M S RAMAIAH INSTITUTE OF TECHNOLOGY

DEEP LEARNING PROJECT

REPORT

TITLE – CRYPTOGRAPHIC APPLICATION USING NEURAL NETWORKS

by

JEEVAN N R – 1MS19IS053

Under The Guidance

Of

Dr. VIJAY KUMAR B P

INTRODUCTION

With the advent of Artificial Intelligence and Internet of Things (IoT) devices, demands for more innovative methods of preserving cybersecurity are growing. The prospect of utilizing neural networks (convolutional neural networks) for cryptography is relatively new. Recent attempts have aimed to use neural networks for cryptanalysis and have achieved dismal results. Rather than focusing on achieving a successful algorithm, the objective of this project is to explore how the facets of neural networks can be used and tuned for cryptographic operations, including encryption, decryption, signing, and verification.

By encompassing the CIA (Confidentiality, Integrity, and Availability) triad, in this project, I attempt to demonstrate an efficient neural cryptography algorithm (NCA) through tuning and implemented NCA through socket communication (which is a first step towards a a unique UI that allows users/devices to securely and seamlessly communicate). NCA is dynamically self-configurable, which results in a robust, yet terribly inconsistent algorithm, although it is still possible to achieve low error rates multiple times with neural networks.

I have also developed a signature/verification capability calculated based on the hidden states of the neural networks, which has not been accomplished before. The adopted and modified code presented in this project serves as a foundation to further research and improvements to the neural cryptography work. Once fully developed, the cipher will be able to improve its resilience when it is compromised and autonomously restore the confidentiality of the system. There are numerous opportunities for NCA, including the possibility of using NCA for secure communication between autonomous IoT devices. Another opportunity for NCA would be for Artificial Intelligence due to its dynamically self-configurable nature and because it can function efficiently in massively parallel processing environments.

The necessary packages to run this code are:

* NumPy

* TensorFlow (version 1.11.0)

* Pandas

* Math

* matplotlib.pyplot

* Seaborn

* socket

* Sys

METHODOLOGY

The art of secure communication has always brought about the invention of models that challenge existing standards and go beyond. Understanding the basic mechanics of secure communication allows us to understand how traditional models identify issues and solutions and to build upon these algorithms. In this paper, the focus is on symmetric key encryption, although the concepts presented in this paper could be applied to asymmetric key encryption as well.

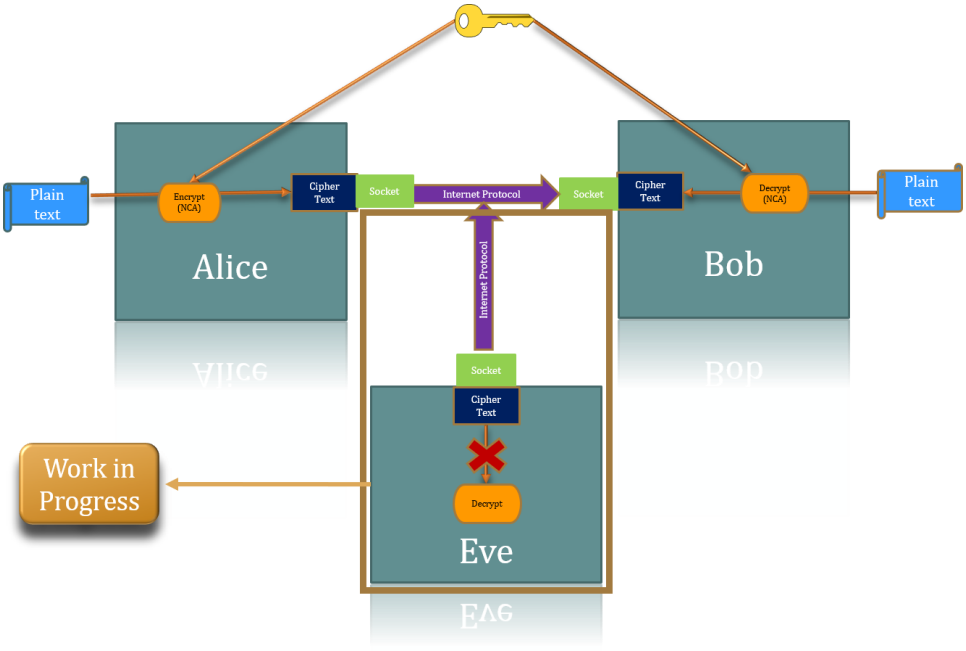
The implementation of “secure communication” can be described as follows:

Generation of the key, the encryption/decryption algorithm, sending the message (platform). An example scenario is as follows: The goal is for Alice to send a message to Bob, and Eve, a party trying to hack into Alice and Bob’s communication, to be unable to read the message. Alice takes a message (plain text) and uses the key she shares with Bob to encrypt plain text into cipher text. This ciphertext is sent to Bob, who consumes the communication and key, and decrypts the cipher text into plain text. Eve intercepts the communication of the cipher text and attempts to recover the plaintext with no knowledge of the keys and in some cases, the encryption algorithm.

The ability to quickly adapt to change and learn from penetrations are gradually becoming characteristics that define robustness in terms of security. Neural networks successfully fulfill this dynamic self-learning capability for many other tasks, such as image recognition for large datasets. Neural networks can be used in three different ways in relation to cryptography: To generate symmetric keys for the protocol (the keys are hidden in the weights), for encryption/decryption (the algorithm is the secret in the model), and for signing/verification. The potential of neural networks lies in the fact that the networks invent their own encryption/decryption and signing/verification algorithms. However, cryptography experts have overlooked the application of neural networks for cryptography because neural networks have a large solution space and are not efficient for cryptanalysis due to the algorithm’s inconsistency. For example, a recent attempt by researchers from Google Brain was performed (Abadi & Andersen 2016) and although they were successfully in training a neural network for cryptanalysis, the algorithm did not meet expectations.

Through this project, I attempt to explore how neural networks could be efficient for cryptography. I present a neural cryptography algorithm for symmetric encryption that can not only guarantee secrecy, but also fulfills the three principles of security (Confidentiality, Integrity, and Availability). Rather than focusing on cryptanalysis (strengthening Eve), I focused on improving the confidentiality of the message being transmitted between Alice and Bob. The goal is to achieve a low error rate by altering the hyperparameters of the algorithm, and to implement the protocol through network sockets and simulate

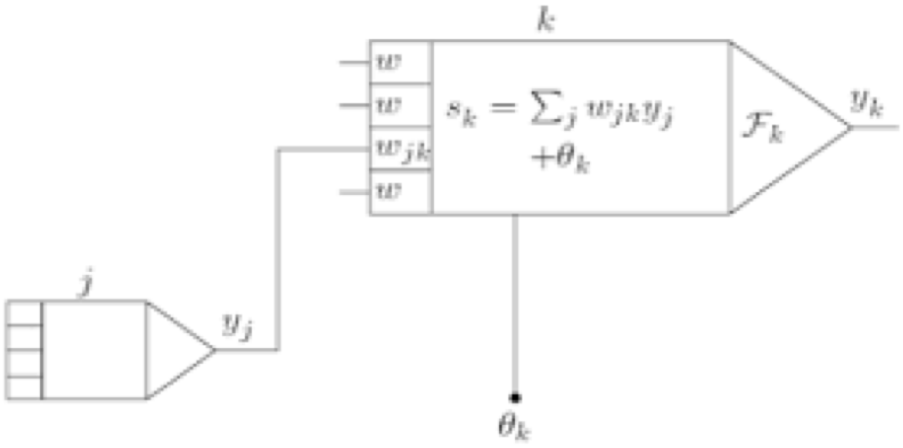
communication between Alice and Bob. I also explore a unique capability of neural networks to ensure integrity and authenticity through digital signatures. The digital signature model is similar to that used for encryption and decryption. It requires key generation, a signing algorithm, and a signature-verifying algorithm. The process for signing and verifying is as follows: Alice sends plaintext and her signature to Bob, who takes Alice's plaintext, computes the signature and compares it to the signature received. If the signature Bob computed and the signature Alice sent along with her plain text message are the same, then the message's integrity can be guaranteed; if not, then Bob will know that Eve or another third party has modified the original message. The project provides an implementation of digital signatures that uses the internal hidden layer states of Bob's and Alice's CNN machines to represent the 32-byte signatures.



CNN MODEL

As mentioned above, I employed Convolutional Neural Networks (CNNs) for the encryption/decryption (128-bit key) with signing/verification features. To create the original algorithm, I modified adopted code from Liam Schoneveld ("Adversarial Neural Cryptography in Theano") and GitHub user Ankesh Anand ("Adversarial Neural Cryptography in TensorFlow") to include the convolutional layers and experimented with the parameters by changing the learning rate of the optimizer, the epochs for training and testing, the initial weights, etc. I used the TensorFlow framework and abandoned implementation with the Theano framework, due to lack of CPU optimization libraries on Windows 10; NCA has been implemented in Theano, TensorFlow, and regular python framework by different authors, however, I experimented with the Adam optimizer,

Adagrad optimizer, and Momentum optimizer, which use the loss functions for minimizing error (the Adam optimizer resulted in the best performance). Next, I designed the UI accompanying the NCA algorithm. The user can enter input in ASCII, which is then converted to binary and structured appropriately in order to be accepted by np.array for both encryption/decryption and signing/verification functions.



For socket communication, the user (Alice) types in a plaintext message, which is then encrypted (cipher text) and sent to Bob's socket. Upon receiving the cipher text, Bob's shared key is used to decrypt the message and Bob prints out the recovered message. Three functions are defined for socket communication:

- sock_send creates a TCP/IP socket for Alice, binds the socket to the port (localhost port 10000) and serializes the data into an array of bytes.
- sock_recv_init` creates a TCP/IP socket for Bob, binds the socket to the port, and makes it listen for incoming connections.
- sock_recv` waits for a connection, and once it receives the data, it deserializes it.

The local sockets can be easily modified for communications between separate devices over the network.

Final Results

Threat Modeling: With NCA, even if the symmetric key is compromised, the algorithm is still hidden in the training and weights. This results in maintenance of the “secrecy is not security” principle, due to the fact that the algorithm can be provided publicly, but the dynamic self-configurable nature of the neural networks makes it hard to hack. However, if a method is devised to extract the weights out of the NCA machines, the hackers will be able to compromise the algorithm until an NCA retraining is forced. This same advantage also serves as a disadvantage since the CNNs are not consistent. However, the cryptography algorithm can be strengthened to meet future needs by hyper-parameter definition/tuning without the use of penetration testing, which is not feasible with existing traditional toolchains for NCA. Key revocation/reissue deployment is not complicated as merely retraining the NCA on legitimate entities/machines is sufficient. As of now, the keys and NCA algorithms may not meet industry standards, but if further improved, NCA can attempt broader adoption. The results demonstrate that NCA training is slower than AES and SHA256 in all tests. NCA also requires high compute power and memory capacity (16 GB RAM and 8-core 3.1 GHz CPU with hyperthreading). However, Massively Parallel Processing (MPP) computers can make NCA training faster and enable crypto-recycling frequently and efficiently.

Future Work

The foremost concern is to **test NCA further and** try and improve the consistency of NCA through further hyperparameter tuning. Future work also includes implementing the sockets over Internet Protocol, instead of localhost. In addition, other goals include implementing Eve and her socket as part of the crypto pipeline. Tests, such as performance benchmarking, can be performed against AES, SHA-256, and other encryption techniques to assess the strengths and weaknesses of the NCA algorithm for further improvement.

Another important field that I would like to explore is the combination of quantum and neural cryptography. Quantum cryptography involves altering the contents of a message in transit as soon as the attacker (Eve) sees it; this guarantees the integrity of the message. For example, quantum cryptography could be used as a potential channel for transmission of messages encrypted through NCA.

References

- Abadi, M & Andersen, D. Learning to Protect Communications with Adversarial Neural Cryptography. October 24 2016. Google Brain.
- Adversarial Neural Cryptography in Theano. https://nlml.github.io/neural-networks/adversarialneural-cryptography/

- Adversarial Neural Cryptography in TensorFlow. https://github.com/ankeshanand/neuralcryptography-tensorflow
- Fernet (symmetric encryption). https://cryptography.io/en/latest/fernet/
- Cryptography using Artificial Intelligence - Jonathan Blackledge
- "Learning to Protect Communications with Adversarial Neural Cryptography" by Abadi, M & Andersen, D.