A SECURE COMMUNICATION WITH Generative Adversarial Nets (GANs) of ADVERSARIAL NEURAL CRYPTOGRAPHY

Shreyash Turkar[1], Nidhi Singh[2],
[1,2]Department of Computer Science & Engineering,
Indian Institute of Information Technology, Nagpur Maharashtra, 440003, India
shryeash.turkar@iiitn.cse.in, nidhi.2592@gmail.com

Abstract: We here discuss whether we can employ neural nets to secure our communication channels. We take classic Alice, Bob and Eve example where Alice is trying to send a message to Bob and Eve is eavesdropping. Instead of using rigid symmetric encryption algorithms like AES, DES, we will create an adversarial neural network where Alice will be trained to encrypt using a shared key, Bob will be trained to decrypt using a shared key and Eve will be trained to reconstruct the message without a shared key.

***Index Terms*—Symmetric cryptography, GANs, CNN**

I. Introduction

Today, there are several types of neural networks which are used for various purposes. As we discover more ways of exploiting their potential, we ask whether they can also be used to make our communication secure and replace the existing rigid cryptographic algorithms. For sake of this paper, we will focus on symmetric encryption algorithms.

Symmetric cryptosystem can be easily explained by a classic example of three people named Alice, Bob and Eve. We assume that Alice and Bob have shared a secret key. Alice wants to send a secret message to Bob which she encrypts using their shared secret key to create ciphertext, Alice sends this ciphertext to Bob through a channel, presumably insecure. Bob uses the same shared secret key to decrypt the ciphertext. When ciphertext was intercepted by Eve but she can't decrypt it without the shared secret key.

II. Symmetric Encryption

Symmetric cryptosystem can be easily explained by a classic example of three people named Alice, Bob and Eve. We assume that Alice and Bob have shared a secret key. Alice wants to send a secret message to Bob which she encrypts using their shared secret key to create ciphertext, Alice sends this ciphertext to Bob through a channel, presumably insecure. Bob uses the same shared secret key to decrypt the ciphertext. When ciphertext was intercepted by Eve but she can't decrypt it without the shared secret key.

A. *Classic Cryptosystem*

A classic scenario in security involves three parties: Alice, Bob, and Eve. Typically, Alice and Bob wish to communicate securely, and Eve wishes to eavesdrop on their communications. We start with a particularly simple instance of this scenario, depicted in 1, in which Alice wishes to send a single confidential message P to Bob. The message P is input to Alice. When Alice processes this input, it produces an output C. ("P" stands for "plaintext" and "C" stands for "ciphertext".) Both Bob and Eve receive C, process it, and attempt to recover P . We represent what they compute by PBob and PEve, respectively. Alice and Bob have an advantage over Eve: they share a secret key K. We treat K as an additional input to Alice and Bob. We assume one fresh key K per plaintext P, but, at least at this abstract level, we do not impose that K and P have the same length.

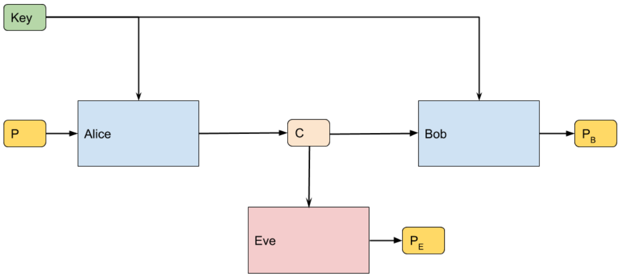

Fig. 1: Classic Alice, Bob and Eve Example

B. *Integrating Neural Network into Cryptosystem*

We train three neural networks Alice, Bob and Eve whose jobs are as follows:

- Alice's job is to take in n-bit message (encoded as a vector of -1 and 1 to represent 0 and 1 respectively) and n-bit key as input to output a n-bit cipher-text.
- Bob's job is to take the n-bit cipher-text created by Alice and use the n-bit key as input to reconstruct the original n-bit message.
- Eve's job is to take only the n-bit cipher-text and try to recreate original n-bit message.

C. *NETWORK ARCHITECTURE*

III. Conclusion

This will be my conclusion

References

[1]
[2]