Deep Learning based Analysis of Stream Ciphers A5/1 and RC4

Aryan Prajapati
Department of Information Technology, Institute of Engineering & Technology
Devi Ahilya University, Indore – 452001, India

ABSTRACT
In this paper, we present the analysis of stream ciphers by using deep learning methods to detect and expose their vulnerability. For this, A5/1 and RC4 stream ciphers are chosen as they both are popular and known to have several security vulnerabilities. In this paper, these vulnerabilities are exposed in the form of 'bias' detection using the deep learning methodology.

General Terms
Security vulnerabilities, Deep learning and Analysis

Keywords
Cryptanalysis, Stream cipher, Symmetric stream cipher, A5/1, RC4, Deep Learning, Dense Neural Network.

1. INTRODUCTION
With the establishment of telecommunication systems and computer networks, data can now be transmitted over significant distances in significantly less time. However, it also makes the user's data susceptible to various attacks causing a threat to the user's privacy and data integrity. To fill this void, predominately stream cipher (a symmetric encryption algorithm) is used as a security measure that ensures data and signaling confidentiality and maintains user secrecy over the transmission channel. In a stream cipher, the data in the form of a 'stream of the digits' (called Plaintext) is encrypted one at a time with the corresponding pseudorandom digit (called Keystreams) that is constructed by following some set of algorithms created by following some mathematical basis. The combination of the plaintext and keystream generates the digits of the ciphertext stream, an unreadable form of data. In practice, the 'stream of digits' is a typical 'stream of bits' and the combining operation is an exclusive-or (XOR). A similar process is used to obtain the Plaintext at the end. The process can be expressed as:

$$C_t = P_t \otimes K_t$$

$$P_t = C_t \otimes K_t$$

Where Ct represents the Ciphertext, Pt represents the Plaintext, Kt represents Keystream, and '⊗' represent the exclusive-or operand. For analysis on the stream ciphers, the primary focus is given to two well-known stream ciphers A5/1 and RC4.

The A5/1 is a symmetric stream cipher that was first introduced in 1987 to provide security to the Global System for Mobile (GSM) cellular telephone standard by protecting the integrity and confidentiality of the user's information over the air interface. The A5/1 encryption algorithm uses three Linear Feedback Shift Registers (LFSRs) along with a clocking mechanism to provide the pseudorandom keystreams.

Likewise, the RC4 or "Rivest Cipher 4" stream cipher was first introduced in 1987 by Ron Rivest. The RC4 is known for its remarkable speed and its peculiar simplicity due to which it is one of the most widely used software stream ciphers. It is used in SSL, WEP, TLS, and WPA-TKIP. For ciphering, the RC4 uses a well-defined algorithm to generate the pseudorandom keystreams.

These stream ciphers' strength is their ability to generate computationally unrecognizable or random keystreams. In cryptography, the non-random events which can be computationally recognized either in the cipher's internal state or in the generated keystream lead to the formation of 'bias', which is strongly undesirable.

In this paper, we present a method based on deep learning methodology along with the essence of supervised learning to detect and expose the presence of 'Bias' in the keystream. For this, we construct a deep neural network with 6 layers and with over '500,000' trainable parameters.

We take a large dataset (greater than 100,000 records) consisting of input-output pairs for the respective. The general essence of our methodology is to detect the hidden bias in the dataset and expose them in terms of a numeric value via by accuracy score by the Deep Neural Network model.

1.1 Security and Related Work on A5/1 Stream Cipher

The A5/1 stream cipher was believed to be a strong encryption algorithm, even among the other members of the A5 encrypting family which provide over-the-air communication privacy in the GSM (Global System for Mobile communication) cellular telephone standard. However, with numerous efforts of researchers, A5/1 was proven to have weaknesses that could be used to bypass the security measure. Thus, threatening the data and signal confidentiality and user secrecy. The general design of A5/1 was leaked in 1994 and by 1999, the complete algorithm was reverse engineered. As a result, many attacks were published by the researchers, which are summarized as follows:

Ross Anderson (in 1994) [1], proposed an attack by guessing 41 bits in shorter registers R1 and R2 and deriving 23 bits from the longer register, R3. The complexity of the attack was found to be *O*(245) and more than one month of time

J. Golic (in 1997) [2] describes an improved attack requiring *O*(240) steps. The attack was based on solving the sets of 240 linear equations which however takes more time than the previous algorithm.

Alex Biryukov, Adi Shamir, and David Wagner (in 2000) [3] proposed two new cryptanalytic attacks on A5/1, in which a single PC could extract the conversation key in real-time from a small amount of generated output. The first attack, called the

biased birthday attack, requires the output of the A5/1 algorithm during the first two minutes of the conversation, and computes the key in about one second, whereas the second attack called the random Subgraph attack requires the output of two seconds of conversation and takes several minutes to computes the key. The success probability of these attacks was claimed to be 60%.

Patrik E. and Thomas J. (2002) [4], proposed an attack based on the method given by Biryukov, (the time-tradeoff attack in 2000). Their proposal was based on 'identification of correlation'. The complexity of the attack was almost independent of the shift-register length. Their proposed attack explored the weak key initialization, which allowed separation of the session key from the frame number in binary linear expressions. The implementation of the attack takes only 2 to 5 minutes.

Maximov (2004) [5], Proposed an improved version of the attack that was published by Patrik E. and Thomas J. (2002) [4]. The improved result makes the attack to have less than one minute of computation.

Barkan (2006) [6], proposed Guess and determine attack. This attack was made to work with all A5 family members. In the pre-computation phase of this attack, a huge amount of data is needed to be computed and stored.

1.2 Security and Related Work on RC4 Stream Cipher

RC4 is known to be one of the simplest and most widely adopted ciphers. But its simplicity causes the RC4 to become susceptible to various security attacks. Since the first appearance of the RC4 design with the general public, when it got anonymously released on emails and newsgroups in 1994, several attacks were proposed by the researchers, which are briefly mentioned as follows:

Paul and Subhamoy Maitra (In 2007) [7], proved the permutation-key correlation, which was first observed in 1995, by Andrew Roos [8], (also called Roos' biases). The work includes an algorithm for complete key reconstruction from the final permutation after the 'key scheduling algorithm', without any prior assumption.

Bias attack by Itsik Mantin and Adi Shamir (In 2001) [9], shows that the second output byte of the cipher is more inclined toward zero with a probability close to 1/128, instead of a theoretical 1/256.

Fluhrer, Mantin, and Shamir (In 2001) [10] proposed an attack that was based on their discovery that among all the possible RC4 keys, the statistics for the first few bytes of the output keystream are strongly non-random, thus leading to leaking information about the key. Further, they used the attack to break the WEP (wired equivalent privacy) which used RC4 for encryption.

Erik Tews, Ralf-Philipp Weinmann, and Andrei Pychkine (In 2007) [11] published a tool 'aircrack-ptw' which was created based on an analysis published by Andreas Klein [12] referring to the correlation between RC4 keystream and key. The tool was able to crack 104-bit RC4 used in 128-bit WEP in under a minute.

Royal Hollway attack (In 2013) [13][14], was proposed by a group of security researchers at the Information Security Group at Royal Holloway, University of London. The attack targeted RC4 in TLS and WPA/TKIP. The attack allows an attacker to recover limited plaintext from a TLS and WPA/TKIP connection when RC4 encryption is used. The attack works on the principle of statistical flaws in the RC4 encryption.

NOMORE attack (in 2015) [15] was proposed by security researchers from KU Leuven. The Numerous Occurrence Monitoring & Recovery Exploit (NOMORE) attack targeted RC4 in TLS and WPA-TKIP. The attack against TLS could decrypt a secure HTTP cookie within 75 hours and the attack against WPA-TKIP could be completed within an hour and allowed an attacker to decrypt and inject arbitrary packets.

2. DESCRIPTION OF STREAM CIPHERS

In this section, we describe the descriptions of stream ciphers: A5/1 and RC4 along with the process of generation of keystreams.

2.1 Description of A5/1 Stream Cipher

A5/1 stream cipher (also commonly referred to as GSM stream cipher) in Fig 1 is a symmetric stream cipher constituting three short linear feedback shift registers (or LFSRs) of lengths 19, 22, and 23 bits, denoted as R1, R2, and R3 respectively. These registers are controlled and manipulated by predetermined feedback polynomials and a clocking mechanism that depends on the majority rule function (or simply the majority function). The majority rule determines which register is to be shifted by a 'majority poll'. Additionally, these registers have internal sub-groups on 'bits' that are described and summarized in Table 1.

Table 1. A5/1 Registers Feedback Polynomials and Parameters

Linear Feedback Shift Registers (or LFSRs)	LFSRs Length (in bits)	Feedback Polynomials	Tapping Bits (T_i's)	Clocking Bits(C_i 's)	Output Bits
R1	19	$1+ x^{14} + x^{17}+ x^{18} + x^{19}$	13, 16, 17, 18	8	18
R2	22	$1 + x^{21} + x^{22}$	20, 21	10	21
R3	23	$1 + x^{8} + x^{21} + x^{22} + x^{23}$	7, 20, 21, 22	10	22

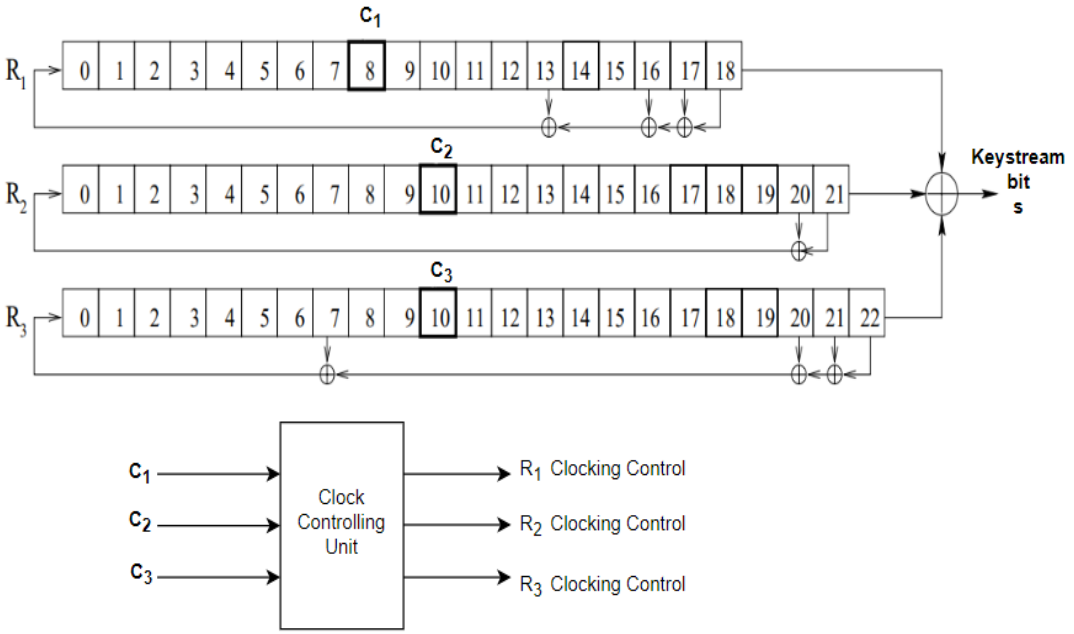


Fig 1: A5/1 Stream Cipher

The shift cycle in these registers is always in favor of removing the most significant bit (MSB) i.e., the highest label bits and updation or insertion from the lowest significant bit (LSB) i.e from the lowest label bits. The cipher output is given in a bit-by-bit manner by XOR-ing all MSBs after each clock cycle is completed.

Along with LFSRs, A5/1 stream cipher also constitutes a 64-bit long session key, and a publicly known 22-bit long frame vector (also known as an initialization vector (IV)) for initializing these registers. These 64+22 bits determine the strength of the stream cipher as a weakened combination of these vectors will eventually lead to a weaker internal state for the cipher. A poor combination makes A5/1 stream cipher more susceptible to attacks.

2.1.1 Process of Keystream Generation

Before generation of the keystream, A5/1 stream cipher undergoes an initialization phase or a warm-up phase in which all of its registers are initialized by the session key and the frame vector. Initially, all the bits of the registers are set to zero. Then, they all are clocked 64 times, which is equivalent to the length of the session key. In each cycle, a bit from the session key is XORed with the result from the predefined feedback polynomials and inserted into the LSBs of the registers concurrently. A similar is then repeated for the 22-bit long frame vector. Thus, all registers have clocked a total of 86 times after which the internal state of the cipher is obtained. After obtaining the internal state, LFSRs undergo clocking (constraint by the majority function) in which all the registers are shifted for a total of 100 times each. This is done to eliminate the internal dependency present among the bits of the register. The following is regarded as the warm-up phase.

After the completion of the warm-up phase, the stream cipher is ready to generate the keystream. The register undergoes several clocking process constraints by the majority function. The majority function uses the clocking bits C1, C2, and C3 of the register R1, R2 and R3 respectively, to determine the value of the majority bit, m. This value will decide which register is to be clocked next.

M = Maj (C1, C2, C3)

Table 2. Majority Rules Results.

C1	C2	C3	m	Register to be clocked
1	1	1	1	R1, R2, R3
1	1	0	1	R1, R2
1	0	1	1	*R1, R3*
1	0	0	0	R2 R3
0	1	1	1	R2, R3
0	1	0	0	R1, R3
0	0	1	0	R1, R2
0	0	0	0	R1, R2, R3

If among the three clocking bits, two or more bits are 0, then the value of the majority bit, m is 0. Similarly, if the majority lies with bit 1, then the value of the majority bit, m is 1. Then, the register is clocked if its clocking bit value matches with the majority bit, m. All possible scenarios and their outcomes are shown in Table 2. As in each cycle, at least two or three registers are clocked, hence each register is clocked with a probability of 3/4. This cycle is continued till the total of N cycles is achieved, where N is equivalent to the length of the plaintext. After each loop, each output bit of the registers XORed to obtain a bit of the N-length keystream. The overall encryption process is described in Fig 2.

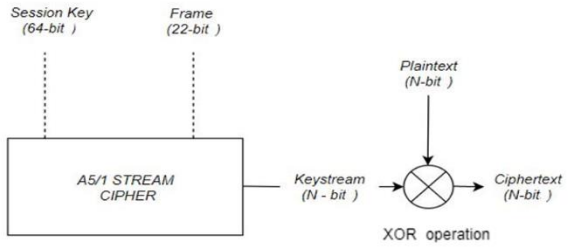


Fig 2: Schematic diagram of A5/1 Stream Cipher

For the GSM transmission, the value of N is 228, where the first 114 bits are for the downlink and the second 114 bits are for the uplink. Also, the transmission is done by sending a sequence of these N-length bits, called frames.

2.2 Description of RC4 Stream Cipher

RC4 (or Rivest Cipher 4) stream cipher is also a symmetric stream cipher but unlike the A5/1 stream cipher, it doesn't involve any hardware components. Instead, it uses data structures (arrays or vectors) to generate a keystream. For encryption, the stream cipher uses two byte-arrays, a State Vector 'S' of size 256 (S[0], S[1], ... ,S[255]) and a user-selected variable-length key 'K' of size l (K[0] ,K[1], ..., K[l]) where l ϵ {1,256}. These are controlled and updated by Key Scheduling Algorithm (KSA, described in Algorithm 1, Fig 3a) and Pseudo Random Generation Algorithm (PRGA, described in Algorithm 2, Fig 3b).

Algorithm 1 Key Scheduling Algorihtm

```
initialization
for i from 0 to n − 1 do
    S[i] := i
end for
j := 0
generate a random permutation
for i from 0 to n − 1 do
    j := (j + S[i] + K[i mod l]) mod n
    Swap S[i] and S[j]
end for
```

Fig 3a: RC4 algorithm: KSA

2.2.1 Process of Keystream Generation

The process of keystream generation in RC4 stream cipher is done in two phases: the key scheduling phase and the generation phase. In the key scheduling phase, the Key Scheduling Algorithm (KSA, described in Algorithm 1, Fig 3a) initializes the State vector S by using the random key K which defines the internal state of the stream cipher. The initialization task is carried out in two loops. In the first loop, the state vector S is initialized by its index value, and in the second loop the state vector S is randomized by using the random key. Initially, the key is in 'string' format which further is converted into byte-array by ASCII encoding.

Algorithm 2 Pseudo Random Generation Algorithm

```
initialization
i := 0
j := 0
generate pseudo random sequence
loop
    i := (i + 1) mod n
    j := (j + S[i]) mod n
    Swap S[i] and S[j]
    k := (S[i] + S[j]) mod n
    print S[k]
end loop
```

Fig 3b: RC4 algorithm: PRGA

The generation phase in which the Pseudo Random Generation Algorithm (PRGA, described in Algorithm 2, Fig 3b) modifies the internal state and outputs the byte of the keystream. The total number of iterations performed in the algorithm is equivalent to the length of the plaintext, m (where m can be any positive integer). The algorithm uses two integers 'i' and 'j' to access the array elements in a pseudo-random manner. The overall encryption process is described in Fig 4.

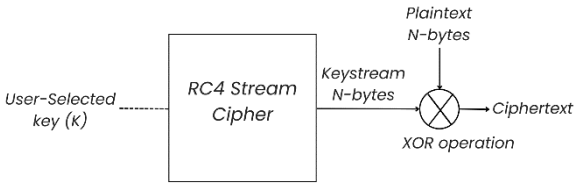


Fig 4: Schematic diagram of RC4 Stream Cipher

3. ANALYSIS AND EXPERIMENTAL RESULTS OF STREAM CIPHERS

In this section, we introduce the implementation of the Deep Neural Network (DNN) to analyze the stream ciphers: A5/1 and RC4. The general idea of our approach is to use the Deep Neural Network model as a prediction model to predict the outcomes of the given stream ciphers. The model takes the encryption key as its input, the generated keystream as its target, and traces its efficiency as an accuracy score. If the obtained accuracy score is more than the theoretically possible score, then we can claim the presence of bias in the keystream. From the perspective of machine learning, the following can be seen as a multi-class classification problem using the Deep Neural Network as a classifier in the supervised machine learning environment.

3.1 Analysis of A5/1 Stream Cipher and its Result

The published work of many researchers indicates that there exists some weakness in the encryption algorithm in A5/1 stream cipher, firmly in its use of LFSRs. This aspect is taken as a base for our analysis. In our analysis, we have tested the encryption process from the perspective of machine learning, rather than the standard statistical perspective. For analysis, we convert the cryptanalysis problem to a machine learning problem by making it a classification problem. To do so, we focus our attention on the initial vector (referring to session key

and frame vector), which initializes the LFSRs and the bit-by-bit keystreams.

We have recreated A5/1 stream cipher in a software-based environment by using data structures, as a replacement for the shift register. The authenticity of the stream cipher is tested by the test vectors provided by Marc Briceno, Ian Goldberg, and David Wagner in their work, "A pedagogical implementation of A5/1" [16]. For the analysis, we collected a large number of records containing distinct initial vectors and their corresponding generated keystreams. As among the session key and frame vector, the frame vector is publicly known, we only changed the session key in each new record. Also, for generalization, we only generated an 18-bit long keystream instead of a 228-bit keystream, which is used in GSM transmission.

The generated dataset from above has dimension: 262144 ×82, in which the 82 columns show the 64-bit long session key and 18-bit keystream and the total such records are: 2^18=262144. This generated dataset is fed to the deep neural network model.

The specification of the deep neural network used in the analysis is described in Fig 5. The model takes the first 64 columns i.e., the session key as a decision variable to predict the specific bits of the 18-bit keystream present in the remaining columns. The compiling parameters are: Loss = 'categorical_crossentropy', Optimizer = 'Adam' (Adam optimizer) with learning rate = 0.001, and Metric = 'accuracy'. The train and test split on the dataset is standard 0.8 (80 % training set and 20% testing set). The number of epochs taken in the training phase is 50 with callbacks as a Regularizer.

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 64)	4160
dense_1 (Dense)	(None, 512)	33280
dense_2 (Dense)	(None, 512)	262656
dense_3 (Dense)	(None, 256)	131328
dense_4 (Dense)	(None, 256)	65792
dense_5 (Dense)	(None, 128)	32896
dense_6 (Dense)	(None, 2)	258

Total params: 530,370
Trainable params: 530,370
Non-trainable params: 0

Fig 5: Deep Neural Network Model specification for A5/1 stream cipher analysis

We perform the analysis with several permutation of the session key to properly analyze the use of the LFSRs in the stream cipher. First, we generate all 64- bits of the session key randomly in each record, to make all LFSRs have different initialization with each record. Second, we generate only 20 bits randomly, and the rest of the bits are set to be constant in each of the records. Similarly, we generate 32 bits randomly in each of the records, and the rest of the bits constant as done previously, and finally, we generate 45 bits randomly in each of the records keeping the rest to be constant. Through this, the activity of LFSRs gets initialized and analyzed with a different permutation of the keystreams. The results of our analysis are summarized in Table 3. The listed value is the accuracy score given by the prediction model i.e., the deep neural network model which was determined during the testing Phase.

Table 3. Results observed during analysis of A5/1 Stream Cipher

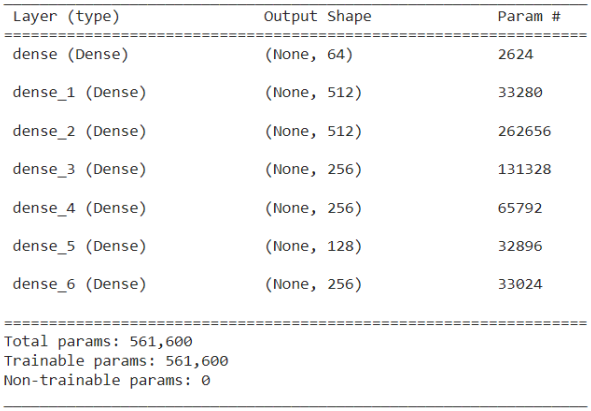
Targeted i^{th} bit of the keystream	Accuracy Score for the randomly generated 64 bits	Accuracy Score for the randomly generated 20 bits	Accuracy Score for the randomly generated 32 bits	Accuracy Score for the randomly generated 45 bits
Bit -1	0.4977	0.5029	0.5010	0.5028
Bit -2	0.5014	0.4989	0.5047	0.4998
Bit -3	0.5001	0.4954	0.4973	0.5006
Bit -4	0.5000	0.4991	0.4991	0.5004
Bit -5	0.4995	0.5028	0.5017	0.4977
Bit -6	0.5010	0.4986	0.4995	0.5000
Bit -7	0.4996	0.5003	0.5021	0.5031
Bit -8	0.4988	0.5021	0.4970	0.5017
Bit -9	0.5003	0.4981	0.5008	0.5002
Bit -10	0.5015	0.4993	0.5026	0.5008
Bit -11	0.4942	0.5030	0.5021	0.4994
Bit -12	0.5049	0.5022	0.4999	0.4992
Bit -13	0.4995	0.5025	0.4977	0.5015
Bit -14	0.5004	0.4999	0.5017	0.4963
Bit -15	0.5015	0.4986	0.4973	0.4969
Bit -16	0.5026	0.5009	0.4992	0.4967
Bit -17	0.5039	0.5061	0.4981	0.4982
Bit -18	0.5002	0.5024	0.5012	0.4995

3.2 Analysis of RC4 Stream Cipher and its Result

The 'bias-detection' on the generated keystream is not new in the research domain of the RC4 stream cipher. Many researchers succeeded in their efforts and also published complete proof for it [10][17]. But the analysis was based on a statistical perspective. In our analysis, we perform our 'bias detection' from the perspective of machine learning. To do so, we focus our attention on the user-selected key and the produced byte-by-byte keystream.

We recreate RC4 stream cipher in the software-based environment using the same original algorithm. The authenticity of the stream cipher is tested by using well-known test vectors [18]. For the analysis, we collect several large number of records each containing distinct but different byte-length of keys and the corresponding generated keystreams.

The generated dataset from above has dimensions: 262144×50, 262144×90, and 262144×138. Each column represents a 5-byte or 40 bits key, 10-byte or 80 bits key, and a 16-byte or 128 bits key combined with produced byte size keystreams of length 10 respectively. These generated datasets are fed to the deep neural network model. Since the output type is a 'byte', the following analysis can be seen as a 256-Class Classification problem.



Layer (type)	Output Shape	Param #
dense (Dense)	(None, 64)	2624
dense_1 (Dense)	(None, 512)	33280
dense_2 (Dense)	(None, 512)	262656
dense_3 (Dense)	(None, 256)	131328
dense_4 (Dense)	(None, 256)	65792
dense_5 (Dense)	(None, 128)	32896
dense_6 (Dense)	(None, 256)	33024

Total params: 561,600
Trainable params: 561,600
Non-trainable params: 0

Fig 6: Deep Neural Network Model specification for RC4 stream cipher analysis

The specification of the deep neural network used in the analysis is described in Fig 6. The model takes the columns of the user-selected key as a decision variable to predict the byte of the 10-length keystream present in the remaining columns. The compiling parameters are: Loss = 'categorical_crossentropy', Optimizer = 'Adam' (Adam optimizer) with learning rate =0.001, and Metric='accuracy'. The train and test split on the dataset is standard 0.8 (80% training set and 20% testing set). The number of epochs taken in the training phase is 50 with callbacks as a regularizer.

Table 4. Results observed during analysis of RC4 Stream Cipher

Targeted i^{th} - byte of the keystream	Accuracy score for 5-byte key	Accuracy score for 10-byte key	Accuracy score for 16-byte key
Byte -1	0.0039	0.0036	0.0037
Byte -2	0.0069	0.0068	0.0066
Byte -3	0.0040	0.0042	0.0037
Byte -4	0.0039	0.0038	0.0038
Byte -5	0.0043	0.0039	0.0042
Byte -6	0.0041	0.0036	0.0039
Byte -7	0.0043	0.0041	0.0039
Byte -8	0.0040	0.0036	0.0042
Byte -9	0.0038	0.0042	0.0041
Byte -10	0.0039	0.0038	0.0040

To generalize our analysis on the RC4 stream cipher we perform the analysis by considering three scenarios, when the user-selected key, K is 5-byte long, 10-byte long, and 16-byte long respectively, and with produced byte size keystreams of length 10. The results of our analysis are summarized in the Table 4. The listed value is the accuracy score given by the prediction model i.e the Deep Neural Network model which was determined during the testing Phase.

4. CONCLUSIONS

From the analytical result of the A5/1 stream cipher, we found that the prediction model, the deep neural network predicts each bit of the keystream in every scenario with an average accuracy score of 0.5 or 1/2. This score matches completely with the theoretical score for A5/1 stream cipher. As theoretically for an ideal stream cipher, any prediction on the bit of keystream will always be a random guess of its possible states. From the above, we claim that the generated keystream is unbiased and no direct relation exists between the session key with the produced bit of the keystream when LFSRs are in the initialization phase.

In RC4 stream cipher, theoretically, every prediction model gives an accuracy score of 1/256=0.0040 in every possible scenario. But from the analytical results of the RC4 stream cipher, we found that the prediction model i.e, the deep neural network, predicts each byte of the keystream in every scenario with the same accuracy score as the intended, except from the second byte of the keystream. The model predicted the second byte with an accuracy score of approximately 0.0070, which is nearly double the expected value. Thus, we claim that the generated keystream is biased and a direct or indirect relation must exist in the user-key with the produced byte of the keystream, particularly for the second byte of the keystream.

5. FUTURE WORK

The analytical results obtained through the deep learning methodology on the stream ciphers provide enough evidence to prove that the deep learning methodology can be used in the field of cryptanalysis. Alongside using it as a tool to perform 'attacks', it can also be used as a 'test parameter' along with NIST statistical tests [19] to check the strength of a keystream generator. Also, its 'self-learning' and 'customizable' aspects give more flexibility to testers. Apart from deep neural networks (DNNs), the Deep Learning methodology has a lot more to offer such as Recurrent Neural Networks (RNNs) and Long Short Term Memory networks (LSTMs) that are computationally stronger and have more features than DNNs, which can be used to provide an aid in cryptanalysis in future.

6. DECLARATION OF COMPETING INTEREST

The author declares that he has no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

7. ACKNOWLEDGMENT

The author would like to extend his deepest appreciation to Dr. Devendra Yadav and Dr. Girish Mishra for their invaluable guidance and mentorship throughout the course of this work. The author also thanks to the SAG Laboratory of the Defence Research and Development Organisation (DRDO), New Delhi, India for providing a supportive research environment and workplace. Moreover, the author is truly grateful for the opportunity to work with such esteemed organization.

8. REFERENCES

[1] Ross Anderson (17 June 1994). "A5 (Was: HACKING DIGITAL PHONES)".

[2] J. Golic, “Cryptanalysis of an alleged A5 stream cipher,” Advances in Cryptology. EUROCRYPT97. LNCS, Vol. 1233, 1997, pp. 239-255, Springer-Verlag

[3] Biryukov, Alex; Adi Shamir; David Wagner. "Real Time Cryptanalysis of A5/1 on a PC". Fast Software Encryption—FSE 2000: 1–18.

[4] Patrik E., Thomas J. (2002) “Another Attack on A5/1” citeseerx library, pages (284-289)

[5] Maximov, Alexander; Thomas Johansson; Steve Babbage (2004). "An Improved Correlation Attack on A5/1". Selected Areas in Cryptography 2004: 1–18.

[6] Barkan, E., Biham, E., and Keller, N., Instant Ciphertext-only Cryptanalysis of GSM Encrypted Communication, Technical Report CS-2006-07, Technion, 2006.

[7] Goutam Paul, Siddheshwar Rathi, and Subhamoy Maitra. On Non-negligible Bias of the First Output Byte of RC4 towards the First Three Bytes of the Secret Key. Proceedings of the International Workshop on Coding and Cryptography (WCC) 2007, pages 285–294.

[8] Andrew Roos. A Class of Weak Keys in the RC4 Stream Cipher (1995).

[9] Itsik Mantin, Adi Shamir (2001). A Practical Attack on Broadcast RC4, FSE 2001. pp. 152–164

[10] Fluhrer, Scott R.; Mantin, Itsik; Shamir, Adi (2001). "Weaknesses in the Key Scheduling Algorithm of RC4", pp 1 -24.

[11] Erik Tews, Ralf-Philipp Weinmann, Andrei Pyshkin." Breaking 104-bit WEP in under a minute".

[12] A. Klein, Attacks on the RC4 stream cipher, Designs, Codes and Cryptography (2008) 48:269–286.

[13] Al Fardan; et al. (8 July 2013). "On the Security of RC4 in TLS and WPA", Information Security Group, Royal Holloway, University of London.

[14] "On the Security of RC4 in TLS and WPA", Information Security Group, Royal Holloway, University of London, Retrieved 6 September 2013.

[15] Mathy Vanhoef and Frank Piessens (9 August 2015). "RC4 NOMORE: Numerous Occurrence Monitoring & Recovery Exploit"

[16] Marc Briceno, Ian Goldberg, and David Wagner (1998-1999), "A pedagogical implementation of A5/1", featured in cryptome.org/jya/a51-pi.htm#PI

[17] Basu, Riddhipratim; Ganguly, Shirshendu; Maitra, Subhamoy; Paul, Goutam (2008). "A Complete Characterization of the Evolution of RC4 Pseudo Random Generation Algorithm". Journal of Mathematical Cryptology.

[18] RC4: Test Vectors, https://en.wikipedia.org/wiki/RC4, // last accesed on 22 Jan 2023.

[19] NIST SP 800-22, A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, random number generator, hypothesis test, P-value, by NIST Computer Security Division (CSD) (In 2010).