Abstract

In this paper we discuss security problems in some modern stream ciphers. As we observe sometimes that a designer claims that the algorithm designed is more secure but when it comes to open literature we find a number of problems. We discuss SNOW, Scream and Rabbit. Some efforts have been made to overcome the problems those were pointed out in these cryptosystems by different cryptanalysts. The stream ciphers are faster and efficient than block ciphers but comparatively less secure. Our emphasis in this paper is to make some compromise on efficiency but to get more security.

Index Terms - Stream ciphers, Block ciphers, SNOW, Scream, Rabbit, chaotic.

I. Introduction

First we discuss SNOW 1.0. The idea for its design is taken from the classical summation generator [1]. The design of the cipher is quite simple, consisting of LFSR i.e. linear shift feedback register feeding a finite state machine. The new version of this stream cipher is SNOW 2.0[2], which is more secure and a bit faster as compared to SNOW 1.0. Although some minor changes have been made in SNOW 1.0 as the word size has not been changed (32 bit) and the LFSR length is again 16 [2]. It is claimed that SNOW 2.0 is more secure and has more resistance against guess and determine attacks. Although some minor changes have been made in original SNOW and we got its more secure version. So, by making very few but careful changes we can close the entry points for ciphers. Then we discuss Scream, which is considered to be a more secure SEAL [9]. We have made some changes to make it even more secure as some of the analysts have found out weaknesses in the original Scream.

Then we discuss Rabbit, which is considered to be a high performance stream cipher [3]. We analyze the security of this model against different possible attacks and the resistance of the model against these attacks; we found out some weaknesses, those are removed by making some changes in the model. The organization of this paper is as follows:

In section II we discuss the shortcomings in original cryptosystem and their resistance against different attacks. In section III we present our work that how these systems can be improved with less overhead and minimal changes. In section IV we analyze these improved models that why these are so logical and providing more strength to the cryptosystems. We conclude and summarize our work in section V.

II. Analyzing the models

In this section we discuss the shortcomings of the cryptosystems SNOW 1.0, Scream and Rabbit respectively.

A. SNOW 1.0

The SNOW 1.0 is a word oriented stream cipher with word size of 32 bits [1]. The cipher is described with two possible sizes i.e. 128, 256 bits. As usual encryption starts with key initialization, giving the components of the cipher their initial key value [1, 2]. The guess and determine attacks has the data complexity of 295 words and the process complexity of 2224 operations [4], if we have clever initial choices then of the complexity can be decreased even more. There are some weaknesses in SNOW 1.0, which also reduces the complexity of the attack below the exhaustive key search [2].

Finite State Machine (FSM) has only one input function s (1). It enables an attacker to invert the operations in FSM to derive more unknowns from only a few guesses. There is an unfortunate choice of
feedback polynomial in SNOW 1.0. The linear recurrence equation is given by:

\[ S_{t+16} = \alpha (S_{t+9} + S_{t+3} + S_t) \]  

(1)

There is a distance of 3 words between \( S_t \) and \( S_{t+3} \) and a distance of 6 = 2.3 between \( S_{t+3} \) and \( S_{t+9} \). Thus by squaring

\[ S_{t+32} = \alpha^2 (S_{t+18} + S_{t+6} + S_t) \]  

(2)

We can see that \((S_{t+i} \oplus S_{t+i+6})\) can be considered as a single input to either equation. Hence, the attacker does not need to determine \( S_{t+i} \) and \( S_{t+i+6} \) explicitly but only the XOR sum to use in both ones [2].

The choice of the feedback polynomial emerges when considering bitwise linear approximations. Using the same techniques as in [11] we can take the 2\(^{32}\)th power of the feedback polynomial i.e.

\[ P(x) = x^{16} + x^{13} + x^7 + x^{-1} \in F_2^{32}[x] \]  

(3)

\[ P(x) = x^{16}, 2^{32} + x^{13}, 2^{32} + x^7, 2^{32} + x^{-1}, 2^{32} \in F_2^{32} \]  

[x]  

(4)

Since \( \alpha^{-1} \in F_2^{32} \) we have \( \alpha^{-1}, 2^{32} = \alpha^{-1} \) summation of \( p(x) \) and \( p^{-32}(x) \) yields

\[ x^{16}, 2^{32} + x^{13}, 2^{32} + x^7, 2^{32} + x^{16} + x^{13} + x^7 \]  

(5)

dividing this equation by \( x^7 \) we get linear recurrence equation satisfying

\[ S_{t+16}, 2^{32} + S_{t+13}, 2^{32} + S_{t+7}, 2^{32} + S_{t+9} + S_{t+6} + S_t = 0 \]  

(6)

We derive the linear recurrence equation that holds for each single bit position. Hence, any bitwise correlation found in the FSM can be turned into a distinguishing attack. For correlation and distinguishing attacks we need about \( 2^{35} \) words of output and the computational complexity about \( 2^{100} \) [6]. This discussion proves that SNOW 1.0 is a very weak cryptosystem and some changes were made to improve its quality; those were addressed in the design of SNOW 2.0 [2], which is also having some problems. We discuss it in section III-A

B. Scream: A software efficient stream cipher

Scream is a software efficient stream cipher that was designed to be a more secure SEAL [7, 9]. This model resembles in many ways a block-cipher design but it offers a significantly higher level of security. As SEAL is specialized in encrypting small messages and data authenticity, that has a number of weaknesses [7]. One of the proposed attack [8] that requires 230 “samples”, each 4-words long to distinguish SEAL from random function. The new version SEAL 3.0 was proposed but it was also having some weaknesses, so based on the design of SEAL a new model Scream was proposed in different versions. The first version Scream that is so-called toy cipher due to its simplicity has many more weaknesses.

For scream family of cipher two distinguishing attacks are proposed. The best one has the complexity around 280 [8]. Although if we implement distinguishing attack on scream then the attack uses 2105 output words and has complexity of a similar size. We can decrease the complexity of the attack against scream by improving the model of attack [10]. Some other attacks proposed for scream are linear attacks and low diffusion attacks [9], show that what ever the claims are, we can distinguish the cipher from a truly random sequence which tends the system to cryptographic weaknesses and invite new attacks.

The main problem is in the for-loop where we find X block which is 16 bytes long, where it is XORed directly with Y and we send it to the function F( ) as argument, that may allow the ciphers to distinguish the key from a truly random sequence.

C. Rabbit

The design of Rabbit was inspired by the complex behavior of real-valued chaotic maps. These maps are primarily characterized by an exponential sensitivity to small perturbations causing iterates of such maps to seem random and long-time unpredictable [11]. These properties have also previously leaded to suggestions that chaotic systems can be used for cryptographical purposes [12, 13]. Due to this modern method used for designing the model makes cryptanalysis difficult but even then efforts have been made to analyze it. The cryptanalysis of the Rabbit is resulted in the following:

To investigate the possibilities for Divide-and-Conquer and Guess-and-Determine types of attacks, an algebraic analysis was performed with special attention on the nonlinear parts of the next state function, as they are the main sources of mixing the input bits [14, 15]. Some of the attacks like correlation and distinguishing attacks have got some sort of success against it. The literature gives us some
clue the there are some deficiencies in the model through which we can peep in to the interior of the model specifically due to some weaknesses in the key scheduling algorithm of the model.

There is a possibility of related key attacks that exploit the symmetries of the next state and key setup function. For instance consider two keys $K$ and $K'$ related by $K[i] = K'[i+32]$ for all $i$. This lead to the relation, $X_{j,0} = X'_{j+2,0}$ and $C_{j,0} = C'_{j+2,0}$. In the same way this symmetry could lead to a set of bad keys, i.e. if $K[i] = K[i+32]$ for all $i$, then $X_{j,0} = X_{j+2,0}$ and $C_{j,0} = C_{j+2,0}$ and in this way the key can be traced.

We need to take care of the brute force attack, and other classical and modern attacks. Due to advancement in mathematics we cannot guarantee that a cryptosystem will always be secure and foolproof but we can give more tough time to cryptanalysts.

III. Recommendations for improvements in models

In this section we present the changes those we have made in the original models. In section II we discussed the problems in the original models. To minimize their severity we make the following changes.

A. SNOW 1.0

In section II-A the design analysis of SNOW 1.0 has been discussed in more detail and it has been shown that SNOW 1.0 is a very weak stream cipher. So, these weaknesses were addressed and removed in new version of SNOW i.e. SNOW 2.0 [2], but very soon a number of weaknesses were addressed by different analysts [4, 6] and proved that this version is also not so secure.

So, we make some changes in the basic design of SNOW 2.0 as SNOW 1.0 has already been improved to release its next version. Both versions of SNOW have basically the same design but some minor changes, so the basic design remains same for SNOW.

a. Convert the LFSR property to partial NLFSR:
   i. Take circular left shift after XORing $\alpha$ (alpha) with $S_{t+2}$.
   ii. Take circular left shift again after XORing $\alpha^{-1}$ (alpha inverse) with $S_{t+11}$.

b. Before XORing with $R_2$ take circular left shift of ($R_1 S_{t+13}$) once again.

We represent the model diagrammatically with the above improvements. The diagram clearly represents that we need memory buffers at three different places to store the bit stream on temporary basis.

Figure 1: The new model of SNOW 2.0

B. Scream: A software efficient stream cipher

Some of the attacks although less efficient have been found, changes have been made in it [9] to improve the security but the efficiency have been lost by 10% --15% although [10] analyze it for security. So, we make some changes as such to improve the security as well as less compromise on performance 2% --5%.

The main weaknesses are in the for-loop where we find $X$ block that is 16 bytes long, where it is XORed directly with $Y$ and we send it to the function $F()$ as argument, that may allow the ciphers to distinguish the key from a truly random sequence. The changes that we have made are as follows:

a. Divide $X$ block into 16 chunks, each one byte long.
b. Re-compute each byte by XORing it with the next byte.
c. Before XORing with the next byte, rotate it one bit to the left.
d. With the last byte the first recently computed byte is XORed.
e. Recombine all the bytes to get a final 16 bytes block.

f. Compute X = F(X ⊕ Y).

In this way we enjoy the stream cipher advantages 16 byte blocks. As stream ciphers are speedily computable, so the above computation takes less time as well as the non-linearity has also been maintained and thus security is improved.

C. Rabbit

We see in section II-C that there are some weaknesses in the key scheduling of Rabbit as we make some changes in its design that generates the key stream as such to minimize the attacks on it.

In the next state functions every X affects the next second state which is easily traceable so we make the following changes:

a. Make the map as such that the current state X affects the next X state and that X affects the next second state and that affects the next third state. Thus there is no linear increase in the state scheduling. At the fifth state the process is repeated once again as for the first four states.

b. The process continues till all states change their values at least once.

As in SNOW 2.0 α (alpha) and α⁻¹ (alpha inverse) are XORed but even then they can be traced if we are able to succeed in finding α. So, left circular shift makes it difficult to trace the sequence as every time the sequence is circulated that is stored inside temporary buffer at three different places in our model. The input to the FSM is (S_{t+15}, S_{t+5}) but S_{t+15} is the sequence number after left circular shift, so the output of the FSM denoted by F_t, is computed as:

\[ F_t = ((\text{LeftCircularShift} (S_{t+15}) \oplus R1_t) \ll n \oplus R2_t) \]

IV. Analysis of the improved models

In this section we prove that the changes we make are very logical and important. The security of the cryptosystems has been improved by making these efforts.

A. SNOW 1.0

Both versions of SNOW are based on LFSR so, the linear attacks are more efficient to analyze and crack them. We convert their design to Partial Nonlinear Feedback Shift Registers i.e. PNLSRs, so the linear attacks are mostly discouraged. We give a partial touch of chaotic map [13] in our model. Chaotic maps give more strength to the security of cryptosystems. It also discourages the guess and determine attacks. The overhead that we make decreases the performance of the system. In the original model the linear recurrence equation is given by:

\[ S_{t+16} = \alpha (S_{t+9} + S_{t+3} + S_t) \]

(7)

There is a distance of 3 words between \( S_t \) and \( S_{t+3} \) and a distance of 6 = 2.3 between \( S_{t+3} \) and \( S_{t+9} \). Thus by squaring:

\[ S_{t+32} = \alpha^2 (S_{t+18} + S_{t+6} + S_t) \]

(8)

As in SNOW 2.0 α (alpha) and α⁻¹ (alpha inverse) are XORed but even then they can be traced if we are able to succeed in finding α. So, left circular shift makes it difficult to trace the sequence as every time the sequence is circulated that is stored inside temporary buffer at three different places in our model. The input to the FSM is (S_{t+15}, S_{t+5}) but S_{t+15} is the sequence number after left circular shift, so the output of the FSM denoted by F_t, is computed as:

\[ F_t = ((\text{LeftCircularShift} (S_{t+15}) \oplus R1_t) \ll n \oplus R2_t) \]
where n is a chosen prime number.
The linear or distinguishing attacks do not affect the model even if we have more clever attacks. The key sequence can not be determined by linear sequence. The guess and determine attacks are also discouraged as these are based on initial guesses, but we see that due to repeated circular shift the guess is in contradiction with the exact value as the cipher is always confused about the exact location of the bit due to its circular shifts.
We do not make severe changes in the basic design of the cryptosystem to keep the thing same as in the original model. The model is so weak that by making changes we can decrease the severity of the attack but we can not fully stop attacks on it. The drawback in our model is that we store the key sequence at three different locations in temporary buffers that makes the model a little inefficient.

**B. Scream: A software efficient stream cipher**

It is an advanced stream cipher that provides better security. It is more efficient than SEAL, but even then we look in Scream-0 that some security flaws are there, specially in statement X := F(X ⊕ Y). As X and Y both are XORed in F ( ), which give way to ciphers for tracing the key. The F ( ) function has linear approximations that approximate only three of the 8-by-8 S-boxes. Since the S-boxes in Scream-0 are based on the Rijndael S-box, the best approximation of them has bias 2^3. So, we can probably get a linear approximation of the F ( ) function with bias 2^9.

For linear approximation we need to eliminate the linear masking, as each of these masks is used 16 times before it is modified. For each step the cipher looks a pair (X ⊕ Y ⊕ W[i], F (X) ⊕ Z ⊕ W[i+1]), where X is random. So if X is found once then we can trace the whole sequence. So in scream-F this problem was removed but it lost the efficiency by 10-15%.

So we make changes such that to improve the security as was challenged by direct XORing of X and Y. The basic quality of the model has also been restored. The theory of chaotic maps has been introduced in the model which provides better security by splitting the X in to 16 individual bytes and each one has been XORed with the next one. So, it makes it difficult for cipher to trace the sequence by applying the linear approximation and the (X ⊕ Y ⊕ W[i], F (X) ⊕ Z ⊕ W[i+1]) has been converted to even more complex shape i.e. (X_i ⊕ X_{i-1} <<< 1) ⊕ Y ⊕ W[i], F (X) ⊕ Z ⊕ W[i+1]). The pair will never give the required result to cipher as it changes all the times. So, the cipher will not easily trace the sequence as well as the efficiency has also been restored up to some extant that was the basic purpose of the model to provide speedy encryption.

The basic purpose of the model has been restored along with the improved security to stop linear, differential and guess and determine attacks.

**C. Rabbit**

It is based on chaotic map but there are very little design weaknesses that allows cipher to peep in to the model. We see that every current state affect linearly the next second state. We can determine how this state affects the coming state as mathematically represented in the section II-C

In original model the g-function is computed by making some operation on the state of the map [11]. We see that every state is computed by XORing the previous two states as follows.

\[
X_{0,i+1} = g_{0,i} + (g_{i,j}<<<16) + (g_{6,i}<<<6) \quad (10)
\]
\[
X_{1,i+1} = g_{1,i} + (g_{6,i}<<<8) + g_{7,i} \quad (11)
\]
\[
X_{2,i+1} = g_{2,i} + (g_{j,i}<<<16) + (g_{0,i}<<<16) \quad (12)
\]
\[
X_{3,i+1} = g_{3,i} + (g_{2,i}<<<8) + g_{1,i} \quad (13)
\]

The g-function is the computed in straight forward mathematical function.

\[
g_{ji} = ((X_{j,i} + C_{ji})^2 \oplus ((X_{j,i} + C_{ji})^2 \gg 32)) mod 2^{32} \quad (14)
\]

This g-function can be summarized as follows.

\[
g(Y) = (Y^2 \oplus (Y^2 >> 32)) mod 2^{32} \quad (15)
\]

This g-function has a very huge complexity and only with very small word length it has been examined [14]. This discussion reveal that the non-linear order of this function is maximal and only through high-level differential it can be examined due to some weaknesses mentioned here.

In our model every current state affects the next states in different manner which makes it difficult to analyze it in a linear and even in differential way. The guess and determine attacks have also been discouraged.

The differential analysis is impossible because we are not always sure after how much iteration every state is affected, if we increase the number of states for
achieving more security the mathematical representation is only possible by making some logics. So, even a clever attacker is always confused about the state of the art. The speed of the cryptosystem is not affected as graphical and mathematical model is changed through logics only.

V. Conclusion

We discussed some of exemplary stream ciphers those are claimed by their designers to be more secure, but we saw that every model has been traced when they came to open literature. We observed that every model has left some very minor weaknesses those may be considered as ignorance of designers or there may be some other things that every designer had left some peeping point to make it always possible for them to crack their own models. We saw that with minor efforts those peeping points have been closed. The world is invited for any suggestion related to the paper will be encouraged to improve our models.

References