Defence Science Journal, Vol. 71, No. 4, July 2021, pp. 499-506, DOI : 10.14429/dsj.71.16209

Deep Learning based Cryptanalysis of Stream Ciphers

Girish Mishra[#,@], Indivar Gupta[#], S.V.S.S.N.V.G. Krishna Murthy[@], and S.K. Pal[#]

[#]*DRDO-Scientific Analysis Group, Delhi - 110 054, India*
[@]*DRDO-Defence Institute of Advanced Technology, Pune - 411 025, India*
[*]*E-mail: gmishratech28@gmail.com*

ABSTRACT

Conventional cryptanalysis techniques necessitate an extensive analysis of non-linear functions defining the relationship of plain data, key, and corresponding cipher data. These functions have very high degree terms and make cryptanalysis work extremely difficult. The advent of deep learning algorithms along with the better and efficient computing resources has brought new opportunities to analyze cipher data in its raw form. The basic principle of designing a cipher is to introduce randomness into it, which means the absence of any patterns in cipher data. Due to this fact, the analysis of cipher data in its raw form becomes essential. Deep learning algorithms are different from conventional machine learning algorithms as the former directly work on raw data without any formal requirement of feature selection or feature extraction steps. With these facts and the assumption of the suitability of employing deep learning algorithms for cipher data, authors introduced a deep learning based method for finding biases in stream ciphers in the black-box analysis model. The proposed method has the objective to predict the occurrence of an output bit/byte at a specific location in the stream cipher generated keystream. The authors validate their method on stream cipher RC4 and its improved variant RC4A and discuss the results in detail. Further, the authors apply the method on two more stream ciphers namely Trivium and TRIAD. The proposed method can find bias in RC4 and shows the absence of this bias in its improved variant and other two ciphers. Focusing on RC4, the authors present a comparative analysis with some existing methods in terms of approach and observations and showed that their process is more straightforward and less complicated than the existing ones.

Keywords: Stream cipher; Cryptanalysis; RC4; Deep learning

1. INTRODUCTION

Deep Learning is a new field of machine learning techniques where the learning methods are generally based on the Artificial Neural Network framework. Deep learning, invariably known as deep neural networks, has a variety of architectures such as Recurrent Neural Network (RNN), Deep Belief Network (DBN), and Convolution Neural Network (CNN). These architectures are applied in diverse domains to solve different problems like natural language processing, machine translation, medical image processing, computer vision, etc.[1,2]. These deep learning architectures have produced excellent results in respective domains, which have sometimes outperformed human experts. Deep learning also has an extra edge over traditional machine learning algorithms due to its capability of representation learning. Representation learning is a term used in machine learning to denote a class of techniques that automatically discover the representations required for feature detection from raw data. Establishing representation learning techniques is motivated by the fact that any machine learning task, such as classification or regression, generally require input that is computationally convenient and meaningful to process. To summarise, deep learning is a class of machine learning algorithms[3] that uses multiple hidden layers of artificial neurons, which helps feature learning from the raw input dataset.

There are many existing, well-established methods for cryptanalysis. However, the foremost step in applying these methods is to observe an inherent flaw in the cipher algorithm and then exploit it to mount the cryptanalytic attack, which is generally very complex. The advantage of deep learning methods to have automatic feature engineering capabilities from raw data can help to observe the flaw, as mentioned above. Sometimes it may also assist in the cryptanalysis task. Moreover, the stream cipher generated keystream should not have any biases. The adversaries can exploit the same in mounting cryptanalytic attacks on the cipher to get complete or partial information about the key or the plain data. So, before employing any of the available cryptanalytic techniques, which are very complicated in nature, it would be crucial to know such biases by using a deep learning approach that directly works over the raw data.

The application of machine learning in cryptography, especially of deep learning, has been explored in a very confined manner. Rivest, in his landmark survey paper[4] emphasised over the fact that how cryptography and machine learning contributed ideas and techniques to each other. He perceived machine learning and cryptanalysis as sister fields as both of them share the common goal of learning an unknown function from input-output pairs. Recently, Abadi

Received : 21 August 2020, Revised : 26 November 2020
Accepted : 22 February 2021, Online published : 01 July 2021

and Andersen[5], established a milestone work when exploring whether, in a multi-agent system, neural networks can learn to use secret keys to protect information from other neural networks. Hesamifard et al.[6] attended the problem of ensuring the privacy of raw data and developed new techniques for providing solutions to run deep neural networks over encrypted data. Picek et al.[7] analysed the convolutional neural network's performance for side-channel analysis while posing a problem to researchers whether the deep learning networks are better suited for side-channel analysis compared to other machine learning techniques. In other work, Wang[8] carried out a side-channel analysis of block cipher AES using deep learning. In latest developments, Gohr[9] came up with improved attacks on round-reduced lightweight block cipher Speck32/64[10]. This work[9] is perhaps the first such work that combined neural networks with robust conventional cryptanalysis techniques and demonstrated a neural network-based attack on a symmetric cryptographic primitive and improved upon the published state of the art. Gohr's work was carried forward by Baksi et al.[11], Jain et al.[12], and Yadav et al.[13]. All these works clubbed conventional differential cryptanalysis with machine learning and achieved better results. Recently, Xiao et al.[14] quantified a cipher's strength by measuring the difficulty for a neural network in mimicking the cipher algorithm. They defined several new metrics such as cipher match rate, training data complexity, and training time complexity to assess the cipher strength quantitatively. This work showed that a popular stream cipher Hitag2, used in modern cars' security, is weaker than the 3-round DES cipher. The literature survey indicates that the crypto research community has started an extensive exploration of deep learning applicability in cryptography in recent times.

In this paper, authors proposed a deep learning based method. The method is in line with the advantages of deep learning techniques as these techniques do not require feature extraction or feature selection task, which otherwise can not be escaped in any other mathematical, statistical, or analytical procedures. It progressively learns the necessary features from raw data. The proposed method considers the problem in the black-box scenario, as shown in Figure 1. Black-box as a device can be interpreted in terms of its inputs and outputs without any knowledge of its internal workings. In our experiments, the prior assumption is to consider a stream cipher algorithm as a black box and deal with the problem as a classification problem. The objective is to predict the value at a specific location in the output from an available input.

Following are authors main contributions in this paper:

- The findings from this paper show that the deep learning based method is more straightforward and advantageous compared to existing methods for the analysis of stream ciphers.
- The validation of the proposed method through extensive analysis of RC4 stream cipher and its improved variant RC4A establishes the applicability of deep learning methods in cipher data.
- The performance analysis indicates towards the generalisation of the method to other stream ciphers.

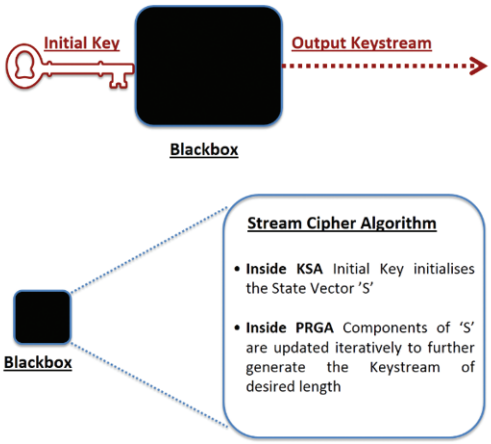

Figure 1. Black-box analysis model for proposed method.

2. PRELIMINARIES

This section provides a brief description of deep learning fundamentals, which are necessary for understanding the proposed method. It also contains a design description of RC4 stream cipher.

2.1 Deep Learning Methods

Deep learning is a class of machine learning algorithms[3,15] that successively uses multiple layers of artificial neurons to extract higher-level features from the raw input. For example, in image processing, initial lower layers identify edges, whereas higher layers identify more meaningful features such as digits, letters, or faces. Figure 2 shows a general deep learning architecture containing an input and output layer with multiple intermediate hidden layers.

Due to the vanishing gradient problem in neural networks[16] and computational limitations until a few years back, it was not possible (or computationally infeasible) to train multiple-layered neural networks. But, recent developments in the last decade, such as increased processing with GPU (Graphical Processing Unit) and TPU (Tensor Processing Unit) from Google and introduction of Rectified Linear Unit (ReLU) as an activation function in place of the classical sigmoid function, made it possible to stack multiple hidden layers in the network[17]. ReLU and Sigmoid activation functions are defined as $f(x) = \max(0, x)$ and $g(x) = \frac{1}{1+e^{-x}}$ respectively. The input layer takes the raw input, and intermediate hidden

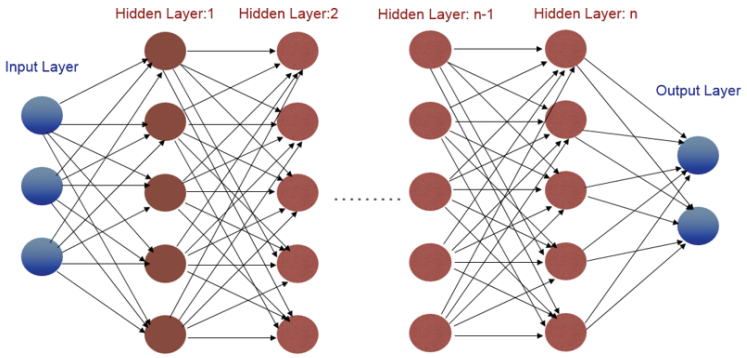

Figure 2. General Deep Learning Architecture.

layers progressively process the input data further by using an activation function. After that, the output layer generally uses Softmax function for final classification. The Softmax function is given below:

$$F(X_i) = \frac{e^{x_i}}{\sum_{i=0}^{k} e^{x_i}} \quad where\ i = 0,1,2,...,k$$

Here k denotes the number of possible outputs. Stacking many intermediate layers between input and output layers allows networks to learn an abstract representation of the mapping from input data to the corresponding target output. Some popular deep learning techniques are Convolution Neural Networks (CNN), Recurrent Neural Networks (RNN), and Long Short Term Memory networks (LSTM)[18].

2.2 Brief Description of Stream Ciphers

Symmetric key cryptography comprises of Block ciphers and Stream ciphers. In stream ciphers, the keystream of the desired length is generated using an initial key (invariably known as a secret key). The generated keystream is then masked (generally, using exclusive-or) with the plaintext bits to provide a ciphertext. In the early era of modern cryptography, stream ciphers were widely used due to their simplicity and speed than that of block ciphers. Stream ciphers have been the preferred choice in applications where extremely high throughput is needed or only low complexity hardware is available for usage. Keeping in mind the fundamental requirement in cryptography, the stream cipher design should ensure the desired cryptographic requirement of randomness in the output keystream. One of the criteria of fulfilling this is to ensure that the occurrence of any value in the range 0 to 255 at a byte position in stream cipher generated keystream should have equal probability. In other words, the occurrence of any value (in the given range) should have a probability 1/256. If the probability of the occurrence of any value at a byte position differs from it significantly, the keystream is adjudged to be biased at that specific byte position. The existence of this bias makes the stream cipher vulnerable to cryptanalytic attacks.

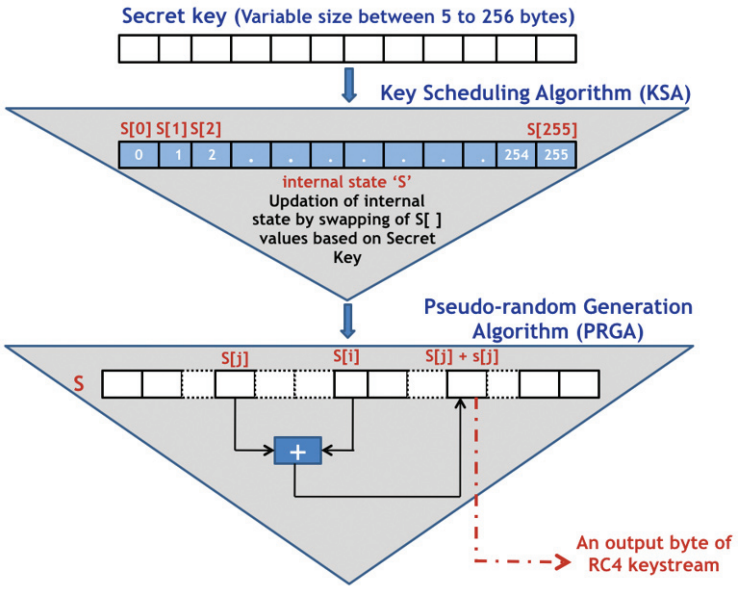

Figure 3. A Simple Layout of RC4.

RC4 is a stream cipher that iteratively generates the keystream in a byte-wise form. The generated keystream was claimed to possess pseudo-random characteristics. The exclusive-or (XOR) of keystream bits with the plaintext bits is done for encryption. The decryption is performed on similar lines by bit-wise XOR of ciphertext bits with the keystream bits. RC4 algorithm is composed of two phases: Key Scheduling Algorithm (KSA) and Pseudo Random Generation Algorithm (PRGA). The cipher description is given in Algorithm 1, and a simple layout of RC4 stream cipher is shown in Figure 3.

Algorithm 1 (RC4 Stream Cipher)

Key Scheduling Algorithm

1. *procedure KSA*()
2. *for* $i = 0$ *to* $N-1$ *do*
3. $s[i] \leftarrow i$
4. *end for*
5. $j \leftarrow 0$
6. *for* $i = 0$ *to* $N-1$ *do*
7. $j \leftarrow j + s[i] + key[i \bmod keylength] \bmod N$
8. *swap* $(s[i]$ *and* $s[j])$
9. *end for*

Pseudo Random Generation Algorithm

11. *procedure PRGA*()
12. *while* $i \geq 0$ *do*
13. $i \leftarrow i+1$
14. $j \leftarrow j + s[i]$
15. *swap* $(s[i]$ *and* $s[j])$
16. *output* $(s[s[i]+s[j]])$
17. *end while*
18. *end procedure*

In the algorithmic flow of RC4, '+' denotes the *addition* mod*ulo* N operation. In KSA, N pairs of the array $s[\,]$ are swapped based on the values of the secret key (*key*). At the end of KSA phase, an initial state is achieved for PRGA. After that, the output keystream of the desired length is generated in PRGA phase of the algorithm.

Paul and Praneel[19] found a weakness in RC4 stream cipher, and later on, to improve the security of the cipher, they proposed a strengthened RC4 variant, which they named RC4A. RC4A stream cipher uses two key components instead of using only one to generate two state arrays s_1 and s_2 by following the Key Scheduling Algorithm (KSA) of RC4. Other steps of RC4A pseudo-random generation algorithm[19] are similar, as performed in RC4.

As discussed earlier, we carry out the same experiment over two more stream ciphers namely Trivium[20] and TRIAD[21]. Trivium was designed to explore how much

a stream cipher can be made simple while maintaining the desired level of security and also ensuring the speed and flexibility. Since then, Trivium has been a well-tested stream cipher for its security analysis and till date no cryptanalytic attacks superior to brute-force are reported. So, we select this cipher to see if our method can learn and predict any bias in Trivium generated keystream. On the other hand, TRIAD-SC is relatively a new stream cipher based on which a family of lightweight symmetric-key schemes are proposed under NIST Lightweight Cipher competition[23]. No significant cryptanalytic work has been reported on TRIAD-SC as on date, therefore we experiment over this stream cipher to explore any possibility of finding the biases in the keystream generated by this cipher. Trivium is a synchronous stream cipher designed specifically for hardware usage, although it performs reasonably good in software implementation as well. Trivium was submitted to the eSTREAM competition[22] by its designers[20] and was later selected for profile 2 of the eSTREAM project. Profile 2 was targeted for low area hardware ciphers. Trivium generates keystream output of size up to 2^{64} bits from a given key and IV, each of 80-bit size. We here avoid more details and the same can be had at reference[20].

TRAID, a stream cipher based family, consists of an authentication encryption mode (TRAID-AE) and a hash function (TRIAD-HASH). TRIAD-AE is basically an encryption-then-mac construction, in which a stream cipher TRIAD-SC is meant for performing encryption module. TRIAD was submitted to the lightweight crypto standardisation process of NIST[23]. TRIAD-SC takes 128-bit secret key and a 96-bit nonce as input along with one constant value (0xFFFFFFFE) and generates the keystream of desired length for encrypting the plaintext. We are not providing any further details about the algorithm and the same can be found in[21].

3. PROPOSED METHODOLOGY FOR CRYPTANALYSIS OF STREAM CIPHER

In this section, we propose a method that is primarily based on deep learning. We adopt this method for the cryptanalysis of stream ciphers with the target of finding biases in the generated keystream. The detailed stepwise algorithm is as follows:

0. Let the initial parameters be represented as:
a) Initial Key: K
b) Initial State: S
c) Keystream: KS

(Considering the black-box scenario, the updation of S is not known to our method)

1. Let $K = k_1 k_2 ... k_{n*8}$; where n denotes the length (in bytes) of the Initial Key.

2. With K as input, run stream cipher algorithm to generate the keystream KS

3. Iterate Step-1 and Step-2, N times for N different values of K to generate $"K - KS"$ pairs, i.e., *"InitialKey − Keystream"* pairs

4. Create a dataset consisting of these N pairs

5. Divide dataset into two parts, namely Training dataset and Test dataset having 80% and the remaining 20% of above-mentioned N pairs respectively

6. Pass the dataset to deep learning network by considering Initial Key K as input and 1st byte of corresponding KS as target

7. Train the network using training dataset to create a deep learning model

8. Validate the model on the test dataset and collect the result which shows the prediction probability of the model for 1st byte of KS

9. Repeat the process mentioned in steps from 6 to 8 for experimenting with 2nd, 3rd, 4th, . . . bytes of KS

10. i^{th} byte of KS is biased If Prediction Probability (for i^{th} byte of KS) is significantly more than 1/256 (0.0039), else it is random

4. ANALYSIS OF STREAM CIPHERS USING PROPOSED METHODOLOGY

The experimental set-up, as explained in Section 3, is used for cryptanalysis of RC4 by taking the following values:

length of Initial Key (n) = 5

#Samples/ Pairs (N) = 4,00,000

The experiment has been performed for predicting 1st, 2nd, 3rd, 4th, . . . bytes of RC4 keystream. As explained earlier, RC4 takes a variable-length Initial Key as input (length typically being in the range between 5 and 256 bytes) and iteratively generates a pseudorandom Keystream in byte-wise form as output. In our experimental setup, initially, we fixed the input key size to 5 bytes and generated 4,00,000 input-output pairs. The size of State Vector/ Initial State, which gets initialised by the initial key and generates keystream by its regular updation, is taken to be 256 for our experiments. It is re-emphasised that the variable Initial Key is the input, and the j^{th} byte of the Keystream denotes the target value in our experimental set-up. In other words, for one training/test pair, the Initial Key is the input, and j^{th} byte of correspondingly generated Keystream is the target value.

Initially, deep learning based prediction model was set-up for predicting the first output byte of RC4 keystream. For each sample in our supervised deep learning model, five bytes of initial key are taken as five neurons in input layer and first byte of the keystream is taken as target in output layer. For example, if initial key is *0xEB9F72AE1C* (in Hex format) and generated keystream is *0xBBF316E8D940AF0AD3* . . . (in Hex format), then *0xEB, 0x9F, 0x72, 0xAE* and *0x1C* are five neurons in input layer and first byte *0xBB* is the target in output layer. In total, 4,00,000 such input-output pairs were taken for creating the deep learning model. This data set was divided in 80% (3,20,000 in numbers) training and 20% (80,000 in numbers) test instances for validation of the method. Seven dense layers, each with 10 neurons and *ReLU* activation function, followed by an output layer with 256 neurons with Softmax activation function have been taken in creating the model. The model is compiled with *adam* optimizer and *categorical_crossentropy* as loss function[3]. Loss denotes the error between actual target and output value predicted by DL network. Adam optimizer, an extension to stochastic gradient descent, has found broader adoption in recent times after increasing popularity of deep learning techniques in diversified domains. Categorical

crossentropy is a loss function used in multi-classs classification tasks. Multi-class classification is related to the tasks where a given sample can belong only one of many possible categories. This actually quantifies the difference between two probability distributions. Ours is a supervised learning problem as the model is first trained with the help of the samples for which the classes are known. The process of backpropagation is used for fine tuning the weights for minimizing the loss. Training is done in five epochs and after five epochs results are collected for training and validation data. Similar prediction models were also created for predicting each of the first ten output bytes of RC4 keystream. The deep learning arrangement employed in our work is shown in Figure 4.

The similar arrangement of experiment has been used for cryptanalysis of RC4A stream cipher except the following parameter:

length of Initial Key (n) = 10

The length of initial key in case of RC4A is taken large in comparison to that for RC4 as the initial key for the former stream cipher is a combination of two key components. The rest of experimental set-up and the amount of data are exactly same in both cases.

For experimenting with two other stream ciphers, Trivium and TRAID-SC, we generated 2^{17} keystream samples for different respective inputs for each cipher. The objective was to compute the prediction accuracy for each of the first 256 bits of the keystream. We also performed the experiments for predicting the first 32 bytes of the keystream.

5. EXPERIMENTAL RESULTS AND COMPARATIVE ANALYSIS

The implementation of the code for the proposed method has been done on Google Colab[24]. Colaboratory (Colab in short) is a free cloud service provided by Google in Jupyter notebook environment where availability of GPUs and TPUs may be used for the development of codes for problem-solving. It allows developing deep learning algorithms using popular libraries such as PyTorch, TensorFlow, and Keras[25].

The broad objective of the experiment is to create deep learning model to predict the output bytes of RC4 keystream when the initial key is known to us. For validating the proposed method, we construct ten different deep learning models, one each for the first 10 bytes of the keystream. Table 1 shows the results of our experiments. We used 4,00,000 samples (input-output pairs) for constructing each deep learning model for the prediction of output bytes as the prediction rates were stable with this sample size. For this amount of samples, average execution time of the code/model over GPU runtime in Google Colab was observed to be 16 seconds. We also carried out the experiments with fewer samples and observed that the model was continually providing reasonable prediction rates even with 30,000 samples. But below this number, the results (prediction rates) were not consistent.

Theoretically, if a well-designed stream cipher generates a random keystream, then it may safely be assumed that the occurrence of a specific value at any output byte position in

Table 1. Prediction Probability for different output byte positions in RC4 Keystream for input secret keys of size 5 bytes

Byte Position in RC4 Keystream	Training Accuracy (Prediction Probability for Training Data)	Testing Accuracy (Prediction Probability for Test Data)
1	0.0040	0.0040
2	**0.0079**	**0.0073**
3	0.0040	0.0040
4	0.0041	0.0040
5	0.0040	0.0039
6	0.0040	0.0042
7	0.0040	0.0039
8	0.0039	0.0042
9	0.0040	0.0040
10	0.0040	0.0040

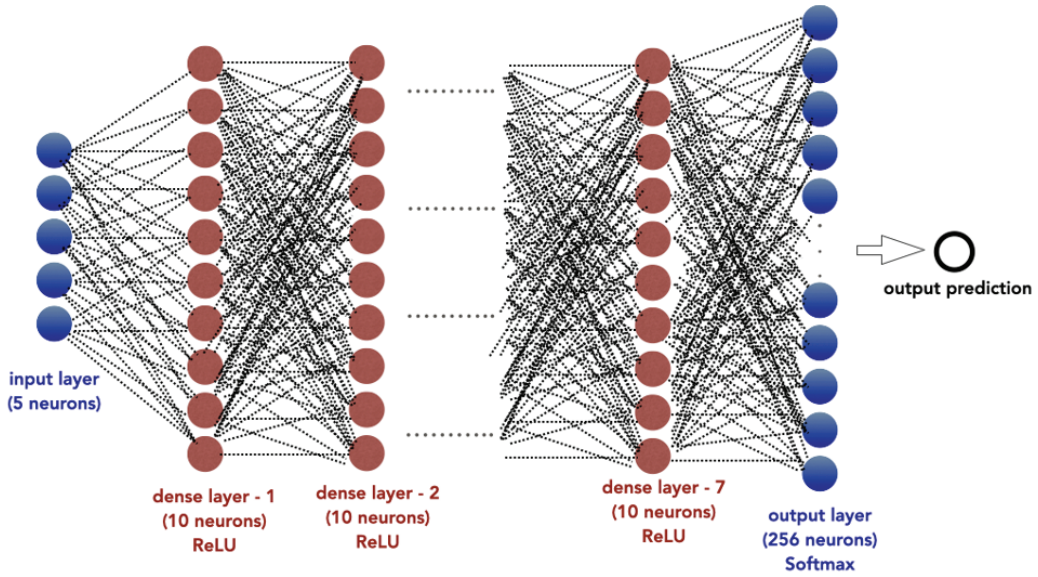

Figure 4. Deep Learning arrangement used in the experiment.

the keystream could not be predicted with better than chance probability (chance probability means the occurrence of each possible output should be equiprobable, like chances of occurrence of either head or tail in tossing a coin should be with equal probability i.e., 0.5).

Similarly, in our case, had RC4 stream cipher generated a random keystream, no method would have predicted the occurrence of a specific value at any output byte position of keystream with significantly better than 1/256 = 0.0039 probability (as a byte may take any value from 0 to 255). Our experimentation results show that the deep learning model has been able to predict the second output byte with a significantly good probability (i.e., 1/128 = 0.0073). In contrast, it could not learn to predict the occurrence of other output bytes with better than chance probability. For other output bytes of RC4 keystream, the prediction probability is approximately 1/256=0.0039. In other words, our model could learn the process of mapping the input initial key to the second byte of the keystream with a biased probability. This finding exposes the presence of bias at second byte position in RC4 keystream, which can enable the adversaries in mounting a practical ciphertext-only attack in some applications. We carried out similar experiments for four other different input initial key sizes of 8, 16, 24, and 32 bytes. In all the experiments, we obtained similar findings that that the model could learn the biased behavior at 2nd output byte of RC4 keystream with the probability around 1/128. For other output bytes, the prediction probability was approximately 1/256, which is the desired randomness for output byte of RC4 keystream. The overall results for our experiment with RC4 stream cipher are shown in Table 2.

The experiments on similar lines were also carried out for RC4A stream cipher. The experimental results for the same are presented in Table 3. The results clearly validate the designers claim that they have successfully eliminated the bias at second output byte in their improved variant of RC4 stream cipher (i.e. RC4A). In case of stream ciphers Trivium and TRIAD, the achieved accuracy was almost equal to chance probability. As nothing significant was observed for two ciphers, we omit showing the results here.

Mantin and Shamir[26] performed a comprehensive statistical analysis of RC4 keystream. Their result shows that the probability of the second output keystream byte of RC4 being zero is approximately double than expected if the initial permutation is randomly chosen. Paul and Praneel[19], after doing rigorous analysis, presented a new statistical bias in the distribution of the first two output bytes of RC4 keystream generator. Pudovkina[27] analytically attempted to detect a bias in the distribution of first and second output values of RC4 keystream considering certain uniformity assumptions. The methods used by these researchers and us along with respective observations are shown in Table 4. Several other researchers through statistical and mathematical means and experiments tried to find biases in RC4 and other stream ciphers by making

Table 3. Prediction Probability for different output byte positions in RC4A Keystream for input secret keys of size 5 bytes

Byte Position in RC4A Keystream	Training Accuracy (Prediction Probability for Training Data)	Testing Accuracy (Prediction Probability for Test Data)
1	0.0040	0.0042
2	**0.0044**	**0.0037**
3	0.0041	0.0046
4	0.0042	0.0040
5	0.0048	0.0033
6	0.0049	0.0046
7	0.0041	0.0036
8	0.0044	0.0044
9	0.0039	0.0037
10	0.0044	0.0041

Table 2. Prediction Probability for outputs bytes in RC4 Keystream for different input key sizes

Key Size in Bytes	Prediction Probability for Different Output Bytes of RC4 Keystream									
	1st Byte	2nd Byte	3rd Byte	4th Byte	5th Byte	6th Byte	7th Byte	8th Byte	9th Byte	10th Byte
5	0.0040	**0.0079**	0.0040	0.0041	0.0040	0.0040	0.0040	0.0039	0.0040	0.0040
8	0.0039	**0.0079**	0.0041	0.0037	0.0046	0.0038	0.0041	0.0035	0.0038	0.0042
16	0.0040	**0.0077**	0.0039	0.0042	0.0033	0.0042	0.0042	0.0038	0.0036	0.0042
24	0.0041	**0.0075**	0.0042	0.0041	0.0040	0.0042	0.0036	0.0044	0.0035	0.0041
32	0.0043	**0.0075**	0.0038	0.0038	0.0038	0.0038	0.0037	0.0039	0.0038	0.0036

Table 4. Comparative Analysis of different methods for finding weakness of RC4 in terms of approach and observations

Type	Observation	Source
Statistical analysis	The probability of second output word of RC4 being 0 is approximately double than the expected probability. Other output words have uniform distribution	Ref.[26]
Statistical analysis	The first two output words of RC4 are equal with probability that is significantly less than the expected probability	Ref.[19]
Exhaustive probabilistic model	The distribution of first, second output values of RC4 and digraphs are not uniform	Ref.[27]
Deep learning method	The prediction probability of second output word of RC4 is almost double than the expected probability. Other output words have prediction probability as expected in uniform distribution	This work

certain assumptions. Thus, the most of the findings have been based on manual calculation of the probability distribution in keystream data, which required much in-depth analysis of the cipher structure. On the other side, our proposed method does not require prior assumptions and any complex mathematical or statistical analysis for finding the bias. It simply needs input-output pairs of the stream cipher algorithm and passes them to the deep learning framework. Therefore, it can evidently be concluded that the proposed deep learning based method provides a straightforward approach for cryptanalysis of stream ciphers in terms of finding the biases.

6. CONCLUSION

The foremost application of the proposed work is to perceive distinguishability in different stream ciphers, which simply ensures mounting of distinguishing attack on stream ciphers. In other words, the distinguishing attack means if certain number of black boxes generate the pseudo-random keystreams from a known input key, then the black box behaving as a specific stream cipher can be identified amongst all. Thus, a deep learning based method has successfully been applied in finding the bias and further exploration of distinguishability in stream ciphers. The proposed method has been validated on RC4 stream cipher to ascertain its simplicity in comparison to earlier sophisticated mathematical and statistical methods. The observations and analysis throughout the paper clearly indicate that the emergence of new deep learning techniques will always go hand in hand with cryptographic analysis. The prediction accuracy can be impacted by many factors, including the network design, the training data volume, and the training time. We also carried out the experiments over three other stream ciphers RC4A, Trivium and TRIAD, but did not observe anything significant. In future, more experiments will be performed to gain better insights into the experimental setting. The aim will also be to analyze several other stream ciphers and explore better representation of cryptographic data which may become more relevant to deep learning framework.

REFERENCES

1. Bahdanau, D.; Cho, K. & Bengio, Y. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473, 2014, https://arxiv.org/abs/1409.0473 [Accessed on 01 Aug 2020].
2. Chen, C.; Seff, A.; Kornhauser, A. & Xiao, J. Deepdriving: Learning affordance for direct perception in autonomous driving. *In* Proceedings of the International Conference on Computer Vision, *IEEE*, 2015, pp. 2722-2730. doi: 10.1109/iccv.2015.312
3. Deng, L. & Yu, D. Deep learning: methods and applications. Foundations and trends in signal processing, Now Publishers Inc. Hanover, MA, USA, 2014, 7(3–4), pp. 197-387. doi: 10.1561/9781601988157
4. Rivest, R.L. Cryptography and machine learning. *In* Proceedings of the International Conference on the Theory and Application of Cryptology, Springer, Berlin, Heidelberg, 1991, pp. 427-439. doi: 10.1007/3-540-57332-1_36
5. Abadi, M. & Andersen, D.G. Learning to protect communications with adversarial neural cryptography. arXiv preprint arXiv:1610.06918, 2016, https://arxiv.org/abs/1610.06918 [Accessed on 31 July 2020].
6. Hesamifard, E.; Takabi, H. & Ghasemi, M. Cryptodl: Deep neural networks over encrypted data. arXiv preprint arXiv:1711.05189, 2017, https://arxiv.org/abs/1711.05189 [Accessed on 12 July 2020]. doi: 10.1145/3292006.3300044
7. Picek, S.; Samiotis, I.P.; Kim, J.; Heuser, A.; Bhasin, S. & Legay, A. On the performance of convolutional neural networks for side-channel analysis. *In* Proceedings of the International Conference on Security, Privacy, and Applied Cryptography Engineering, Springer, Cham, 2018, pp. 157-176.
8. Wang, H. Side-channel analysis of aes based on deep learning. https://www.diva-portal.org/smash/get/diva2:1325691/FULLTEXT01.pdf, 2019 [Accessed on 07 Aug 2020].
9. Gohr, A. Improving attacks on round-reduced speck32/64 using deep learning. *In* Proceedings of Annual International Cryptology Conference, Springer, 2019, pp. 150–179. doi: 10.1007/978-3-030-26951-7_6
10. Beaulieu, R.; Shors, D.; Smith, J.; Treatman-Clark, S.; Weeks, B. & Wingers, L. The SIMON and SPECK Families of Lightweight Block Ciphers. IACR Cryptol. ePrint Arch., 2013, p.404, https://eprint.iacr.org/2013/404.pdf [Accessed on 07 Aug 2020].
11. Baksi, A.; Breier, J.; Dong, X. & Yi, C. Machine learning assisted differential distinguishers for lightweight ciphers. *IACR Cryptol. ePrint Arch.*, *2020*, p.571, https://eprint.iacr.org/2020/571.pdf [Accessed on 14 Oct 2020].
12. Jain, A.; Kohli, V. & Mishra, G. Deep learning based differential distinguisher for lightweight cipher PRESENT. Cryptology ePrint Archive, Report 2020/846. https://eprint.iacr.org/2020/846. [Accessed on 14 Oct 2020].
13. Yadav, T. & Kumar, M. Differential-ML distinguisher: machine learning based generic extension for differential cryptanalysis, https://eprint.iacr.org/2020/913.pdf. [Accessed on 14 Oct 2020].
14. Xiao, Y.; Hao, Q. & Yao, D. D. Neural cryptanalysis: metrics, methodology, and applications in cps ciphers. *In* Proceedings of the Conference on Dependable and Secure Computing (DSC), IEEE, 2019, pp. 1-8. doi: 10.1109/dsc47296.2019.8937659
15. Maghrebi, H.; Portigliatti, T. & Prouff, E. Breaking cryptographic implementations using deep learning techniques. *In* Proceedings of the International Conference on Security, Privacy, and Applied Cryptography Engineering, Springer, Cham, 2016, pp. 3-26. doi: 10.1007/978-3-319-49445-6_1
16. Hochreiter, S. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *J. of Uncertain. Fuzziness Knowl.-based Systems,* **6**(02), 1998, pp. 107–116. doi: 10.1142/s0218488598000094
17. Jarrett, K.; Kavukcuoglu, K.; Ranzato, M.A. & LeCun, Y.,

September. What is the best multi-stage architecture for object recognition?. *In* Proceedings of the international conference on computer vision, IEEE. 2009, pp. 2146-2153.
doi: 10.1109/iccv.2009.5459469

18. Brownlee, J. Long Short-term Memory Networks with Python: Develop Sequence Prediction Models with Deep Learning. Machine Learning Mastery, 2017.
19. Paul, S. & Preneel, B. A New Weakness in the RC4 Keystream Generator and an Approach to Improve the Security of the Cipher. *In* Proceedings of the International Workshop on Fast Software Encryption, 2004, February, pp. 245-259. Springer, Berlin, Heidelberg.
doi: 10.1007/978-3-540-25937-4_16
20. De Canniere, C. & Preneel, B. Trivium. In New stream cipher designs. 2008, pp. 244-266. Springer, Berlin, Heidelberg.
21. Isobe, S.T., Meier, W. and Zhang, B., TRIAD v1., https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/TRIAD-spec.pdf, accessed on 01 Feb 2021.
22. Robshaw, M. The eSTREAM project. In New Stream Cipher Designs. 2008. pp. 1-6. Springer, Berlin, Heidelberg.
23. McKay, K., Bassham, L., Sönmez Turan, M. ^ Mouha, N. Report on lightweight cryptography (No. NIST Internal or Interagency Report (NISTIR) 8114 (Draft)). 2016, National Institute of Standards and Technology.
24. Bonner, A. Getting started with google colab. https://towardsdatascience.com, 2019 [Accessed on 07 Aug 2020].
25. Erickson, B.J.; Korfiatis, P.; Akkus, Z.; Kline, T. & Philbrick, K. Toolkits and libraries for deep learning. *J. of Digital Imaging*, 2017, **30**(4), pp. 400-405.
doi: 10.1007/s10278-017-9965-6
26. Mantin, I. & Shamir, A. A practical attack on broadcast RC4. In International workshop on fast software encryption, 2001, April, pp. 152-164. Springer, Berlin, Heidelberg.
doi: 10.1007/3-540-45473-x_13
27. Pudovkina, M. Statistical weaknesses in the alleged RC4 keystream generator. IACR Cryptol. ePrint Arch., 2002, p.171, https://eprint.iacr.org/2002/171.pdf [Accessed on 07 July 2020].
doi: 10.1007/3-540-44706-7_2

CONTRIBUTORS

Mr Girish Mishra completed his M.Sc. and M.Phil. in Mathematics from University of Rajasthan Jaipur, India. He has been working as a Scientist in Scientific Analysis Group, Defence R&D Organisation since 2003. He has worked and contributed in various areas related to cryptography, information security and machine learning. He has published more than 15 research papers in various international journals and conferences. He received DRDO Young Scientist award in 2010 and Lab Scientist award in 2007. His current areas of interest include machine learning, blockchain technology and cryptography.
In the current study, he contributed in development, execution, and analysis of the concept and the algorithm.

Dr Indivar Gupta completed his PhD from IIT Delhi, India. He has been working as a Senior Scientist in Scientific Analysis Group, DRDO since 2000 and has research contributions in various areas related to cryptography and information security. He has published more than 20 research papers in various international journals and conferences. Presently, his areas of research include computational algebra, number theory, cryptography, information security, and high-performance computing.
In the current study he made a contribution in the development of concept and technique.

Dr S.V.S.S.N.V.G. Krishna Murthy obtained his Post-graduation from Osmania University, Hyderabad and Doctoral degree from Indian Institute of Technology Kanpur, Kanpur. He has 21 years of experience in teaching. He is currently Associate Professor & Head, Department of Applied Mathematics, Defence Institute of Advanced Technology, Deemed University, Pune. His research interests include Mathematical Modelling, Finite Element Analysis in fluid flow through Porous Media, Numerical Method's for Partial differential equations, Numerical parallel Algorithms, Computational Fluid Dynamics. He was a recipient of Erasmus Mundus – WILL Power (Window India Learning Link Power) Fellowship `2010-11'. He has authored more than 30 publications in reputed journals other than conferences.
In the current study, he did the development of the concept and technique.

Dr Saibal Kumar Pal is working as a Senior Scientist and Divisional Head at Defence Research & Development Organisation, Scientific Analysis Group, Delhi. He has also worked as the Chief Information Security Officer (CISO) of DRDO during the period 2017 – 2019. Dr. Pal completed his PhD from University of Delhi in the area of Information Security. He has co-authored 3 books on Electronic Governance, AI & Data Science & has more than 250 research publications in peer-reviewed journals & international conference proceedings. His areas of interest are Cryptography, Cyber Security, Computational Intelligence and Information Systems. Dr. Pal is a recipient of Lab Scientist Award in 2010 and DRDO Scientist of the Year Award in 2012.
In the current study, he contributed in the development of the concept and technique.