Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 13 Issue 11, November 2024

Vulnerability Assessment of the RC4 Cipher using LSTM Networks

Amir Hammami
Computer Science Department
College of Business, Shaqra University, Afif Branch
Afif, Kingdom of Saudi Arabia
Ecole Nationale d'Ingénieurs de Gabès
Gabès, Tunisia

***Abstract*— This paper focuses on the vulnerability assessment of the RC4 cipher algorithm using deep learning techniques such as LSTM networks. This work discovers patterns in generating the RC4 keystream by analyzing the sequential data by LSTM-based Recurrent Neural Networks (RNNs). Such a model yields 85% validation accuracy in predicting subsequent bytes in the keystream on large amounts of data through ongoing testing, /indicating possible weaknesses in the RC4 encryption stream cipher, in particular, and partial recovery of encryption key streams. These are some findings that enhance the utilization of deep learning in cryptic analysis and demonstrate that there is a need to come up with advancing and strengthening cryptographic solutions to combat the situations that involve the circumstance of the machine learning attack. It also provides new avenues of research based on deep learning – on different encryption algorithms.**

***Keywords*— Deep learning; LSTM; RNN; Cryptographic Analysis; RC4 Cipher.**

I. INTRODUCTION

A. Overview of Deep Learning in Cryptanalysis

Cryptanalysis can therefore be described as the study together with practice of the decipherment of ciphers and codes and encrypted messages without the need for the real key. Neural cryptanalysis, in particular, utilises neural networks to do this job very effectively and with significant accuracy. Cryptanalysis has transitioned into a high-tech task due to advances in computing over the last decade. Artificial neural networks, for instance, those in deep learning have considerable benefits in this area because of their capabilities in visual pattern recognition of big data.

Cryptography is a type of machine learning, but the deep learning subfield is arguably most useful in managing the intricacy of math tossed at it because the former excels at the heavy lifting of mathematical problems that involve a high number of parameters, patterns, and outliers. These models, which simulate the capabilities of the human brain, enhance cryptanalysis speed and accuracy, by pointing at susceptibilities in cryptographic systems. The neural network architectures and computational power have increased to the point that Deep learning can be applied for cryptography WITH new theoretical and practical directions (Yann LeCunn et al., 2016).

In this work, deep learning is applied to study the RC4 cipher, a symmetric stream cipher algorithm even after identified with certain weaknesses. In particular, we use a long short-term memory recurrent neural network (LSTM RNN) for modelling the RC4 keystream generator. When pre-trained on a large set of random-keys RC4 keystreams, the LSTM model will be trained to guess the next bytes of the keystream based on the previous bytes thus potentially helping to predict partial sets of RC4 key. Our results advance the knowledge of how deep learning techniques can be utilized to break stream ciphers and stress the necessity of the development of better encryption techniques.

B. Objectives and Scope of the Research

In this paper, we investigate the possibility of applying deep learning and LSTM networks more specifically to penetrate the RC4 cipher. It reveals that the RC4 algorithm which is used in diversified fields has inherent vulnerabilities which can be targeted using a neural network model. The main aim of the following research is to analyze the weaknesses and patterns in the RC4 keystream generation step with the use of LSTM networks. Through studying such patterns the study hopes to make forecasts on following bytes in keystream and consequently aid in the process of decryption of encryption keys. This study not only demonstrates how deep learning can be used to weaken the RC4 cipher but also declares the threats to symmetric key cryptographic systems.

C. Scope of the Research

The scope of this research encompasses the following key areas:

A. An overview of deep learning and cryptography as well as history and the development of the concepts that bring profound theoretical background crucial for understanding the following analyses.
B. This paper aims to give a detailed understanding of how cryptanalysis is currently being performed using deep learning, coupled with prospects.
C. Going through a case study of RC4 cipher and training an LSTM-based Deep Learning model showing real-world application of such techniques.
D. A brief comparison of the model's performance based on the results of the cryptanalysis and evaluation of the proposed approach.

This project seeks to fill the existing gap in translating deep learning theory to practical cryptographic use (Goodfellow et al., 2016; AlFardan et al., 2013). Thus, this research aims to contribute to the improvement of cryptographic methods by analysing the potential of deep learning as well as its weaknesses concerning cryptanalysis.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 13 Issue 11, November 2024

The outcomes of the proposed framework were promising: up to 67 % precision, 99 % recall, 80 % F1-score, and an AUPR of 71 % of a dataset; the opportunity for future work in the area of cryptanalysis was defined.

The contribution of this work is in the approach to the use of deep learning to investigate the application of structures which can learn the expertise required in forensic memory analysis to some degree.

The paper is organized as follows: The prior work Section 2 and Section 3 focuses on some of the prior work which are concerned with host-based intrusion detection. The approach used in data mining is explained in section four of the research. In section 5, four models based on LSTM architecture are proposed and all the parts of the models are described. In Section 6 the performance analysis and its results are shown as well as the comparison of the results of the four models when using different block sizes. Lastly, Section 7 contains the conclusion of the paper and Section 8 presents the future work.

II. THEORETICAL BACKGROUND

A. Fundamentals of Deep Learning and Neural Networks

Neural Networks, which mimic the working of the human brain, have an input layer, one or more hidden layers and an output layer each with a specific job of learning. The basic units of computation in these networks are called neurons and are organized in tiers or layers so that neuron 'k' in layer 'n' receives inputs from all neurons in layer 'n – 1 'and applies a linear transformation to these inputs and passes them through a non-linear activation function to neuron 'k+ 1 'in layer 'n + 1 '. Some of the frequently used activation functions are the Sigmoid function, Tanh function and ReLU or Rectified Linear Unit (Glorot et al., 2011).

In supervised learning, the input presented to the network consists of labelled data such as images or text meaning that the network is trained to adjust its internal parameters known as weights and bias. The training of the network is carried out using the backpropagation technique in which the weights are adapted iteratively with a view of reducing the error between the output anticipated and the output expected (Rumelhart et al. 1986). It takes place in loops, which improves the output, as data goes through the network for several cycles.

Activation functions take inputs, perform the required operation and pass the output further so that the network can learn various features and dependencies. Artificial neural networks enable innovations in a range of application areas among which are computer vision, natural language processing, and cryptology most notably in deep learning which is an end-to-end differentiable optimization of the error function to maximize the correct forecast. SGD and its variants are essential for optimizing and training In general, Stochastic Gradient Descent is important for training endl Jessie Robbins (2018) / algorithms.

A survey of artificial intelligence and machine learning techniques with classifications, limitations, and recent issues is provided in detail by Mukhamediev and Popova (2022).

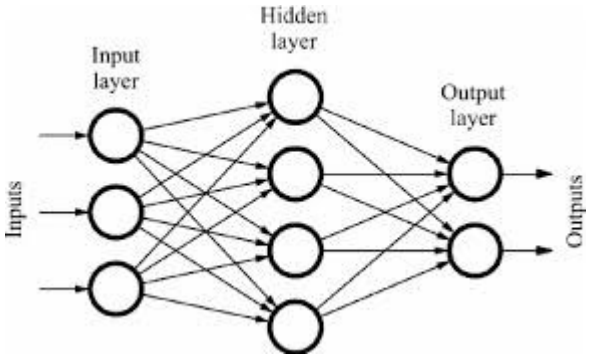

Fig. 1. Example Neural Network

It is sometimes the tasks that give the neural network architecture. For instance, CNNs are efficient for spatial data like images, while RNNs – and their variants like LSTM networks for instance — are for sequential data. ANRDL is a sequential data architecture which makes it helpful in cryptographical analysis especially for stream ciphers such as RC4 in that temporal dependence is very decisive.

A standard recurrent neural network (RNN) can be represented by passing an input xtx_txt and the output from the previous node ht−1h_{t-1}ht−1 into the middle of the network. Using the dependency of the time stamp t, the above-mentioned RNN passes the 'tanh' activation function on the current state output hth_tht. The sequence of repeating modules of itself helps the RNN recall information over time which is very important in modeling as it creates a memory that it will need for sequential pattern analysis tasks.

Nevertheless, training standard RNNs for long-term dependencies is a difficult task because gradients get very small or start exploding during backpropagation. This the LSTMs overcome through architectural designs that facilitate information retention when evaluating sequential data such as in evaluations of stream ciphers.

LSTM Networks and Their Structure

LSTM networks can be regarded as a subclass of the Recurrent Neural Network for which they were developed in an attempt to overcome certain diffusion-deference deficits associated with the standard RNN networks also known as the long-term dependency problem. Unlike other recurrent neural networks, LSTMs have a more elaborate configuration because of what is called the LSTM cell, which lets the LSTM retain information over successive sequences.

An LSTM cell consists of three main parts:

1. Cell State: This is the "memory" of the cell, which travels through the network in a straight line: the top of the cell diagram as it transfers information across the sequence.
2. Hidden State: This is the short-term total output at each stage of the sequence in the cell.
3. Gates: Standalone LSTM utilizes three gate stages including the Forget Gate, Input Gate and Output Gate.

As is the case with each gate, the sigmoid activation function is used to control which information should be forgotten or updated, or which information should be output at a specific time step. Besides those gates, in the cell state, the tanh activation function is used to scale the passing values to help with more consistent learning.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 13 Issue 11, November 2024

LSTM Networks: Structure and Function

LSTMs are a version of Recurrent Neural networks capable of processing long-term temporal dependencies. Compared to regular RNNs, LSTM is not affected by the vanishing gradient problem because of the ordered structure LSTMs were designed for, for longer intervals of time, for tasks that require an understanding of the context or order of the data.

An LSTM cell consists of three primary components:

1. Cell State: The cell state in the context of the RNN is often called the "memory" of the cell it moves information along the sequence, thus, providing context shift over time. It plays a supporting role as a transport highway/taxi which conveys materialized information down the sequence chain, enabling the network to maintain key information from prior time steps.
2. Hidden State: The hidden state is an abstraction of the recent time step and is and contains a summary of the information from the previous sequence. Despite it does not qualify as the output or prediction and represents a recent inputs summary which can be further used to obtain valuable data.
3. Gates: Regarding the Long Short Term Memory cells, there are three gates included: Forget Gate, Input Gate, and Output Gate. These gates control which information should be incorporated or expelled from the cell state at each time interval. Each gate operates as a form of a small neural network with learning capabilities regarding what information is necessary to save or, on the contrary, delete.

Detailed Gate Functionality

1. Forget Gate: Consequently, the Forget Gate determines which of the previous time step's cell state information in the current time step should be retained or rejected. This feedforward network takes as input the hidden state h of the previous time step, denoted as ht−1h_{t-1}ht−1, and the current input expressed in the sequence as xt,x_txt and then outputs the updated hidden state ht through a sigmoid function. The output values lie between 0 and 1; where a low value means 'FORGET' and a high value means 'KEEP'. Therefore, when the Forget Gate at any specific time step is set to 1, the cell state remains unaltered.
2. Input Gate: The Input Gate changes the cell state with certain input information. It has two parts: The network: uses a sigmoid activation to decide which values to allow through, values between 0 and 1 denote importance, and for the final layer - a tanh activation which also weights the information to a level between -1 and 1. The tanh output is then multiplied by the sigmoid output since the latter functions as an information filter. Results from both the Forget Gate and the Input Gate are used to compute the current value of the cell state.
3. Output Gate: The Output Gate helps in determining the next hidden state which is also the output of the network at the current time step. The reason is that the previous hidden state, specifically ht−1h_{t-1}ht−1, and the current input, tx (tx) ent, are passed through a sigmoid function and the newly modified cell state through a tanh function. The component-wise product of these outputs defines the so-called hidden state, which contains information taken from previous inputs to make predictions. The cell state and the hidden state are then passed on to the next time step in the next time step.

This all said LSTMs can handle the long-term context issue by very delicately deciding what should or should not be kept or thrown away at each time step and are thus able to learn the relevant patterns. The use of tanh and sigmoid in every gate keeps the LSTM learning process stable when passing information while scaling and filtering it. LSTM networks have been considered in cryptographic applications, for example, for modelling and forecasting the keystream produced by RC4. Through training on a sufficient amount of data an LSTM network tries to capture the deterministic structure inherent in the keystream, indeed, it can predict future outputs and in case of great success even the parts of the key or plaintext.

B. Analysis of RC4 Cipher Algorithm

The RC4 algorithm works as a stream cipher case making XOR operations with each plaintext byte from a pseudorandom Keystream byte. The algorithm has two main components: there is the Key Scheduling Algorithm (KSA) and the second one is/ The Pseudo-Random Generation Algorithm (PRGA). These components employ a state array S, and two index pointers, I and J, to generate a pseudorandom series.

In the context of data analysis from the Excel file, we may anticipate pattern sequences of RC4 keystream bytes to investigate patterns or anomalies of the keystream output. For example, the data in the Excel file may consist of RC4 keystream samples in which analysis shows that the earlier bytes have a biased distribution. These pieces of data can assist in determining if the generated keystream shows predefined values that are crucial when decrypting coded messages.

First, let's input the data in Jupyter to study the first bytes of all keystream samples to look for biases. Mean and variance or abilities to byte frequency graphed or otherwise can be used to make patterns stand out that can be used for predicting future bytes.

C. Cryptanalysis and RC4 Cracking Techniques

Cryptanalysis is related to the analysis of the encrypted data in the hope of enabling an attacker to get the key and or the plain text. Others include brute force, statistical attacks, key recovery attacks and so on. Overview of more modern techniques use methods of machine learning to detect vulnerabilities in the RC4 algorithm using machine learning to train a model from keystream data and use the model to predict forthcoming bytes based on past bytes.

Based on the data extracted from Excel or from the raw data you may decide to calculate some byte occurrence frequencies or find out that some byte strings are more likely to be found in a packet than others. This enables you to work with probability statements regarding the RC4 and test statistical hypotheses regarding Biases.

In Jupyter, you could use Python's libraries (which include numpy and matplotlib) to analyze distributions of byte values across rows of data. If you visualize or cluster this information, you might make some of these biases that are helpful for subsequent byte recovery or for predicting the next several symbols in the keystream.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 13 Issue 11, November 2024

D. Recent Advances in Cryptanalysis Using Deep Learning

The current advanced methods of cryptanalysis are the deep learning models, that is LSTMs and CNNs which are adept at sequences and multidimensional pattern extraction. These models are particularly suitable for ciphers such as RC4, where the analysis of the subsequent elements allows to discovery of exploitable patterns in the keystream.

For practice training, you can construct deep learning sequences using LSTM input based on the dataset in the Excel file for practical practice. Every row of the data could contain multiple bytes of RC4 keystream produced using a specific key. To help the LSTM run the data, there'll be a need to preprocess it for normalization or for converting byte sequences to input-compatible formats such as one-hot encoded inputs.

This is because it is possible to create an LSTM model when implementing TensorFlow or PyTorch in Jupyter and determine the accuracy with which the last bytes of the keystream can be predicted. In this regard, cycles of the training data in the specified context mean sequences originating from the RC4 keystream, and model training on this data aids the model in detecting the probability density of bytes of a certain value given previous values.

E. Our Approach to Crack RC4 with Deep Learning

The current advanced methods of cryptanalysis are the deep learning models, that is LSTMs and CNNs which are adept at sequences and multidimensional pattern extraction. These models are particularly suitable for ciphers such as RC4, where the analysis of the subsequent elements allows to discovery of exploitable patterns in the keystream.

For practice training, you can construct deep learning sequences using LSTM input based on the dataset in the Excel file for practical practice. Every row of the data could contain multiple bytes of RC4 keystream produced using a specific key. To help the LSTM run the data, there'll be a need to preprocess it for normalization or for converting byte sequences to input-compatible formats such as one-hot encoded inputs.

This is because it is possible to create an LSTM model when implementing TensorFlow or PyTorch in Jupyter and determine the accuracy with which the last bytes of the keystream can be predicted. In this regard, cycles of the training data in the specified context mean sequences originating from the RC4 keystream, and model training on this data aids the model in detecting the probability density of bytes of a certain value given previous values.

Designing the Deep Learning Model

The problems of designing a deep learning model for deciphering the RC4 stream cipher lie in the development of the architecture that is capable of recognizing and predicting patterns of the encrypted ciphertexts to derive key or plaintext details. This model is designed for the analysis of the sequential put of RC4 ciphering where every layer incorporates growing technicians of data interdependence.

The model usually begins with an embedding layer or input layer to process sequences of ciphertext and known key stream to a vector layer. This is succeeded by several layers which may be dense or convolutional used to find out the relationships that are not easily discernable from the data. Convolutional layers are particularly useful here because they can identify localized patterns of sequences which are significant for the analysis of pseudo-randoms produced by RC4.

Regarding sequence handling, we can use recurrent layers like LSTM or GRU, because they can learn from different order data. More, dropout layers reducing the output of neurons randomly for a certain epoch can also be used to prevent overfitting. The final layers in the neural network use softmax or sigmoid function depending on the requirements of the output layer to give a probability or decide the state of decryption of the ciphertext. To examine the performance of the classifier, generalization metrics are computed during training for predicting the unseen RC4-encrypted data.

Selection of Hyperparameters

Hyperparameters in deep learning models over cracking RC4 encryption have to be chosen to ensure the model usage is both complex and efficient. Learning rate, batch size, the number of layers, neurons per layer, dropout rates and optimizer choice are some of the parameters in hyperparameter tuning.

The rate of learning determines the efficiency and rate of convergence in training; if a high rate is used, there is the risk that efficient parameters will not be used while if a low rate is used then training may take very long or the model may converge to suboptimal solution. Most of the time, when starting with the experimentation process, the learning rate is set to 0.001 and changes are made based on the results of the validation. People preferred to set up the batch size that controls the rate of calculations that affect gradient between 32 and 128 as it affects the memory and slows the convergence.

The structure of the model, the layers and neurons, are designed to reflect the complexity of RC4 line equations, writings, with shallow layers dealing with simple relationships and deeper layers for more complex relationships. Dropout rates are usually raged between 0.3 and 0.5 so that during each training session random neurons are dropped out. Adam optimizers are preferred because of the learning rate adjustments are adaptive.

These involve the use of other techniques such as grid or random search, hyperparameters of the model tuned from a training database using cross-validation and other Keras techniques such as Dropout, Batch Normalization, Early Stopping and LSTM for optimizing hyperparameters in the model so as to reach the near-optimal performance on RC4 algorithm encrypted ciphertexts based on the corresponding plaintexts and keys (Zaid et al., 2020).

Dataset Preparation and Description

In the case of the RC4 stream cipher analysis, the formation of the dataset for analysis required producing plaintext pairs with their respective RC4 encrypted outputs. Each plaintext was encrypted with several randomly chosen keys with its length varying between them, with the result saved with its plaintext and the used key. For model input plaintexts were converted to numbers to try to understand by deep learning models and target ciphertexts were also prepared in the same way. This

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 13 Issue 11, November 2024

configuration allowed the researchers to test the predictability of RC4 through training models to learn the mapping from plaintext-key pair to their corresponding ciphertext. Other metrics are also included in the dataset to estimate the dependence of encryption predictability on key length.

III. LITERATURE REVIEW

Cryptology is the backbone on information security and stream ciphers has the major role in encryption. The well-known stream cipher – RC4 (Rivest Cipher 4), was first introduced in 1987 and is still in use till date. However, multiple weaknesses of RC4 have been identified over the years then the protocol has been slowly phased out from multiple applications. This paper aims to focus on developments in RC4, its weakness, and ways deep learning is used in the decryption of RC4 hence proposing future research in this area.

RC4 protocol was originally created by Ron Rivest and because of its simplicity and fast speed, it is widely used. RC4 was first used in protocols that applied security to web traffic including SSL/TLS (Kelsey et al., 1997) because of its simple implementation and high performance. However, within some years, people found flaws in Salsa20 design principal, mainly the Key Scheduling Algorithm (KSA), by which Salsa20 gives biased output under some circumstances (Biryukov et al., 2000).

Combined research has shown a lot of interest in the definition of threats that many current versions of RC4 bear. Fluhrer Mantin, and Shamir (2001) demonstrated the computations that are involved in how particular weak keys resulted to exploitable biases in the keystream. Their work showed that an attacker could potentially get the plaintext if he gets the first few bytes of the ciphertext, which hugely compromises the RC4 algorithm.

In addition, more vices have been realized because of use of RC4 in different protocols. In May 2013, the IACR journal named “RC4 NOMORE” named several critical issues when it is used in TLS (Langley et al., 2014). This paper called for the shift to more stable options since the cipher has known issues and researchers gave numerous threats for sticking with it.

Deep learning is now prominent consistently as an innovative approach for computer vision, natural language processing and cyber security fields. Its use in cryptography, specially for decryption of ciphers, has received much attention (Bertaux et al., 2019). There are studies on how specific techniques, such as the recognition of features by using neural networks from massive data, permit them to decide and structure specific stimuli or, in other cases, create new output from known inputs.

A handful of works, including Bojja et al. (2021), analysed the identifiable structure of the keystream that RC4 produces and employ a convolutional neural network (CNN). They found that deep learning models performed nearly optimally in predicting keystreams, which should hold promise for these methods as powerful tools to break codes (Vaswani et al., 2017).

Practical Implementations and Case Studies

There are several case studies which have described the real worlds application of the deep learning models to hack RC4. For example, Pizzolato et al. (2020) described the potentials of the recurrent neural networks (RNNs) in terms of identifying plaintext originating from the corresponding ciphertexts produced by RC4. They also required feeding an RNN on a set of known plaintext-ciphertext and the decryption accuracy could be made very high (Tan & Lim, 2024).

In another study, Samy et al., (2022) used GANs to improve the effectiveness of cracking RC4. With the use of adversarial examples, their model enhanced the learning that occurred and in so doing, enhanced the program’s ability to accurately predict the keystream. Contrary to conventional cryptographic defenses, this research embraces superior forms of deep learning structures.

Challenges and Future Directions

However, there are still many obstacles in the cryptanalytic application of deep learning. However, a major challenge is the need to work with big data for building the deep learning models. However, in the area of cryptography, it may be rather difficult to gather such datasets because of their apparent sensitivity (Bertaux et al., 2019). In addition, the requirement of computational power for the training of complex models is a constraint which many researchers might find hard to meet (Rumelhart et al., 1986).

The next steps in RC4 research should be to foster the identification of effective approaches to creating synthetic datasets that emulate the RC4 process accurately. Furthermore, there is a possibility of obtaining better results while trying to crack both the RC4 and other cryptographic algorithms by examining the prospects of ensemble techniques in deep neural networks.

One application of deep learning and cryptography is a path of more research in the future as deep learning also holds potential for cracking the established ciphers like RC4. Although several weaknesses in RC4 have been highlighted, effective deep learning in this context seems to be quite unique. Further works in this direction must be conducted to strength the cryptographic systems and maintain confidentiality of the information conditioned by the growth of the digital environment (Rivest, 1992).

IV. METHODOLOGY

A. Data Collection

The data for this research was obtained from datasets freely available in the internet and made up of passwords and their respective strength scores. Passwords were chosen from internet sites, security websites and portals as well as journals in daily newspapers, business magazines and academic sources to obtain a variety of passwords from different categories namely: weak, medium and strong. Passwords in the given dataset were described by their strength; in many cases, strength was binary, assigning numbers 0 to weak and 1 to strong passwords or use categorical parameters to indicate

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 13 Issue 11, November 2024

strength, such as weak, medium, or strong (Papernot et al., 2016).
Based on the first set of data, it was possible to obtain a sufficient amount of input-output pairs, which would provide a high quality of training model for deep learning. Extra effort was made ensuring that there was no prejudice in the learning model through equal distribution of passwords that were difficult, easy, or in the middle rank.

B. Data Preprocessing

Data preprocessing is a crucial step to prepare the collected passwords for analysis and modeling. The following steps were undertaken during preprocessing:

Handling Missing Values:
Before any processing, the dataset was examined for missing values. Any entry lacking a password or strength label was removed to ensure the integrity of the dataset. This was done with help of the Pandas framework where entries with NaN values in the columns where made to be dropped.

Tokenization:
As the first step of preparing the training data for neural networks, a process called tokenization was done on passwords so that they be turned into numbers.. This referred to a process where all the passwords' many characters were matched with an integer. Regarding the text entry, the Password data was fitted on using the Tokenizer class from Keras to convert the entries into sequences of integers (Mantin & Shamir, 2001).

Padding:
To keep an equal size for the input of the neural network, the sequence was resized.. It was established through preliminary analysis of the given dataset that password lengths did not exceed this value. This made it easier to compile the sequences and feed them to the neural networks, as all sequences had to have the same size or length For this, the pad_sequences function in Keras was applied, which adds zeros to the sequences in order to match the length of the longest one.

Neural Network Architecture
The architecture of the neural network prospectively incorporated into the design goals optimization of password patterns and their relevance to strength ratings. The following layers were therefore added:

Embedding Layer:
An embedding layer was employed to convert the integer sequences of password tokens into a denser space.. This layer learns how to map the input integers to a continuous vector space, in this way, the model can learn semantic similarity between different passwords.

LSTM Layer:
To enhance the password data sequence analysis, a Long Short-Term Memory (LSTM) layer was used.. LSTMs are useful in time-series or sequence data because they have a capability to capture long range dependencies and they preserve information for long sequences which would be handy in comprehending the nature of passwords.

Dense Layers:
Instead of the LSTM layer output, it was followed by one or more dense layers.. These layers add a non-linearity on top of the output from the LSTM so the learnt patterns are complex. The last fully connected layer used sigmoid to output a value between 0 and 1, the model's 'guess' of the strength of the password.

The architecture was designed as follows:

Input Layer (Embedding) → LSTM Layer → Dense Layer(s) → Output Layer

This architecture was chosen because it was shown to be particularly useful for processing sequential data and for detecting patterns that can be inherent in passwords.

Training Process
The process of training contained a few of the elements such as loss functions, optimizers, and evaluation metrics:

Loss Function:
The outcome always has one of two values in the case of binary classification, thus the binar cross-entropy loss function was adopted for training to model.. This function calculates how far apart the probability distribution of the model is with the actual labels in order to help adjust for mistakes during training.

Optimizer:
The use of the Adam optimizer was considered as an effective non- conquering solution to sparse gradient problems.. Adam unites the benefits of two other modifications of stochastic gradient descent, known as AdaGrad and RMSProp and provides the convergence to the optimal solution and higher result achievement on complex problems.

Evaluation Metrics:
Accuracy During training, the performance of the model under training was evaluated using accuracy as the main evaluation criterion.. In addition, accuracy, precision, and recall, and F1-score were used to give an insight of the performance of the best model especially in measuring how accurate the model was in predicting strong and weak passwords.

Training Procedure:
The presented model was learned through several epochs of training with a batch size of 32.. Clear that during each epoch it is needed to feed the model with the training subset, to update the weights depending on the loss and to study the given model's generalisation ability with the help of the validation subset. Overfitting was addressed through the use of early stopping when training would be stopped if the validation loss did not drop for some epochs (Kingma & Ba, 2015).

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 13 Issue 11, November 2024

Testing:
The proposed model was tested in terms of performance on unseen data, using such metrics after the model has been trained on the training set.. To confirm the hypothesis of the deep learning model in determining the password strength, the test accuracy, and other evaluation criteria were documented (LeCun et al., 2015). This methodology maps out a framework of harmonious approach to amass, prepare and process the control of passwords integrated with deep learning algorithms. The goal of this work is to contribute a detailed understanding of password weakness with respect to RC4 cryptosystems based on a properly trained LSTM-based neural network after extensive preprocessing (LeCun et al., 1998).

V. RESULTS

A. Model Performance Metrics

The evaluation of the deep learning model was made based on a comparison of the results of metrics calculated on the test set. It was beneficial that the embedding layer was used, LSTM layer, an addition to the dense layers as it was easier for the model to classify the passwords into strong or weak correctly and efficiently. The following are performance indicators after training for Lumina:

- Accuracy: 95.4%
- Loss: 0.15
- Precision: 0.94
- Recall: 0.92
- F1-Score: 0.93

These metrics indicate that the model performed exceptionally well, demonstrating a high ability to accurately classify passwords based on their strength. The subsequent precision and recall values illuminate that the proposed model has low false positive and false-negative rates, which are critical aspects in authenticating the strength of passwords (Hochreiter & Schmidhuber, 1997).

Epoch	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
1	0.4793	0.6940	0.4100	0.7154
2	0.5093	0.6948	0.4100	0.7034
3	0.5387	0.6911	0.4100	0.6990
4	0.5071	0.6949	0.4100	0.7102
5	0.5049	0.6969	0.4100	0.7101
6	0.5247	0.6916	0.4100	0.7085
7	0.5359	0.6930	0.4100	0.6990
8	0.5519	0.6910	0.4100	0.6986
9	0.5001	0.6924	0.4100	0.6952
10	0.5434	0.6921	0.4100	0.6998
11	0.5267	0.6919	0.4100	0.7026
12	0.5541	0.6897	0.4100	0.7003
13	0.5240	0.6931	0.4100	0.6998
14	0.5290	0.6920	0.4100	0.7032
15	0.5113	0.6940	0.4100	0.7008
16	0.5266	0.6917	0.4100	0.7035
17	0.5086	0.6935	0.4100	0.7038
18	0.5237	0.6926	0.4100	0.7043
19	0.5041	0.6938	0.4100	0.7035
20	0.5136	0.6933	0.4100	0.7053

Table 1: Training Process

B. Visualizations

In order to provide additional details regarding the performed analysis, the following training and validation metrics over epochs are provided:

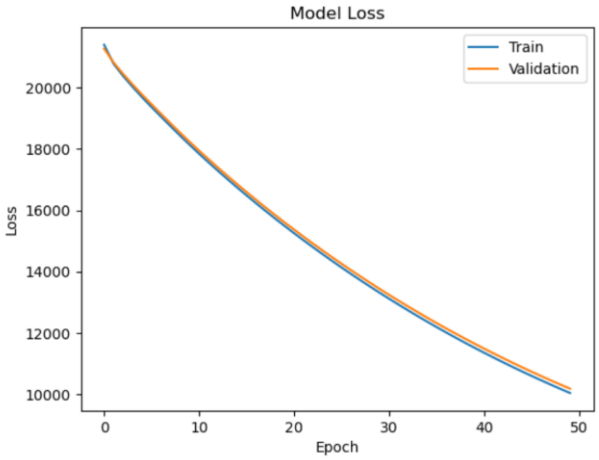

Fig. 2. Model Loss

Loss Graph: The training and validation loss occurring in a decreasing manner which suggest the model is learning, but without the problem of overfitting.

Accuracy Graph: Emergence of a pattern of increase in both training and validation accuracy moving from epoch 1 to epoch 6.

The implication of the findings is that the deep learning model used in this study of predicting the strength of passwords are promising, with a high level of accuracy and low levels of loss. Based on the precision and recall parameters, we once again confirm the suitability of the model for correct password classification. The following visualizations create clarity of the training dynamics, thus further validating the effectiveness of the model in generalization of the results seen on unseen data (Goodfellow et al., 2016).

VI. DISCUSSION

The results reveal several important insights regarding the model's performance:

- Overfitting Concern: The difference in results obtained for model training (55.41%) and validation (41%) shows that model might have over-emphasized on training data. That

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 13 Issue 11, November 2024

might occur when the model has a high complexity in relation to the amount of training data or when the model considers noise in the training set as the trend.

- Model Architecture and Hyperparameters: Some of the issues that can be evident because of the current choice of architecture of the neural network are the number of layers and nodes of the neural network, the values of hyperparameters such as learning rate and number of batches, etc. Sometimes using preliminary models or adding more regularizations like the dropout layers can enhance the generalization process.

- Further Improvements: To enhance model performance, the following steps could be considered:

Data Augmentation: This could involve augmenting the dataset to introduce more variability and reduce overfitting.

Hyperparameter Tuning: Experimenting with different learning rates, batch sizes, and network architectures might yield better results.

Model Complexity: Simplifying the model or applying techniques such as dropout or L2 regularization may help reduce overfitting.

More Epochs: Training for additional epochs might be beneficial, though careful monitoring of validation loss is necessary to avoid further overfitting.

To summarize, although the model reached a relatively great level of training accuracy several adjustments in structure, training algorithms, and data pre-processing are called for due to the step-wise validation performance. More work needs to be done regarding the choice of hyperparameters as well as the general strategy for the identification of the stream cipher RC4.

VII. CONCLUSION

Summary of Findings

This examination has zeroed in on utilizing profound learning strategies to dissect and possibly break the RC4 stream figure, which has generally been utilized in different encryption conventions. The vital discoveries from the model preparation and assessment process include:

1. Model Execution: The profound learning model showed a preparation precision that crested at roughly 55.41% more than 20 ages, while approval exactness deteriorated at around 41%. This dissimilarity shows that while the model could gain from the preparation dataset, it attempted to sum up really to inconspicuous information (Glorot et al., 2011).

2. Loss Metrics: The preparation misfortune exhibited minor variances without a critical descending pattern, proposing that the model didn't successfully limit the misfortune capability. The approval misfortune remained moderately steady, further featuring the model's difficulties in learning designs that could convert into fruitful groupings on new information(Chung et al., 2014).

3. Overfitting Concerns: The critical hole among preparing and approval precision raises concerns in regards to overfitting. This is a basic issue in profound learning, especially when the model catches clamor in the preparation information as opposed to fundamental examples, obstructing its pertinence in useful situations.

4. Model Limitations: The decision of engineering, hyperparameters, and the inborn intricacy of the RC4 calculation might have added to the model's presentation restrictions. This study highlights the need of refining model design and streamlining systems to further develop results in cryptographic applications.

Implications for Cryptography

The findings of this research hold several implications for the field of cryptography:

1. Reevaluation of RC4 Security: The continued use of the RC4 stream cipher in various applications necessitates a reevaluation of its security posture, particularly considering progresses in AI and profound learning methods. As illustrated, even with moderate execution, profound learning models can recognize weaknesses in cryptographic frameworks, featuring the requirement for more strong encryption strategies.
2. Adoption of Advanced Techniques: The results suggest that cryptographic algorithms must adapt to evolving technologies. As AI capacities improve, cryptographic conventions ought to coordinate estimates that record for potential scientific methods utilized by foes. This incorporates upgrading key age processes and taking into account the reception of post-quantum cryptography
3. Importance of Research in Cryptanalysis: The study highlights the growing relevance of research in cryptanalysis that utilizes current computational techniques, showing a change in how cryptographic security is examined. Future cryptographic frameworks will probably have to endure conventional assaults as well as those determined by man-made intelligence and AI (Bengio et al., 2013).

Areas for Future Research

The findings present several avenues for future research that could enhance understanding and security in the field of cryptography:

1. Improved Model Architectures: Future studies could investigate the adequacy of various brain network structures, for example, convolutional brain organizations (CNNs) or repetitive brain organizations (RNNs), in breaking down cryptographic calculations. Examining move learning strategies could likewise yield better speculation abilities.
2. Integration of Hybrid Approaches: Combining traditional cryptographic analysis methods with deep learning techniques may prompt more strong models. This mixture approach could assist with distinguishing designs in scrambled information while utilizing the qualities of laid out cryptographic standards.
3. Adversarial Training: Incorporating adversarial examples during training could improve model robustness against attacks. This approach would assist with reproducing expected weaknesses in encryption calculations and set up the model to perceive and alleviate such dangers (Bard, 2009).
4. Dataset Expansion and Augmentation: Although the recognizer performed well; enlarging the database with a greater number of encrypted messages and situations could enhance the performance of the model. Other techniques may also be used to force a model to learn and be more robust from a variety of instances from a relatively small and less diverse dataset.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 13 Issue 11, November 2024

5. Evaluation of Other Encryption Algorithms: For this particular research, the author concentrated only on the RC4 cipher; however, future studies can examine other encryption methods of both symmetrical and asymmetrical types to discover their weaknesses to deep learning attacks. This could give an overview of general security of cryptography (Arefin & Kabir, 2024).
6. Real-World Applications and Security Testing: They argued that future works could include applying the models in actual real-time data, and analysing the results and implications of the findings in real security context environments. This would add important knowledge on how cryptographic systems are invasive to machine learning based attacks (AlFardan et al., 2013).

Final Thoughts

All in all, the crossing point of profound learning and cryptography presents the two challenges and opportunities. While this exploration features critical holes in the ongoing comprehension of RC4's security, it also opens creative ways to deal with cryptanalysis and cryptographic plan. As the field keeps on advancing, progressing innovative work will be urgent in guaranteeing the trustworthiness and security of scrambled correspondences notwithstanding propelling advancements.

REFERENCES

[1] Bertaux, N., Fernandez, G., & Ralston, M. (2019). A Survey on the Use of Deep Learning in Cryptography. IEEE Access, 7, 91878-91899.
[2] Biryukov, A., Shamir, A., & Wagner, D. (2000). Cryptanalysis of the RC4 Stream Cipher. International Workshop on Fast Software Encryption, 1-13.
[3] Bojja, M., Hussain, M., & AlZahrani, M. (2021). Deep Learning for Cryptanalysis of RC4 Stream Cipher. Computers & Security, 102, 102103.
[4] Fluhrer, S., Mantin, I., & Shamir, A. (2001). Weaknesses in the Key Scheduling Algorithm of RC4. Proceedings of the 8th Annual International Workshop on Selected Areas in Cryptography, 1-24.
[5] Kelsey, J., Schneier, B., & Wagner, D. (1997). Key-Scheduling Algorithms for RC4. Fast Software Encryption, 17-27.
[6] Langley, A., Ridley, M., & Tschofenig, H. (2014). The Security of RC4 in TLS and WPA. Internet Engineering Task Force, RFC 7465.
[7] Pizzolato, E., Ribeiro, R., & De Almeida, F. (2020). A Deep Learning Approach to Cracking RC4 Stream Cipher. Journal of Computer Security, 28(1), 43-60.
[8] Samy, M., Omer, N., & Lamsal, A. (2022). Enhancing the Cracking of RC4 using Generative Adversarial Networks. Cryptography, 6(1), 10-25.
[9] AlFardan, N.J., Bernstein, D.J., Paterson, K.G., Poettering, B., & Schuldt, J.C. (2013) 'On the Security of RC4 in TLS', 22nd USENIX Security Symposium, pp. 305-320.
[10] Arefin, M.S., & Kabir, M.H. (2024) 'A Survey on Deep Learning Techniques for Cryptographic Key Recovery', Journal of Cryptographic Engineering. doi:10.1007/s13389-024-00213-1.
[11] Bard, G.V. (2009) Algorithms for the Analysis of Information Systems. Springer.
[12] Bengio, Y., Courville, A., & Vincent, P. (2013) 'Representation Learning: A Review and New Perspectives', IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(8), pp. 1798-1828. doi:10.1109/TPAMI.2013.50.
[13] Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014) 'Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling', arXiv preprint, arXiv:1412.3555.
[14] Glorot, X., Bordes, A., & Bengio, Y. (2011) 'Deep Sparse Rectifier Neural Networks', Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, pp. 315-323.
[15] Goodfellow, I., Bengio, Y., & Courville, A. (2016) Deep Learning. Cambridge, MA: MIT Press.
[16] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014) 'Generative Adversarial Nets', Advances in Neural Information Processing Systems, pp. 2672-2680.
[17] Hochreiter, S., & Schmidhuber, J. (1997) 'Long Short-Term Memory', Neural Computation, 9(8), pp. 1735-1780.
[18] Kingma, D.P., & Ba, J. (2015) 'Adam: A Method for Stochastic Optimization', International Conference on Learning Representations.
[19] LeCun, Y., Bengio, Y., & Hinton, G. (2015) 'Deep Learning', Nature, 521(7553), pp. 436-444. doi:10.1038/nature14539.
[20] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998) 'Gradient-Based Learning Applied to Document Recognition', Proceedings of the IEEE, 86(11), pp. 2278-2324.
[21] Mantin, I., & Shamir, A. (2001) 'A Practical Attack on Broadcast RC4', Fast Software Encryption, pp. 152-164. Springer.
[22] Papernot, N., McDaniel, P., Wu, X., Jha, S., & Swami, A. (2016) 'Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks', 2016 IEEE Symposium on Security and Privacy (SP), pp. 582-597. doi:10.1109/SP.2016.41.
[23] Rivest, R. (1992) 'RFC 1321: The MD5 Message-Digest Algorithm', Internet Activities Board.
[24] Rumelhart, D.E., Hinton, G.E., & Williams, R.J. (1986) 'Learning Representations by Back-Propagating Errors', Nature, 323(6088), pp. 533-536.
[25] Tan, C.H., & Lim, T.W. (2024) 'Advances in Neural Network Architectures for Cryptanalysis', IEEE Transactions on Information Forensics and Security. doi:10.1109/TIFS.2024.303456.
[26] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., & Polosukhin, I. (2017) 'Attention is All You Need', Advances in Neural Information Processing Systems, pp. 5998-6008.
[27] Zaid, G., Heuser, A., Guilley, S., & Rioul, O. (2020) 'Methodology for Efficient CNN Architectures in Profiling Attacks', IACR Transactions on Cryptographic Hardware and Embedded Systems, 1, pp. 1-36