American Journal of Scientific Research
ISSN 2301-2005 Issue 74 September, 2012, pp.87-107

http://www.eurojournals.com/ajsr.htm

Minimizing the Intensity of Guess and Determine Attack on Snow 2.0 using Intelligent Algorithms

Arshad Iqbal
IBMS Khyber Pakhtunkhwa Agricultural University Peshawar
E-mail: arshadiqbal84@hotmail.com

Abdusalam
Corresponding Author, Lecturer, Institute of Management Studies
University of Peshawar, Pakistan
E-mail: salamkhan4u@gmail.com

Muhammad Amin
Assistant Professor, Department of Computer Science & Telecom
Iqra National University Peshawar
E-mail: mamin@inu.edu.pk

Zahoor Jan
Assistant Professor, Department of Computer Science
Abdul Wali Khan University Mardan
E-mail: zahoor.jan@awkum.edu.pk

Syed Irfan Ullah
Director Noor Software, Silicon Center Peshawar
E-mail: syedirfan_phd@yahoo.com

Abstract

SNOW 2.0, a software oriented stream cipher proposed by T. Johansson and P. Ekdahl in 2002 as an enhanced version of the NESSIE finalist SNOW 1.0, is usually considered as one of the strongest Stream Ciphers designed so far. This study described Guess and Determine (GD) Attack on SNOW 2.0. In this study, we located the specific bit positions in key using comparison algorithm, those can be traced by Guess and Determine (GD) Attack. After locating the specific bit positions in key, we developed Intelligent Algorithm to avoid the traceable bit locations for minimizing the intensity of Guess and Determine (GD) Attack on SNOW 2.0.

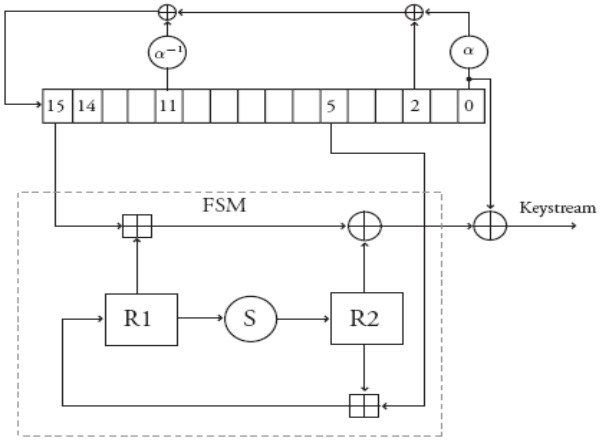
Keywords: Guess and Determine, AGD, Linear Feedback Shift Register, Finite State Machine

1. Introduction

1.1. Snow 2.0

SNOW 2.0 is an improved version of SNOW 1.0.The word size remains same (32 bits) and length of Linear Feedback Shift Register is again 16, but the feedback polynomial is different. The Finite State Machine has two input words instead of one, taken from the Linear Feedback Shift Register, and the running key is generated by XOR between FSM output and the last entry of the Linear Feedback Shift Register, 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 as well as to also give initial values to the internal Finite State Machine (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 the second version it is read out after the cipher is clocked once (Ekdahl and Johansson, 2002). In SNOW 2.0 there are 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.

Figure 1.1: A schematic model of SNOW 2.0



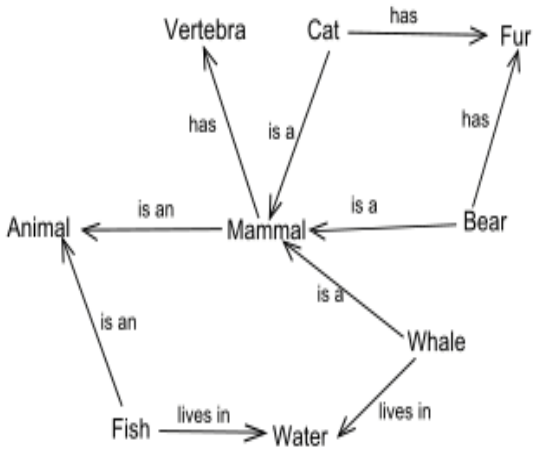
1.2. Guess and Determine (GD) Attack

GDattacks are one of the general attacks which have been effective on some Stream Ciphers. As it comes from the name, in GD 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 (Ahmadi and Salehani, 2004).

1.4. Semantic Network

A Semantic Network is a network, which represents semantic relations between the concepts. This is often used as a form of knowledge representation. It is a directed or undirected graph consisting ofvertices, which represent concepts, and edges (Allen and Frisch, 1982).

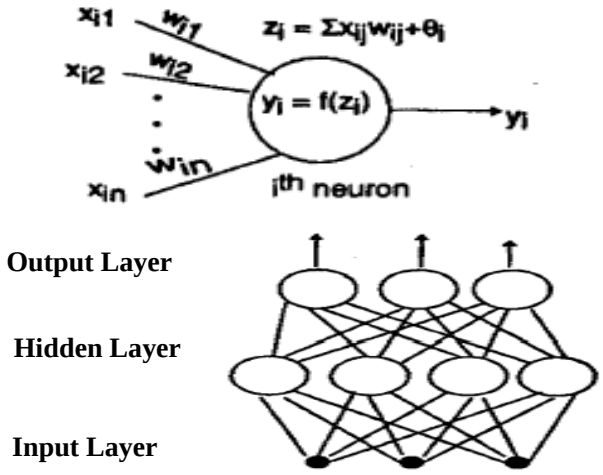
Figure 1.4: Example of a semantic network



1.5. Artificial Neural Network

Artificial Neural Network (ANN) is the field of science that tries to replicate brain-like computing. The brain is understood to use massively parallel computations where each computing element (a neuron or brain cell in the terminology of this science) in the massively parallel system is envisioned to perform a very simple computation, such as Yi = *f (zl),* where *zi*is assumed to be a real valued input, *Yi* is either a binary or a real valued output of the i^{th} neuron, and f a nonlinear function (figure 1.5). The nonlinear function f, also called a node function, takes different forms in different models of the neuron; a typical choice for the node function is a step function or a sigmoid function. The neurons get their input signals from other neurons or from external sources such as various organs of the body like the eyes, the ears and the nose. The output signal from a neuron may be sent to other neurons or to another organ of the body (Roy, 2000).

Figure 1.5: Example of an Artificial Neural Network



1.6. Motivation

Without good cryptographic protection of data communication, it is practically effortless and perhaps even routine for an opponent to intercept the data. Especially those sent through a modem or email system. For this purpose many designs of Stream Ciphers have been proposed in an effort to find a proper candidate to be chosen as a world standard for data encryption. These Stream Ciphers should be secure and trustworthy. And the trustworthiness of Stream Ciphers may well be verified by method of cryptanalysis. Over the last few years, a growing but limited numbers of papers have been published

proposing two or more designs of single Stream Cipher and state that, their version is more reliable the previous one. So it is essential to give information to the open world that how can minimize the intensity of GD Attack on Stream Cipher i.e. SNOW 2.0 using Intelligent Algorithms.

1.7. Problem Statement

The problem with existing stream cipher i.e. SNOW 2.0 is that it suffered from Guess and Determine (GD) Attack. A GD Attack guesses some internal valuesand then exploits the relationships (such as the recurrence relationship in a shift register) to determine other internal values. The cipher is "broken" when a complete internal state has been determined from the guessed valuesso the cryptanalyst easily decrypts the cipher text into plain text. Due to GD Attack, the integrity of cipher text is lost.

1.8. Proposed Solution

The following are possible proposed solutions of the above mention problem,

- To create Original Keystream using SNOW 2.0.
- To design GD Attack and to apply on SNOW 2.0 in order to analyze the impact.
- To create Comparison Algorithm so that to locate the traceable bit positions in key produced by SNOW 2.0.
- Adopting Intelligent Algorithms to avoid the traceable bit locations in key produced by SNOW 2.0 so that to minimize the intensity of Guess and Determine (GD) Attack on SNOW 2.0.

2. Previous Research

Nybergand Wallen (2006) proposed new and more accurate estimates of the biases of the linear approximation of the Finite State Machine (FSM) of the Stream Cipher SNOW 2.0. Based on improved bias estimates we also find a new linear distinguisher with bias 2^{-869} that is significantly stronger than the previously found ones by Watanabe *et al.* (2003) and makes it possible to distinguish the output keystream of SNOW 2.0 of length 2^{174} words from a truly random sequence with workload 21^{74}. This attack is also stronger than the recent distinguishing attack by Maximov and Johansson (2005). We also investigate the diffusion properties of the MixColumn transformation used in the Finite State Machine (FSM) of SNOW 2.0 and present some evidence why much more efficient distinguishers may not exist.

Kiyomoto*et al.*(2005) discussed Guess and Determine (GD) Attackson Clock-Controlled Stream Ciphers.The main focus was an experimental analysis of the influence of irregular clocking and presented Guess and Determine (GD) Attacks on AA5,and the nimplemented miniature models of Stream Ciphersin order to evaluate the proposed attacks and demons tratethattheseattackswereapplicable.Furthermore,theproposedschemewasshowntobe applicabletootherClock-ControlledStreamCiphers,andpropertiesofClock-Controllersandtheireffectivenessagainst Guess and Determine (GD) Attacks.

Olivier and Henri (2005), SNOW 2.0, a software oriented stream cipher proposed by Johansson and Ekdahl in 2002 as an enhanced version of the NESSIE finalist SNOW 1.0, is usually considered as one of the strongest stream ciphers designed so far. It investigate the resistance of SNOW 2.0 against algebraic attacks. This is motivated by the fact that the main source of non-linearity in SNOW 2.0 comes from a permutation build upon the Advance Encryption Standerd (AES) S-box, which inputs and outputs are well known to be related by numerous quadratic equations.

Johansson and Spaanenburg (2005)Stream Ciphers like SNOW 2.0 are very promising techniques for encryption in trusted hardware, but demand specialized IP cores to enhance

conventional architectures. It described the design of such a core that can beadapted to the system needs according to a ratio of throughput and effective slice usage of 3.2 to 3.5.

Ahmadi and Salehani (2004) introduced some criteria of modifying a given Linear Feedback Shift Register (LFSR) based stream cipher with respect to general attacks i.e. Guess and Determine (GD) Attacks and introduced a modified version of the stream cipher SNOW2.0. In order to evaluate the resistance of the modified SNOW2.0 against Guess and Determine (GD) Attacks, they used the application Advanced Guess and Determine (AGD) Attacks. Table 2.1 shows the results of applying Advance Guess and Determine (AGD) Attacks on the Original and Modified SNOW2.0.

Table 2.1: Results of applying AGD attacks on the two versions of SNOW2.0

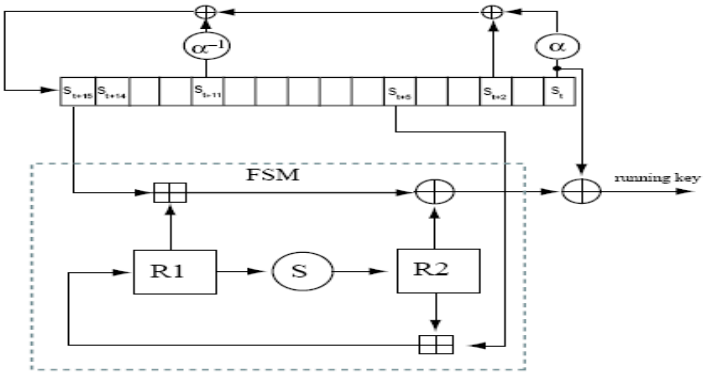
Implementation Result/ Cipher System	Guess and Determine (GD) Attack scenario	Complexity
Original SNOW 2.0	Guess: 8 words + Solving equations	≈ 2267
Modified SNOW 2.0	Guess: 9 words + Solving equations	≈ 2297

The results show that the security of SNOW2.0 against such Guess and Determine (GD) Attacks have increased by 30 bits, using such a modification. In other words, the security margin of the Modified SNOW2.0 is about 41 bits (297-256) which is more trustable than the Original SNOW2.0.

Hawkes and Rose (2003) proposed GD Attacks on the Stream Cipher i.e. SNOW. A GD Attack guesses some internal values and then exploits the relationships to determine other internal values. The cipher is "broken" when a complete internal state has been determined from the guessed values**.**

Ekdahl and Johansson (2002)proposed a new version of Stream Cipher SNOW 2.0. The design is based on the NESSIE proposed SNOW 1.0 and addresses all weaknesses found in the original construction. The implementation is easier and encryption is faster than SNOW 1.0. Typical encryption speed is over 3Gbits/sec on a Intel Pentium 4 running at 1.8 GHz.The new version SNOW 2.0 is schematically a small modification of the original construction (figure 2.1). The word size is unchanged (32 bits) and the LFSR length is again 16, but the feedback polynomial has been changed. The Finite State Machine (FSM) has two inputs, taken from the Linear Feedback Shift Register(LFSR) and the running key is formed as the XOR between the Finite State Machine (FSM) output and the last element of the Linear Feedback Shift Register(LFSR), as done in SNOW 1.0. The operation of the cipher is as follows. First, a key initialization is performed. This operation provides the Linear Feedback Shift Register(LFSR) with a starting state as well as giving the internal Finite State Machine (FSM) resistors R1 and R2 their initial values. Next the cipher is clocked once and the first keystream symbol is read out. Then the cipher is clocked again and the second keystream symbol is read, etc.

Figure 2.1: Schematic picture of SNOW 2.0



Ekdahl and Johansson (2000)developed a new word-oriented stream cipher, called SNOW. The design of the cipher is quite simple, consisting of a Linear Feedback Shift Register (LFSR), feeding a Finite State Machine. The design goals of producing a Stream Cipher significantly faster than AES,

with significantly lower implementation costs in hardware, and a security level similar to AES is currently met.The proposed Stream Cipher is a word oriented additive Stream Cipher, where a word in the specification is chosen to be 32 bits. The cipher is described with two possible key sizes, 128 and 256 bits. As usual, the encryption starts with a key initialization, giving the components of the cipher their initial key values. For the moment, we assume that the key initialization is done, and concentrate on the cipher in operation. The generator is depicted in figure 2.2. It consists of a length 16 LFSR over 2, feeding a finite state machine. The FSM consists of two32 bit registers, called R1and R2, as well as a some operations to calculate the output and the next state (the next value of R1 and R2). The Finite State Machine (FSM) is shown in figure 2.3.

Figure 2.2: The Generator SNOW

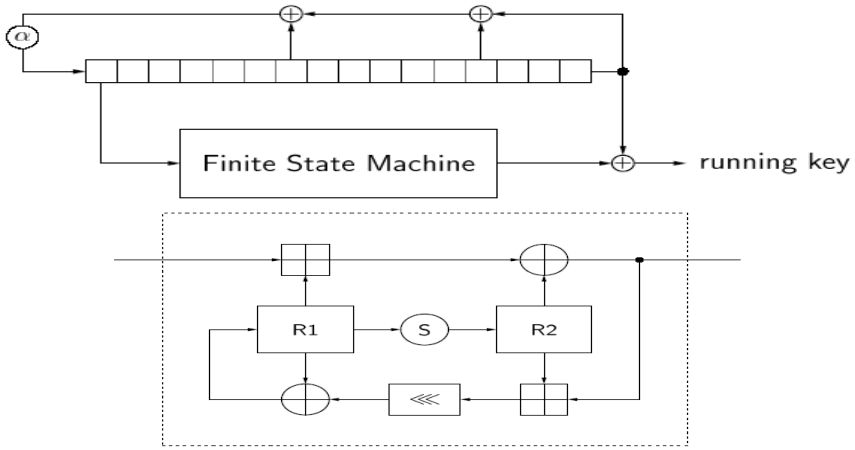
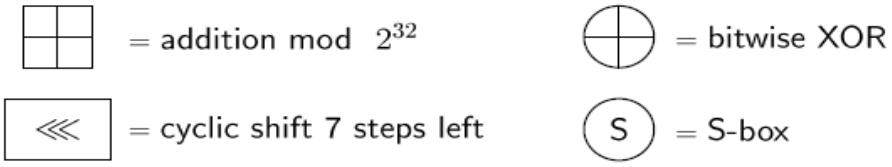


Figure 2.3: The Finite State Machine



The operation of the cipher is as follows. First, key initialization is done. This procedure provides initial values for the Linear Feedback Shift Register (LFSR)a swell as for the R1, R2 registers in the Finite State Machine (FSM). Next, the first 32 bits of the running key is calculated by bit wise adding the output of the Finite State Machine (FSM) and the last entry of the Linear Feedback Shift Register(LFSR). After that the whole cipher is clocked once, and the next 32 bits of the running key is calculated byagain bit wise adding the output of the finite state machine and the last entry of the Linear Feedback Shift Register(LFSR). We clock again and continue in this fashion.

Roy(2000) pointed out some very serious misconceptions about the brain in connectionism and Artificial Neural Networks. Some of the connectionist ideas have been shown to have logical flaws, while others are inconsistent with some commonly observed human learning processes and behavior. For example, the connectionist ideas have absolutely no provision for learning from stored information, something that humans do all the time. The article also argues that there is definitely a need for some new ideas about the internal mechanisms of the brain. It points out that a very convincing argument can be made for a "control theoretic" approach to understanding the brain. A "control theoretic" approach is actually used in all connectionist and neural network algorithms and it can also be justified from recent neurobiological evidence.

3. Research Methodology

3.1. Methodology

To attain the final purpose, the entire development was divided into the following tasks.

Task I: Study and analysis of SNOW 2.0.
Significance: It helped to understand working of SNOW2.0.
Task II: Application of Guess and Determine (GD) Attack on SNOW 2.0.
Significance: To analyze the behavior of cipher after applying attack.
Task III: Comparison between Original keystreams and Attacking Keystreams.
Significance: To locate the traceable bit positions in key produced by SNOW 2.0.
Task IV: Adopting Intelligent Algorithms
Significance: To avoid the traceable bit locations in key produced by SNOW 2.0.

3.2. Main Steps of the Proposed System

3.2.1. Design of SNOW 2.0

SNOW 2.0 has been implemented by P. Ekdahl and T. Johansson, they discussed some aspects of designing of SNOW 2.0 which have high impact in implementation software, as follows,

They started withLinear Feedback Shift Register (LFSR). The field F_2^{32} is defined as an extension field over F_2^8, with $\alpha \in F_2^{32}$ being the root of the degree 4 polynomial

$$x^4 + \beta^{23}x^3 + \beta^{245}x^2 + \beta^{48}x + \beta^{239} \in F_2^8[x] \quad (1)$$

Hence, they have the degree reduction of α given by

$$\alpha^4 = \beta^{23}\alpha^3 + \beta^{245}\alpha^2 + \beta^{48}\alpha + \beta^{239} \quad (2)$$

In the feedback loop, multiplication with α and α^{-1} can be implemented as a simple byte shift plus and additional XOR with one of 256 possible patterns. This can be seen from the representation of a word as a polynomial in F_2^8 [x] using (α^3, α^2, α, 1) as base. Thus, any element w in F_2^{32} can be written as

$$w = c_3\alpha^3 + c_2\alpha^2 + c_1\alpha + c_0, \quad (3)$$

where (c_3, c_2 , c_1 , c_0) are the bytes of w, c_0 being the least significant byte. Multiplying w with α, will yield a reduction according to (2) as follows

$$\alpha w = c_3\alpha^4 + c_2\alpha^3 + c_1\alpha^2 + c_0\alpha \quad (4)$$

$$= (c_3\beta^{23} + c_2)\alpha^3 + (c_3\beta^{245} + c_1)\alpha^2 + (c_3\beta^{48} + c_0)\alpha + c_3\beta^{239} \quad (5)$$

Similar calculations can be done for the multiplication with α^{-1}. Thus, to get a fast implementation of the Linear Feedback Shift Register (LFSR) feedback, one can use precomputed tables

$$MUL_\alpha[c] = (c\beta^{23}, c\beta^{245}, c\beta^{48}, c\beta^{239}) \quad (6)$$

$$MUL_\alpha^{-1}[c] = (c\beta^{16}, c\beta^{39}, c\beta^{6}, c\beta^{64}) \quad (7)$$

Where c runs through all elements in F_2^8. The pseudo-code for the multiplication would be

```
// Multiplication w*alpha ("<<" is left shift, ">>" is right shift)
result = (w<<8) XOR MUL_a[w>>24];
//Multiplication w*alphaˆ-1
result = (w>>8) XOR MUL_ainverse [w and 0xff];
```

The S-box is implemented using the same techniques as done in Rijndael (Daemen and Rigmen, 2002) and Scream (Coppersmith *et al*., 2002). Recall the expression for the S-box, r = S (w)

$$\begin{pmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \end{pmatrix} = \begin{pmatrix} x & x+1 & 1 & 1 \\ 1 & x & x+1 & 1 \\ 1 & 1 & x & x+1 \\ x+1 & 1 & 1 & x \end{pmatrix} \begin{pmatrix} S_R[w_0] \\ S_R[w_1] \\ S_R[w_2] \\ S_R[w_3] \end{pmatrix} \quad (8)$$

The matrix multiplication can be split up into linear combinations of the columns

$$\begin{pmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \end{pmatrix} = S_R[w_0] \begin{pmatrix} x \\ 1 \\ 1 \\ x+1 \end{pmatrix} + S_R[w_1] \begin{pmatrix} x+1 \\ x \\ 1 \\ 1 \end{pmatrix} + S_R[w_2] \begin{pmatrix} 1 \\ x+1 \\ x \\ 1 \end{pmatrix} + S_R[w_3] \begin{pmatrix} 1 \\ 1 \\ x+1 \\ x \end{pmatrix}.$$

By using four table of words, each of size 256, defined by

$$T_0[a] = \begin{pmatrix} xS_R[a] \\ S_R[a] \\ S_R[a] \\ (x+1)S_R[a] \end{pmatrix}, T_1[a] = \begin{pmatrix} (x+1)S_R[a] \\ xS_R[a] \\ S_R[a] \\ S_R[a] \end{pmatrix},$$

$$T_2[a] = \begin{pmatrix} S_R[a] \\ (x+1)S_R[a] \\ xS_R[a] \\ S_R[a] \end{pmatrix}, T_3[a] = \begin{pmatrix} S_R[a] \\ S_R[a] \\ (x+1)S_R[a] \\ xS_R[a] \end{pmatrix},$$

They easily implemented the S-box by addressing the tables with the bytes (w_3, w_2, w_1, w_0) of the input word w. In pseudo-code they wrote

// calculate r = S-box (w)

r = T0 [byte0 (w)] XOR T1 [byte1 (w)] XOR T2 [byte2 (w)] XOR T3 [byte3 (w)];where byte0 (w) means the least significant byte of w, etc.

They have two different C implementations, both using tables for feedback multiplication and S-box operations. The first version (version 1) implements the LFSR with an array using the sliding window technique. This version is considered and "easy to read" standard reference version. The second version (version 2) implements the cipher with "hard coded" variables for the LFSR. This version produces 16.32 = 512 bits of keystream in each procedure call, corresponding to 16 consecutive clocking's. Table 4.1 indicates the speed of the two implementations versions. For the key setup in SNOW 1.0, the IV mode is used as reference, since it also uses 32 clocking's in the initialization phase. This accounts for a more reasonable comparison. The tests where run on a PCWith Intel 4 processor running at 1.8GHz, 512 Mb of memory. Each program was compiled using gcc with optimization parameter "-O3" and inline directives in the code (Ekdahl and Johansson, 2002).

Table 3.1: Number of cycles needed for key setup and cycles per word for Keystream generation on a Pentium 4 @ 1.8GHz.

Operation	SNOW 2.0	
	Version 1	Version 2
Key Setup	937	-
Keystream Generation	38	18

3.2.2. Design of Guess and Determine (GD) Attack

After the designing of SNOW 2.0, the next step was 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 was designed in a way that 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 was used to guess the secret key and initialization values. It is described below:

Step I: Guess of secret key and IV values was made. For this purpose we used the random function in which random values from given range were taken.

Step II: Next step was to 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: Binary form of guessed key was transformed into 32-bits because 32 bit registers were used for storage and performing operations.

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

Step V: After initialization, proper working was started and running keystreams were generated.

3.2.3. Comparison Algorithm

After the generation of keystreams of both SNOW 2.0 and Attack, the next step was to evaluate the attack.For this purpose, there were two possible ways. One way to make the comparison was through Microsoft Excel. And another way was to design an algorithm in any programming language tool. As number of generated keystreams was very large, therefore the handling of keystreams in Microsoft excel was quite difficult and time taking. So we designed an algorithm for comparison of keystreams in C++. Before explanation of comparing algorithm, description of keystream was given. Each keystream comprises of 8 alphanumeric. Keystreams generated by Original SNOW 2.0 and attacks were stored in separate files,

Keystreams generated by Original SNOW 2.0 was stored in a file named Original file.

Keystreams generated by Attack was stored in a file named Attack file.Working of algorithm as follows:

Step I: Input of keystream was taken from Original file and was stored in array of size 8.

Original File
ABCDEFHT
EFDHVFGT
JKLMASDE

Array 1

A	B	C	D	E	F	H	T

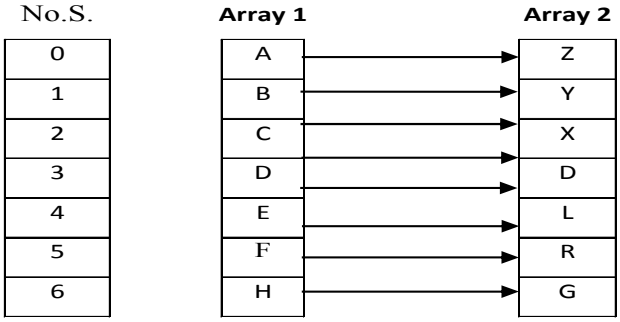
Step II: Input of keystream was taken from Attacking file and was stored in array of size 8.

Attack File
ZYXDLRGT
KGJTEADW
BNXCFSRT

Array 2

Z	Y	X	D	L	R	G	T

Step III: Both arrays were compared index wise.



Comparison was done in a way that if the value at index 1 of Original array was similar to the value of index 1 of attacking array, it was stored '1'in resulting file otherwise '0' was stored.

Table 3.2: Comparison of Keystreams

Index	Value at index of array 1	Matching (Equal/not equal)	Value at index of array 2	Result
0	A	!=	Z	0
1	B	!=	Y	0
2	C	!=	X	0
3	D	= =	D	1
4	E	!=	L	0
5	F	!=	R	0
6	H	!=	G	0
7	T	= =	T	1

Step III: All the keystreams were read from Attack file one by one and compared with the Original Keystream which was taken as input in step I.

Original File
ABCDEFHT
EFDHVFGT
JKLMASDE

Attack File
ZYXDLRGT
KGJTEADW
BNXCFSRT

Step IV: The resulting bitswere stored in another file for further calculations.This procedure continued until all attacking keystreams were being compared with first keystream of Original file.

Original Keystream

A	B	C	D	E	F	H	T

Attack File
ZYXDLRGT
KGJTEADW
BNXCFSRT

Comparison of first keystream with all Attacking Keystreams

Original Keystream

E	F	D	H	V	F	G	T

Attack File
ZYXDLRGT
KGJTEADW
BNXCFSRT

Comparison of second keystream with all Attacking Keystreams

Original Keystream

J	K	L	M	A	S	D	E

Attack File
ZYXDLRGT
KGJTEADW
BNXCFSRT

Comparison of third keystream with all attacking keystreams

Step V: The next keystream was taken from the Original file when the matching of first Keystream of Original file was completed with all Attacking keystreams.

The algorithm was executed in this fashion until all keystreams of Original file were compared with each and every keystream of Attack file. This helped to locate the specific bit positions in key, those can be traced by Guess and Determine (GD) Attack.

3.2.4. Adopting Intelligent Algorithms

After locating the specific traceable bit positions in key, Intelligent Algorithms were designed so that to avoid the traceable bit locations in key produced by SNOW 2.0.

3.3. Hardware and Software Requirements

- System with specification of at least 850MHz processor.
- 256 MB of RAM.
- Machine must be running Microsoft Windows Operating System.
- Register size 32.

3.4. Tools Used

C, C++, SPSS, and Microsoft Excel.

3.5. Functional Requirements

All functional requirements were described in the above mentioned tools.

3.6. Non Functional Requirements

These requirements include:

Reliability: Algorithms and implementation techniques must be reliable so that anyone can use it without any hesitation.

Usability: Usability addressed the relationship between a software tool and users. It should be simple. It should be readable and easily understandable.

Portability: Users should run on any machine there should be no hardware limitations

4. Implementation

The objective is to implement all tasks which are essential to accomplish the final aim of the thesis. C and C++ is the implemented platform for the application.

4.1. Implementation of SNOW 2.0

Implementation of SNOW 2.0 included the following steps,

1. Key Initialization
2. Clocking of cipher without producing any keystream.
3. Working of Finite State Machine (FSM).
4. Keystream Generation.

4.1.1. Key Initialization

First of all key initialization was performed, this operation provided starting values to LFSR as well as to internal registers of FSM that were R1 and R2.The function snow_loadkey() was used for this purpose. This function was taken six parameters, one was secret key k, second was fixed size of key (i.e. 128 or 256) and last four parameters were Initialization Values (IV0, IV1, IV2, IV3) (Annex-B).

4.1.2. Clocking of Cipher

After initialization of Linear Feedback Shift Register (LFSR) and Finite State Machine (FSM), the next step was clocking of cipher without producing any output symbol. Instead the output of the Finite State Machine (FSM) was incorporated in the feedback loop.

For this purpose the function of snow_feedback_clock() was used (Annexure-B).

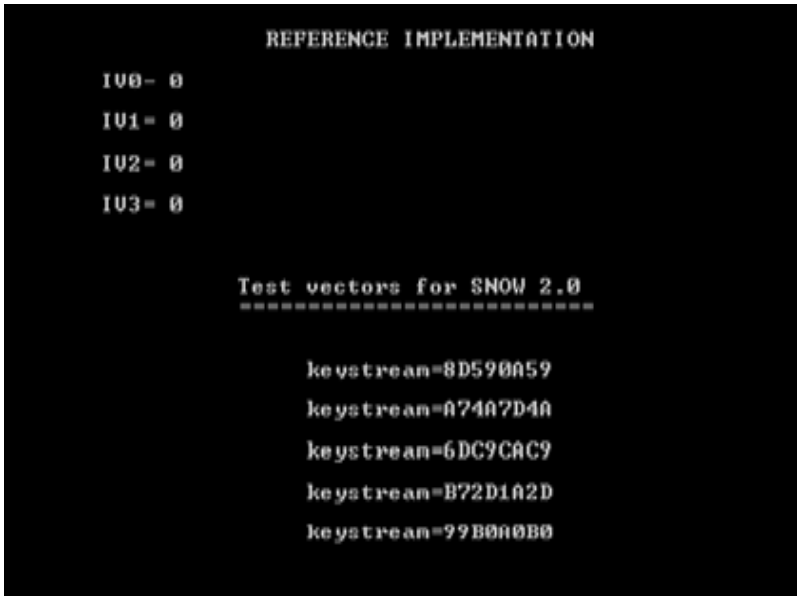
4.1.3. Working of Finite State Machine (FSM)

Snow_update_internals( ) was the function which was used to update the internal values of R1 and R2 which were two 32 bits registers of Finite State Machine (FSM) (Annexure-B).

4.1.4. KeystreamGeneration

After 32 clocks the cipher turned back to normal operation and was clocked once more before the key stream symbol was produced. Next the running key stream was calculated by bitwise adding the output of Finite State Machine (FSM) and last entry of the Linear Feedback Shift Register (LFSR).The function snow_keystream was used for the implementation of this step (Annexure-B).

Figure 4.1: Keystreams of Original SNOW 2.0

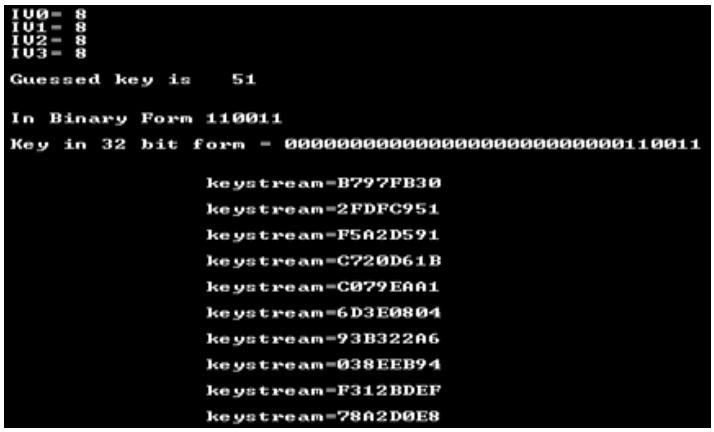


4.2. Implementation of Guess and Determine (GD) Attack

Guess and Determine (GD) Attack can be considered as general attack on Stream Ciphers. These attacks have often been implemented heuristically.It was obvious from section 4.1 that the implementation of SNOW 2.0 started with initialization process, that gave initial values to LFSR and FSM and further operations was performed by using these initial values.

The process of initialization was implemented by using snow_loadkey( ) function, since this function took six parameters, second parameter was fixed but remaining five were variables (i.e. secret key and IV values) that played vital role in giving initial values to Linear Feedback Shift Register (LFSR) and FSM. If an attacker traces out secret key and IV values then he/she can easily generate the running keystream that is originally generated. It can be concluded that, the security of whole cipher can be measured from initialization of LFSR process. GD Attack is well thought as serving instrument for determining the security of any cipher. Implementation of attack made guess on secret key and initialization values, and by using these guessed values it initialized the LFSR and FSM. After initialization of cipher the remaining procedure was same as followed by SNOW 2.0 (Annexure-B).

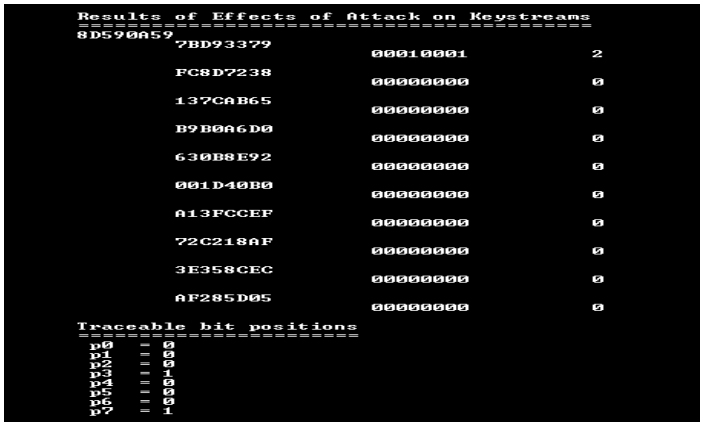
Figure 4.2: Keystreams of Guess and Determine (GD) Attack



4.3. Implementation of Comparison Algorithm

After implementation of Guess and Determine (GD) Attack, next step was to implement Comparison Algorithm using C++ programming language (Annexure-B).

Figure 4.3: Comparision of Original Keystreams and Attacking Keystreams and Results of traceable bit positons

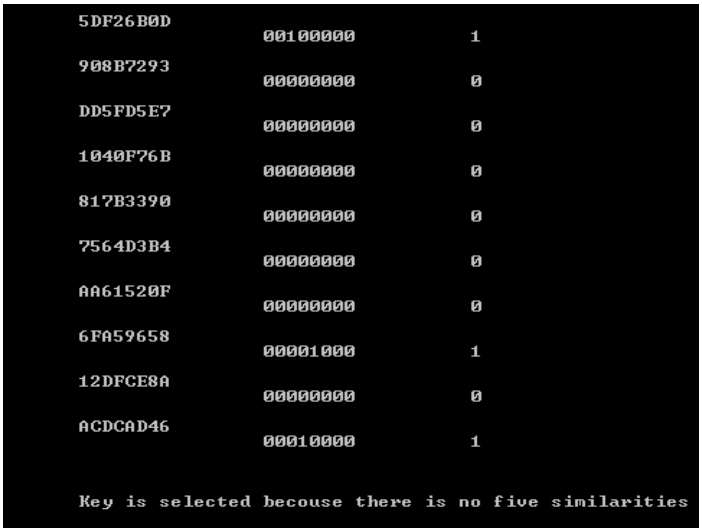


4.4. Implementation of Intelligent Algorithm

After implementation of Comparison Algorithm, we implemented Intelligent Algorithm for miniminzing of Guess and Determine (GD) Attack on SNOW 2.0 (Annexure-B).

Figure 4.4: Key is discarded due to finding five similarities

Figure 4.5: Now key is selected because there are no five similarities



5. Results and Discussion

This chapter presents the results of the analysis conducted in light of objectives outlined for the study. Necessary discussions have also been added along with the findings at appropriate places.

5.1. Experimental Analysis before using Intelligent Algorithm

A large number of experiments were conducted but to conserve space, only representative samples of the results were presented in this study. Our aim was to conduct these experiments to find the traceable bit positions in key. For this purpose we compared the Original Keystream with Attacking Keystream through comparison algorithm. So we started with the comparison of small data and then increased it. The complete analysis included two phases.

Phase I

In order to determine the traceable bit positions in key, 4 experiments have been done in this phase. Each experiment has one Original Keystream and 500, 2000, 5000 and 10,000 Attacking Keystreams respectively. The results are given below;

In first experiment the guess key is 3 and initialization values (IV) are IV0=2, IV1=2, IV2=2, IV3=2, and compared one Original Keystream (table 5.1) with 500 Attacking Keystreams (table 5.2), at row wise there are 126 times of zero similarities, 56 times of one similarity, and 35 times of two similarities, statistically Mean is 0.5804, Std. Deviation is 0.75412 and Variance is 0.569.

Now positions wise (column wise), there are 24 similarities on position zero, 24 similarities on position one, 32 similarities on position two, 33 similarities on position three, 24 similarities on position four, 23 similarities on position five, 35 similarities on position six and 22 similarities on position seven, statistically Mean is 3.5115, Std. Deviation is 2.22592 and Variance is 4.955 (Annexure-A).

Table 5.1: Keystream of SNOW 2.0

Original SNOW 2.0 **One Keystream**
BD5DA15D

Table 5.2: 500 Keystreams generated by GD Attack

Attacking Keystream				
500 Keystreams				
1887D6BC	29508881	754BF29D	89F3F573	FB0C5EAA
C8802C06	FBDDB6B2	0E5DA2D6	F32CD071	7FEC6753
F37D1CE1	F56E33D2	FE285C0E	C8889646	73A3645E
69BBF060	3B508011	9BD2C35A	2610E819	1F6F0DFC
.	.	.	.	.
.	.	.	.	.

In second experiment the guess key is 87 and initialization values (IV) are IV0=6, IV1=1, IV2=1, IV3=1 and applied 2000 attacks (table 5.3) on same Original Keystream (table 5.1), at row wise there are 1195 times of zero similarities, 645 times of one similarity, 140 times of two similarities, 19 times of three similarities and one time of four similarities, statistically Mean is 0.4930, Std. Deviation is 0.67318 and Variance is 0.453.

Now positions wise (column wise), there are 113 similarities on position zero, 120 similarities on position one, 123 similarities on position two, 133 similarities on position three, 131 similarities on position four, 103 similarities on position five, 130 similarities on position six and 133 similarities on position seven, statistically Mean is 3.5649, Std. Deviation is 2.28576 and Variance is 5.225 (Annexure-A).

Table 5.3: 2000 Keystreams generated by GD Attack

Attacking Keystream				
2000 Keystreams				
B13AF5D6	66D54BD6	72483655	3E36158D	55612A73
3345B0FE	4DABFD0D	E5BF8C42	912C7112	D771AFF9
5BE2DA08	22F2A300	FF54C2DF	4C10695D	2484C816
36DA9872	76FBA0BC	7F12DA4B	D6D23972	C1F66ED9
.	.	.	.	.

In third experiment the guess key is 44 and initialization values (IV) are IV0=2, IV1=6, IV2=6, IV3=6, and we increased attacks into 5000 attacks (table 5.4) and applied on same Original Keystream (table 5.1), at row wise there are 3031 times of zero similarities, 1567 times of one similarity, 348 times of two similarities, 53 times of three similarities and one time of four similarities, statistically Mean is 0.4852, Std. Deviation is 0.67459 and Variance is 0.455.

Now positions wise (column wise), there are 301 similarities on position zero, 336 similarities on position one, 296 similarities on position two, 298 similarities on position three, 326 similarities on position four, 300 similarities on position five, 307 similarities on position six and 262 similarities on position seven, statistically Mean is 3.4221, Std. Deviation is 2.26108 and Variance is 5.112 (Annexure-A).

Table 5.4: 5000 Keystreams generated by GD Attack

Attacking Keystream				
5000 Keystreams				
F9527786	952494BD	FB36ECC3	D61FBEB9	9ADA0658
AD215826	8288B3D6	FA29DC31	F98A5BE0	1492F7D2
C633EAB8	D02973E7	D0E42710	6A581DDE	416FE412
D075E8AC	5C1E9340	FB28DF9C	A12A0DAA	DCE4D295
82387D6A	6195EA04	E800A7EE	99DCAF6A	4816309E
B97F3FDF	22D89D7B	7DAF36AF	250BAB2B	C5737C70
A79DDA30	77689704	E220706D	4C900FED	37E05883
58E32026	069D894A	B0C4E722	9F23F6A5	336005DB

Now in fourth experiment the guess key is 76 and initialization values (IV) are IV0=3, IV1=3, IV2=8, IV3=8 and applied 10,000 attacks (table 5.5) on same Original Keystream (table 5.1), at row wise there are 5942 times of zero similarities, 3208 times of one similarities, 744 times of two similarities, 100 times of three similarities and 6 time of four similarities, statistically Mean is 0.5020, Std. Deviation is 0.68267 and Variance is 0.446.

Now positions wise (column wise), there are 639 similarities on position zero, 640 similarities on position one, 636 similarities on position two, 654 similarities on position three, 612 similarities on position four, 615 similarities on position five, 620 similarities on position six and 604 similarities on position seven, statistically Mean is 3.4552, Std. Deviation is 2.28577 and Variance is 5.225 (Annexure-A).

Table 5.5: 10,000 Keystreams generated by GD Attack

Attacking Keystream				
10,000 Keystreams				
7817E02B	FEC39B47	08F30D91	C03E95B0	C4F5CC0F
F5CBB3F3	ABF45E67	ED98706A	69D68694	EA512FD3
369841FB	57006A59	7263ABAB	A6A3D92C	C7DD2503
3C2AFFED	E93C5CC2	BFE493A4	714636CA	69A83B3A
E9092978	31157012	3D5D2F28	7CC38C8D	83223B02
ACE1EA0B	5B0CB212	AC108B65	E7FC9A8D	583CC843
CAF72C54	B3E1ED68	7C4AE8AE	FB84BC9F	41D32784
7E9141E4	F164A9CD	98AAA056	763AA637	0DF770FF

Phase II

In this phase, 5 experiments have been done. Each experiment has one Original Keystream and 40,000, 60,000, 80,000, 90,000 and 100,000 Attacking Keystreams respectively. The results are given below;

In first experiment the guess key is 99 and initialization values (IV) are IV0=1, IV1=5, IV2=5, IV3=5 and applied 40,000 attacks (table 5.6) on same Original Keystream (Table 5.1), at row wise there are 23762 times of zero similarities, 12924 times of one similarities, 2902 times of two similarities, 376 times of three similarities, 31 times of four similarities and 5 times of five similarities, statistically Mean is 0.5001, Std. Deviation is 0. .68067 and Variance is 0.463.

Now positions wise (column wise), there are 2512 similarities on position zero, 2515 similarities on position one, 2473 similarities on position two, 2427 similarities on position three, 2490 similarities on position four, 2584 similarities on position five, 2483 similarities on position six and 2521 similarities on position seven, statistically Mean is 3.5075, Std. Deviation is 2.29649 and Variance is 5.274 (Annexure-A).

Table 5.6: 40,000 Keystreams generated by GD Attack

Attacking Keystream				
40,000 Keystreams				
52DBF275	C1B96E26	C2137A02	B344B7B2	AA8C77D3
2B923ED1	647F8EED	8A981C0D	AD5B7698	EE21A13C
EAD70891	3A9A89AD	EA97C0CD	80478368	F0852F3F
46A7BD79	A05C4DF8	E1FA5205	14048794	E736CE38
25BA3BB4	64D990FB	D1D33D7E	301FF70C	0BFA666D
1B035DAC	B35E5A22	E2B7A910	76090414	5555C05F
.	.	.	.	.

In second experiment the guess key is 112 and initialization values (IV) are IV0=8, IV1=8, IV2=4, IV3=4 and applied 60,000 attacks (table 5.7) on same Original Keystream (table 5.1), at row wise there are 35705 times of zero similarities, 19232 times of one similarities, 4423 times of two

similarities, 601 times of three similarities, 34 times of four similarities and 5 times of five similarities, statistically Mean is 0. .5007, Std. Deviation is 0.68265 and Variance is 0.466.

Now positions wise (column wise), there are 3686 similarities on position zero, 3675 similarities on position one, 3756 similarities on position two, 3772 similarities on position three, 3767 similarities on position four, 3851 similarities on position five, 3728 similarities on position six and 3807 similarities on position seven, statistically Mean is 3.5232, Std. Deviation is 2.28639and Variance is 5.228 (Annexure-A).

Table 5.7: 60,000 Keystreams generated by GD Attack

Attacking Keystream				
60,000 Keystreams				
3089FD12	A10E1DEB	CC268F3A	4F36D125	26A7C1CC
D0F9E50D	281F5365	C792FE9E	4100DB2E	C87AFAB1
51DBB1E4	A7D77AFF	C497ECEC	0B1BDA93	10EF7E2F
6AA1CAF4	3740B619	30CA3D66	4A2918AC	B53C1219
E9635F13	73D156AC	70AF25E2	979B8455	A74A50C6
A432EE29	C1B0914E	A10B1C22	E45374B1	17597D15

In third experiment the guess key is 606 and initialization values (IV) are IV0=2, IV1=6, IV2=6, IV3=6 and applied 80,000 attacks (table 5.8) on same Original Keystream (table 5.1), at row wise there are 47687 times of zero similarities, 25478 times of one similarities, 5943 times of two similarities, 815 times of three similarities, 73 times of four similarities and 4 times of five similarities, statistically Mean is 0.5015, Std. Deviation is 0.68677 and Variance is 0.472.

Now positions wise (column wise), there are 5111 similarities on position zero, 4987 similarities on position one, 4969 similarities on position two, 4998 similarities on position three, 4958 similarities on position four, 4988 similarities on position five, 5153 similarities on position six and 4957 similarities on position seven, statistically Mean is 3.4971, Std. Deviation is 2.29656 and Variance is 5.274 (Annexure-A).

Table 5.8: 80,000 Keystreams generated by GD Attack

Attacking Keystream				
80,000 Keystreams				
E4F62D4C	00EB2BE1	3CA3B010	1E89825B	DDFAFE1B
6A02A03F	43B215E5	7A4FB357	59FA5DAD	90BF4F38
9F994977	B7139CFD	0DB887CA	33B26F0C	AF077443
BC19C2D7	12155727	CA96D2B8	5B449D8B	5490CCB3
FD67EF58	B81487BA	DF00516D	30BA093E	5AC7BB90
167FEE0E	961AC34B	105732D0	FBAE7BE2	584B7A6A
.	.	.	.	.

In fourth experiment the guess key is 814 and initialization values (IV) are IV0=6, IV1=6, IV2=1, IV3=1 and applied 90,000 attacks (table 5.9) on same Original Keystream (table 5.1), at row wise there are 53643 times of zero similarities, 28660 times of one similarities, 6742 times of two similarities, 875 times of three similarities, 75 times of four similarities and 5 times of five similarities, statistically Mean is 0.5010, Std. Deviation is 0.68503 and Variance is 0 .469.

Now positions wise (column wise), there are 5446 similarities on position zero, 5598 similarities on position one, 5745 similarities on position two, 5588 similarities on position three, 5661 similarities on position four, 5641 similarities on position five, 5721 similarities on position six and 5694 similarities on position seven, statistically Mean is 3.5234, Std. Deviation is 2.28584 and Variance is 5.225 (Annexure-A).

Table 5.9: 90,000 Keystreams generated by GD Attack

Attacking Keystream				
90,000 Keystreams				
0456FB09	F3CB3132	C06E98E4	892D8209	CFAC8EDE
9742DAA3	7CEF4BE7	FA2FD7F5	E01C3ACA	E5ECF44F
895A3D3C	687BA1AB	D5CC1CDB	98745F95	5EDA0A4A
DEE8254D	7547DB26	01DFF82D	EFA5FE33	42C614D9
5BD66A6D	91AF00F7	E66F4C08	C4130C91	7B25F8C3
FDA4B5BD	90AE5D7F	6A4AAF98	E3A8BA63	A3215C43
.	.	.	.	.

In fifth experiment the guess key is 449 and initialization values (IV) are IV0=6, IV1=6, IV2=6, IV3=2 and applied 100,000 attacks (table 5.10) on same Original Keystream (table 5.1), at row wise there are 59697 times of zero similarities, 31641 times of one similarities, 7523 times of two similarities, 1053 times of three similarities, 79 times of four similarities and 7 times of five similarities, statistically Mean is 0.5020, Std. Deviation is 0.68885 and Variance is 0 .475.

Now positions wise (column wise), there are 6393 similarities on position zero, 6387 similarities on position one, 6154 similarities on position two, 6165 similarities on position three, 6327 similarities on position four, 6225 similarities on position five, 6223 similarities on position six and 6323 similarities on position seven, statistically Mean is 3.4907, Std. Deviation is 2.30008 and Variance is 5.290 (Annexure-A).

Table 5.10: 100,000 Keystreams generated by GD Attack

Attacking Keystream				
100,000 Keystreams				
280C4D21	11EC733E	4D1021CB	E6E99632	FBF8E67A
E0960F3D	FADF69F1	264BB0DF	1752850A	B4BBCDEA
045683CE	1DE687BE	F4ACB0F7	8A874A39	43BE6F96
1DC9BF71	20B0E040	D58FEE4C	01281F12	6F5B8F59
1DA6EE5D	48BDE37A	5AD804D4	BA72F33F	F1F6E96D
8F61DA24	CB9AFEB4	366208F3	2AF0EF79	B625D394
.	.	.	.	.

5.2. Comparative Analysis

The following are comparative analysis between Phase I and Phase II:At Row Wise:

Table 5.11: Comparative analysis between Phase I and Phase II at Row Wise

	Phase I				**Phase II**				
Similarities	**Similarities in 500 attacks**	**Similarities in 2000 attacks**	**Similarities in 5000 attacks**	**Similarities in 10,000 attacks**	**Similarities in 40,000 attacks**	**Similarities in 60,000 attacks**	**Similarities in 80,000 attacks**	**Similarities in 90,000 attacks**	**Similarities in 100,000 attacks**
0	126	1195	3031	5942	23762	35705	47687	53643	59697
1	56	645	1567	3208	12924	19232	25478	28660	31641
2	35	140	348	744	2902	4423	5943	6742	7523
3	0	19	53	100	376	601	815	875	1053
4	0	1	1	6	31	34	73	75	79
5	0	0	0	0	5	5	4	5	7

Position Wise (column wise):

Table 5.12: Comparative analysis between Phase I and Phase II at Position Wise

	Phase I				Phase II				
Positions	**Similarities in 500 attacks**	**Similarities in 2000 attacks**	**Similarities in 5000 attacks**	**Similarities in 10,000 attacks**	**Similarities in 40,000 attacks**	**Similarities in 60,000 attacks**	**Similarities in 80,000 attacks**	**Similarities in 90,000 attacks**	**Similarities in 100,000 attacks**
0	24	113	301	639	2512	3686	5111	5446	6393
1	24	120	336	640	2515	3675	4987	5598	6387
2	32	123	296	636	2473	3756	4969	5745	6154
3	33	133	298	654	2427	3772	4998	5588	6165
4	24	131	326	612	2490	3767	4958	5661	6327
5	23	103	300	615	2584	3851	4988	5641	6225
6	35	130	307	620	2483	3728	5153	5721	6223
7	22	133	262	604	2521	3807	4957	5694	6323

5.3. Experimental Analysis after using Intelligent Algorithm

The process was carried out for 90,000 Attacking Keystreams on individual Original Keystream. When five similarities were found between Original Keystream and Attacking Keystream that key was discarded (table 5.1 & figure 4.4) and selected when there were no five similarities (table 5.13 & figure 4.5) using Intelligent Algorithm.

Table 5.13: Original Keystream of SNOW 2.0.

Original SNOW 2.0 One Keystream
E7FC94FC

6. Summary, Conclusion and Future Work

- SNOW were designed by Patrick Ekdahl and Thomas Johansson and submitted to NESSIE project. After the report of distinguishing and Guess and Determine attacks on SNOW 1.0, a new version SNOW 2.0 was proposed by Patrick Ekdahl and Thomas Johansson in 2002.
- GD Attacks are one of the general attacks which have been effective on some Stream Ciphers.
- Guess and Determine (GD) Attacks are traced bit positions in key when applied on SNOW 2.0.
- In order to determine the traceable bit positions in key, the experimental analysis were divided into two phases.
- In first phase, 4 experiments were carried out. Each experiment has one Original Keystream and 500, 2000, 5000 and 10,000 Attacking Keystreams.
- In first experiment, maximum similarities (traceable bits) were found on position six and at row wise, maximum 35 times of two similarities were found.
- In second experiment, maximum similarities (traceable bits) were found on position three & seven and at row wise maximum one time of four similarities were found.
- In third experiment, maximum similarities (traceable bits) were found on position one and at row wise, maximum one time of four similarities were found.
- In fourth experiment, maximum similarities (traceable bits) were found on position three and at row wise, maximum 6 times of four similarities were found.
- Now in second phase, 5 experiments were carried out. Each experiment has the same Original Keystream as in first phase and 40,000, 60,000, 80,000, 90,000 and 100,000 Attacking Keystreams.
- In first experiment, maximum similarities (traceable bits) were found on position five and at row wise, maximum 5 times of five similarities were found.

- In second experiment, maximum similarities (traceable bits) were found on position five and at row wise maximum 5 times of five similarities were found.
- In third experiment, maximum similarities (traceable bits) were found on position six and at row wise, maximum 4 times of five similarities were found.
- In fourth experiment, maximum similarities (traceable bits) were found on position two and at row wise, maximum 5 times of five similarities were found.
- In fifth experiment, maximum similarities (traceable bits) were found on position zero and at row wise, maximum 7 times of five similarities were found.
- After locating the traceable bit positions in key, Intelligent Algorithms were used to avoid the traceable bit locations; those were traced by Guess and Determine (GD) Attacks.
- Using Intelligent Algorithm, we discarded the Original Keystream produced by SNOW 2.0, when finding five similarities (figure 4.4) and selected another Original Keystream because there were no five similarities (figure 4.5).

In order to determine the traceable bit positions in key, the experimental analysis were divided into two phases. In first phase, 4 experiments were carried out. Each experiment has one Original Keystream and 500, 2000, 5000 and 10,000 Attacking Keystreams.In second phase, 5 experiments were carried out. Each experiment hasthe same Original Keystream as in first phase and 40,000, 60,000, 80,000, 90,000 and 100,000 Attacking Keystreams. After analysis of both phases, maximum five similarities were found in Original Keystream. We applied 90,000 Attacking Keystreams on two different Original Keystreams (table 5.1 & table 5.13). Using Intelligent Algorithm, we discarded first Original Keystream due to finding five similarities (figure 4.4) and selected second Original Keystream because there were no five similarities (figure 4.5). So we conclude that the intensity of GD Attack on SNOW 2.0 can be minimized using Intelligent Algorithms.

After analyzing the results, we found that there were maximum five similarities in key. So it is highly recommended to use Intelligent Algorithms to avoid the traceable bit locations in key for minimizing the intensity of GD Attack on SNOW 2.0.

References

[1] Allen, J. and A. Frisch. 1982. Semantic Network. Proceedings of the 20th Annual meeting of ACL(Association for Computational Linguistics), University of Toronto, Toronto. pp. 19-27.

[2] Anand, S. and G. V.Ramanan. 2006. Periodicity,ComplementarityandComplexityof2-adic Feedback-with-Carry Shift Register CombinerGenerators. Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security (**ASIACCS). pp**. 275-282.

[3] Ahmadi, H. and Y. E. Salehani. 2004. A Modified Version of SNOW2.0. Proceedings of the 2007 International CSI (Computer Society of Iran) Computer Conference, Iran.http://homepages.ucalgary.ca/~hahmadi/index_files/snow_CSICC.pdf.

[4] Ahmadi, H. and T. Eghlidos. 2005. Advanced Guess and Determine Attacks on Stream Ciphers. Proceedings of the 2005 International Symposium on Telecommunications (IST). pp. 87-91.

[5] Coppersmith, D., S. Halevi and C. Jutla. 2002. Scream: a software-efficient stream cipher. In Fast Software Encryption (FSE), Lecture Notes in Computer Science Journal. 2365: 195-209.

[6] Deliyanni, A. and A. R. Kowalski. 1979. Logic and Semantic Networks. Communications of the Association for Computing Machinery (CACM) Journal.22 (3): 184-192.

[7] Denning, R.and D.Elizabeth. 1945. Cryptography and Data Security.Library of Congress Cataloging in Publication Data. pp. 1-20.

[8] Daemen, J. and V. Rigmen. 2002. The design of Rijndael. Springer Verlag Series on Information Security and Cryptography, ISBN 3-540-42580-2.

[9] Ekdahl, P.2003. On LFSR based Stream Ciphers Analysis and Design. Ph.D. Thesis, Department of Information Technology, Lund University, Sweden.

[10] Ekdahl,P. and T. Johansson. 2000. SNOW – a new stream cipher. Proceedings of first open NESSIE Workshop, Heverlee, Belgium.

[11] Ekdahl,P. and T. Johansson. 2002. A new version of the stream cipher SNOW.Lecture Notes in Computer Science Journal. 2595: 47-61.

[12] Hawkes, P. and G. Rose. 2003. Guess and Determine Attacks on SNOW. Lecture Notes in Computer Science Journal. 2595: 37-46.

[13] Johansson, W. H. F. and T. Spaanenburg. 2005. SNOW 2.0 IPcore for trusted hardware. Proceeding of 2005 International Conference on FPL,Tampere University of Technology, Finlandpp. 281- 286.

[14] Kiyomoto, S., T. Tanaka and k. Sakurai. 2005. Experimental Analysis of Guess-and-Determine Attacks on Clock-Controlled Stream Ciphers. The Institute of Electronics, Information and Communication Engineers (IEICE) Transactions on Fundamentals of Electronics, Communications and Computer Sciences Journal.E88-A (10): 2778-2791.

[15] Nyberg, K. and J.Wallen. 2006. Improved Linear Distinguishers for SNOW 2.0. Lecture Notes in Computer Science Journal. 4047: 144-162.

[16] Olivier, B. and G. Henri. 2005. Resistance of SNOW 2.0 against algebraic attacks. Lecture Notes in Computer Science Journal. 376: 19-28.

[17] Roy, A. 2000.Artificial Neural Networks - A Science in Trouble.Proceedings of theACM (Association for Computing Machinery) SIGKDD(Special Interest Group (SIG) on Knowledge Discovery & Data Mining) Explorations Newsletter.1 (2): 33 - 38.

[18] Watanabe, D., A. Biryukov and C. D. Canniere. 2003. A Distinguishing Attack of SNOW 2.0 with Linear Masking Method. Journal of IEIC (Institute of Electronics,Information and Communication Engineers) Technical Report.102(746): 23-28.

[19] Zimmermann, P. 1990. Encryption and Decryption.Introduction to Cryptography.p.11.www.eie.fceia.unr.edu.ar/ftp/Comuicaciones/AN%20INTRODUCTION%2TO%20CRYPTOGRAPHY.PDF.