Adversarial Learning for Cross Layer Security

Hesham Mohammed and Dola Saha
{hhussien,dsaha}@albany.edu
University at Albany, SUNY
Albany, New York, USA

ABSTRACT

Spectrum access in the next generation wireless networks will be congested, competitive, and vulnerable to malicious intents of strong adversaries. This compels us to rethink wireless security for a cross-layer solution addressing it as a joint problem for encryption and modulation. We propose a novel neural network generated cross-layer security algorithm where the trusted transmitter encodes a secret message using a shared secret key to generate a secured waveform. This encrypted waveform remains undeciphered by the adversary while the intended receiver can recover the secret. Cooperative learning is introduced to enable our trusted pair to defeat the adversary and learn the encryption and modulation jointly. The model can encode any modulation order and improves both reliability and secrecy capacity compared to prior work. Our results demonstrate that the trusted pair succeeds in achieving secure data transmission while the adversary can not decipher the received cipher data.

CCS CONCEPTS

• **Security and privacy → Cryptography**; • **Theory of computation → Cryptographic protocols**; • **Mathematics of computing → Information theory**.

KEYWORDS

Wireless Security, Adversarial Learning, End-to-end Encryption, Information Theoretic Analysis

ACM Reference Format:
Hesham Mohammed and Dola Saha. 2021. Adversarial Learning for Cross Layer Security. In *3rd ACM Workshop on Wireless Security and Machine Learning (WiseML '21), June 28-July 2, 2021, Abu Dhabi, United Arab Emirates.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3468218.3469043

1 INTRODUCTION

The growing need for spectrum to support the next generation (xG) wireless networks can only be satisfied by coexistence of frequency-agile, cognitive and heterogeneous nodes. In this scenario, securing mobile networks is an essential requirement to achieve trusted communication among users. Classical cryptography has always been an add-on feature for communication as a result of the layered approach to network design. Physical layer security has gained attention as a technology to secure wireless communication, but is chosen to complement cryptography, not as a replacement. The disconnect between higher layer cryptography and physical layer signal generation is predominantly due to the fact that encryption algorithm is designed using binary operations with security as the primary goal, whereas the communication signals are designed to maximize the spectral efficiency. Cryptography guarantees security regardless of the signal reception capability of adversary but under the assumption of constrained computational power of the adversary. On the other hand, physical layer security finds its roots in information theory and holds without any restriction of the adversary's computation capability, but with the constraint on adversary's reception quality. Individually, each one of these technologies have significant advantages that has been very well studied in the literature. However, the combination of these two areas has not been embraced for improving security in a practical wireless environment.

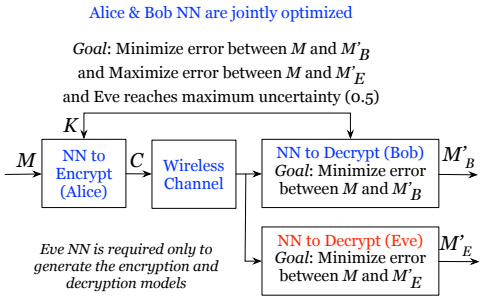


Figure 1: Three neural networks trained adversarially to achieve cross-layer security.

Recent years have experienced tremendous growth in deep learning research and its capabilities are leading to breakthroughs [20] in several domains of science and technology. Data driven deep neural networks (DNNs) are known to learn complex functions, may require less computation to approximate certain functions and operate on real numbers, unlike traditional cryptography. Specifically, adversarial learning [12] have shown tremendous potential in generating realistic images, videos, speech, handwritten text and even used in wireless and medical applications. We leverage these capabilities of NNs to create a novel cross-layer integrated cryptography algorithm for use in a 'zero trust' electromagnetic environment.

Cryptography is beyond simple mapping of bits to another domain. To ensure secrecy and integrity of information, it is important that an eavesdropper (Eve) listening to the cipher data (communication between Alice, the sender, and Bob, the receiver) will not be able to decipher it without a key. Hence, we design Alice and

WiseML '21, June 28-July 2, 2021, Abu Dhabi, United Arab Emirates

ACM ISBN 978-1-4503-8561-9/21/06…$15.00
https://doi.org/10.1145/3468218.3469043

Bob to each have a neural network to learn the encryption algorithm, as shown in Figure 1. It is only possible to obtain higher levels of secrecy when we can also emulate an adversary, a passive eavesdropper in our model, and cooperatively train Alice and Bob to defeat Eve. Hence, we introduce Eve as another neural network during the training phase that is constantly trying to decipher the data and the model is trained in an adversarial manner to converge to a system, where Alice and Bob can defeat Eve. The goals of both Bob and Eve networks are to recover the message transmitted by Alice. Alice and Bob are jointly trained such that Eve reaches maximum uncertainty. The input of the encryption block is in bit domain, whereas the output is a complex signal; thus creating a cross-layer encryption technique.

Thus, our contributions in this paper can be listed as:

- We propose a novel neural network generated cross-layer security algorithm that encrypts messages with a shared secret key and converts bits directly to secured signal, represented in complex numbers, through an adversarial learning.
- We utilize the capability of bi-directional recurrent neural network in learning sequences to generate encrypted signal that depends on both the past and the future output. This improves both the reliability and secrecy with a shorter key length compared to feature extraction networks.
- We design a neural network architecture, which can be parameterized based on the modulation order, such that encrypted constellation can represent one or multiple bits of plaintext data.
- We have formulated secrecy capacity of our model based on information-theoretic analysis.
- Our results show that the proposed model succeeds in learning a secure waveform that remains unrecognizable to Eve and achieves a higher secrecy level compared to prior work.

2 RELATED WORK

Recent years have experienced increasing use of deep learning in wireless communication systems. An autoencoder (AE) model is used in [19] to learn the different modulation orders without any prior mathematical formulation. NNs have also been used to learn end-to-end OFDM systems for wireless [24]communications. In [16], authors use recurrent neural networks (RNNs) to decode convolutional codes. In [6], authors propose an AE model to learn a joint source-channel coding algorithm such that optimal codes can be achieved.

The idea of using adversarial neural network in cryptography was introduced in [1], where one bit of plain text is encrypted using adversarial learning to generate a floating point value. However, representation of one bit with floating point value is not quite useful for storage purposes or transmission over the air as it increases the size of the cipher data by N times, where N-bits are used for floating point representation (in fixed point representation). In [8], the authors reduce the model complexity used in [1] by using fully connected layers. Although the results show that Alice and Bob could exchange confidential data successfully in fewer training iterations, the model is imperfectly secured due to reduction of complexity of the encryption algorithm. In [3, 10, 17], the authors use autoencoder model and train it in a Gaussian wiretap channel environment to achieve key-less encryption between Alice and Bob. However, they assumed that Eve's channel is worse compared to the main channel. Moreover, the results show that Eve can decode the secret if the received SNR at Eve is higher than Bob. We choose this opportunity to jointly redesign the encryption and wireless signal generation, such that encrypted floating point cipher data can be mapped to complex domain for spectrally efficient waveform generation.

The closest work to ours is [18], where authors introduced a cooperative learning model for creation of secured modulation. The model uses three neural networks: a trusted sender (Alice), a trusted receiver (Bob), and an eavesdropper (Eve). The three networks are implemented using convolutional neural networks (CNNs), where the secured modulation is created based on the secret message and a shared key. A cooperative objective function is introduced between the trusted pair to learn the encryption algorithm while defeating Eve. A discrete activation function is also introduced to support lossy media transmission. Although the results are promising, there are several issues in this prior work. *First*, the learning model was restricted for accepting only one bit of information to be encrypted (i.e., modulation order $m = 1$). *Second*, the encrypted modulation is restricted to a finite number of levels based on the discrete $tanh$ function. This reduces the randomness of the secret modulation and consequently the secrecy level of the transmitted waveform is reduced. *Third*, the system was restricted to accept equal size of message and key in each iteration, which might be impractical in many scenarios. In this paper, we present a recurrent neural network based model that is empowered with memory units to remember prior states. Our work overcomes the limitations of prior work and enables a practical cross-layer encryption model.

3 SYSTEM MODEL

The wiretap channel is a three-node network model, which involves three nodes: a trusted sender (Alice), a trusted receiver (Bob), and an eavesdropper (Eve), as shown in Figure 1. Alice transmits a confidential message M to Bob in the presence of Eve. In this work, we consider symmetric key encryption. Thus, Alice encodes M using a shared secret key K to produce the cipher data C, which is transformed to a complex vector X, to be sent over a broadcast channel to Bob. Both Bob and Eve receive complex cipher data Y_{Bob} and Y_{Eve}, which are X after passing through the channel. They recover the real representation of cipher data C'. Bob decodes C' using K to obtain the predicted plain data $\hat{M}_{Bob}$. However, Eve uses C' only to obtain $\hat{M}_{Eve}$, which is the predicted output for M. In this paper, we use the degraded Gaussian wiretap channel. Hence, if Alice transmits X, the received symbols for both Bob and Eve are given by:

$$Y_{Bob} = X + N_{Bob}, \qquad Y_{Eve} = X + N_{Eve} \tag{1}$$

where $N_{Bob}, N_{Eve} \sim \mathcal{CN}(0, \sigma^2)$ are the added complex noise vector and σ^2 depends on the received signal to noise ratio (SNR). Note that both the channels for Bob and Eve have the same statistical properties.

Alice, Bob and Eve apply neural networks with parameters θ_A, θ_B and θ_E respectively. Alice's network is designed to accept M and K in bits. In other words, $M, K \in \mathcal{B}$, where $\mathcal{B} = \{0, 1\}$. It also accepts modulation order, m. Hence, $M \in \mathcal{B}^{N \times m}$, where N is the width of the input symbol. Similar to M, K's inner dimension (m_k) can

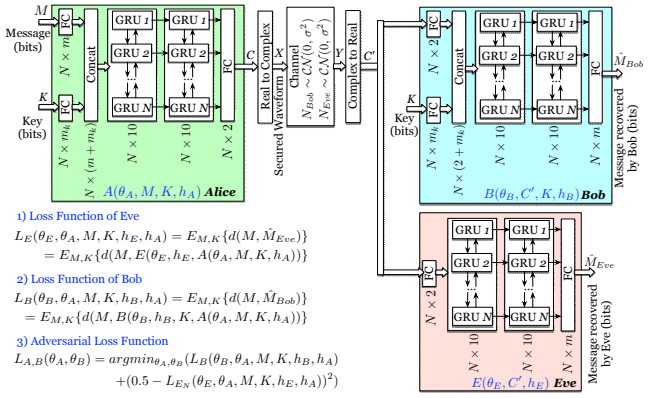


Figure 2: Neural Network Architecture

be extended to increase the secrecy of C (i.e., $K \in \mathcal{B}^{N \times m_k}$). Since neural networks operate on real data, Alice's network outputs real data as cipher, C, via two output channels (i.e., $C \in \mathcal{R}^{N \times 2}$). Thus, the transmitted cipher symbol X is given by $X = C_1 + jC_2$, where C_i is the output dimension of the Alice network in real domain.

For practical implementation, we constrain C within the range (-1,1). Bob's network is designed to accept the real C' as well as K and outputs $\hat{M}_{Bob} \in \mathcal{R}^{N \times m}$. On the other hand, Eve's network accepts C' only, and outputs $\hat{M}_{Eve} \in \mathcal{R}^{N \times m}$. At the end of successful training process, $\hat{M}_{Bob}$ should converge to $\mathcal{B}$, while $\hat{M}_{Eve}$ should not. Hard decision decoding is performed on both $\hat{M}_{Bob}$ and $\hat{M}_{Eve}$ to covert them from $\mathcal{R}$ to $\mathcal{B}$.

According to Wyner [23], Bob can reliably decode C to obtain M with a vanishing probability of error (i.e., $P_e = Pr[\hat{M}_{Bob} \neq M] = 0$), while Eve can not decode C if the transmission rate R between Alice and Bob is below or equal to secrecy capacity C_{sec}, which is given by:

$$C_{sec} = \max_{M} \{I(M; \hat{M}_{Bob}) - I(M; C)\} \tag{2}$$

where $I(X;Y)$ is the mutual information between X and Y. In the security context, Alice and Bobs' networks should be optimized using equation (2). There are challenges in directly using (2) in a neural network architecture and design of loss functions. First, it is intractable to estimate the mutual information from data samples. Secondly, secrecy capacity, C_{sec} in (2) is derived for key-less encryption, which is not our goal. This paper implements a shared secret key based encryption technique, due to which objective functions need to be derived as an approximation for C_{sec}.

4 NEURAL NETWORK ARCHITECTURE

Neural networks can be classified into two types: feature extraction and sequential networks. Feature extraction networks are capable of learning local correlation and dimension reduction. Hence, output of these networks depends on current input and is given by:

$$Y_t = f(X_t, \theta) \tag{3}$$

where $f(.)$ is the network's mapping function, X_t and Y_t are the input and the output at time instant t, while θ is the network parameters. Feed forward networks (FFNs) and convolutional neural networks (CNNs) [11] are the most commonly used networks for feature extraction. However, these networks are not capable of learning sequences or track time-domain dependencies. On the other hand, sequential networks are capable of learning data sequences and capturing the time-domain dependencies of the output on the current input and the current state of the network, which depends on the previous inputs' history. Thus, the output of these networks can be expressed as:

$$(Y_t, h_t) = f(X_t, \theta, h_{t-1}) \tag{4}$$

where $f(.)$ is the network mapping function, θ is the network parameters, Y_t and h_t are the network output and the current state respectively, which depend on the current input X_t and the previous state h_{t-1} that encodes the history of the network. Recurrent neural networks (RNNs) [5] are the most common type in sequence learning. However, RNNs can not capture long-term dependencies and suffer from vanishing gradient problem. Gated Recurrent Units (GRUs) [4] and Long Short Term Memory networks (LSTMs) [15] are special kinds of RNNs that are capable of capturing the long-term dependencies and solving the vanishing gradient problem.

In this work, we consider a bidirectional GRU (BiGRU) as the base unit for building the neural network for Alice, Bob, and Eve respectively. BiGRU consists of 2 GRU cells [14]. The first one processes input sequence in the forward path ${h_t}^f = f(X_t, \theta, h_{t-1})$, while second processes the backward path ${h_t}^b = f(X_t, \theta, h_{t+1})$, where ${h_t}^f, {h_t}^b$ are the current states for forward and backward paths respectively. The output of BiGRU cell can be expressed as:

$$Y_t = < {h_t}^f, {h_t}^b > \tag{5}$$

where $< X, Y >$ is the concatenation output of X and Y. BiGRU's structure increases the encrypted symbol's secrecy level in the security context since the encrypted symbol depends on the whole input sequence. Moreover, BiGRU increases the decoding capability at the receiver side, which improves the communication reliability.

4.1 Neural Network Structure

The learning model consists of three neural networks: Alice (encryption model), Bob (decryption model), and Eve (adversarial model), as shown in Figure 2. Alice accepts M and K in bits and output real cipher data C, while Bob accepts the received real cipher data C' and K and outputs $\hat{M}_{Bob}$. Eve accepts only C' and tries to infer the secret message through $\hat{M}_{Eve}$.

Alice's network starts by two fully connected (FC) layer for both $M \in \mathcal{B}^{N \times m}$ and $K \in \mathcal{B}^{N \times m_k}$, where N is the symbol length, while m and m_k represents the modulation order per sample for both M and K respectively. The role of the FC layer to perform initial permutation for both M and K, then change the domain of M and K form $\mathcal{B}$ to $\mathcal{R}$ to increase the mapping space and avoid singularities (zero mappings). The outputs of the FC layers are concatenated over the inner dimension. In other words, the output vector of the concatenation stage has a dimension of $N \times (m + m_k)$ then fed as an input to the GRU layers.

Two BiGRU layers are used to perform sequence encoding for the input vector. Unlike convolutional networks, BiGRU guarantees that each entry of the encoded vector depends on the previous, current, and future inputs (i.e., each entry of the output vector is a function of the whole sequence) to increase the randomness of the encoded vector and improves the reliability at the decoder side. Moreover, the output of the encoded vector depends on the system state h_t, which depends on the previous input symbols.

Hence, Alice can increase C_{sec} with smaller K, which guarantees that the system is robust and is not restricted to learn one-time pad encryption [21]. The final layer is the FC layer to reduce the dimension of the learned features to the channel dimension n. In this work, we assume a single channel (i.e., $n = 2$); hence the output real cipher data C consists of two real vectors corresponding to in-phase (I) and quadrature-phase components (Q). $Tanh(x)$ activation function is used for both the GRU layers and the final layer to ensure that the output range between (-1,1) satisfies the power constraint.

Bob's network is similar to Alice's with some minor differences. First, the rule of the GRU layers is to decode the cipher data and compensate the channel effects to recover the original message. Second, Sigmoid activation function is used in the last layer to ensure $\hat{M}_{Bob}$ within the range of (0,1). Eve's network has a similar structure except for the concatenation layer since Eve predicts the secret form C' only. Xavier initializationis used to initialize all the parameters of the learning model to accelerate the convergence of the networks and avoid gradients saturation.

4.2 Objective Functions

In this section, we formulate the objective of each entity within the learning model. Alice encodes M using K, which is shared to Bob, and outputs C. Bob tries to decode C' using K to obtain M by minimizing the error between the predicted output $\hat{M}_{Bob}$ and M, while Eve infers M using only C', by minimizing the error between M and $\hat{M}_{Eve}$. Informally speaking, Alice and Bob cooperate to find out a secure way of confidential data exchange and defeat Eve such that Eve can not recover the secret from the received cipher data. Thus, the three networks should be trained in an adversarial manner. In the rest of this section, we derive the loss function that supports the three networks' adversarial behavior.

We define $A(\theta_A, M, K, h_A), B(\theta_B, C', K, h_B), E(\theta_E, C', h_E)$ are the mapping functions for Alice, Bob, and Eve respectively where h_A, h_B, h_E represent the state vectors for the three networks which depend on the input history of the three networks. In addition, we define $d(M, \hat{M})$ as the L2 norm between M and $\hat{M}$. Intuitively, Eve loss function can be formulated as:

$$\begin{aligned} L_E(\theta_E, \theta_A, M, K, h_E, h_A) &= E_{M,K}\{d(M, \hat{M}_{Eve})\} \\ &= E_{M,K}\{d(M, E(\theta_E, h_E, A(\theta_A, M, K, h_A))\} \end{aligned} \quad (6)$$

where $E_{M,K}\{.\}$ is the expected value over M and K set.
It is noted here that Eve has full access to M during the training process such that Eve can reach the most adversarial behavior. Similarly, Bob's loss function is given by:

$$\begin{aligned} L_B(\theta_B, \theta_A, M, K, h_B, h_A) &= E_{M,K}\{d(M, \hat{M}_{Bob})\} \\ &= E_{M,K}\{d(M, B(\theta_B, h_B, K, A(\theta_A, M, K, h_A))\} \end{aligned} \quad (7)$$

As shown in (7), L_B minimizes the distance between M and $\hat{M}_{Bob}$ that guarantees the communication reliability between Alice and Bob. However, L_B is not sufficient to achieve secrecy between Alice and Bob. Thus, we define a cooperative loss function between Alice and Bob to guarantee both reliability and secrecy as:

$$\begin{aligned} L_{A,B} = argmin_{\theta_A,\theta_B}\{&L_B(\theta_B, \theta_A, M, K, h_B, h_A) \\ &- L_E(\theta_E, \theta_A, M, K, h_E, h_A)\} \end{aligned} \quad (8)$$

As shown in (8), $L_{A,B}$ minimizes Bob's loss L_B to achieve communication reliability between Alice and Bob and maximizes Eve's loss L_E to achieve secrecy. Moreover, $L_{A,B}$ optimizes the network parameters of both Alice and Bob only (i.e., θ_A and θ_B) such that cooperative learning takes place only between the trusted partners. Note that, Both L_E and L_B depend on Alice's network (i.e., θ_A), which means that Alice's network tries to learn a secure pattern that Bob can recognize this pattern while Eve can not. Thus, during each epoch in the training process, θ_A are frozen during updating Eve's parameters (i.e., θ_E)) using (6), then Both θ_A and θ_B are updated simultaneously based on (8) while θ_E are frozen.

The cooperative loss function introduced in (8) is similar to the min-max optimization problem in GANs [13]. Thus, the cooperative loss function needs to be formulated such that it can be implemented easily. According to the entropy definition introduced by Shannon [22], Eve reaches the maximum uncertainty if Eve's error probability of decoding the received signal equals 0.5. Thus equation (8) can be reformulated as:

$$\begin{aligned} L_{A,B}(\theta_A, \theta_B) = argmin_{\theta_A,\theta_B}(&L_B(\theta_B, \theta_A, M, K, h_B, h_A) \\ &+ (0.5 - L_{E_N}(\theta_E, \theta_A, M, K, h_E, h_A))^2) \end{aligned} \quad (9)$$

where L_{E_N} is the normalized loss function of Eve.
As shown in (9), the first component minimizes the decoding error between Alice and Bob to achieve communication reliability, while the second component minimizes the difference between the normalized error (i.e., probability of error) at Eve and 0.5 to achieve the maximum uncertainty.

5 INFORMATION THEORETIC ANALYSIS

In this section, we formulate the secrecy capacity C_{sec} of the learning model and show that C_{sec} increases if we use sequential networks (i.e., RNN, LSTM, or GRU) compared to feature extraction networks (i.e., FFNs, and CNN).

As shown in (2), C_{sec} can be obtained by maximizing $I(M; \hat{M}_{Bob})$, while minimizing $I(M; C)$. Starting by the second term $I(M; C)$ can be given by:

$$I(M; C) = H(M) - H(M/C) \quad (10)$$

where $H(.)$ is the binary entropy function.
The conditional joint entropy $H(M, K/C)$ is given by:

$$H(M, K/C) = H(K/C) + H(M/K, C) = H(K/C) \quad (11)$$

since $H(M/K, C) = 0$. Then, the conditional entropy $H(M/C)$ can be bounded by:

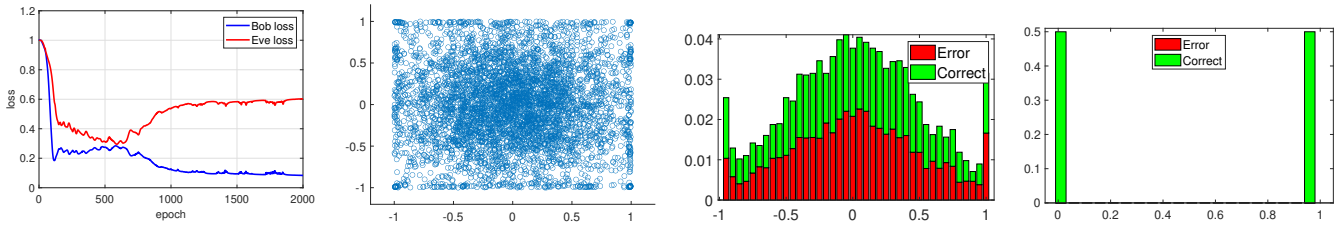
$$H(M/C) \leq H(K/C) \leq H(K) \quad (12)$$

$$I(M; C) \geq H(M) - H(K) \geq N(H(P_m) - H(P_K)) \quad (13)$$

where P_m and P_K are the probability density function of each entry in M and K respectively.
On the other hand, $I(M; \hat{M}_{Bob})$ is given by:

$$\begin{aligned} I(M; \hat{M}_{Bob}) &= H(\hat{M}_{Bob}) - H(\hat{M}_{Bob}/M) \\ &= H(\hat{M}_{Bob}) - H(\hat{M}_{Bob}/Y) \end{aligned} \quad (14)$$

since Alice's encoding function is deterministic after the training process. To infer the second term in (14) (i.e., $H(\hat{M}_{Bob}/Y)$), we use the concept of typical sequence [9] to calculate the probability of the correct decoding (i.e., $P(M = \hat{M}_{Bob})$). Feature extraction networks



(a) Loss function. **(b) C Constellation.** **(c) C distribution.** **(d) $\hat{M}_{Bob}$ distribution.**

Figure 3: Loss function and data distribution for $m = 2$ at the training SNR.

are one-to-one mapping. Assuming uniform distribution over the set of transmitted symbols $\mathcal{M}$, the probability of decoding $\hat{M}_{Bob}$ from received vector Y_{Bob} is given as:

$$P(\hat{M}_{Bob}/Y_{Bob}) = 1/Q$$

where Q is the number of all possible sequences that Y can take. On the other hand, if a sequential model is used, which has $2^{k\times N\times L}$ states, then

$$P(\hat{M}_{Bob}/Y_{Bob}) = 2^{k\times N\times L}/Q$$

where L is the number of sequential layers of Alice, k is the number of bits used to represent the value inside each sequential unit, and the number of units in each layer equals N . Thus $I(M;\hat{M}_{Bob})$ can be given by:

$$I(M;\hat{M}_{Bob}) = N(H(P_m) - (1/N)\log Q + C_{eff}) \tag{15}$$

where C_{eff} effective model capacity which can be expressed as:

$$C_{eff} = \begin{cases} 0, & \text{FFN, CNN} \\ k \times L, & \text{RNN, LSTM, GRU} \end{cases} \tag{16}$$

By maximizing $I(M;\hat{M}_{Bob})$ and minimizing $I(M;C)$, C_{sec} is given:

$$C_{sec} = N(H(P_K) + k \times L - (1/N)\log Q) \tag{17}$$

Consequently, sequential models' secrecy rate R_{sec} can achieve C_{sec}, which reduces the secrecy-reliability trade-off. Moreover, the randomness of the transmitted cipher data can be increased due to the sequential model's high capability of decoding the received sequences through learning optimal decoding schemes.

6 PERFORMANCE EVALUATION

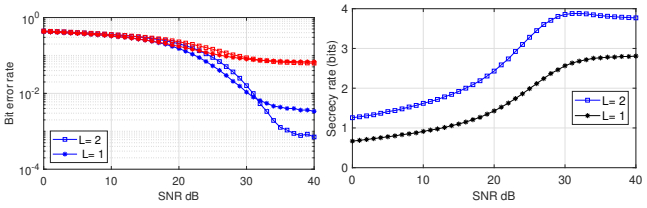
6.1 Experimental Setup

We implement our experiments in Tensorflow [2]. In our experiments we choose the symbol length $N = 48$, such that N is similar to 64-FFT as in Wi-Fi implementation [7]. In our experiments, the training set plain data, M, has 20000 symbols. The batch size is 8000, and the learning rate equals 0.001. We also train the learning model to accept different modulation order m. We choose the training signal-to-noise ratio SNR_t equals 20 dB and the order of encryption key $m_k = 1$ for the modulation orders $m = \{1, 2\}$. On the other hand, $SNR_t = \{25, 30\}$ and $m_k = \{2, 3\}$ for $m = \{4, 6\}$. We maintain m_k to be less than or equal to m. Moreover, we repeat a finite set of key symbols over the whole plain data set to ensure that the learned algorithm is robust enough and it is not restricted to learn one-time pad encryption [21]. The key to data set ratio is 0.005. The number of training epochs is 2000. The three networks are trained simultaneously in each epoch such that Eve's weights and biases are frozen during updating the weights and biases of both Alice and Bob based on the cooperative loss function introduced in (9). Alice and Bob's weights are frozen while Eve updates her weights and biases using her objective function introduced in (6). In the testing phase, we construct a plain data test set consists of 1000 symbols, then a key test set is constructed, which has the same key to plain data ratio used for training. The test data set for both M and K are constructed from different seeds from those used in the training phase so that we ensure that the system is not data sensitive.

6.2 Evaluation

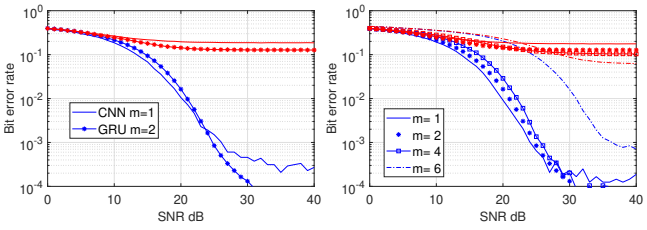
Figure 3 shows the loss function, the learned cipher constellation C, and the distribution of the decoded data $\hat{M}_{Bob}$ for $m = 2$ after a successful training process. Figure 3a shows the loss functions for Alice-Bob and Eve. Both the loss functions decrease at the same rate, which means that both Bob and Eve can partially decode Alice's received pattern. After some time, Alice and Bob succeed in finding a confidential way for secure data exchange. However, Eve's loss increases until she reaches the maximum uncertainty at the end of the training process. Thus, cooperative learning between Alice and Bob succeeds in figuring out their way of secure data transmission. Figure 3b shows the learned cipher constellation, C, transmitted over an AWGN broadcast channel. The cipher constellation looks like a Gaussian noise with zero mean and unity variance. Thus, there is no recognizable pattern that can be detected by Eve. Moreover, we use $tanh(x)$ activation function for power normalization, which increases the constellation's randomness. The cipher constellation's randomness can be observed in figure 3c, which shows C's distribution. The distribution has the shape of the Gaussian distribution with zero mean and unity variance. Thus, C does not carry any statistical properties of the plain data, M. Furthermore, if we apply hard decision decoding on the cipher data C, the aggregate probability of error equals 0.5, which guarantees that C has the maximum uncertainty property. Figure 3d shows the distribution of the decoded plain data at Bob, $\hat{M}_{Bob}$, which indicates that Bob can decode the data correctly, and $\hat{M}_{Bob}$'s distribution converges to the bit values. Hence, Alice and Bob succeed in finding a way to encrypt/decrypt the confidential data without any information leakage in C. For the bit error rate (BER) curves presented in this section, we choose to show Bob's BER in *blue* color, while Eve's BER in *red* color.

Effect of number of GRU units: Figure 4 shows the BER and the corresponding secrecy rate R_{Sec} for $m = 6$ to analyze the effect of reducing the layers of GRU units, L. Figure 4a shows the BER for $m = 6$ with two different models (i.e., $L = 2$ and $L = 1$). Bob's BER for $L = 2$ is lower than $L = 1$ indicating increase of communication reliability with more number of GRU units Also, Eve's BER (in



(a) BER for $m = 6$. **(b) Secrecy rate R_{Sec} for $m = 6$.**

Figure 4: BER and secrecy capacity for AWGN channel.



(a) CNN vs GRU. **(b) Different modulation orders.**

Figure 5: BER for CNN and GRU for AWGN channel.

red) for $L = 2$ is higher than $L = 1$ indicating higher secrecy with more GRU units. Thus the secrecy-reliability trade-off decreases by increasing L. This result is reflected in figure 4b. It is shown that R_{sec} for $L = 2$ is always higher than $L = 1$. This result verifies the C_{sec}'s expression derived in (17).

Comparing with prior work: To show the advantage of the proposed GRU model, we have implemented the CNN model proposed in [18]. Figure 5 shows the BER for the CNN and GRU models for different modulation orders m. Figure 5a shows that the GRU model achieves a reliable communication with higher modulation order (i.e., $m = 2$) compared to a CNN based model with lower modulation order (i.e., $m = 1$). Thus, the GRU model increases the achievable secrecy rate, R_{sec}, between the trusted pairs by accepting higher modulation orders and learning optimal encoding/decoding algorithms compared to the CNN model. Figure 5b shows the BER curves for all modulation order for the proposed GRU model. The figure shows that the GRU model enables us to transmit M with different values of m. However, the CNN model was restricted to $m = 1$ only. Moreover, the GRU model maintains a higher secrecy level such that Eve (curve in red) can not decode the data even at higher SNR. Also, the performance degradation between different modulation orders is relatively small since R_{sec} depends on K and the number of GRU layers L. R_{sec} can be increased by increasing either the depth of the encryption key, m_k, or the number of GRU layers, L, or both.

7 CONCLUSION

This paper shows the power of sequential neural network models (GRU) to learn a novel cross-layer encryption algorithm. To achieve a high secrecy level, we train the trusted pair in an adversary's presence to maximize both the communication reliability between the trusted pairs and the error probability at the adversary. The model ensures practical design constraints where the system accepts higher order modulation for the data and a relatively short key size. Our results showed that GRU models increase the secrecy rate compared to the convolutional models as they use higher order modulation and increases the randomness of the transmitted cipher.

REFERENCES

[1] Martín Abadi and David G Andersen. 2016. Learning to protect communications with adversarial neural cryptography. *arXiv preprint arXiv:1610.06918* (2016).

[2] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*.

[3] Karl-Ludwig Besser, Pin-Hsun Lin, Carsten R Janda, and Eduard A Jorswieck. 2019. Wiretap code design by neural network autoencoders. *IEEE Transactions on Information Forensics and Security* 15 (2019), 3374–3386.

[4] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Association for Computational Linguistics, Doha, Qatar, 103–111. https://doi.org/10.3115/v1/W14-4012

[5] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[6] Kristy Choi, Kedar Tatwawadi, Aditya Grover, Tsachy Weissman, and Stefano Ermon. 2019. Neural joint source-channel coding. In *International Conference on Machine Learning*. PMLR, 1182–1192.

[7] IEEE Computer Society LAN/MAN Standards Committee et al. 2007. IEEE Standard for Information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *IEEE Std 802.11ˆ* (2007).

[8] Murilo Coutinho, Robson de Oliveira Albuquerque, Fábio Borges, Luis García Villalba, and Tai-Hoon Kim. 2018. Learning perfectly secure cryptography to protect communications with adversarial neural cryptography. *Sensors* 18, 5 (2018), 1306.

[9] Thomas M Cover. 1999. *Elements of information theory*. John Wiley & Sons.

[10] Rick Fritschek, Rafael F Schaefer, and Gerhard Wunder. 2019. Deep learning for the Gaussian wiretap channel. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 1–6.

[11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (Eds.). 2672–2680.

[13] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial networks. *arXiv preprint arXiv:1406.2661* (2014).

[14] Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural networks* 18, 5-6 (2005), 602–610.

[15] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (Nov. 1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[16] Hyeji Kim, Yihan Jiang, Ranvir Rana, Sreeram Kannan, Sewoong Oh, and Pramod Viswanath. 2018. Communication algorithms via deep learning. *arXiv preprint arXiv:1805.09317* (2018).

[17] Thomas Marchioro, Nicola Laurenti, and Deniz Gündüz. 2020. Adversarial Networks for Secure Wireless Communications. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE.

[18] Hesham Mohammed and Dola Saha. 2020. Learning Secured Modulation With Deep Adversarial Neural Networks. In *2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)*. 1–7. https://doi.org/10.1109/VTC2020-Fall49728.2020.9348833

[19] Timothy O'Shea and Jakob Hoydis. 2017. An introduction to deep learning for the physical layer. *IEEE Transactions on Cognitive Communications and Networking* 3, 4 (2017), 563–575.

[20] Maithra Raghu and Eric Schmidt. 2020. A Survey of Deep Learning for Scientific Discovery. arXiv:2003.11755 [cs.LG]

[21] Frank Rubin. 1996. One-time pad cryptography. *Cryptologia* 20, 4 (1996), 359–364.

[22] Claude Elwood Shannon. 1948. A mathematical theory of communication. *Bell system technical journal* 27, 3 (1948), 379–423.

[23] Aaron D Wyner. 1975. The wire-tap channel. *Bell system technical journal* 54, 8 (1975), 1355–1387.

[24] Zhongyuan Zhao, Mehmet C. Vuran, Fujuan Guo, and Stephen D. Scott. 2021. Deep-Waveform: A Learned OFDM Receiver Based on Deep Complex-valued Convolutional Networks. arXiv:1810.07181 [eess.SP]