Deep Learning Models as Moving Targets to Counter Modulation Classification Attacks

Naureen Hoque and Hanif Rahbari
Rochester Institute of Technology, NY, USA
{naureen.hoque, rahbari}@mail.rit.edu

***Abstract*—Malicious entities abuse advanced modulation classification (MC) techniques to launch traffic analysis, selective jamming, evasion, and poison attacks. Recent studies show that current defenses against such attacks are static in nature and vulnerable to persistent adversaries who invest time and resources into learning the defenses, thereby being able to design and execute more sophisticated attacks to circumvent them. In this paper, we present a moving-target defense framework to support a novel modulation-masking mechanism we develop against advanced and persistent MC attacks. The modulated symbols are first masked using small perturbations to make them appear to an adversary in a state of ambiguity about the model as if they are from another modulation scheme. By deploying a pool of deep learning models and perturbation-generating techniques, our defense strategy keeps changing (moving) them as needed, making it difficult (cubic time complexity) for adversaries to keep up with the evolving defense system over time. We show that the overall system performance remains unaffected under our technique. We further demonstrate that, over time, a persistent adversary can learn and eventually circumvent our masking technique, along with other existing defenses, unless a moving target defense approach is adopted.**

***Index Terms*—Moving target defense, modulation classification.**

I. Introduction

Digital modulation in wireless communication is used to convert binary data into analog signals. A transmitter typically deals with varying channel quality by adjusting the modulation (and coding) scheme it uses for transmission. As those signals inherently carry characteristics that reveal the modulation scheme, adversaries, when unable to obtain this information directly from the frame header, use modulation classification (MC) to identify the modulation scheme in order to launch advanced attacks, e.g., traffic analysis [1], selective jamming [2], transmitter fingerprinting [3], and breaching user privacy [4]. Moreover, there is a growing concern regarding adversaries targeting the deep learning (DL) models used in benign MC techniques, and by extension, their wide civil and military applications [5]–[11]. These adversaries require knowledge of the underlying classification parameters to launch disruptive attacks, including poisoning and evasion, by creating adversarial examples [12]–[15]. It is imperative to develop a robust defense against a plethora of MC-based attacks.

This paper is the first to focus on efficiently counteracting the evolving tactics of *persistent* MC attackers. Existing countermeasures that obfuscate the modulation scheme effectively circumvent MC attacks for several hundred modulated symbols without relying on encryption [16], but given sufficient time to analyze long symbol sequences, a persistent adversary can eventually extract the true modulation scheme [17]. Another defense technique involves making controlled perturbations, in the complex value of the symbols, that are designed to mislead the classifier of intruders [18]. However, adversaries can train DL classifiers to bypass this defense and abuse the same concept to further reduce the bit error rate (BER) at the receiver [19]. Other existing defenses are attack-specific (e.g., evasion and poison attacks) using adversarial training, in which the defender trains its own DL model with adversarial examples in advance so that an adversary cannot mislead it [20], [21]. However, as the defender does not usually generate adversarial examples using the same perturbation technique or model as the adversary, the defense is more likely to fail over time when facing persistent adversaries capable of repeated evasion and poison attacks [22]. All of these defense approaches share a common weakness: their static nature becomes a vulnerability over time, as it provides attackers ample opportunity to refine their methods.

To prevent a persistent adversary from patiently probing a static defense mechanism and eventually identifying the hidden modulation scheme, we hypothesize that a dynamic defense approach is necessary. Therefore, in this paper, we aim to answer this question: *how can the principles of a dynamic approach be applied to protect a wireless communication system from modulation classification attacks without sacrificing the overall system performance?* To answer this, we first propose a technique with negligible BER penalty for *masking* the true modulation scheme by adding perturbations to the modulated symbols to make them appear as if they belong to another modulation scheme. A deep learning decision function is used to obtain the amount of perturbations. This effectively misleads the classification model of an adversary, while the receiver's model remains unaffected since it is aware of the amount of perturbation. However, as we will demonstrate, even this mechanism can be compromised over time under a gray-box scenario against a persistent attacker.

Therefore, on top of this base, we develop a moving target defense (MTD) strategy as a proactive defense. The fundamental concept behind MTD is to continuously modify the attack surface by changing the attributes of a system, forcing attackers to frequently restart their learning attempts in case a given technique is not robust against persistent attackers [23].

In our proposed MTD against Modulation Classification Attacks (MTD-MCA) framework, we consider a *pool* of trained

DL models (neural networks of various layers) for the intended receivers to classify the modulation schemes along with several perturbation-generating algorithms we use to generate adversarial examples for our masking technique. The pool is shared securely between the transmitter and receiver once. Each time, a different pair of model and perturbation algorithm is selected for a bounded time to ensure that repeated attacks consistently fail, after which the system moves to the next model and algorithm. Put differently, before a persistent adversary can collect enough masked traffic and meticulously learn the operating learning model, the transmitter will have already switched to an alternative model and perturbation technique. After a certain period, the deployed model pool expires and is seamlessly replaced by a new set of pre-generated models.

To this end, we also design two key components of an MTD-based approach: how and when to move the target (the pair). To do that, we introduce a *Selector* and a *Scheduler* into our framework. Our *Selector* bootstraps by generating multiple random samples from our pool of learning models and perturbation algorithms, evaluating these combinations, and then selecting the best pair based on the aggregated results. Next, the *Scheduler* uses a random walk approach by transitioning probabilities between time steps to select the next time step to move. To communicate the chosen model and the perturbation algorithm to the receiver, each combination is assigned a unique ID, and the transmitter sends this ID over the preamble (for backward compatibility) or a header field, similar to having a Modulation and Coding Scheme (MCS) index in the plaintext header. The ID allows the receiver to uniquely identify the specific combination of the learning model and perturbation algorithm used for the transmitted data. The receiver, being aware of the chosen combination, will be able to discern the true modulation scheme. As a result, it can achieve an equivalent BER performance compared to a system without any defense mechanism in place.

Contributions— Our main contributions are as follows:

- We develop a novel modulation masking technique that employs an adversarial examples approach for perturbation generation, concealing the true modulation scheme of transmitted symbols to appear as a different but specified modulation scheme. Our rigorous evaluations through simulation demonstrate that it maintains the same BER as a system lacking any defense mechanism under a wide range of signal-to-noise ratio (SNR) levels.
- We design MTD-MCA on top of our perturbation-based masking technique to prevent persistent attackers from learning and circumventing the masking defense by continuously modifying system attributes to keep adversaries at bay. By treating the decision function of learning models and perturbation-generating algorithms as moving targets, MTD-MCA significantly increases the complexity for persistent adversaries (now cubic in time) to learn system patterns and launch successful attacks.
- Our evaluation results further show that MTD-MCA significantly mitigates the effectiveness of modulation classification attacks, with the strongest attacker achieving no more than approximately 40% success rate.

Paper Outline— We provide the preliminaries to understand the rest of the paper in Section II and the system model in Section III. Sections IV and V discuss and evaluate our proposed defense, respectively. We discuss related work in Section VI before concluding the paper in Section VII.

II. Preliminaries

For a coherent discussion, we first provide a brief overview of modulation classification, adversarial examples in DL, and the principles of moving target defense. Table I lists the important notations with their meanings for this section.

A. Modulation Classification

In a system where the receiver cannot use the conventional means of learning about the modulation type (e.g., when the headers are encrypted or undecodable, the communication protocol is unknown (like in a war zone), or there is no such field in the header), it applies MC to identify the modulation scheme. Let X denote the received modulated signal samples defined as $X = (X_i, y_i) : i \in \{1, \dots N\}$, where X_i represents a series of d symbols and $y_i \in \{1, \dots m_c\}$ denotes its modulation scheme (m_c is the dimension of the output space, i.e., the number of classes/modulation schemes in the system).

B. Adversarial Examples

Adversarial examples are specially crafted signals designed to cause an ML/DL system to make incorrect decisions or behave unexpectedly. These adversarial examples can lead to communication errors, interference, and compromise of the integrity, confidentiality, and availability in wireless networks and their associated devices [12]–[15]. In this paper, we utilize adversarial examples for a different purpose—to mask the true modulation scheme to defend against MC attacks.

In the MC domain, adversarial examples are generated by introducing perturbations to the modulated symbols. The amount of perturbation is influenced by the specific characteristics of the classification model and the perturbation-generating algorithm (see below). When training the model by updating its parameters, denoted by θ, the primary objective is to minimize the expected loss[1] $L(.)$ across all (X_i, y_i) pairs. This objective is mathematically expressed as:

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L(\theta, X_i, y_i). \tag{1}$$

The optimal solution to this loss minimization problem generally leverages an optimization algorithm (e.g., stochastic gradient descent (SGD)). It iteratively updates the model parameters θ following the gradient of the loss with learning rate η and batch size b, formally denoted as ∇_θ, as follows:

$$\theta_t = \theta_{t-1} - \nabla_{\theta_{t-1}} \cdot \frac{\eta}{b} \sum_{i=1}^{b} L(\theta_{t-1}, X_i, y_i). \tag{2}$$

[1]The loss function measures the discrepancy between predicted outputs and the actual labels in the training data to minimize the classification error.

The objective of the learning model at the receiver is to perform the mapping $f_\theta : X \to y$. The output of f_θ for each X_i is an m_c-dimensional vector, and each dimension represents the likelihood of input belonging to the corresponding modulation scheme. Suppose $d = 1$. The decision function $f_\theta(x_i)$ maps an input symbol $x_i \in X$ to a class label y_i that exhibits the highest likelihood. Then, $x_i' = x_i + \delta$ is called an adversarial example with perturbation δ if: $f_\theta(x_i') = y_i' \neq y_{\text{true}}$ and $||\delta|| < \delta_{max}$, where y_{true} is the ground truth, $||.||$ is a distance metric, and δ_{max} is the maximum allowable perturbation that preserves the semantic integrity of x. Semantic integrity is domain- and/or task-specific. In our problem, each x_i' must appear to belong to a modulation class that the system supports. Among many existing perturbation-generating algorithms, the following are a few that we use and customize in this paper.

1) Projected Gradient Descent (PGD): This algorithm is an iterative optimization method used to generate adversarial examples [24]. In our problem, it applies small step-wise perturbations to the input signal in the direction that maximizes the model's loss, aiming to find a perturbation δ required to cause misclassification.

$$\delta^{(t)} = \text{Clip}_{\delta_{max}} \left(\delta^{(t-1)} + \alpha \cdot \text{sign}(\nabla_\theta L(\theta, x_i, y_{\text{true}})) \right) \quad (3)$$

where α is the step size, t is the iteration index, $\text{Clip}_{\delta_{max}}$ is a function that clips the perturbation to ensure it stays within the allowed maximum perturbation budget, and $\nabla_\theta L(\theta, X, y_{\text{true}})$ is the gradient of the loss with respect to the model's parameters.

2) DeepFool: DeepFool is an adversarial attack algorithm that finds adversarial perturbations by linearizing the decision boundary of the classification model [25]. It iteratively moves the input signal in the direction of the nearest linear boundary of the model until misclassification is achieved. The δ generation for DeepFool can be approximated as follows:

$$\delta^{(t)} = -\frac{f_\theta(x_i^{(t)})}{||\nabla_{x_i} f_\theta(x_i^{(t)})||_2^2} \nabla_{x_i} f_\theta(x_i^{(t)}) \quad (4)$$

where $||\delta||_2$ denotes the L_2 norm of the perturbation.

3) Momentum Iterative Method (MIM): It is a variant of the PGD algorithm that introduces momentum into the perturbation update process [26]. It accumulates past perturbations to have a smoother update direction, which can lead to faster convergence. The δ generation for MIM can be expressed as:

$$\begin{aligned} r^{(t)} &= \mu \cdot r^{(t-1)} + \frac{\nabla_\theta L(\theta, x_i, y_{\text{true}})}{||\nabla_\theta L(\theta, x_i, y_{\text{true}})||_1} \\ \delta^{(t)} &= \text{Clip}_{\delta_{max}} \left(\delta^{(t-1)} + \alpha \cdot \text{sign}(r^{(t)}) \right) \end{aligned} \quad (5)$$

where r is the momentum and μ is the momentum factor.

4) Basic Iterative Method (BIM): This is another variant of the PGD algorithm that performs multiple iterations of small perturbations to generate adversarial examples [27]. It aims to increase the confidence of misclassification over iterations. The δ generation for BIM can be expressed as:

TABLE I
IMPORTANT NOTATIONS USED IN THE PRELIMINARIES.

Notation	Definition
X	Batch of modulated signals X_i
d	Number of symbols in X_i
N	Total number of labeled batches X_i
$f(.)$	Learning model's decision function
y	Predicted modulation scheme (label) of the signal X_i by $f(.)$
m_c	Number of modulation schemes supported by the system
θ	Learning model parameters (decision boundary)
t	Iteration index
η	Learning rate
$L(.)$	Loss function
x'	Adversarial example of input signal x
y'	Predicted label of x' by $f(.)$
δ	Perturbation amount
δ_{max}	Maximum allowable perturbation
α	Step size for perturbation algorithms

$$\delta^{(t)} = \text{Clip}_{\delta_{max}} \left(\delta^{(t-1)} + \alpha \cdot \text{sign}(\nabla_\theta L(\theta, x_i + \delta^{(t-1)}, y_{\text{true}})) \right) \quad (6)$$

5) NewtonFool: The NewtonFool method perturbs an input sample to cross the classifier's decision boundary, thereby altering the classifier's output [28]. Each iteration updates the input sample X by adding a small perturbation δ, which is aligned with the direction of the classifier's decision boundary at that sample:

$$\delta^{(t)} = \delta^{(t-1)} - \eta \cdot \frac{\nabla_{\delta^{(t-1)}} L(f(\delta^{(t-1)}, y_{\text{true}}))}{||\nabla_{\delta^{(t-1)}} L(f(\delta^{(t-1)}, y_{\text{true}}))||_2^2} \quad (7)$$

C. Moving Target Defense (MTD)

MTD is a proactive and dynamic defense strategy that aims at increasing the adversary's uncertainty and effort by dynamically changing (moving) the attack surface. Hence, this strategy makes the adversary unable to take their time to learn about and eventually break the target system. MTD was originally developed for the detection and prevention of cyber-intrusion attacks as a part of the Intelligence Driven Defense® model [29], but it is also applied in other domains, such as denial-of-service (DoS) attacks [30] and data exfiltration [31].

MTD has three key elements that determine the strategy and implementation of this technique: *what* to move, *when* to move, and *how* to move [32]–[35]. *What* refers to the elements of the system that will be moved in response to a detected threat before it occurs, such as IP addresses, port assignments, file system layouts, virtual machines, software components, etc. [32], [36]. *When* refers to the timing and frequency of the changes to the elements of the system. These changes may be triggered in response to a detected threat, or they may be executed on a predetermined schedule to proactively prevent attacks. Finally, *how* indicates the specific techniques to move the target elements. The choice of what, when, and how to move in an MTD framework will depend on the specific needs and requirements of the problem space and the nature of the security threats. An effective MTD framework must be able to adapt to changing threats and make decisions about what, when, and how to move in real time, to provide a dynamic and effective cyber defense.

III. System & Adversary Model

We consider a typical wireless communication system comprising a transmitter and a receiver that adapt their transmission rate based on the quality of the channel. The receiver may train a DL classifier to identify the modulation scheme of incoming signals more efficiently than alternative methods.

In this system, we assume the adversary operates from a gray-box perspective. This implies that the adversary might be aware of the general system configurations (e.g., protocol and supported modulation schemes) and the defense mechanisms (e.g., the overall strategy and algorithms). However, they are not privy to the secrets and specific details of the MTD strategies, such as the exact timing and nature of the changes in system configurations or classification parameters. We assume the persistent adversary to be tenacious and adaptive, demonstrating a willingness to invest significant effort in understanding the configurations and parameters while continuously probing the defense mechanisms.

The adversary's primary goal is to accurately identify the true modulation scheme. To achieve this, it can create a substitute learning model that meticulously mirrors the behavior of the actual system. This is accomplished by observing the system's traffic and iteratively refining the substitute model. This process involves multiple attempts and interactions with the system to gather information and adapt its model over time.

IV. Proposed Moving Target Defense against Modulation Classification Attacks

We first propose our modulation-masking technique to conceal the true modulation scheme from adversaries. We achieve this by adding perturbations to the modulated symbols, making them appear as if they belong to another modulation scheme. For the simplicity of exposition, we begin with a system involving a single model and a single perturbation algorithm, which we call fixed target defense against MC attacks (FTD-MCA). Next, we build upon this foundation by integrating our moving target defense strategy, leading to the development of our proposed framework, MTD-MCA. Table II lists the important notations used in this section.

A. Fixed Target Defense against MC Attacks (FTD-MCA)

To hide the true modulation scheme from an unintended receiver, we mask modulated symbols by adding perturbations obtained using a learning decision function. This involves generating a vector in the input feature space (the IQ values) to effectively mislead the adversary's classification model in a state of ambiguity about that learning model.

Unlike other domains (e.g., image classification), where each element might have a different label, in our context, all $d > 1$ symbols in X_i (e.g., symbols in a single wireless frame) are modulated using the same scheme. Accordingly, we customize the perturbation generation methods (3)-(7) by replacing x_i with X_i. This modification ensures that the perturbations are consistent with the modulation scheme of the entire signal, rather than varying across different symbols.

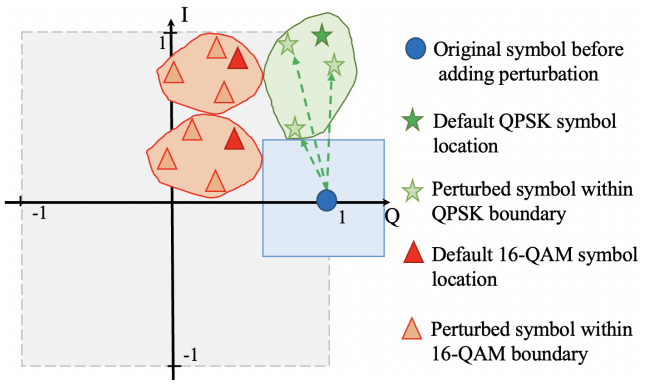


Fig. 1. Illustrative example of a targeted perturbation using PGD, where multiple BPSK symbols at (1,0) are perturbed to fall within a model-specific QPSK decision boundary, outside that of 16-QAM symbols, where minimum and maximum I/Q perturbation budgets (δ_{max}) are set based on that boundary.

We also modify the perturbation generation methods to allow all of the symbols in X_i masked by a specific, higher modulation scheme, denoted by y_{sp}. The perturbation amount is based on the distance between the model's decision boundary (θ_{tsp}) and that of the true modulation scheme, y_{true}, and the specified one (y_{sp}). In the following, we show the modified equation of PGD. Likewise, we modify the rest of the (3)-(7).

$$\delta^{(t)} = \text{Clip}_{\delta\text{max}}\left(\delta^{(t-1)} + \alpha \cdot \text{sign}\left(\nabla_{\theta_{tsp}} L(\theta_{tsp}, X_i, y_{\text{true}})\right)\right) \tag{8}$$

Fig. 1 illustrates an example of our strategy. Suppose y_{true} is BPSK and the system supports two more modulation schemes, namely, QPSK and 16-QAM. For a BPSK symbol originally at $(1, 0)$, targeted perturbation is applied to steer it towards the y_{sp} model's decision boundary for y_{sp}, that is QPSK in this example. Without targeting, the perturbation could shift the symbol towards either a QPSK or 16-QAM demodulation decision boundary within the clipped range of $[-1, 1]$ on both axes. The receiver first addresses noise or channel impact elimination, then reverses the known perturbation, and subsequently demodulates the BPSK symbols in a manner consistent with regular system operations under any given SNR level.

B. Moving Target Defense against MC Attacks (MTD-MCA)

Our proposed framework contains three key components: *what*, *how*, and *when*. The *what* component represents the target that MTD-MCA moves, and it is a tuple τ (f, p), where f is the model's decision function and p is a perturbation algorithm. We consider a pool of models (that can include any type of layers, such as transformers, convolutional, fusion, etc.) and a pool of perturbation-generating algorithms such as the five algorithms defined above. Our *how* component is the *Selector*, which decides the target using randomization. The third component is the *Scheduler*, which decides *when* to move the target. This dynamic approach introduces unpredictability into the timing of target movements, enhancing the effectiveness of the MTD strategy by making it difficult for potential adversaries to anticipate when the target will be switched.

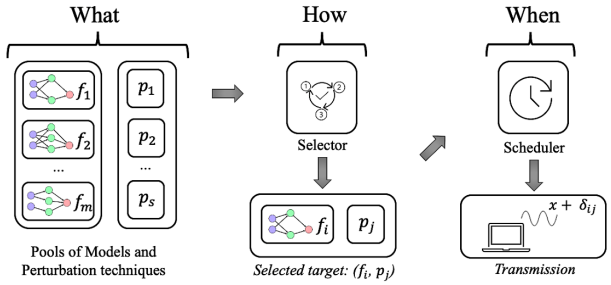


Fig. 2. Proposed MTD-MCA framework: what, how, and when to move.

1) Communicating the operating target: In the proposed system, a table containing the combinations of models and perturbation algorithms is generated by a *Selector* (discussed in the following subsection IV-B3), where each combination is assigned a unique ID. The decision functions of the combinations are securely pre-shared with the transmitter and receiver. The transmitter uses the plaintext ID of the chosen target and embeds it in the preamble using, e.g., the embedding technique in [37] (for backwards compatibility) or in the frame headers. This approach is similar to indicating a Modulation and Coding Scheme (MCS) index in the physical-layer header.

In our case, the transmitter needs to communicate the unique ID of the specific combination used for a given frame to the receiver. For m models and s perturbation algorithms, the systems need $s \times m$ number of unique IDs. For the number of bits needed to represent the combination ID, we need $\lceil \log_2(s \times m) \rceil$ bits. It allows creating pools consisting of 2^4 to 2^{16} combinations. By regularly renewing the pools with totally new sets of models and perturbation algorithms, the system gains an inherent level of uncertainty, making it much more difficult for an adversary to associate any ID with a pattern.

2) What to move: Instead of using a single fixed target model, the MTD-MCA framework utilizes a pool of m models $(f_1, f_2, ..., f_m)$ and s perturbation-generating algorithms $(p_1, p_2, ..., p_s)$. MTD-MCA randomly selects a model and a perturbation-generating algorithm pair from the pool to perturb the modulated symbols and conceal the true modulation scheme before transmission. The receiver identifies the chosen f_i and p_j using the ID and uses it to remove the perturbations applied by the transmitter. This step enables the retrieval of the true modulation scheme of the received symbols, facilitating the subsequent demodulation process.

3) How to move: The *Selector* operates within two subpools: one containing various models, and the other containing perturbation algorithms. It employs bootstrapping techniques, a resampling method widely used in statistics and machine learning, to make well-informed decisions.

Using bootstrapping, the *Selector* creates multiple random samples from the pools of learning models, denoted by $\mathcal{F}$ (f represents a specific learning model), and perturbation algorithms, denoted by $\mathcal{P}$ (p represents a specific perturbation algorithm). Let S_i denote the ith sample, where each sample consists of a combination of learning models and perturbation algorithms:

$$S_i = \{(f_{i1}, p_{i1}), (f_{i2}, p_{i2}), ..., (f_{ij}, p_{ij})\} \tag{9}$$

where (f_{ij}, p_{ij}) represents the jth combination of a learning model f_{ij} and a perturbation algorithm p_{ij} in the ith sample.

The *Selector* then evaluates the performance of the system using these combinations. Let $\Psi(S_i)$ represent the performance metric obtained from the ith sample S_i. The goal is to identify the combination that demonstrates the highest overall performance or meets specified optimization criteria. This can be achieved by aggregating and analyzing the performance metrics obtained from each sample:

$$\Psi(S) = \Psi(S_1), \Psi(S_2), ..., \Psi(S_n) \tag{10}$$

where $\Psi(S)$ is the set of performance metrics from all n samples. The final step is to select the best-performing combination based on the aggregated performance metrics. Let S^* represent the sample with the highest overall performance:

$$S^* = \arg\max_{S_i \in S} \Psi(S_i) \tag{11}$$

The combination in S^*, denoted as (f^*, p^*), is the chosen target for the system. The steps are outlined in Algorithm 1. The overall time complexity of the algorithm is $\mathcal{O}(n)$.

4) When to move: The scheduler operates using discrete time steps represented as $T = t_1, t_2, \ldots, t_n$, where n is the number of time steps. A transition probability matrix Pr defines the likelihood of transitioning from one time step t_i to another t_j in a random walk scenario, subject to the condition that the probabilities satisfy $\sum Pr(i, j) = 1, \forall i$. The scheduler's decision-making process involves a current time step t_{cur}, initially set to t_1, and a random selection procedure based on transition probabilities to determine the next time step t_{next}. To achieve this, the scheduler randomly selects a time step t_j from T using $Pr(t_{\text{cur}}, t_j)$. The random selection process is repeated for M iterations (i.e., time steps), leading the scheduler to a specific time step t_k. Subsequently, the target's movement is scheduled to occur after M frames. The steps are outlined in Algorithm 2. The overall time complexity of the algorithm, dominated by the for-loop, is $\mathcal{O}(M)$.

TABLE II
NOTATIONS USED TO FORMULATE THE PROPOSED SCHEME.

Notation	Definition
τ	Tuple representing the target in MTD-MCA.
$\mathcal{F}$	Pool of learning models.
$\mathcal{P}$	Pool of perturbation-generating algorithms.
f_i	Learning model from $\mathcal{F}$.
p_i	Perturbation-generating algorithm from $\mathcal{P}$.
$\Psi(S_i)$	Performance metric from the i-th sample.
S_i	i-th sample with learning model and perturbation.
S^*	Chosen target with the highest performance.
T	Set of discrete time steps $\{t_1, t_2, \ldots, t_n\}$.
Pr	Transition probability matrix for the scheduler.
t_{cur}	Current time step in the scheduler's decision-making.
t_{next}	Next time step randomly selected by the scheduler.
t_k	Time step when the target's movement is scheduled.

Algorithm 1: Selector with Bootstrapping

Input: $\mathcal{F}$, $\mathcal{P}$
Output: (f^*, p^*)
Initialize: $S_0 \leftarrow \emptyset$
for $i \leftarrow 1$ **to** n **do**
  Randomly select (f_{ij}, p_{ij}) from $\mathcal{F} \times \mathcal{P}$ to form S_i;
end
for $i \leftarrow 1$ **to** n **do**
  Evaluate $\Psi(S_i)$ for each sample S_i;
end
$S^* \leftarrow \arg\max_{S_i \in S} \Psi(S_i)$;

Algorithm 2: Scheduler with Random Walk

Input: T, Pr
Output: t_k
Initialize: $t_{cur} \leftarrow t_1$
for $i \leftarrow 1$ **to** M **do**
  Randomly select t_{next} from T using $Pr(t_{\text{cur}}, t_{\text{next}})$;
  $t_{cur} \leftarrow t_{next}$;
end
$t_k \leftarrow t_{cur}$;

C. Computational Complexity: Defender versus Attacker

In FTD-MCA or any fixed target defense mechanism, let's denote the attacker's complexity to compromise the system by $\mathcal{O}(t_a)$ (t_a is defined as the amount of time for an adversary to break a system with single, fixed defense). Conversely, our MTD-MCA approach significantly increases this complexity by dynamically altering both the decision function and the perturbation algorithm across m models and s perturbation algorithms at discrete intervals, leading to a non-linear increase in the attacker's adaptation complexity. The time complexity for the attacker in MTD-MCA is approximately $\mathcal{O}(m \times s \times t_a)$. This elevates the time complexity to *cubic*, substantially higher than the linear complexity of $\mathcal{O}(t_a)$ in MTD-MCA, prolonging the time required for an adversary to adapt effectively, thereby enhancing the system's security.

V. Performance Evaluation

We evaluate the performance of the proposed modulation masking technique and assess the defense strength of FTD-MCA and MTD-MCA when facing a persistent attacker. We implemented our framework, including the *Selector* and *Scheduler*, the neural network models, and the perturbation-generating algorithms defined in Section IV using Keras as the front-end and TensorFlow as the back-end in Python.

A. Dataset & Metrics

The data is generated using MATLAB Communications Toolbox™ and includes four standard baseband modulation schemes: BPSK, QPSK, 16-QAM, and 64-QAM. This dataset contains a balanced set of $500,000$ signals. It is then divided into two halves, one for evaluating the proposed technique by a defender and the other for the adversary, enabling a defense assessment analysis. To make sure neither dataset is biased against or in favor of the defender/adversary, we switch the datasets and achieve comparable results.

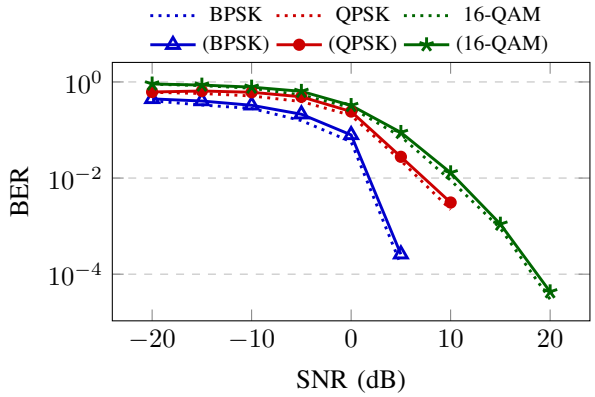


Fig. 3. BER performance of different modulation schemes under various SNR levels, where (x) represents a masked modulation scheme as 64-QAM. The results demonstrate that the BER is nearly the same as the system without any masking (shown as dotted lines).

To evaluate the proposed modulation masking techniques proposed in Section IV, we utilize the defender's dataset to generate modulation-masked symbols. These masked symbols, along with the original modulated symbols, are transmitted over an additive white Gaussian noise (AWGN) channel with varying SNR levels ranging from -20 dB to 20 dB. The attacker dataset undergoes the same process. For reliable evaluation, each experiment is repeated $1,000$ times, and the results presented here are the averages of those repetitions.

Both datasets are further divided into training (70%), validation (10%), and testing (20%) set to accurately train and validate the proposed techniques. We assess the performances based on BER under various SNR levels, and collected metrics including accuracy, loss, F1-score, and confusion matrices.

B. Models & Perturbation-generating Algorithms

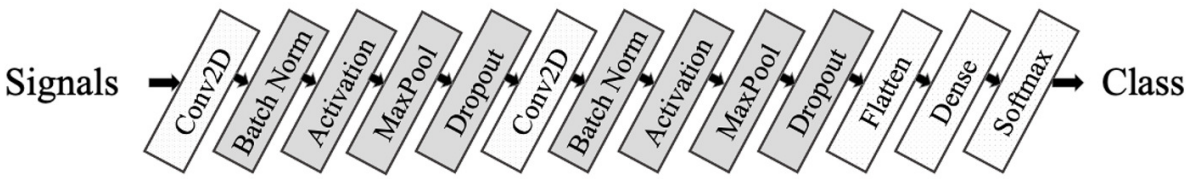
To evaluate the performance of our proposed system, we begin by creating a diverse pool of convolutional neural network (CNN) models and perturbation algorithms. Hyperparameter tuning is employed to identify the top-performing models based on their performance metrics to select the models for this pool. The size of the model pool can be of any number. To select and train multiple models cost-effectively, knowledge distillation[2] [38] can be applied as well. After the tuning process, we select the five best-performing models to evaluate the MTD-MCA's assessment against repeated attacks (details in the subsections V-C and V-D). The pool of these five models and five perturbation techniques are presented in Table III. The table outlines the key characteristics of each model, providing an overview of their architectures and configurations. Our model takes the received symbols as input and returns the class (i.e., modulation scheme).

[2]A technique where a complex "teacher" model trains simpler "student" models to replicate its performance efficiently at a much lower training cost.

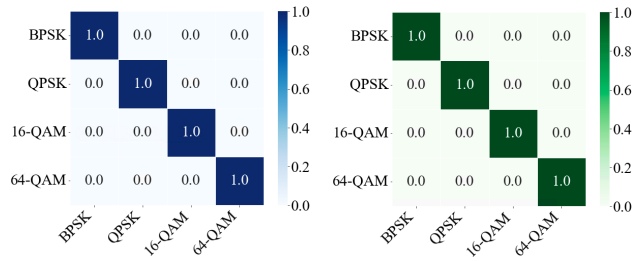
TABLE III
THE POOL OF MODELS AND PERTURBATION-GENERATING ALGORITHMS WE CONSIDERED IN THE USE-CASE.

Model Pool					
Parameters	f_1	f_2	f_3	f_4	f_5
Optimizer	RMSprop	SGD	SGD	SGD	SGD
Learning Rate	0.1	0.1	0.1	0.01	0.01
Momentum	0.1	0.2	0.2	0.6	0.6
Batch size	32	32	32	32	32
Activation	LReLU	ELU	ELU	ELU	ELU
Filter number	$24, 58$	$80, 118$	$22, 36$	$64, 26$	$30, 68$
Kernel size	$2, 8$	$4, 6$	$2, 10$	$2, 4$	$4, 2$
MaxPool size	$6, 2$	$6, 4$	$6, 4$	$4, 2$	$6, 6$
Dropout rate	$0.2, 0.4$	$0.1, 0.1$	$0.2, 0.1$	$0.3, 0.4$	$0, 0$
Trainable param.	29810	113726	10866	19874	11962

Main Structure of the Models in the Pool



Perturbation Generator Pool					
Perturbation	p_1	p_2	p_3	p_4	p_5
Algorithms	PGD	DeepFool	MIM	BIM	NewtonFool



(a) Without any masking. (b) Masked as 64-QAM.

Fig. 4. Confusion matrix comparison between a system without any masking applied with the proposed MTD-MCA. Despite the true modulation schemes being masked as QPSK, the receiver effectively removed the amount of perturbation and accurately identified the correct modulation scheme.

C. System Performance at the Intended Receiver

We now investigate the impact of masking on the modulation classification accuracy and BER performance under various SNR levels. The results showcase the effectiveness of the proposed work by comparing the system's performance without any masking against the results achieved with our proposed technique.

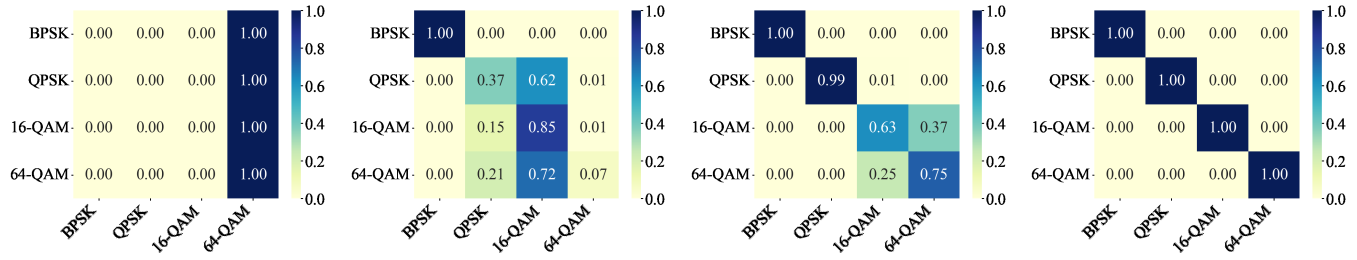
Fig. 3 highlights BER across three modulation schemes: BPSK, QPSK, and 16-QAM. As the SNR increases, the BER shows a decrease for all modulation schemes and closely approaches the performance of the system without any masking, as indicated by the dotted lines.

Fig. 4 showcases a comparison of confusion matrices between two systems: one without any masking applied and the other with the proposed MTD-MCA. The confusion matrix highlights the effectiveness of the receiver in accurately identifying the correct modulation scheme, even when the true modulation schemes were masked as 64-QAM. Despite the presence of perturbations, the receiver efficiently removed them, leading to precise and reliable modulation scheme identification. This result demonstrates the potential of the MTD-MCA approach in enhancing modulation recognition in wireless communication systems.

D. Defense Performance against Attacks

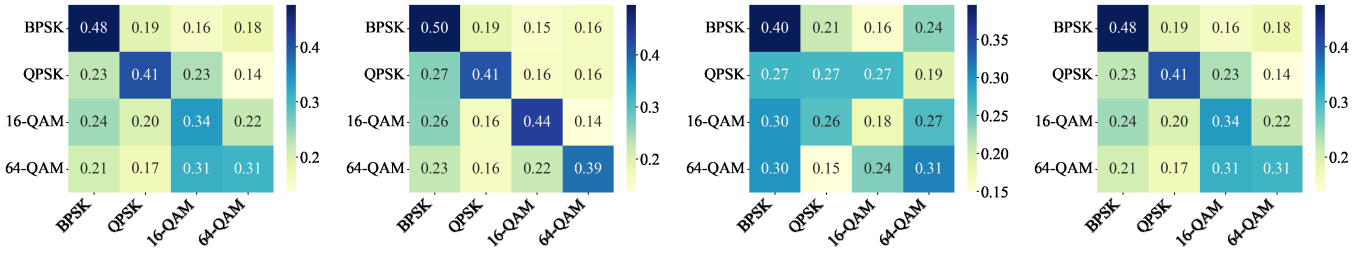
For brevity, we focus on a specific use case where the defender's pool consists of five models and five perturbation algorithms in the MTD-MCA scenario. The persistent attacker's objective is to target both the FTD-MCA and MTD-MCA systems with repeated attempts, carefully analyzing the outcome of each attempt to refine their attack approach. In this assessment, multiple attack attempts, coupled with the various models and perturbation algorithms considered in both attack scenarios, make the assessment representative and reflective of a wide range of potential scenarios, making it applicable to generic use cases.

1) Attack Attempts against FTD-MCA: For the FTD-MCA assessment, we consider every combination of five models and



(a) First attack attempt. (b) Intermediate attack attempt. (c) Continued attack attempt. (d) Final attack attempt.

Fig. 5. Persistent adversary's attack performance over time in identifying the true modulation scheme under FTD-MCA.



(a) Initial attack attempt. (b) Intermediate (but stagnant) attempt. (c) Continued (but stagnant) attempt. (d) Final (and still stagnant) attempt.

Fig. 6. Persistent adversary's attack performance over attempts in identifying the true modulation scheme under MTD-MCA.

the five perturbation algorithms (f_i and p_j) from Table III and report the average scores in Fig. 7.

For the first attempt against the FTD-MCA system, the attacker selects a long short-term memory (LSTM) model, leveraging its ability to process entire sequences of data at once. Unfortunately for the attacker, this initial attempt, as shown in Fig. 5(a), turns out to be a complete failure.

Analyzing the results, the persistent attacker decides to change their strategy entirely instead of refining the same model. For the next attempt, they opt for a one-layer two-dimensional CNN model. One of the outcomes of this intermediate attempt (Fig. 5(b)) shows some progress for the attacker, motivating them to refine their models further. They add another layer of convolution to improve their approach. One of the continuations of the attempts, presented in Fig. 5(c), reflects even more success for the attacker. However, despite the two-layer convolutional model, the attacker observes that the performance is still not satisfactory. As a result, they decide to refine their approach once again, this time by considering adversarial examples and incorporating them during model training. In the final attempt, illustrated in Fig. 5(d), the persistent attacker achieves 100% success in their attacks against the FTD-MCA system. The iterative refinement process and the incorporation of adversarial examples have allowed the attacker to overcome the defenses and successfully compromise the system. Throughout this assessment, the attacker's persistent approach, coupled with their adaptability in changing model architectures and refining strategies, demonstrates the importance of robust and effective defense mechanisms to thwart persistent attackers.

2) Attack Attempts against MTD-MCA: For this assessment, we consider every combination of five models and the five perturbation algorithms (f_i and p_j) from Table III and report the average scores in Fig. 7. For illustration purposes, we provide details of one use case in the following.

In the assessment of the MTD-MCA system, we consider that the persistent attacker initiates their attempt with a two-dimensional CNN equipped with a single convolutional layer. The attacker collected data to attack the MTD-MCA system when its target was the (f_5, p_2) pair. As depicted in Fig. 6(a), the initial attack achieves only around 38% success in overall accuracy. Analyzing the results, the attacker decides to enhance their approach by considering the use of two convolutional layers. However, during this time, the MTD-MCA system shifts its target to the (f_4, p_5) pair. In the subsequent attempt, shown in Fig. 6(b), the attacker's success rate remains approximately 40% in overall accuracy. Before the next attack, the attacker incorporates adversarial examples and trains their model accordingly. By the time they refine their model for the third attack, the MTD-MCA system's target has shifted to the (f_3, p_4) pair. The attacker continues refining their approach, adding a third layer of convolution, and training with adversarial examples. In the fourth attempt with (f_2, p_3) pair, depicted in Fig. 6(d), the attacker's success rate remains at approximately 40%. Throughout these attempts, the attacker's accuracy does not exceed 40%, showcasing the efficacy of our MTD-MCA approach in defending against persistent attacks. The MTD-MCA system effectively adapts its target features and perturbation strategies, making it challenging for the attacker to devise a successful and consistent attack.

We compare the defense strength of both systems in terms of attack performance, specifically focusing on accuracy and F1 score. The results are depicted in Fig.7. While FTD-MCA's accuracy and F1-score gradually degrade due to its static nature, MTD-MCA's dynamic adaptation proves to be highly effective in mitigating the threat. Since the adversary fails to

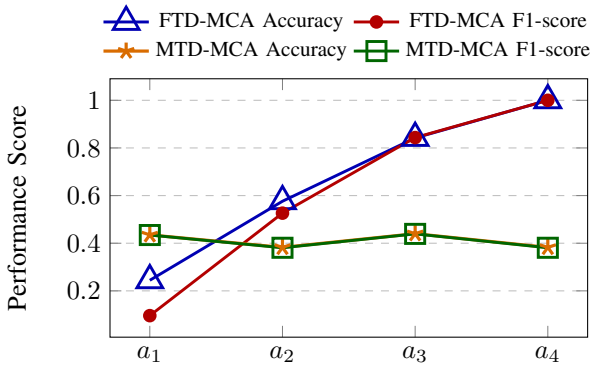


Fig. 7. Attack performance– initial (a_1), intermediate (a_2), continued (a_3), and final (a_4) attempts.

discern the underlying classification parameters of the system, they are unable to launch other disruptive attacks such as poisoning and evasion attacks.

VI. Related Work

A. Defenses against Modulation Classification Attacks

To prevent an adversary from identifying the modulation order, some works proposed modulation concealment techniques [16], [39] to circumvent MC without hampering the quality of communication with its intended receiver.

In modulation obfuscation, symbols are camouflaged in the constellation map of the highest-order modulation scheme to conceal the true modulation order using a secret sequence known only to the intended receiver [16], making statistical MC approaches ineffective for MC and preventing traffic analysis attacks [40]. However, as indicated in [17], persistent adversaries can eventually discern the true modulation scheme. If we use this approach as the basis to apply MTD by alternating between two or more secret sequences, it may delay adversary learning, it ultimately does not prevent it due to the underlying same obfuscation method, allowing adversaries to adapt with more complex models like long learning memory systems (LLMS) to extract the unique patterns of each modulation scheme over longer series of symbols.

In [18], minimum perturbations are added to the transmitted signal to reduce classification accuracy at the intruder while maintaining a low BER at the intended receiver with an assumption that the transmitter knows the intruder's model, channel, and location. However, the authors in [19] show that adversaries can counter this defense over time. If we use MTD on top of this approach by varying perturbation amounts, it will primarily increase the computational load without significantly hindering an adversary's adaptability because the perturbations must remain minimal to maintain a low BER at the receiver.

B. Adversarial Examples in Wireless Communications

Adversarial examples pose significant challenges in wireless communications, as demonstrated by several studies. Davaslioglu *et al.* introduced a Trojan attack in a wireless signal classifying system, where a slight manipulation in the training data inserts Trojans by changing sample labels to a target label [13]. Bahramali *et al.* utilized universal adversarial perturbations (UAP) in a black-box scenario to generate perturbations on a proxy model and then used them to attack the target classification model [41]. Meanwhile, Shi *et al.* proposed a spectrum data poisoning attack, enabling the adversary to infer the transmitter's activity [42].

C. Moving Target Defense Approaches

Multiple research papers have highlighted MTD as a proactive defense strategy in different domains. For instance, Zhang *et al.* [43] demonstrated MTD's ability to counter false data injection attacks in power grids by dynamically adjusting branch susceptances to minimize attack opportunities. The paper [44] introduced a secure hardware architecture that employed ensembles of moving target defenses and churn to continuously obfuscate sensitive information, such as code and pointer representations, effectively thwarting control-flow attacks. In [36], a MTD strategy was applied to enhance the security of deep learning-based visual sensing against adversarial example attacks by generating multiple new deep models after deployment.

VII. Conclusion

In this paper, we tackled the challenge of defending wireless communication systems against MC attacks through a proactive and dynamic defense strategy following MTD principles. Our proposed strategy, MTD-MCA, includes a modulation-masking technique using adversarial examples to conceal the true modulation scheme. The defense continuously modifies system attributes, treating the decision function and perturbation algorithms as moving targets, making it computationally expensive (cubic time complexity) for persistent attackers to learn system patterns and launch successful attacks. Extensive simulations demonstrate that MTD-MCA effectively maintains the same BER as a system without defense mechanisms across varying SNR and significantly mitigates the effectiveness of MC attacks, with the strongest attack achieving no more than 40% success rate.

For future work, we aim to explore other channel models and conduct over-the-air experiments to gain insights into the performance and robustness of the MTD-MCA defense strategy in real-world scenarios.

Acknowledgment

This work is supported in part by the ESL Global Cybersecurity Institute at Rochester Institute of Technology.

References

[1] J. S. Atkinson, O. Adetoye, M. Rio, J. E. Mitchell, and G. Matich, "Your WiFi is leaking: Inferring user behaviour, encryption irrelevant," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Shanghai, China, Apr. 2013, pp. 1097–1102.

[2] G. Noubir, R. Rajaraman, B. Sheng, and B. Thapa, "On the robustness of IEEE 802.11 rate adaptation algorithms against smart jamming," in *Proc. ACM Conf. Secur. Privacy Wireless Mobile Netw. (WiSec)*, Hamburg, Germany, Jun. 2011, pp. 97–108.

[3] W. Xiong, P. Bogdanov, and M. Zheleva, "Robust and efficient modulation recognition based on local sequential IQ features," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Paris, France, Apr. 2019, pp. 1612–1620.

[4] Y. Shi, K. Davaslioglu, and Y. E. Sagduyu, "Over-the-air membership inference attacks as privacy threats for deep learning-based wireless signal classifiers," in *Proc. ACM Workshop Wireless Secur. Mach. Learn. (WiseML)*, Virtual, Jul. 2020, pp. 61–66.

[5] S. Hanna, C. Dick, and D. Cabric, "Signal processing-based deep learning for blind symbol decoding and modulation classification," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 1, pp. 82–96, Jan. 2022.

[6] C. Hou, G. Liu, Q. Tian *et al.*, "Multisignal modulation classification using sliding window detection and complex convolutional network in frequency domain," *IEEE Internet of Things J.*, vol. 9, no. 19, pp. 19 438–19 449, 2022.

[7] A. P. Hermawan, R. R. Ginanjar, D.-S. Kim, and J.-M. Lee, "CNN-based automatic modulation classification for beyond 5G communications," *IEEE Commun. Letters*, vol. 24, no. 5, pp. 1038–1041, 2020.

[8] J. Pawlak, Y. Li, J. Price *et al.*, "A machine learning approach for detecting and classifying jamming attacks against UAVs," in *Proc. ACM Workshop Wireless Secur. Mach. Learn. (WiseML)*, Abu Dhabi, UAE, Jun. 2021.

[9] E. Perenda, S. Rajendran, G. Bovet, S. Pollin, and M. Zheleva, "Learning the unknown: Improving modulation classification performance in unseen scenarios," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Virtual, May 2021.

[10] A. Smith, M. Evans, and J. Downey, "Modulation classification of satellite communication signals using cumulants and neural networks," in *Cogn. Commun. Aerosp. Appl. Workshop (CCAA)*, Cleveland, USA, Jun. 2017.

[11] T. D. Hoang, C. Park, M. Son *et al.*, "LTESniffer: An open-source LTE downlink/uplink eavesdropper," in *Proc. ACM Conf. Secur. Privacy Wireless Mobile Netw. (WiSec)*, Guildford, UK, May 2023, pp. 43–48.

[12] Y. Lin, H. Zhao, Y. Tu, S. Mao, and Z. Dou, "Threats of adversarial attacks in DNN-based modulation recognition," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Virtual, Jul. 2020, pp. 2469–2478.

[13] K. Davaslioglu and Y. E. Sagduyu, "Trojan attacks on wireless signal classification with adversarial machine learning," in *Proc. IEEE Int. Symp. Dynamic Spectrum Access Netw. (DySPAN)*, Newark, NJ, USA, Nov. 2019.

[14] B. Flowers, R. M. Buehrer, and W. C. Headley, "Communications aware adversarial residual networks for over the air evasion attacks," in *Proc. IEEE Military Commun. Conf. (MILCOM)*, Norfolk, VA, USA, Nov. 2019, pp. 133–140.

[15] P. Qi, T. Jiang, L. Wang, X. Yuan, and Z. Li, "Detection tolerant black-box adversarial attack against automatic modulation classification with deep learning," *IEEE Trans. Reliability*, vol. 71, pp. 674–686, 2022.

[16] H. Rahbari and M. Krunz, "Full frame encryption and modulation obfuscation using channel-independent preamble identifier," *IEEE Trans. Inf. Forensics and Security*, vol. 11, no. 12, pp. 2732–2747, 2016.

[17] N. Hoque and H. Rahbari, "Circumventing the defense against modulation classification attacks," in *Proc. ACM Conf. Secur. Privacy Wireless Mobile Netw. (WiSec)*, Surrey, UK, May 2023, pp. 377–382.

[18] M. Z. Hameed, A. György, and D. Gündüz, "Communication without interception: Defense against modulation detection," in *IEEE Global Conf. Signal Info. Processing (GlobalSIP)*, Hawaii, USA, Dec. 2019.

[19] M. Z. Hameed, A. György, and D. Gündüz, "The best defense is a good offense: Adversarial attacks to avoid modulation detection," *IEEE Trans. Inf. Forensics Secur. (TIFS)*, vol. 16, pp. 1074–1087, 2021.

[20] P. F. de Araujo-Filho, G. Kaddoum, M. C. B. Nasr, H. F. Arcoverde, and D. Campelo, "Defending wireless receivers against adversarial attacks on modulation classifiers," *IEEE Internet of Things J.*, vol. 10, no. 21, pp. 19 153–19 162, Nov. 2023.

[21] B. Kim, Y. E. Sagduyu, K. Davaslioglu, T. Erpek, and S. Ulukus, "Channel-aware adversarial attacks against deep learning-based wireless signal classifiers," *IEEE Trans. Wireless Commun.*, vol. 21, no. 6, pp. 3868–3880, Jun. 2022.

[22] Y. Kaya, M. B. Zafar, S. Aydore, N. Rauschmayr, and K. Kenthapadi, "Generating distributional adversarial examples to evade statistical detectors," in *Proc. Int. Conf. Machine Learn. (ICML)*, vol. 162, Jul. 2022, pp. 10 895–10 911.

[23] Executive Office of the President, National Science and Technology Council. (2011) Trustworthy cyberspace: Strategic plan for the federal cybersecurity research and development program. [Online]. Available: https://www.nitrd.gov/pubs/Fed_Cybersecurity_RD_Strategic_Plan_2011.pdf

[24] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Int. Conf. Learn. Representations (ICLR)*, Vancouver, BC, Canada, Apr. 2018.

[25] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," in *IEEE Conf. Comput. Vision Pattern Recog. (CVPR)*, Los Alamitos, CA, USA, Jun. 2016, pp. 2574–2582.

[26] Y. Dong, F. Liao, T. Pang *et al.*, "Boosting adversarial attacks with momentum," in *IEEE Conf. Comput. Vision Pattern Recog. (CVPR)*, Los Alamitos, CA, USA, Jun. 2018, pp. 9185–9193.

[27] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," in *Int. Conf. Learn. Representations (ICLR)*, Toulon, France, Apr. 2017.

[28] U. Jang, X. Wu, and S. Jha, "Objective metrics and gradient descent algorithms for adversarial examples in machine learning," in *Proc. Annu. Comput. Secur. Appl. Conf. (ACSAC)*, Orlando, FL, USA, 2017, pp. 262–277.

[29] E. M. Hutchins, M. J. Cloppert, R. M. Amin *et al.*, "Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains," *Leading Issues in Information Warfare & Security Research*, vol. 1, no. 1, p. 80, 2011.

[30] M. Wright, S. Venkatesan, M. Albanese, and M. P. Wellman, "Moving target defense against DDoS attacks: An empirical game-theoretic analysis," in *Proc. ACM Workshop Moving Target Defense (MTD)*, Vienna, Austria, 2016, pp. 93–104.

[31] S. Venkatesan, M. Albanese, G. Cybenko, and S. Jajodia, "A moving target defense approach to disrupting stealthy botnets," in *Proc. ACM Workshop Moving Target Defense*, ser. MTD '16, 2016, p. 37–46.

[32] J.-H. Cho, D. P. Sharma, H. Alavizadeh *et al.*, "Toward proactive, adaptive defense: A survey on moving target defense," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 1, pp. 709–745, Jan. 2020.

[33] A. Amich and B. Eshete, "Morphence: Moving target defense against adversarial examples," in *Proc. Annu. Comput. Secur. Appl. Conf. (ACSAC)*, Virtual, 2021, pp. 61–75.

[34] S. Sengupta, T. Chakraborti, and S. Kambhampati, "MTDeep: Boosting the security of deep neural nets against adversarial attacks with moving target defense," in *Decision and Game Theory Secur.: Int. Conf. (GameSec)*, Stockholm, Sweden, 2019, pp. 479–491.

[35] Q. Song, Z. Yan, and R. Tan, "DeepMTD: Moving target defense for deep visual sensing against adversarial examples," *ACM Trans. Sen. Netw.*, vol. 18, no. 1, Oct. 2021.

[36] ——, "Moving target defense for embedded deep visual sensing against adversarial examples," in *Proc. Conf. Embedded Netwo. Sensor Sys. (SenSys)*, New York, NY, USA, 2019, pp. 124–137.

[37] H. Rahbari and M. Krunz, "Exploiting frame preamble waveforms to support new physical-layer functions in OFDM-based 802.11 systems," *Trans. IEEE Wireless Commun.*, vol. 16, no. 6, pp. 1545–1554, 2017.

[38] Y. Qian, Y. Guo, Q. Shao *et al.*, "EI-MTD: Moving target defense for edge intelligence against adversarial attacks," *ACM Trans. Priv. Secur.*, vol. 25, no. 3, May 2022.

[39] T. D. Vo-Huu and G. Noubir, "Mitigating rate attacks through crypto-coded modulation," in *Proc. ACM Intl. Symp. Mobile Ad Hoc Netw. Comput. (MobiHoc)*, Hangzhou, China, Jun. 2015, pp. 237–246.

[40] N. Hoque and H. Rahbari, "POSTER: A tough nut to crack: Attempting to break modulation obfuscation," in *Proc. ACM Conf. Comput. Commun. Secur. (CCS)*, Virtual, Nov. 2021, pp. 2402–2404.

[41] A. Bahramali, M. Nasr, A. Houmansadr, D. Goeckel, and D. Towsley, "Robust adversarial attacks against DNN-based wireless communication systems," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, Virtual/Republic of Korea, Nov. 2021, pp. 126–140.

[42] Y. Shi, T. Erpek, Y. E. Sagduyu, and J. H. Li, "Spectrum data poisoning with adversarial deep learning," in *Proc. IEEE Military Commun. Conference (MILCOM)*, 2018, pp. 407–412.

[43] Z. Zhang, R. Deng, D. K. Y. Yau, P. Cheng, and J. Chen, "Analysis of moving target defense against false data injection attacks on power grid," *IEEE Trans. Info. Forensics Secur.*, vol. 15, pp. 2320–2335, 2020.

[44] M. Gallagher, L. Biernacki, S. Chen *et al.*, "Morpheus: A vulnerability-tolerant secure architecture based on ensembles of moving target defenses with churn," in *Proc. Int. Conf. Architectural Support Prog. Lang. Operating Sys. (ASPLOS)*, Providence, RI, USA, 2019, pp. 469–484.