD) Attack is used to guess the secret key and initialization values. It is described below.

Step I: Make guess of secret key and IV values. For this purpose we have used the random function, which will take random values from given range.

Step II: Convert the guessed key into binary form because SNOW is additive Stream Cipher (i.e. a cipher in which plain text, cipher text and secret key must be in binary form).

Step III: Transform binary form of guessed key into 32-bits because 32 bit registers are used for storage and operations.

Step IV: Now secret keys and IV values are ready to initialize the cipher.

Step V: Proper working has been started and keystream are generated and stored in an attacking file.

Comparison Algorithm: Read the next keystream from original file when first keystream of original file is compared with all attacking keystream.

```
Key_comparison
    {
    Set IV ← input through keyboard
    Key = snow2 ( IV )
    vcount = 0
    j = 0
    do{
    hcount = 0

    for i ← 0 to n
    {
    if (key(i) = = dic_key(i,j))
    {
    put 1 to file
    hcount = hcount++
    vcount(i,j)++
    }
```

```
else
{
put 0 to file
}

}
j++
}
while (!eof)
}
```

The algorithm executes in this fashion to compare the original key with each and every keystream of attack file so that to locate the specific byte positions in key, those can be traced by Guess-and-Determine (GD) attack.

Experimental Analysis: A large number of experiments were conducted and its data was analyzed through statistical tools to find out the frequency of each byte position, so as to achieve the goal of finding traceable bit patterns in key. For this purpose we compare the original keystream with attacking keystream through comparison algorithm. Different data sets were taken to minimize biasness in the population; therefore the experiment was conducted on small datasets as well as large data sets. The complete analysis includes two phases.

In phase I, we determine the traceable bit patterns in key, a number of experiments were conducted on comparatively small randomly generated dictionaries and we find the following facts:

The GD attack with low intensity i.e. only with 500 guess keys, it was found that the frequency of 0,1,2,3 was 64.2, 29.2, 5.6 and 1.0 respectively, i.e. out of 500 guess keys 64.2% keys were not matching at byte position, but 1.0% of the keys had the successful match at 3 locations, i.e. 5 out-of 500 keys has 3 byte similarities which is too less if we guess the entire key.

$$f(x_i) \in f(x'_i),$$

where i is the byte position where one byte is equal to 8 bits, so if we compare it one key only which has 3 matches has 24 bits similarity with a specific position, which means that 24 bits have been traced in a pattern with a population of size 500. Figure 3 shows the frequency of byte similarity.

Figure 4 represents a graph shows the similarity of the key with the dictionary bit position wise which may give clue to traceable bit stream patterns.

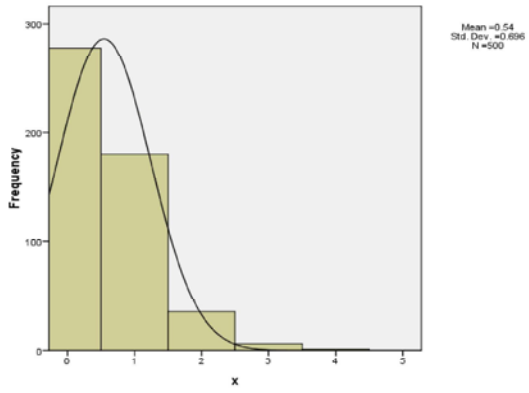

Fig. 3: Frequency graph for dictionary of 500 elements

Fig. 4: Frequency graph for bit stream patterns for population of size 500

The graph in Figure 4 shows that byte positions 2, 3, 6 are more frequent as compared to the others, the dictionary is tested for a number of data sets and it was shown that these byte positions are more vulnerable to the cryptanalyst.

When the population is increased to 200000, it is found that more than 60% of the key is traced; by having 5 byte similarities in 8 byte long key.

In a population of 200000 guess keys, it was found that the frequency of 0, 1, 2, 3, 4, 5 was 59.756, 31.75, 7.43, 0.9775, 0.0835 and 0.0030 percent respectively, i.e. out of 200000 guess keys 59.756% keys were not matching at byte position, but 0.0030% of the keys had the successful match at 5 locations, i.e. 6 out-of keys 200000 has 5 byte similarities which is too less if we guess the entire key.

The graph in Figure 5 represents the similarity of keys with the dictionary byte position wise which may give clue to traceable bit stream patterns.

Table 1: Frequency table for a dictionary of 200000 attacking keys

Row-wise			
		Frequency	
Percent		Valid Percent	
Cumulative Percent			
Valid	0	119512	59.
	59.8	59.8	
	1	63500	31.
	31.8	91.8	
	2	14860	7.
	7.4	98.9	
	3	1955	0.
	1.0	99.9	
	4	167	1.
	.1	100.0	
	5	6	0.
	.0	100.0	
	Total	200000	100.
	100.0		

Table 2: Byte position frequency for dictionary of 200000 attacking keys

Position			
		Frequency	
Percent		Valid Percent	
Cumulative Percent			
Valid	0	12343	12.
	12.4	12.4	
	1	12516	12.
	12.5	24.9	
	2	12569	12.
	12.6	37.5	
	3	12463	12.
	12.5	50.0	
	4	12522	12.
	12.5	62.5	
	5	12583	12.
	12.6	75.2	
	6	12405	12.
	12.4	87.6	
	7	12382	12.
	12.4	100.0	
	Total	99783	100.
	100.0		

Table 3: Cumulative frequency for data of different sizes

	Byte Position								
Sample Size	0	1	2	3	4	5	6	7	Total
500	11.1	11.1	14.7	15.2	11.1	10.6	16.1	10.1	100
1000	12.4	10.4	13.8	13.4	11.6	12.2	14.1	12.0	99.9
2000	11.5	12.2	12.5	13.5	13.3	10.4	13.2	13.5	100.1
5000	12.4	13.8	12.2	12.3	13.4	12.4	12.7	10.8	100
10000	12.7	12.7	12.7	13.0	12.2	12.3	12.4	12.0	100
20000	12.3	12.1	12.7	12.9	12.1	12.9	12.2	12.8	100
30000	12.6	12.6	12.2	13.0	12.1	12.1	12.5	12.9	100
40000	12.6	12.6	12.4	12.1	12.4	12.9	12.4	12.6	100
50000	12.6	12.4	12.5	12.5	12.3	12.6	12.5	12.6	100
60000	12.3	12.2	12.5	12.6	12.5	12.8	12.4	12.7	100
70000	12.4	12.3	12.3	12.3	12.5	12.9	12.8	12.4	99.9
80000	12.7	12.4	12.4	12.5	12.4	12.4	12.8	12.4	100
90000	12.4	12.1	12.7	12.4	12.6	12.5	12.7	12.6	100
100000	12.7	12.7	12.3	12.3	12.6	12.4	12.4	12.6	100
200000	12.4	12.6	12.5	12.5	12.6	12.5	12.4	12.4	99.9
C. %age	12.34	12.28	*12.69333*	*12.83333*	12.38	12.26	*12.88571*	12.29333	

The following table represents frequency of each byte position and its percentage which represents that positions 2, 5 are more frequent as compared to other byte positions so, we can see that data at position 2 and 5 are more vulnerable to the cryptanalyst.

The graph in Figure 6 shows the similarity of the key with the dictionary byte position wise which may give clue to traceable bit stream patterns.

The graph shows that byte positions 2, 6 are more frequent as compared to the others, the dictionary is tested for a number of data sets and it was shown that these byte positions are more vulnerable to the cryptanalyst.

This graph represents that there is uniformity in each byte position and the frequency is also approximately same, but if we attempt even more clever attacks then we see that some of the byte positions are vulnerable to the cryptanalyst. For this purpose we collect data from different population sizes and we pass it through statistical tests it is observed that data at location 2, 3, 6 is comparatively more frequent.

Cumulative Analysis: The following table contains location based frequency for all experiments and its relative frequencies:

From this graph it is clear that byte position 2, 3 and 6 are more frequent as compared to the other ones.

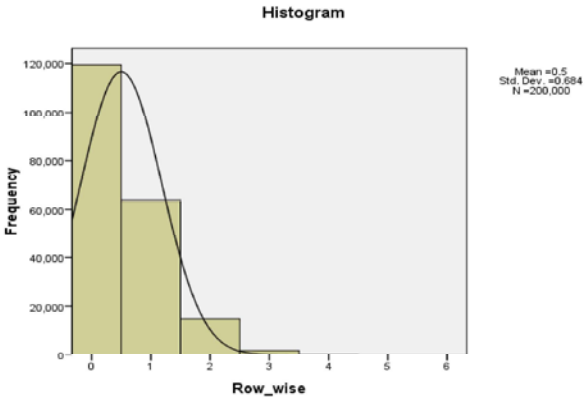

Fig. 5: Frequency graph for dictionary of 200000 elements

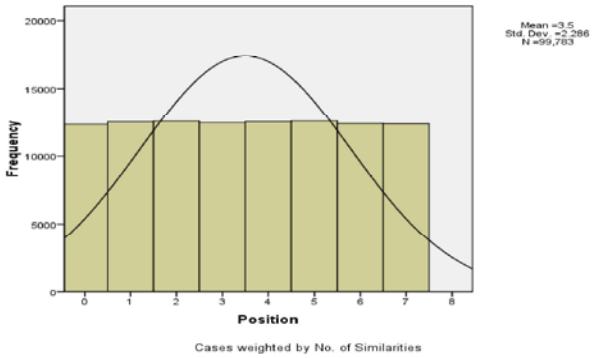

Fig. 6: Frequency graph for bit stream patterns for population of size 200000

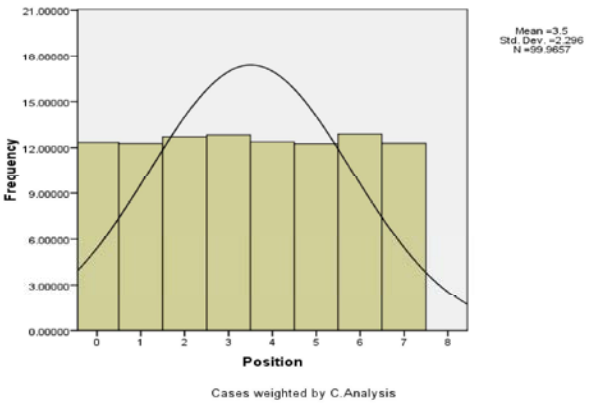

Fig. 7: Cumulative Frequency graph for bit stream patterns based on data in table 3

On comparing the original key with the dictionary we find that in 8 byte long key 5 bytes were traced to be similar with six different attacking keys in a population of 200000 attacking keys which is 0.0030% of the entire population, where as 4 bytes were traced to be similar with 167 different attacking keys which is 0.0835% of the entire populations. On choosing more clever initial guesses.

CONCLUSION

In this paper we have introduced the statistical analysis of the Guess-and-Determine attack on the stream cipher SNOW 2.0, based on advanced Guess-and-Determine attacks. The study is limited to the byte tracing instead of bit tracing, so as to justify the bulk amount of similar bit patterns in the original key. This study opens a way that in Guess-and-Determine attacks if some portion of the key is traced then it is preserved and recycled for tracing other similar bit patterns.

REFRENCES

1. Ullah, S.I., A.A. Tabassam and S.H. Khiyal, 0000. “Cryptographic weaknesses in modern stream ciphers and recommendations for improving their security levels”.
2. Ekdahl, P., 2003. "On LFSR based Stream Ciphers-Analysis and Design", Ph.D. Thesis, Dept. of Information Technology, Lund University.
3. Hawkes, P. and G. Rose, 2002. Guess and Determine Attacks on SNOW. Qualcomm Australia, SAC 2002, LNCS., 2595: 37-46.
4. De Canniere, C., 2001. Guess and Determine Attack on SNOW. NESSIE Public Document,” NES/DOC/KUL/WP5/011/a, See http://www.cryptonessie.org.
5. Mohammadi Chambolbol, A., 2004. Cryptanalysis of Word-Oriented Stream Ciphers, MSc. Thesis, Sharif University of Technology, Sept. 2004.
6. Ahmadi, H. and T. Eghlidos, 2005. “Advanced Guess and Determine Attacks on Stream Ciphers”, IST 2005, pp: 87-91.
7. Maximov, A. and A. Johansson, 2005. “Fast Computation of Large Distributions and Its Cryptographic Applications”. Lecture Notes in Computer Science ASIACRYPT, pp: 313-332.
8. Ahmadi, H. and Y.E. Salehani, 0000. “A Modified Version of SNOW2.0”
9. Ekdahl, P. and T. Johansson, 2002. "A new version of the stream cipher SNOW", Proceedings of Selected Areas in Cryptography (SAC) 2002, Springer, 2002. Available at, citeseer.ist.psu.edu/ekdahl02new.html