Deep Learning Based Cryptography

A major project report submitted in partial fulfilment of the requirement for the award of degree of

Bachelor of Technology

in

Computer Science & Engineering / Information Technology

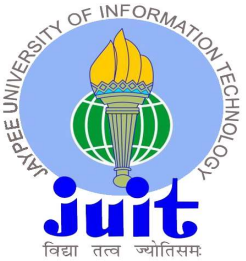
Submitted by

Kushagra Singhal (201501)

Aatish Sharma (201536)

Under the guidance & supervision of

Dr. Rakesh Kanji, Assistant professor (SG)



Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat, Solan - 173234 (India)

Candidate's Declaration

I hereby declare that the work presented in this report entitled **'Deep learning Based Cryptography'** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering / Information Technology** submitted in the Department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology, Waknaghat is an authentic record of my own work carried out over a period from August 2023 to May 2024 under the supervision of **Dr. Rakesh Kanji**(Assistant Professor , Department of Computer Science & Engineering and Information Technology).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Student Name: Kushagra Singhal

Roll No.: 201501

Student Name: Aatish Sharma

Roll No.: 201536

Certificate by Supervisor

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature with Date)

Supervisor Name: Dr. Rakesh Kanji

Designation: Assistant Professor (SG)

Department: Computer Science Engineering

Dated: 15/5/24

Acknowledgement

Firstly, we express our heartiest thanks and gratefulness to almighty God for His divine blessing makes it possible to complete the project work successfully.

We are really grateful and wish my profound indebtedness to Supervisor Dr. Rakesh Kanji, Assistant professor (SG), Department of CSE Jaypee University of Information Technology, Wakhnaghat. Deep Knowledge & keen interest of our supervisor in the field of "Computer Science" to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

We would like to express our heartiest gratitude to Dr. Rakesh Kanji, Department of CSE, for his kind help to finish our project.

We would also generously welcome each one of those individuals who have helped us straightforwardly or in a roundabout way in making this project a win. In this unique situation, we might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated our undertaking. Finally, we must acknowledge with due respect the constant support and patients of our parents.

Group No. - 120

Kushagra Singhal

201501

Aatish Sharma

201536

Abstract

In today's day and age of internet and connectivity the data is generate in rapid pace and it communicated securely over internet. To ensure the security and privacy of the general population, cryptography is employed to hide the data. The advancement in computer sciences has given rise to new ways of processing the data as well. With the availability of big data and higher computational power, the cryptographic techniques and algorithms are challenged with potential threat.

As the AI and ML technology becomes more capable on learning complex patterns in the data, the traditional methods that rely on computational complexity and provide security on the basis of that complexity will become obsolete to certain extent. The AI an ML based cryptography is one of the method that might become key role player in the future cyber security landscape. This technology can provide security not only on the basis of computational complexity but also on the basis of the architecture based security, where the security does not only depend on the complexity of the algorithm but also on the hiding of the way it produces the complexity.

This project aims to dig in to this idea of implementing cryptography using deep learning techniques and show casing the potential usefulness of the AI, ML technology in the field cybersecurity to various levels of cryptography that are required in the current world of data.

TABLE OF CONTENT

Content **Page No**.

Chapter-5 Results and Evaluation

Chapter-6 Conclusions and Future Scope

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1.1 GENERAL INTRODUCTION

The rapid advancements, in data security have coincided with the development of learning techniques leading to the emergence of methods, for strengthening cryptographic systems. A project called "learning based Cryptography" explores the combination of learning and cryptography to enhance data protection. Cryptography forms the basis of communication by encoding information to prevent access and tampering. In the past encryption algorithms have relied on formulas for encryption and decryption.

Deep learning techniques encompass a range of methods that harness the power of networks to analyze and adapt to patterns, in data. This task focuses on learning the utilization of neural networks in cryptography. Neural networks, renowned for their efficacy in image recognition and feature extraction have shown promise in enhancing encryption methods by learning patterns, within data.

The purpose of these studies is to investigate neuro cryptography, which's a subfield of networks that involves communication and decryption. The experimental setup includes a generator and a controller working together to enhance the security of the system through gameplay. This research aims to explore how adversarial neuro cryptography can strengthen data security measures and develop defenses, against attacks.

The core focus of this project is to enhance the efficiency and protection of computer systems by utilizing both cryptographic principles and the advancements in deep learning. By integrating deep learning into cryptography, we aim to combat the constantly evolving cyber threats and pave the path towards a more secure digital world.

1.2 PROBLEM STATEMENT

The main objectives is to study the impact of neural cryptographic structures for attacks, particularly those employing adversarial models, to sophisticated attacks by means of figuring out and mitigating vulnerabilities, this research will help develop greater and dependable cryptographic structures which might be higher included against malicious attacks.

Evaluating the overall and robustness of deep-learning based cryptographic fashions in real-global conditions poses a primary venture. The mission will consciousness on growing the power of encryption algorithms for distinct varieties of data to ensure the effectiveness of diverse applications and deployment environments.

1.3 OBJECTIVES

The primary goal of the "Deep-learning based Cryptography" project is to discover and use the assets of deep learning techniques, in particular neural networks and adverserial neural cryptography, to improve the safety and operation of cryptographic systems. Cryptography, the pillar of communications safety, has continually trusted mathematical algorithms for encryption and decryption. The assignment pursuits to overcome traditional encryption methods with the aid of the usage of the adaptivity of neural networks and mastering capabilities of deep neural networks in context of cryptography.

The following objectives are intended to achieve through this project:

- To develop a deep learning based cryptographic system.
- To implement and explore the use deep learning based cryptography.
- To improve on the developed systems.

1.4 SIGNIFICANCE AND MOTIVATION

Cryptography" project integrates Neural networks and generative adversarial networks (GAN) into the cryptography procedure. The purpose is to increase data protection by using the adaptive skills of deep mastering to derive higher insights. The outcomes of this observe are anticipated to convert data protection and boom the and resilience of the encryption which will lead to prevention of cyber threats.

Significance of Deep-Learning based Cryptography:

The importance of the "Deep learning-based Cryptography" project lies in its ability to convert the process of cryptography by combining deep learning, neural networks and generative adversarial networks (GAN), in addition to conventional encryption. As cyber threats emerge as more usual, the want for security measures and resilience turns into crucial. via integrating neural networks into the encryption process, we intention to improve the reactivity of the encryption manner towards emerging threats and provide powerful protection against block attacks. The results of this looks like to have huge implications for protecting touchy records in various fields along with finance, health, and communications, and ensuring the integrity and confidentiality of data at a time while traditional encryption techniques are vulnerable.

Motivation for Deep-Learning based Cryptography:

The incentive at the back of the "Deep-learning based Cryptography" primarily based on Encryption and Decryption comes from the urgency of addressing the restrictions of traditional encryption-decryption techniques in the face of evolving cyber threats. The traditional system, while operating well, will have difficulty quickly adapting to correct thoughts received from the enemy. Integrating Neural networks and GANs into cryptographic systems represents a thrilling area as it uses the strength of deep gaining knowledge to self-examine and reply to complex styles in cryptographic information.

Motivation is rooted inside the ability to create extra safety and stronger defense mechanisms to live ahead of attack. By exploring the intersection of deep-learning and cryptography, we purpose to supply new solutions to cope with information safety demanding situations and push the bounds of what is possible to guard sensitive statistics in the form of connected digital surroundings.

1.5 ORGANISATION

We have explained the fundamental principles of Deep-learning based cryptography in this project report, as well as how encryption-decryption, a crucial element of cryptography operates. Additionally, The improvements done over time is mentioned in a manner that makes it coherent with the subject of the project.

The fundamental concepts of Cryptography and deep learning is covered in chapter one. Two. The literature review, is divided into two sections explaining existing studies and gaps in those studies. Section three, or the system development portion, is divided into two sections covering design and implementation. The performance analysis and comparisons are presented in Section 4. The conclusion/final solution will be presented in Section 6.

CHAPTER 2: LITERATURE SURVEY

2.1 OVERVIEW OF RELEVANT LITERATURE

Author *Kinzel W.* [9] published his paper about the use of neural networks for the purpose of cryptography. The author gave a basic road map on how one can use artificial neural networks and how real world data can be transformed to make them fit for the artificial neural networks.

Author *Goodfellow I.*[1] in his paper introduced GAN model in the domain of deep learning.This paper was revolutionary and opened various new fields of study. Adverserial neural cryptography is one of those fields that arose from the same study done by the author.

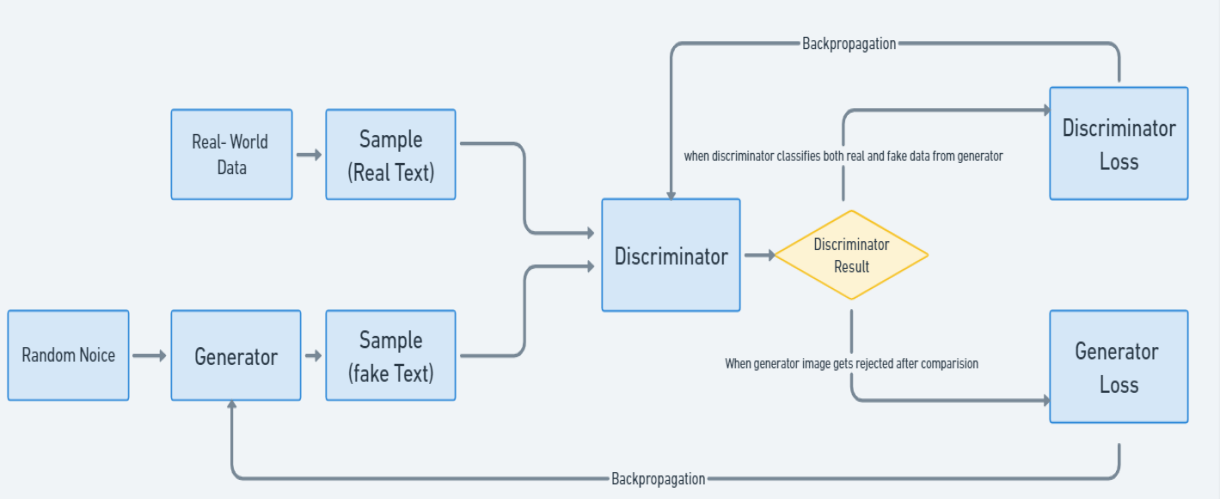


Fig 2.1: Generative Adverserial Network overview.

The authors *Meraouche et al.* [4] of this paper reviewed all the current techniques of neural network based cryptography and compiled a survey of those techniques. The author focuses on pushing the idea of Neural network based cryptography as a light weight, fast and efficient approach that can be useful (suitable modifications and enhancements) in post quantum security measures. The techniques reviewed in the papers were divide into two major eras post 2016,pre 2016. They highlight methodologies ,pitfalls and advantages of each technique they reviewed.They highlighted some major work that impacted this paper such as:

- The GAN inspired cryptographic model[10] that is trained with the adversaly network presence.
- Stenography[11][12][13] models that were also based on [10] showcased the capability of this concept.
- [14][15] Used the GAN approach introduced by *Abadi M et al. [10]* and used multiple adversaries to improve on his work further.

The use of GAN by *Abadi M et al. [10]* demonstrated the applicability of neural networks by encrypting the data in such a way that the adversary network is not able to predict it. They train the model with different level of key availability to the adversary. However, They do not share there dataset on which they trained the model. This work inspired the work of many others and can be seen as the turning point for the future of neural cryptography. They introduced the very idea of encrypting data without using any specific algorithm.

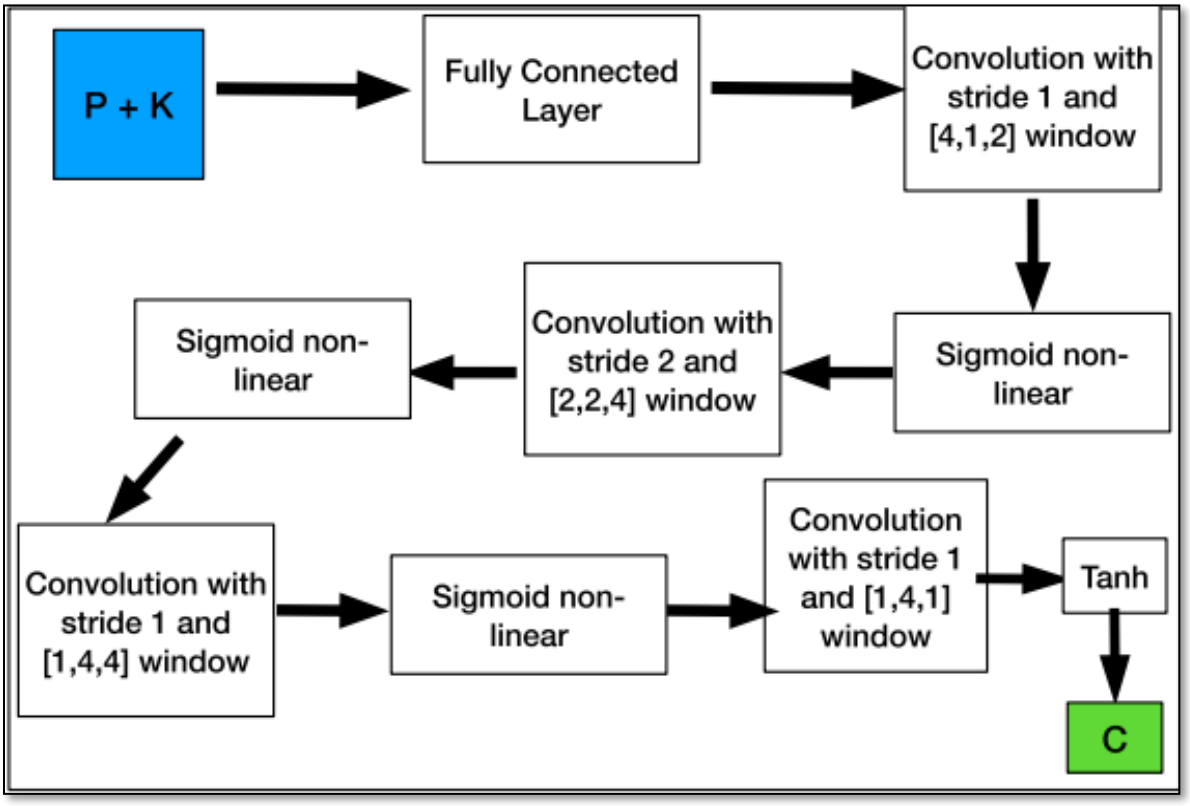


Fig 2.2: Data flow for encryption[10].

The paper also introduced the approach for training as well as decryption.

Deep learning has been a prominent topic of research in the medical field. However ,Its potential in securing data has not been explored to such extent. The use of deep learning for IOT security has also been realised in various fields one such application is demonstrated by *Ding Y. et al.* [8].They have used deep learning to secure Internet of Medical Things(IoMT) by encrypting the data images using cycle-GAN. They train there model on x-ray imagery[16] and claim that the security and efficiency is sufficiently high. They introduce the potential threats on this method of encryption. Those are:

- Hidden parameter leakages.
- Architecture leakages.
- Both Hidden parameters and architecture leakages.

These threats can be generalized to the other models as well.

Homomorphic encryptions are type of encryptions that are not needed to be decrypted in order to process the data and can be used to secure the confidential data even from the processing third party, if the processing of data is with machine learning, artificial intelligence or deep learning then they can be trained using encrypted data and the operations can be done directly ,The output to such operation is also encrypted that does not violate the privacy of the user. *Lee J. W. et al* [2] propose a similar approach using deep neural networks.

Author *Kalsi S. et al* [5] used [17] genetic algorithm along with [18] NW algorithm to generate key for the for encrypting data. They propose a solution to use AI and DNA sequences to make keys for encryption purposes. The author demonstrates the applicability of the technique in the paper. However, The paper lacks tests of proposed solution against established benchmarks in cyber security field.

The approaches that are being mentioned in the literature are still new and are being researched. In post quantum world this approach to cryptography can be used to secure the communications.

S. No.	Paper Title [Cite]	Journal/ Conference (Year)	Tools/ Techniques/ Dataset	Results	Limitation
1.	NEURAL CRYPTOGRAPHY[9]	International Conference on Neural Information Processing(2020)	PyTorch, TensorFlow, Keras libraries. NIST cryptographic benchmark datasets.	Paper propose a new approach to cryptography using ANN that has potential to be more secure, efficient than traditional cryptography	This paper does not include any specific dataset and is an introductory approach to the technique.
2.	CRYPTOGRAPHY BASED ON NEURAL NETWORK[3]	European Conference on Modelling and Simulation(2012)	PyTorch, TensorFlow, Keras, NIST benchmark datasets.	The paper proposes a new approach to cryptography using neural networks that has the potential to be more secure than traditional cryptography	This approach is still under development and needs to be further evaluated in real-world settings to address potential vulnerabilities
3.	Use of Cryptography in Cloud Computing[7]	IEEE International Conference on Control System, Computing and Engineering(2013)	OpenSSL, Botan, libsodium,cloud computing platform, NIST cryptographic benchmark datasets.	Highlighted the importance of cryptography for data protection, and identified promising	Survey is dated and does not cover all recent advances in cryptography for cloud computing

				new areas of research.	
4.	Generative Adversarial Networks[1]	Communications of the ACM (2014)	Generative Adverserial Networks, MNIST dataset, CIFAR-10 dataset.	Generative adversarial networks (GANs) can generate high-quality, diverse, and realistic output	GAN training can be unstable, difficult to interpret.
5.	LEARNING TO PROTECT COMMUNICATION SWITH ADVERSARIAL NEURAL CRYPTOGRAPHY[10]	arXiv preprint(2016)	PyTorch, TensorFlow, Keras, NIST benchmark datasets, OpenSSL, Botan, libsodium libraries.	Authors develop a novel adversarial neural cryptography framework to protect communications from eavesdropping	Adversarial Neural Cryptography (ANC) can be computationally expensive to train and deploy.
6.	DNA Cryptography and Deep Learning using Genetic Algorithm with NW algorithm for Key Generation[5]	Journal of Medical Systems(2018)	PyTorch, TensorFlow, Keras, DEAP, PyGAD, Biopython, NumPy, SciPy libraries. GenBank and Ensembl datasets.	Proposes a new way to generate encryption keys using DNA and AI that is more secure and efficient than traditional methods	The field is still new and needs more testing, and deep learning models require large datasets to train.

7.	DLEDNet: A Deep Learning-based Image Encryption and Decryption Network for IoMT[8]	IEEE Internet of Things Journal(2020)	PyTorch, TensorFlow, Keras, OpenCV, NumPy, SciPy libraries. cloud computing platform.	Deep learning-based image encryption and decryption network that is accurate, efficient, and resistant to attacks for the IoMT.	The network requires a large dataset to train and its security against adversarial attacks needs to be further evaluated
8.	Neural Networks-Based Cryptography: A Survey[4]	IEEE Access(2021)	PyTorch, TensorFlow, KerasNIST benchmark dataset,OpenSSL,Botan, libsodium library.	The paper analysis different neural cryptography approaches.	This paper only gives overview of the techniques.
9.	Privacy-Preserving Machine Learning With Fully Homomorphic Encryption for Deep Neural Network[2]	IEEE Access(2022)	Fully homomorphic encryption (FHE), CIFAR-10 dataset.	The paper shows the feasibility of training DL models on encrypted data, preserving the privacy of users while achieving accuracy in unencrypted models.	This approach is computationally expensive and under development.

10.	Synchronization of Tree Parity Machines using non-binary input vectors[6]	IEEE Transactions (2022)	PyTorch, TensorFlow, Keras, OMNeT++ libraries.	A new approach to synchronize TPM using non-binary input vectors that reduces sync time, improves accuracy, and is feasible for real-world application	The new approach requires more complex implementations and further research is needed to optimize its performance and security

2.2 Key Gaps in the Literature

These papers were good introductory approaches and gave a very good overview to the concepts that can be used. However, they do not talk about various approaches in much details. The literature does not define any one standard approach to mark accuracy and efficiency of the solutions proposed.

CHAPTER 3: SYSTEM DEVELOPMENT

3.1 REQUIREMENTS AND ANALYSIS

The scope of this project is to study deep learning based approaches to perform cryptography and find the appropriate use cases for the developed solutions. The end users for this project are every day normal people who browse the internet and want to secure their communication from potential threats to their privacy.

3.1.1 Functional Requirements

Functional Requirements are the requirements that are necessary and should be of higher priority than other requirements. The scope of a project plays an important role in the selection of functional requirements. With the current project scope following are the functional requirements :-

1. The approach taken to implement the project should be fairly fast and accurate.
2. It should be used/usable in real world applications.
3. The neural architecture used to implement should be appropriate.
4. The randomness and entropy of developed system should be fairly high.
5. The developed system should be evolution friendly i.e It should be able to adapt to the demand of the organization.
6. Computational efficiency should be fairly high.

3.1.2 Non Functional Requirements

Non-functional requirements are the requirements that have less priority in the project scope and do not affect the overall development goals of the project directly. However, they are important part of Software development Lifecycle. The non-functional requirements for the project are following:-

1. The system should meet on standards and bench marks such as encryption speed and throughput.
2. The system should be able to handle large amount of data.
3. The system should be simple to understand and use for end user.

4. The system should be resistant towards invalid inputs and adversarial attempts.
5. The system should meet regulatory compliance and laws for data privacy and information security.

3.1.3 Technical Requirements

These are the requirements that are necessary for the development of the project as the project is built with the help of these requirements. These requirements may change if the project require additional resources. These are the technical requirements for the project:-

1. Security libraries-Cryptography
2. Computer System with minimum 8 GB RAM, at least 100 GB storage and a dedicated graphics card.
3. Amazon Web Service.
4. Integrated Development Environment (IDE) PyCharm.
5. Neural network framework liabraries -Tensor flow keras.
6. Standard development libraries.

3.1.4 Priority chart

Criteria	Priority
Implement a fast and accurate approach.	High
Ensure the usability in real-world applications.	High
Use appropriate neural architecture.	High
Maintain high randomness and entropy.	High
Develop evolution-friendly system.	Medium
Achieve computational efficiency.	Medium
Scalibility of the model.	Low

3.2 PROJECT DESIGN AND ARCHITECTURE

3.2.1 Flow Chart

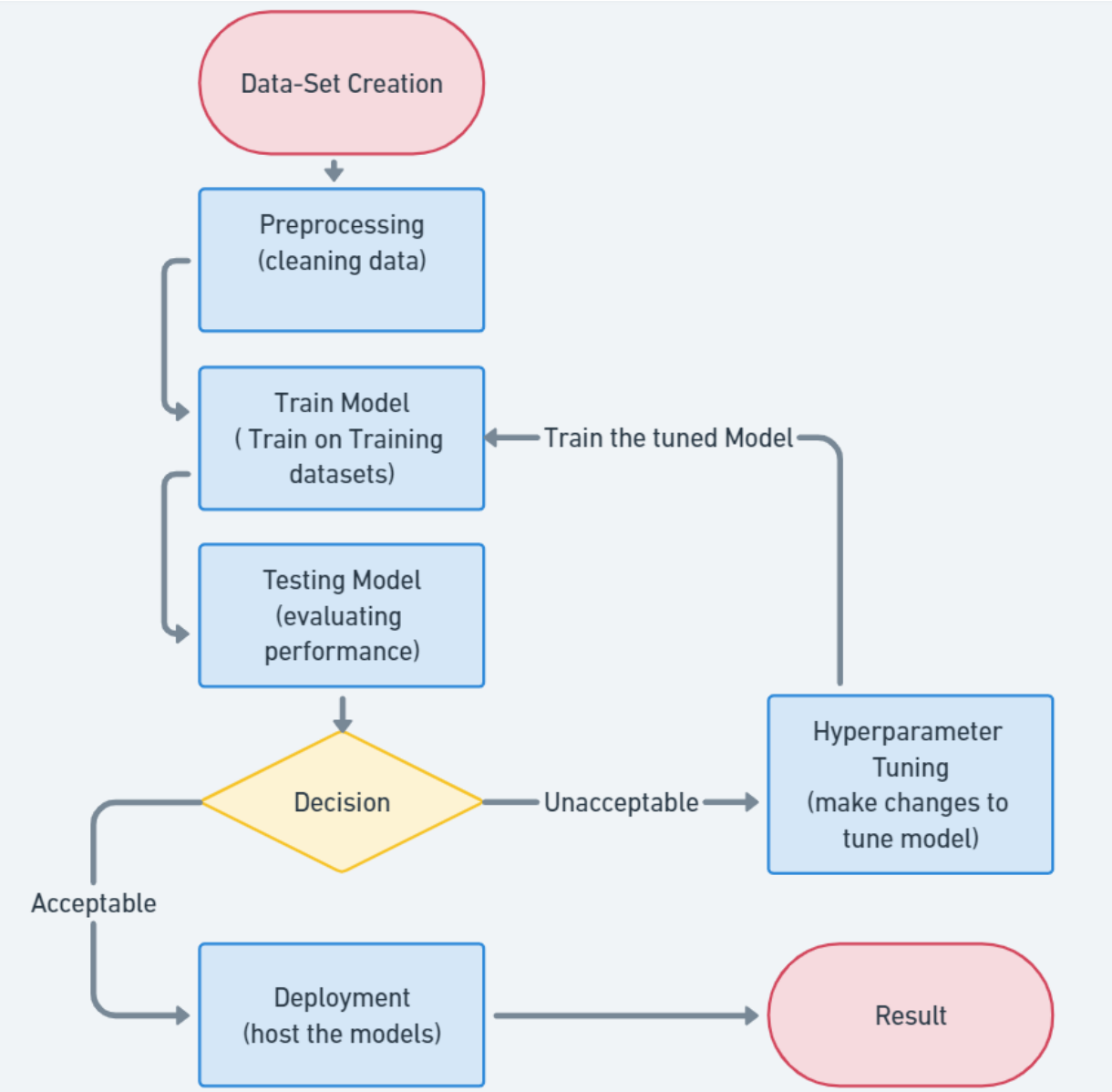


Fig 3.1: Project flow chart

3.2.2 Model Dataflow Diagram

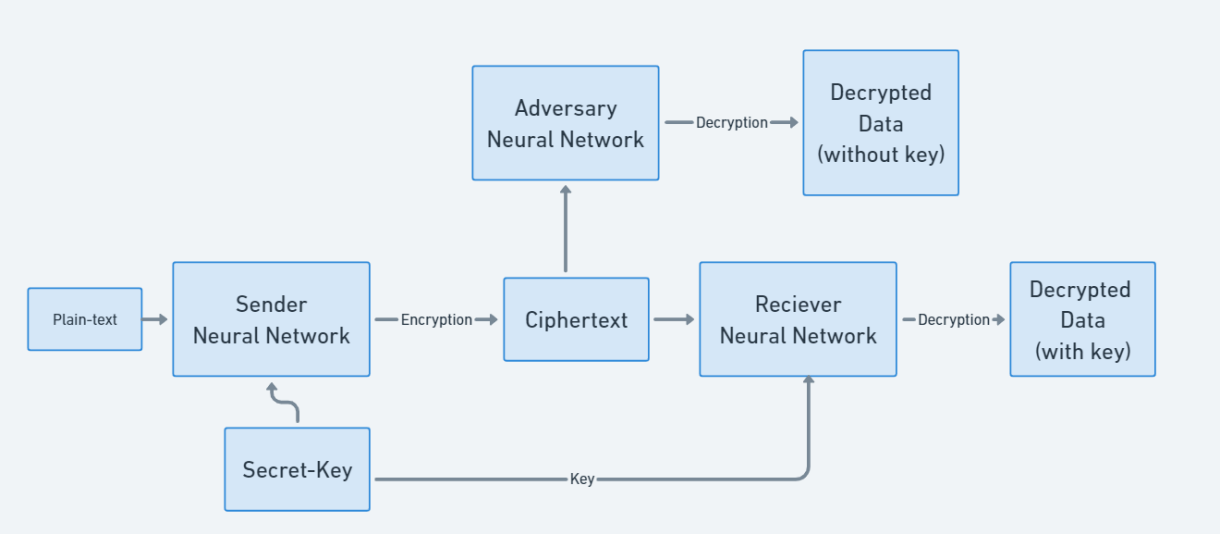


Fig 3.2: Model data flow diagram

Each of these networks have specific architecture. These architecture have been explained in the following:

- The Alice network in Fig: 3.3 is the sender network. Which takes plaintext and key and gives ciphertext as output.
- The Bob network in Fig: 3.4 is the reciver network. Which takes ciphertext and key and gives plaintext as output.
- The Bob network in Fig: 3.5 is the adversary network. Which takes ciphertext tries gives plaintext as output.

The loss of Bob network and Eve network is fed to the Alice network in backpropogation. The Alice network then tries to maximise the Eve loss while reducing the loss of Bob network.

These networks have the specific architectures described in following Figures.

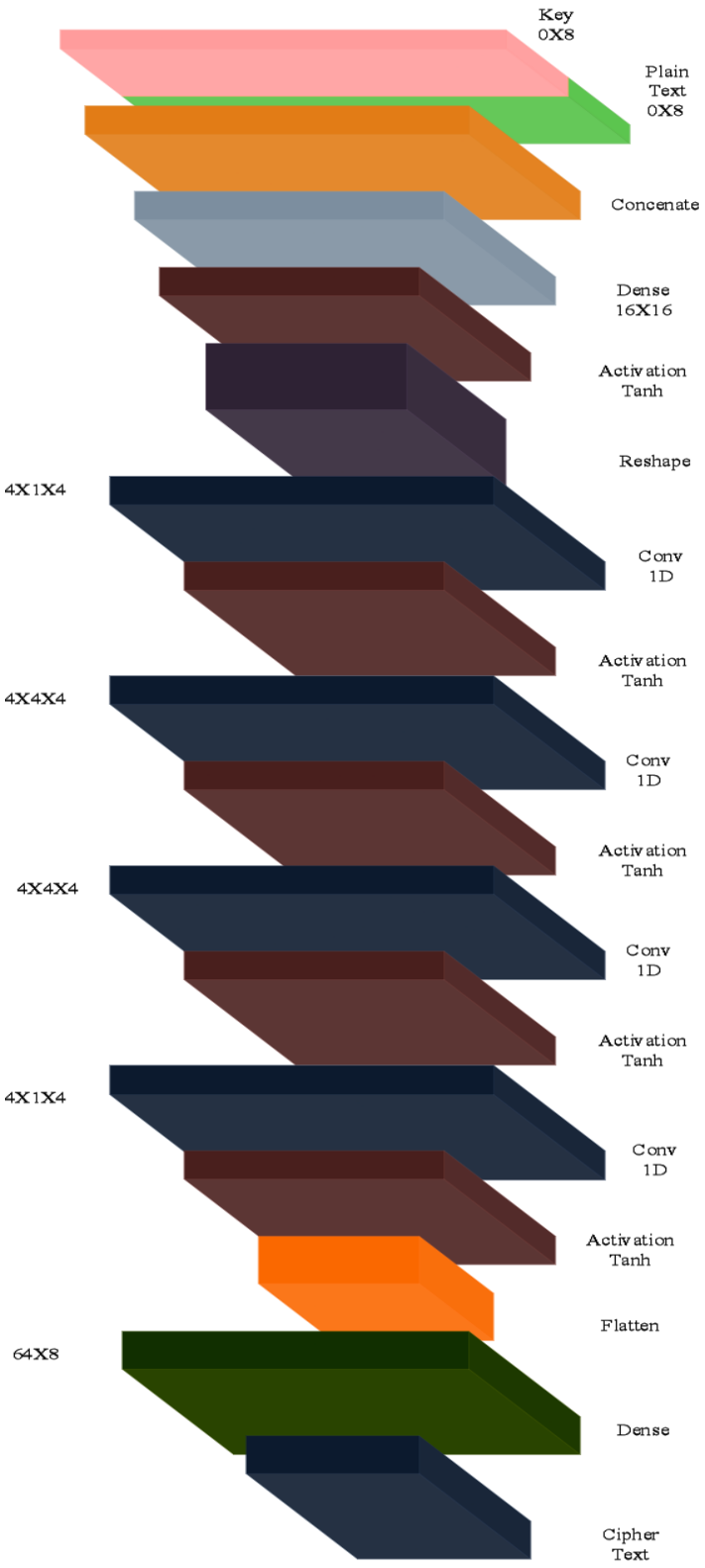


Fig 3.3: Alice architecture(Sender Network)

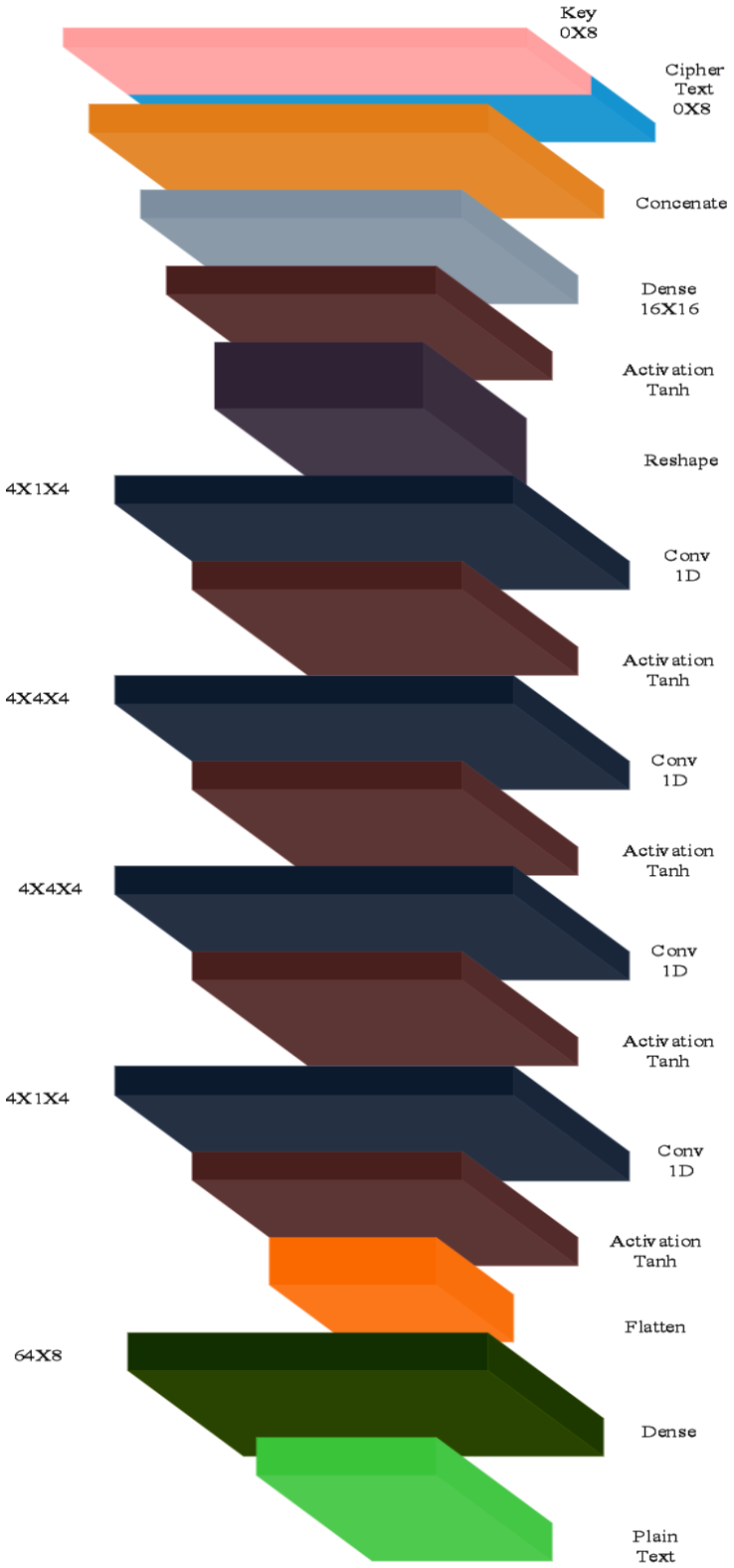


Fig 3.4: Bob architecture (Receiver Network)

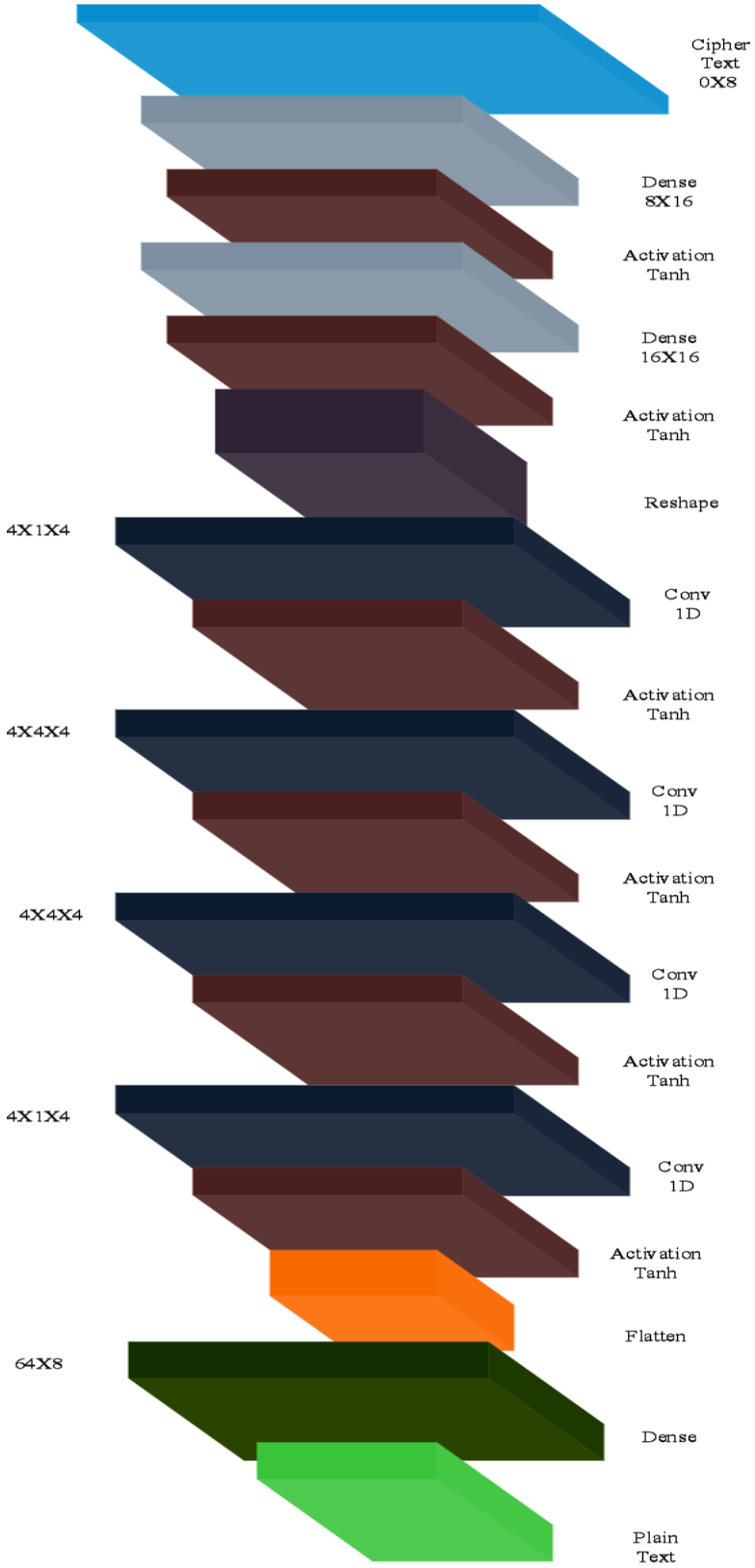


Fig 3.5: Eve architecture(Adversary Network)

3.2.3 Flow chart

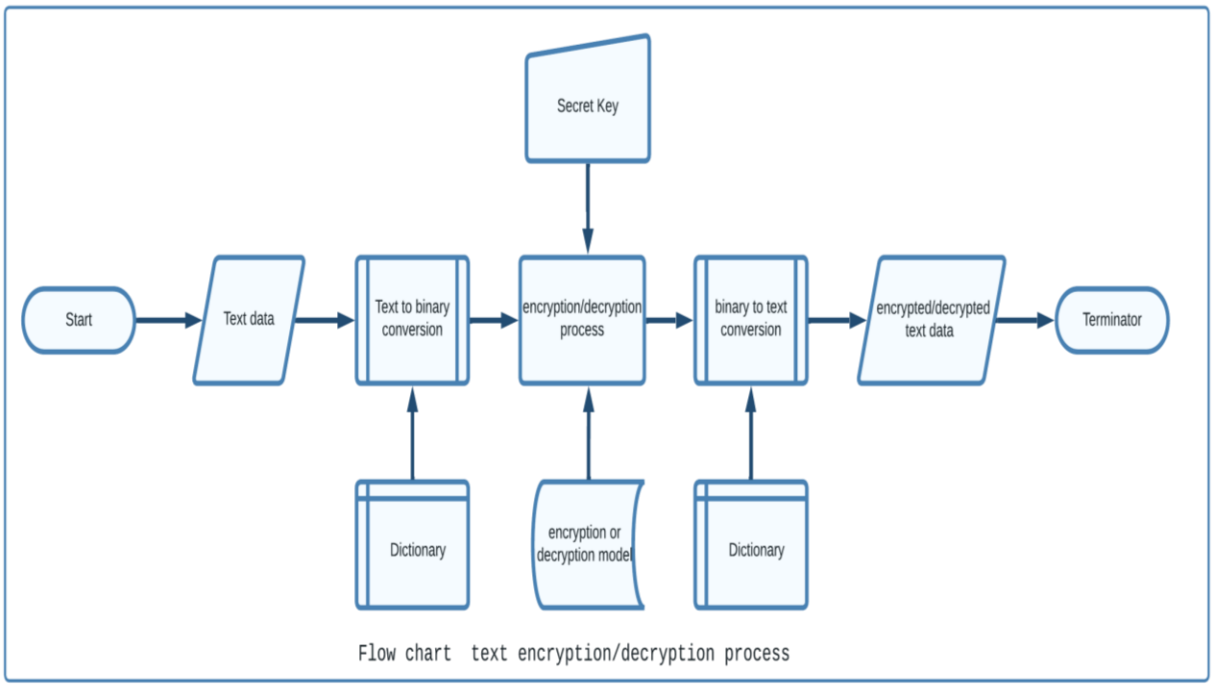


Fig 3.6: Flow chart encryption decryption process

3.2.3 Sequence Diagram

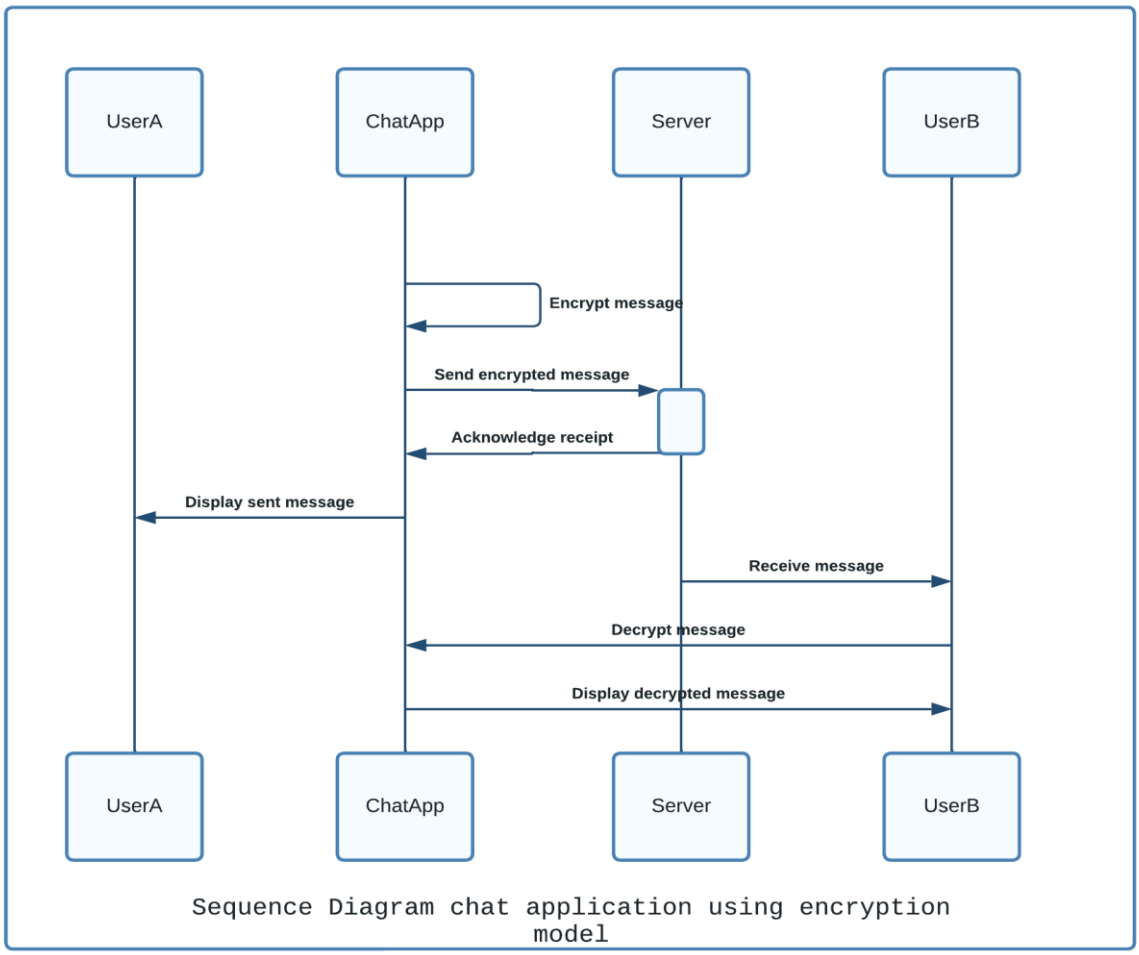


Fig 3.7: Sequence Diagram chat application using encryption model

3.3 DATA PREPARATION

Eight bit encryption model

The data that is being used has only two features each being 8 bits binary input. First feature is the plain text which is a 8 bit random binary and the key which is also 8 bit binary. The cipher text is decided upon the successful computation of loss function. The data set has 65,536 entries.

Sixteen bit encryption model

The data that is being used has only two features each being 16 bits binary input. First feature is the plain text which is a 16 bit random binary and the key which is also 16 bit binary. The cipher text is decided upon the successful computation of loss function. The data set has 50,000 entries generated with the help of python script for keeping the model practical in terms of training time.

3.4 IMPLEMENTATION

All the models that have been trained are of similar architecture,differing in the input size and number of layers.

3.4.1 Code Snippets

```
model_name = 'ANC'

#the standard parameters: message, key, and ciphertext bit lengths
m_bits = 8
k_bits = 8
c_bits = 8
pad = 'same'

# Compute the size of the message space
m_train = 2**(m_bits + k_bits)

alice_file = 'models/crypto/' + model_name + '-alice'
bob_file = 'models/crypto/' + model_name + '-bob'
eve_file = 'models/crypto/' + model_name + '-eve'
```

Fig 3.8: Training parameters for the models

```
##### Alice network #####
#
alice_input0 = Input(shape=(m_bits,)) #message
alice_input1 = Input(shape=(k_bits,)) #key
alice_input = concatenate([alice_input0, alice_input1], axis=1)

alice_dense1 = Dense(units=(m_bits + k_bits))(alice_input)
alice_dense1a = Activation('tanh')(alice_dense1)

alice_reshape = Reshape((m_bits + k_bits, 1,))(alice_dense1a)

alice_conv1 = Conv1D(filters=4, kernel_size=kersize, strides=1, padding=pad)(alice_reshape)
alice_conv1a = Activation('tanh')(alice_conv1)
alice_conv2 = Conv1D(filters=4, kernel_size=kersize, strides=1, padding=pad)(alice_conv1a)
alice_conv2a = Activation('tanh')(alice_conv2)
alice_conv3 = Conv1D(filters=4, kernel_size=kersize, strides=1, padding=pad)(alice_conv2a)
alice_conv3a = Activation('tanh')(alice_conv3)
alice_conv4 = Conv1D(filters=4, kernel_size=kersize, strides=1, padding=pad)(alice_conv3a)
alice_conv4a = Activation('tanh')(alice_conv4)

alice_flat = Flatten()(alice_conv4a)
alice_output = Dense(units=c_bits, activation='tanh')(alice_flat) #ciphertext

alice = Model([alice_input0, alice_input1], alice_output, name='alice')
alice.summary()
```

Fig 3.9: Alice network implementation

```
##### Bob network #####
#
bob_input0 = Input(shape=(c_bits,)) #ciphertext
bob_input1 = Input(shape=(k_bits,)) #key
bob_input = concatenate([bob_input0, bob_input1], axis=1)

bob_dense1 = Dense(units=(c_bits + k_bits))(bob_input)
bob_dense1a = Activation('tanh')(bob_dense1)

bob_reshape = Reshape((c_bits + k_bits, 1,))(bob_dense1a)

bob_conv1 = Conv1D(filters=4, kernel_size=kersize, strides=1, padding=pad)(bob_reshape)
bob_conv1a = Activation('tanh')(bob_conv1)
bob_conv2 = Conv1D(filters=4, kernel_size=kersize, strides=1, padding=pad)(bob_conv1a)
bob_conv2a = Activation('tanh')(bob_conv2)
bob_conv3 = Conv1D(filters=4, kernel_size=kersize, strides=1, padding=pad)(bob_conv2a)
bob_conv3a = Activation('tanh')(bob_conv3)
bob_conv4 = Conv1D(filters=4, kernel_size=kersize, strides=1, padding=pad)(bob_conv3a)
bob_conv4a = Activation('tanh')(bob_conv4)

bob_flat = Flatten()(bob_conv4a)
bob_output = Dense(units=m_bits, activation='sigmoid')(bob_flat) #decrypted message

bob = Model([bob_input0, bob_input1], bob_output, name='bob')
bob.summary()
```

Fig 3.10: Bob network implementation

```
# Eve network
eve_input = Input(shape=(c_bits,)) #ciphertext only

eve_dense1 = Dense(units=(c_bits + k_bits))(eve_input)
eve_dense1a = Activation('tanh')(eve_dense1)
eve_dense2 = Dense(units=(m_bits + k_bits))(eve_dense1a)
eve_dense2a = Activation('tanh')(eve_dense2)

eve_reshape = Reshape((m_bits + k_bits, 1,))(eve_dense2a)

eve_conv1 = Conv1D(filters=4, kernel_size=kersize, strides=1, padding=pad)(eve_reshape)
eve_conv1a = Activation('tanh')(eve_conv1)
eve_conv2 = Conv1D(filters=4, kernel_size=kersize, strides=1, padding=pad)(eve_conv1a)
eve_conv2a = Activation('tanh')(eve_conv2)
eve_conv3 = Conv1D(filters=4, kernel_size=kersize, strides=1, padding=pad)(eve_conv2a)
eve_conv3a = Activation('tanh')(eve_conv3)
eve_conv4 = Conv1D(filters=4, kernel_size=kersize, strides=1, padding=pad)(eve_conv3a)
eve_conv4a = Activation('tanh')(eve_conv4)

eve_flat = Flatten()(eve_conv4a)
eve_output = Dense(units=m_bits, activation='sigmoid')(eve_flat) #code break attempt

eve = Model(eve_input, eve_output, name='eve')
eve.summary()
```

Fig 3.11: Eve network implementation

```
alice.compile(loss='mse', optimizer='sgd')
bob.compile(loss='mse', optimizer='sgd')
eve.compile(loss='mse', optimizer='sgd')
```

Fig 3.12: Compile parameters of the models

```
aliceout = alice([alice_input0, alice_input1])
bobout = bob( [aliceout, binput1] )# bob sees ciphertext AND key
eveout = eve( aliceout )# eve doesn't see the key, only the cipher


eveloss = K.mean(  K.sum(K.abs(alice_input0 - eveout), axis=-1)  )


#
bobloss = K.mean(  K.sum(K.abs(alice_input0 - bobout), axis=-1)  )
abeloss = bobloss + K.square(m_bits/2 - eveloss)/( (m_bits//2)**2 )

# Optimizer and compilation

abeoptim = Adam()
eveoptim = Adam()


# Build and compile the model, used for training Alice-Bob networks
#
abemodel = Model([alice_input0, alice_input1, binput1], bobout, name='abemodel')
abemodel.add_loss(abeloss)
abemodel.compile(optimizer=abeoptim)


# Build and compile the EVE model, used for training Eve net (with Alice frozen)
#
alice.trainable = False
evemodel = Model([ainput0, ainput1], eveout, name='evemodel')
evemodel.add_loss(eveloss)
evemodel.compile(optimizer=eveoptim)
```

Fig 3.13: Implementation Loss calculation and optimisation

```
print("Training for", n_epochs, "epochs with", n_batches, "batches of size", batch_size)

while epoch < n_epochs:
    abelosses0 = [] #epoch-bound losses for text display during training
    boblosses0 = []
    evelosses0 = []
    for iteration in range(n_batches):

        # Train the A-B+E network
        alice.trainable = True
        for cycle in range(abecycles):
            # Select a random batch of messages, and a random batch of keys
            #
            m_batch = np.random.randint(0, 2, m_bits * batch_size).reshape(batch_size, m_bits)
            k_batch = np.random.randint(0, 2, k_bits * batch_size).reshape(batch_size, k_bits)
            loss = abemodel.train_on_batch([m_batch, k_batch, k_batch], None)

        abelosses0.append(loss)
        abelosses.append(loss)
        abeavg = np.mean(abelosses0)

        # Evaluate Bob's ability to decrypt a message
        m_enc = alice.predict([m_batch, k_batch])
        m_dec = bob.predict([m_enc, k_batch])
        loss = np.mean(  np.sum( np.abs(m_batch - m_dec), axis=-1)  )
        boblosses0.append(loss)
        boblosses.append(loss)
        bobavg = np.mean(boblosses0)
```

Fig 3.14: Implementation training process

```
        #
        alice.trainable = False
        for cycle in range(evecycles):
            m_batch = np.random.randint(0, 2, m_bits * batch_size).reshape(batch_size, m_bits)
            k_batch = np.random.randint(0, 2, k_bits * batch_size).reshape(batch_size, k_bits)
            loss = evemodel.train_on_batch([m_batch, k_batch], None)

        evelosses0.append(loss)
        evelosses.append(loss)
        eveavg = np.mean(evelosses0)

        if iteration % max(1, (n_batches // 100)) == 0:
            print("\rEpoch {:3}: {:3}% | abe: {:2.3f} | eve: {:2.3f} | bob: {:2.3f}".format(
                epoch, 100 * iteration // n_batches, abeavg, eveavg, bobavg), end="")
            sys.stdout.flush()

    print()
    epoch += 1

print('Training finished.')
```

Fig 3.15: Implementation training process

```
plt.figure(figsize=(7, 4))
plt.plot(abelosses[:steps], label='A-B', alpha=0.99)
plt.plot(evelosses[:steps], label='Eve', alpha=0.99)
plt.plot(boblosses[:steps], label='Bob', alpha=0.99)
plt.xlabel("Iterations", fontsize=13)
plt.ylabel("Loss", fontsize=13)
plt.legend(fontsize=13, loc='upper right')

plt.savefig("images/" + model_name + "-all.png", transparent=True)
plt.show()
```

Fig 3.16: Implementation plotting loss

```
n_examples = 10000

m_batch = np.random.randint(0, 2, m_bits * n_examples).reshape(n_examples, m_bits)
k_batch = np.random.randint(0, 2, m_bits * n_examples).reshape(n_examples, m_bits)

m_enc = alice.predict([m_batch, k_batch])

m_dec = (bob.predict([m_enc, k_batch]) > 0.5).astype(int)
m_att = (eve.predict(m_enc) > 0.5).astype(int)

bdiff = np.abs(m_batch - m_dec)
bsum = np.sum(bdiff, axis=-1)
ediff = np.abs(m_batch - m_att)
esum = np.sum(ediff, axis=-1)

print("Bob % correct: ", 100.0*np.sum(bsum == 0) / n_examples, '%')
print("Eve % correct: ", 100.0*np.sum(esum == 0) / n_examples, '%')
```

Fig 3.17: Implementation accuracy calculation

```
block_size = block_size_unpadded + block_padding
#Dictonary and corresponding binary
chrlist = [...]
binlist = [...]

def randombits(n):
    if n == 0:
        return ''
    decvalue = np.random.randint(0, 2**n)
    formatstring = '0' + str(n) + 'b'
    return format(decvalue, formatstring)

def encstr(message, block_padding=0):
    cipher = ""
    for c in message:
        binstr = binlist[chrlist.index(c)]
        binstrpadded = randombits(block_padding) + str(binstr)
        cipher = cipher + binstrpadded
    return cipher, len(message)

def decstr(cipher, n, block_padding=0):
    message = ""
    cipherlength = len(cipher)
    block_size = cipherlength // n
    for i in range(n):
        blockpadded = cipher[block_size*i : block_size*i + block_size]
        blockunpadded = blockpadded[block_padding:]
        character = chrlist[binlist.index(blockunpadded)]
        message = message + character
    return message
```

Fig 3.18: Implementation encryption and decryption functions

```
# Function for converting float to 32-bit binary string
def float_to_binary(value):
    binNum = bin(  ctypes.c_uint.from_buffer(ctypes.c_float(value)).value  )[2:]
    binstr = binNum.rjust(32,"0")
    return binstr

def binary_to_float(binstr):
    intvalue = int(binstr, 2)
    floatvalue = ctypes.c_float.from_buffer(ctypes.c_uint(intvalue))
    return floatvalue.value

# Convert a positive integer num into a bit vector of 'bits' size
def bitarray(num, bits):
    return np.array(list(np.binary_repr(num).zfill(bits))).astype(np.int8)
```

Fig 3.19: Implementation float to binary

```
key = np.array([[0,0,0,0,0,0,0,0]])
m = 'aatish'

m_bin, _ = encstr(m, block_padding=3)
m_bin_len = len(m_bin)
print(m_bin, m_bin_len)

ciphertext = ""
for i in range(m_bin_len // m_bits):...

#print(ciphertext, len(ciphertext)) # ciphertext in binary
print(decstr(ciphertext, n=(len(ciphertext)//8), block_padding=3)) #ciphertext as characters

ciphertext_len = len(ciphertext)
plaintextbin = ""
for i in range(ciphertext_len // (c_bits*32)):...

print(plaintextbin)

m_dec = ""
for i in range(len(plaintextbin) // m_bits):
    strbin = plaintextbin[m_bits*i : m_bits*i + m_bits]
    m_dec = m_dec + decstr(strbin, len(strbin)//m_bits, block_padding=3)

print(m_dec)
```

Fig 3.20: Implementation encryption and decryption

3.4.2 Algorithm

3.4.2.1 Algorithm generating encryption decryption model

1. Initialize the neural network models for Alice, Bob and Eve.
2. Generate random plaintext messages and keys.
3. Train Alice and Bob networks and Eve network.
4. Evaluate success of communication.
5. Plot and Analyze loss curves.
6. Repeat from step 3 to 5 until the epochs are completed
7. Test trained model for performance and security.
8. Assess model's ability to resist eavesdropping by Eve.
9. Save trained models for future use.

3.4.2.2 Algorithm encryption decryption process using model generated

1. Take the plaintext/ciphertext input as a parameter and select a key .
2. Iterate through each character in the plaintext/ciphertext.
3. Convert each character into its binary representation with the help of dictionary.
4. Concatenate all the binary representations together to form the binary representation of the entire plaintext.
5. Return the binary representation.
6. Take the binary data as a parameter.
7. Iterate through the binary data in chunks of 8 bits (1 byte) and apply the encryption/decryption in each chunk with a key of same size.
8. Concatenate all the binary representations together.
9. Convert each 8-bit encrypted chunks into corresponding characters as in step
10. Concatenate all the characters together to form the ciphertext/plaintext

3.4.3 Tools and Techniques

- Convolutional Neural Network is a deep-learning architecture which is used for data - processing by usings it's layers to automatically adapt and learn the features form the input data.
- Generative Adversarial Network is machine learning model where generator and discriminator are trained simultaneously to create and evaluate data.
- Kearas is a neural network API which provides user friendly interface for training and building of deep learning models.
- Tensorflow is a framework which provides an ecosystem of tools, libraries and esources for building and deploying machine learning models.
- Python is high level programming language which provides tools and tech such as Numpy, Pandas, Matplotlib, tkinter etc.
- Other tools for developing the web application.

3.5 KEY CHALLENGES

Realization of theoretical concepts in practice was the major key challenge of in the current implementation. However, this was the part of project scope and was fairly addressed in the review process. Other then that following are the significant challenges that we faced in this project.

3.5.1 Dataset generation

The problem with dataset generation was to select appropriate length of both key and plain text. This affected the overall capability of the models to process data directly.

To solve this problem, we use minimum amount of bits that are required to represent all the 256 characters of the ASCII table.

This also allowed us to reduce the complexity of the input that is required for generating the ciphertext using the sender network.

3.5.2 TESTING THE CAPABILITIES OF THE TRAINED MODELS

The models that were generated were trained to accept binary input and the output was also in binary format. This made it hard to realize the outputs in human understandable language.

To solve this problem, we developed functions to process the input and output in model understandable language and then back to human understandable language. Interesting feature of using this method is the customizability of the dictionary the we can use to make the encryptions more utilizable in different scenarios.

CHAPTER 4: TESTING

4.1 TESTING STRATEGY

The cryptographic techniques can be evaluated on the basis of entropy of cipher text and overall difference in frequency distribution of plaintext and cipher text.

The Neural Network approach to cryptography can be evaluate on the basis of reconstruction loss of the ciphertext in terms of alice and bob communication , eves attempt of reconstruction of cipher text without key. Currently we have evaluated the model for the reconstruction loss and accuracy of the model.

4.2 TEST CASES AND OUTCOMES

For testing accuracy of the model we tested it on 10000 different randomly generated bits of plaintext and keys . The accuracy came as followed:

Accuracy of Models over test dataset before increasing the number of layer

Model Name	**Accuracy**
Bob(receiver)	99.87%
Eve(Adversary)	1.95%

Accuracy of Models over test dataset after increasing the number of layers

Model Name	**Accuracy**
Bob(receiver)	99.99%
Eve(Adversary)	1.42%

CHAPTER 5: RESULTS AND EVALUATION

5.1 RESULTS

During the training of the models , while trying to predict the plaintext the eve's(adversary) loss was decreasing initially.However, This changes after a some iterations. The model moves towards increased adversarial loss and decreased accuracy loss . Based on this concept , the neural network arrangement converges to the maximum loss of adversary and minimum reciver loss.

The use of deep learning in the field of encryption is a viable alternate or enhancement over the existing techniques. This technique can be used to enhance the security posture of an organization by improving the security against potential AI threats that run cryptanalysis to get pattern in the encrypted data.

The scalability and customization of this technique that is features of the deep learning methods is one key benefit that can be exploited to further make the cracking of algorithm more difficult.

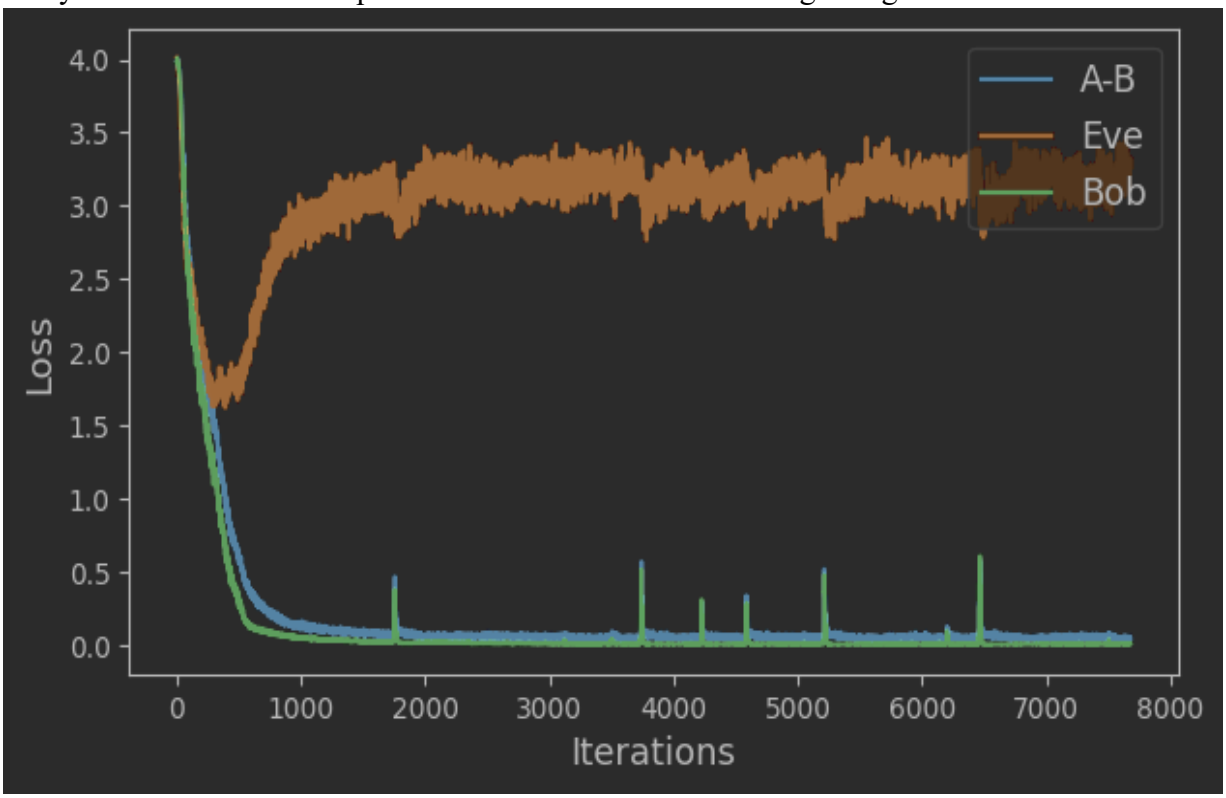


Fig 5.1: Loss of different models in training (8 bit model)

The accuracy of given model on 8 binary inputs is good as shown in the figure below. However, it is also possible to increase the cryptographic strength of the technique by combining it with traditional techniques such as using custom dictionaries.

```
313/313 [==============================] - 2s 5ms/step
313/313 [==============================] - 2s 5ms/step
313/313 [==============================] - 2s 5ms/step
Bob % correct:  99.87 %
Eve % correct:  1.95 %
```

Fig 5.2:Accuracy of decryption on test dataset(8 bit model)

On increasing number of layers the (added two addition layer to each model) the loss over the training is almost the same . However , The nature and accuracy changes to a certain extent. The change can be visualized in the following figures.

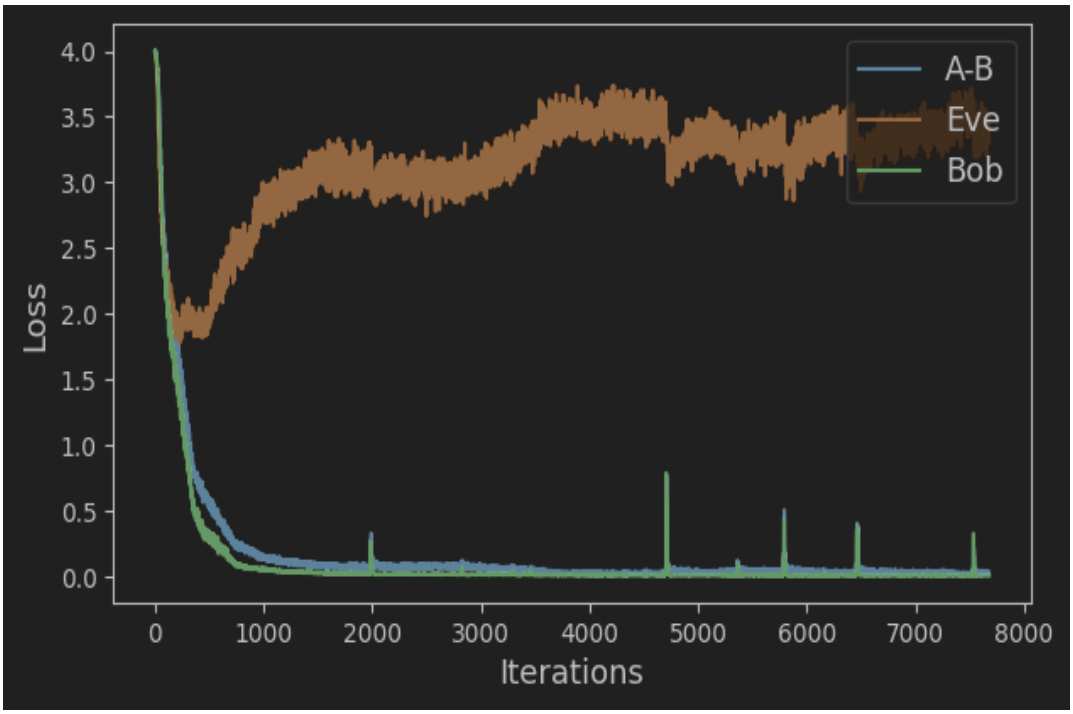


Fig 5.3 Loss of different models in training improved model (8 bit model)

```
313/313 [==============================] - 9s 10ms/step
313/313 [==============================] - 3s 5ms/step
313/313 [==============================] - 2s 5ms/step
Bob % correct:  99.99 %
Eve % correct:  1.42 %
```

Fig 5.4:Accuracy of decryption on test dataset of improved model(8 bit model)

The accuracy for eight bit model was satisfactory and could be used in practical application that is a web app that uses this technique to secure its communication. We experimented with the input size of the model further and gained the following results .

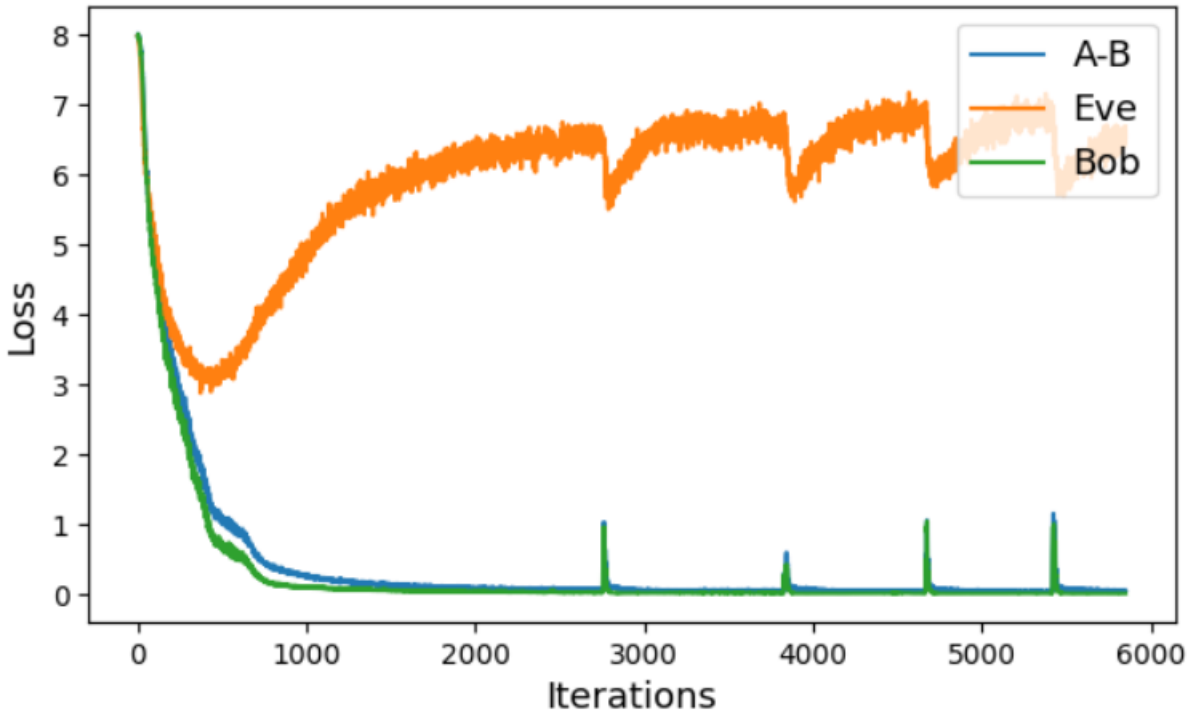


Fig 5.5: Loss of different models in training (16 bit model)

The results were promising the loss of the adversary mode doubled as compared to eight bit model. However, The model was not fit for the practical application in our case.Following are the reason for not using this model-

- The model dealt with sixteen bits which added a lot overhead to text based encryption.

- The model required extensive amount of training time and resources as compared to eight bit model.

Using the improved eight bit model we developed a chat application which provide a secure channel which uses the model developed by us as its encryption algorithm.

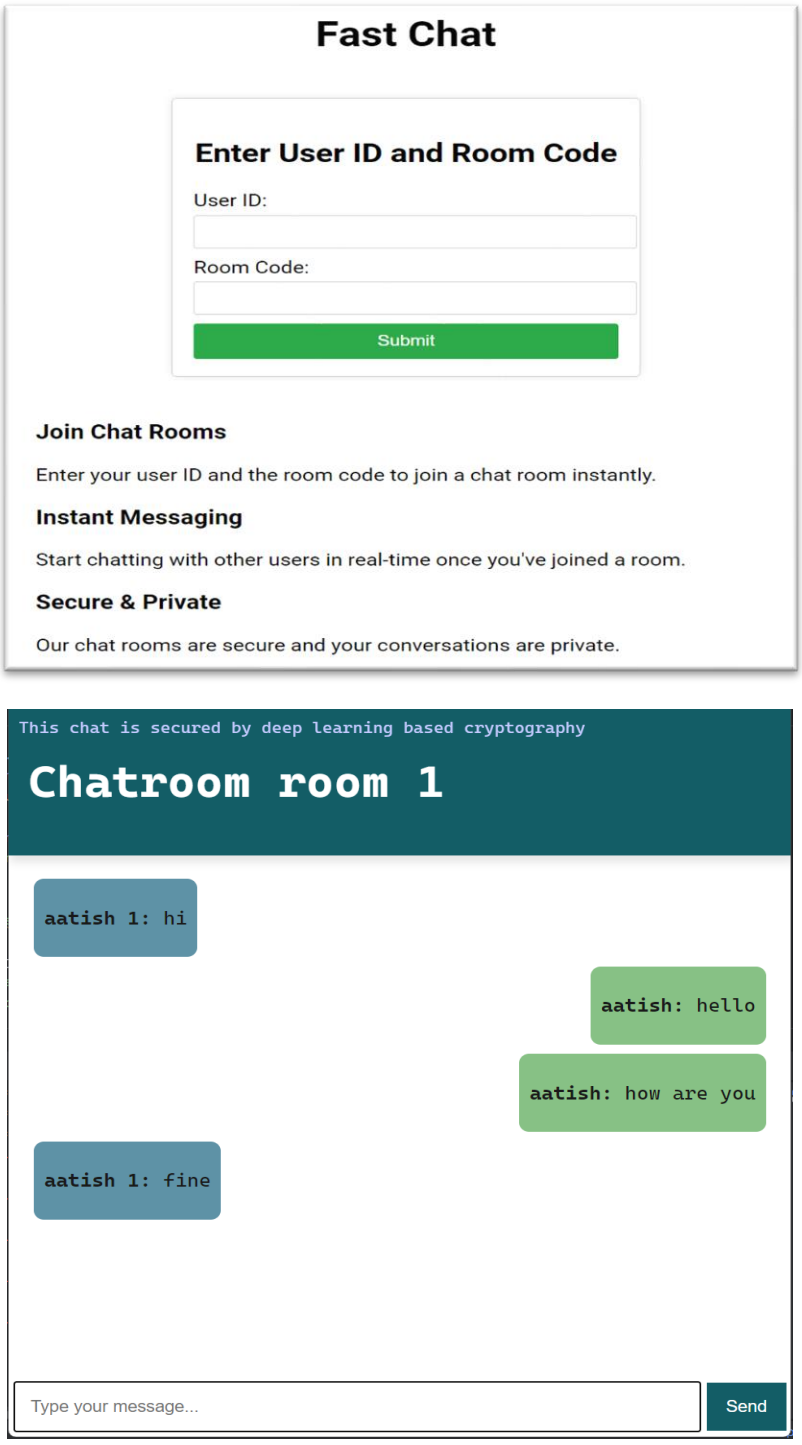


Fig 5.6: Chat application using the cryptographic model

CHAPTER 6: CONCLUSIONS & FUTURE SCOPE

6.1 Conclusion

The field of cybersecurity has seen prolific growth in the previous years and has been touted as one of the niches of the tech world that will bloom in the future by many industrial experts. In simple terms, cybersecurity can be defined as protecting one's electronic devices from viruses, malware and anything that can harm their integrity. The world we live in today has become one humongous haven of data, where millions of gigabytes are exchanged across the internet every single day. In a scenario like this, it is imperative that users can safely and privately share their data without the worry of their data integrity being compromised. This is where the role of cybersecurity comes into play.

A standard practice that has become a mainstay in the arsenal of cyber experts is Data encryption. In layman terms, data encryption can be compared to how a bank locker can be accessed; the customer has one key, the bank has the other key and only a combination of these keys in successive order can allow access to the locker. First, the data that the sender wants to share is converted into plain text and further encrypted into ciphertext. A key is generated and shared with both the sender and the receiver. The ciphertext is then received by the receiver and a key is asked by the system to authenticate the credibility of the receiver. Once it is established that the receiver is credible by verifying the key, the ciphertext is converted into plain text and further into the original data that the sender wanted to originally send.

We got to know that deep learning in cryptography provides a secure communication solution against artificial intelligence threats. Its strength lies in its applicability to scenarios requiring less complex encryption decryption, making it adaptable and evolution-ready. However, limitations include vulnerability to probabilistic attacks and the project's complexity, demanding extensive training and dataset generation efforts. The significant contribution lies in advancing the concept of cryptography through deep learning, emphasizing its potential to enhance security measures in

communication systems. This approach aligns with the evolving landscape of cybersecurity, offering a promising avenue for robust and adaptive cryptographic solutions.

6.2 FUTURE SCOPE

This project is a gave us key insights in the domain of cryptography and deep learning along with the process of how we develop a system from scratch and build an end product out of it. The project taught us the challenges and problems that arise in developing a system and we intend to use this new found knowledge to tackle those challanges in future projects as well.

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, ..., and Y. Bengio, "Generative adversarial networks," Communications of the ACM, vol. 63, no. 11, pp. 139-144, 2020.

[2] J. W. Lee, H. Kang, Y. Lee, W. Choi, J. Eom, M. Deryabin, ..., and J. S. No, "Privacy-preserving machine learning with fully homomorphic encryption for deep neural network," IEEE Access, vol. 10, pp. 30039-30054, 2022

[3] E. Volna, M. Kotyrba, V. Kocian, and M. Janosek, "Cryptography based on neural network," in Proceedings of the European Conference on Modelling and Simulation (ECMS), May 2012, pp. 386-391

[4] I. Meraouche, S. Dutta, H. Tan, and K. Sakurai, "Neural networks-based cryptography: A survey," IEEE Access, vol. 9, pp. 124727-124740, 2021.

[5] S. Kalsi, H. Kaur, and V. Chang, "DNA cryptography and deep learning using genetic algorithm with NW algorithm for key generation," Journal of Medical Systems, vol. 42, pp. 1-12, 2018

[6] M. Stypiński and M. Niemiec, "Synchronization of tree parity machines using nonbinary input vectors," IEEE Transactions on Neural Networks and Learning Systems, 2022.

[7] A. N. Jaber and M. F. B. Zolkipli, "Use of cryptography in cloud computing," in 2013 IEEE International Conference on Control System, Computing and Engineering, November 2013, pp. 179-184

[8] Y. Ding, G. Wu, D. Chen, N. Zhang, L. Gong, M. Cao, and Z. Qin, "DeepEDN: A deep-learning-based image encryption and decryption network for Internet of Medical Things," IEEE Internet of Things Journal, vol. 8, no. 3, pp. 1504-1518, 2020

[9] W. Kinzel and I. Kanter, "Neural cryptography," in Proceedings of the 9th International Conference on Neural Information Processing, 2002. ICONIP'02, vol. 3, November 2002, pp. 1351-1354, IEEE.

[10] M. Abadi and D. G. Andersen, ‘‘Learning to protect communications with adversarial neural cryptography,’’ CoRR, vol. abs/1610.06918, pp. 1–15, Oct. 2016.

[11] M. Yedroudj, F. Comby, and M. Chaumont, ‘‘Steganography using a 3-player game,’’ J. Vis. Commun. Image Represent., vol. 72, Oct. 2020, Art. no. 102910.

[12] J. Hayes and G. Danezis, ‘‘Generating steganographic images via adversarial training,’’ in Proc. Adv. Neural Inf. Process. Syst., vol. 30, 2017, pp. 1954–1963.

[13] Y. Ke, M. Zhang, J. Liu, and T. Su, ‘‘Generative steganography with Kerckhoffs’ principle based on generative adversarial networks,’’ CoRR, vol. abs/1711.04916, pp. 1–8, Nov. 2017.

[14] M. Coutinho, R. de Oliveira Albuquerque, F. Borges, L. G. Villalba, and T.-H. Kim, ‘‘Learning perfectly secure cryptography to protect communications with adversarial neural cryptography,’’ Sensors, vol. 18, no. 5, p. 1306, Apr. 2018.

[15] Z. Li, X. Yang, K. Shen, R. Zhu, and J. Jiang, ‘‘Information encryption communication system based on the adversarial networks foundation,’’ Neurocomputing, vol. 415, pp. 347–357, Nov. 2020. [Online].

[16] S. Jaeger, S. Candemir, S. Antani, et.al., “Two public chest X-ray datasets for computer-aided screening of pulmonary diseases,” Quantitative imaging in medicine and surgery, vol. 4, pp. 475-7, Dec. 2014.

[17]S. D. Immanuel and U. K. Chakraborty, "Genetic Algorithm: An Approach on Optimization," 2019 International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 2019, pp. 701-708, doi: 10.1109/ICCES45898.2019.9002372.

[18]H. Syed and A. K. Das, "Temporal Needleman-Wunsch," 2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA), Paris, France, 2015, pp. 1-9, doi: 10.1109/DSAA.2015.7344785.

crypt

ORIGINALITY REPORT

3%	3%	2%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	ebin.pub Internet Source	<1%
2	clok.uclan.ac.uk Internet Source	<1%
3	arxiv.org Internet Source	<1%
4	Submitted to National School of Business Management NSBM, Sri Lanka Student Paper	<1%
5	repositori.uji.es Internet Source	<1%
6	"Artificial Neural Networks and Machine Learning – ICANN 2018", Springer Science and Business Media LLC, 2018 Publication	<1%
7	Submitted to kitsw Student Paper	<1%
8	export.arxiv.org Internet Source	<1%

9	pdfcookie.com Internet Source	<1%
10	spygirl37.files.wordpress.com Internet Source	<1%

Exclude quotes	On	Exclude matches	< 10 words
Exclude bibliography	On		

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: …….

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report | Paper

Name: _____ __**Department:** _____ **Enrolment No** _____

Contact No. _____**E-mail.** _____

Name of the Supervisor: _____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____

_____

_____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at ……..(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

<table>
<tr><th>Copy Received on</th><th>Excluded</th><th>Similarity Index (%)</th><th colspan="2">Generated Plagiarism Report Details (Title, Abstract & Chapters)</th></tr>
<tr><td rowspan="2"></td><td rowspan="4">• All Preliminary Pages
• Bibliography/Images/Quotes
• 14 Words String</td><td rowspan="2"></td><td>Word Counts</td><td></td></tr>
<tr><td rowspan="2">Report Generated on</td><td>Character Counts</td><td></td></tr>
<tr><td rowspan="2"></td><td>Submission ID</td><td>Total Pages Scanned</td><td></td></tr>
<tr><td></td><td>File Size</td><td></td></tr>
</table>

Checked by
Name & Signature **Librarian**

……

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com