

# Internet of Things (IoT) Security Techniques Based on Machine Learning (ML)

Pranjal Kalita

Assistant Professor, Department of Physics, Madhab Choudhury College, Barpeta, Assam, India, PIN:  
781301

## Abstract:

Machine Learning (ML) techniques including supervised learning, unsupervised learning and reinforcement learning have been widely applied to improve network security, such as authentication, access control, anti-jamming offloading and malware detections [1]-[15]. In this article the author tries to give overview of a few Machine learning techniques for IoT security to facilitate the researchers for further studies.

**Keywords:** IoT, Machine learning, malware

## Introduction:

Internet of Things (IoT) security encompasses the measures, technologies, and practices designed to safeguard Internet of Things (IoT) devices, the data they collect, and the networks they connect to from unauthorized access, misuse, and disruption. This is critical because poorly secured IoT devices can serve as entry points for cybercriminals to access larger networks, steal data, or launch large-scale attacks. Security in the Internet of Things (IoT) involves a combination of strategies to protect connected devices, the data they handle, and the networks they operate within from cyber threats. Given the rapid proliferation and resource constraints of these devices, robust security measures are essential for ensuring privacy, reliability, and safety across various applications.

The unique nature of IoT devices presents specific security challenges:

**Weak Authentication:** Many devices ship with easily guessable or hardcoded default usernames and passwords that users often fail to change.

**Lack of Updates:** Due to limited processing power or a lack of vendor support, many devices do not receive regular software or firmware updates to patch known vulnerabilities.

**Insecure Communication:** Data is often transmitted without proper encryption, making it vulnerable to interception.

**Physical Vulnerabilities:** Devices deployed in public or remote locations are susceptible to physical tampering to extract data or compromise the device.

**Network Invisibility:** Organizations often struggle to maintain an accurate inventory of all connected IoT devices, making it difficult to monitor and manage risks effectively.

**Resource Constraints:** Many devices have limited memory and processing power, making it difficult to implement robust security software like firewalls or antivirus programs on the device itself.

Securing IoT technology requires a combination of strategies from both manufacturers and end-users:

For Users (Home and Business)

**Change Default Credentials:** Always change default passwords to strong, unique ones and enable multi-factor authentication (MFA) whenever possible.

**Keep Software Updated:** Regularly check for and apply firmware and software updates from the manufacturer.

**Network Segmentation:** Isolate IoT devices on a separate guest network or a dedicated Virtual Local Area Network (VLAN) to prevent a compromised device from accessing sensitive data on your primary network.

**Enable Encryption:** Ensure all data is encrypted in transit and at rest.

**Continuous Monitoring:** Keep an inventory of all connected devices and monitor for unusual activity.

For Manufacturers and Developers

**Security by Design:** Integrate security into the product development lifecycle from the start, rather than as an afterthought.

**Secure Defaults:** Ship devices with strong, unique default passwords and secure default configurations.

**Provide Timely Updates:** Establish a clear policy and mechanism for providing regular, secure firmware updates over the air.

**Implement Zero Trust:** Adopt a zero trust security model that requires all users and devices, even those already on the network, to be continuously verified and authorized.

For organizations, frameworks and standards from bodies like the National Institute of Standards and Technology (NIST) and the European Telecommunications Standards Institute (ETSI) help guide secure implementation.

Understanding and mitigating these security risks is crucial as IoT devices become more integrated into our homes, businesses, and critical infrastructure, from smart cars to medical devices.

**Machine Learning (ML) Algorithms in IoT security:** Machine Learning (ML) techniques are used for IoT security through anomaly detection to identify unusual activity, intrusion detection to spot known and new attacks and behavioral analysis to establish baselines to normal operation. Machine learning (ML) algorithms are sets of instructions that allow computers to learn from data and improve their performance over time without being explicitly programmed. They are broadly categorized into three main types: supervised learning, unsupervised learning, and reinforcement learning. The common techniques are described below.

### **Machine Learning techniques for IoT security:**

**Decision Trees:** A decision tree is a non-parametric supervised machine learning algorithm that uses a flowchart-like tree structure to model decisions and their possible consequences for both classification and regression tasks. Each internal node represents a test on an attribute, each branch represents the outcome of the test, and each leaf node (terminal node) represents the final decision or prediction. The process of building a decision tree, also known as recursive partitioning, works by repeatedly splitting the dataset into smaller, more homogeneous subsets based on the most significant features.

1. Start with the Root Node: The process begins with a single root node representing the entire dataset.
2. Select the Best Attribute: The algorithm evaluates different features to find the one that yields the best split. This is typically measured using metrics like Information Gain (maximizing the reduction in uncertainty or "entropy") or the Gini Index (minimizing the probability of misclassification).
3. Split the Data: The dataset is divided into sub-nodes (branches) based on the outcome of the chosen attribute's test.

4. Repeat Recursively: Steps 2 and 3 are repeated for each new sub-node until a stopping criterion is met (e.g., all instances in a node belong to the same class, or a maximum depth is reached).
5. Reach Leaf Nodes: The process ends at the leaf nodes, which provide the final class label (for classification) or a numerical value (for regression).

#### Types of Decision Trees

Decision trees are primarily categorized by the type of target variable they predict:

**Classification Trees:** Used for predicting a categorical outcome (e.g., whether an email is "spam" or "not spam", or if a loan should be "approved" or "rejected").

**Regression Trees:** Used for predicting a continuous numerical outcome

**Support Vector Machines(SVMs):** Support Vector Machines (SVMs) are a powerful set of supervised machine learning algorithms used for both classification and regression tasks. The core idea is to find an optimal hyperplane (a decision boundary) that best separates different classes of data points in a high-dimensional space, maximizing the distance (known as the margin) between the hyperplane and the closest data points.

These closest data points are called support vectors, as they are the only points that matter in defining the position and orientation of the boundary.

The goal of an SVM is to identify the hyperplane that achieves the maximum margin, which generally leads to better generalization performance on unseen data.

**Hyperplane:** In a 2D space, the hyperplane is a line. In 3D space, it's a plane. In N-dimensional space (where N is the number of features), it's an (N-1)-dimensional flat surface.

**Margin Maximization:** The algorithm calculates the distance to the nearest data points (support vectors) from each class and aims to maximize this distance. This "widest possible street" approach makes the model robust to outliers and less prone to overfitting than some other methods.

**Support Vectors:** Only these critical points influence the decision boundary. Other training examples far from the margin can be removed without affecting the final model.

#### Handling Non-Linear Data with the Kernel Trick

Real-world data is often not linearly separable (e.g., a "doughnut" shape of data points where a straight line cannot separate the center from the outer ring). To address this, SVMs use a powerful technique called the kernel trick:

**Mapping to Higher Dimensions:** A kernel function mathematically transforms the input data into a higher-dimensional feature space where the data points might become linearly separable.

**Linear Separation in High-D:** A linear hyperplane is found in this new, higher-dimensional space, which corresponds to a complex, non-linear decision boundary in the original feature space.

**Efficiency:** The "trick" is that the kernel function calculates the similarity between data points in the high-dimensional space without explicitly computing the coordinates themselves, saving significant computational resources. Common kernel functions include the Radial Basis Function (RBF), polynomial, and sigmoid kernels.

SVMs are widely applied in image classification, text categorization (like spam detection), bioinformatics, and medical diagnosis due to their accuracy and robustness in complex data scenarios.

**Bayesian theorem- based algorithm:** Several machine learning algorithms are founded on or heavily utilize the principles of Bayes' theorem to update probabilities and make predictions based on new evidence and prior knowledge. These algorithms are a cornerstone of probabilistic modeling in ML.

#### Key Bayesian Algorithms

1. Naïve Bayes (NB) Classifiers: This is the most well-known and widely used algorithm directly based on Bayes' theorem.

"Naïve" Assumption: It simplifies the complex calculation of probabilities by assuming that all features in a dataset are conditionally independent of each other, given the class label.

Types: There are variations tailored to different data types:

Gaussian Naïve Bayes: Used for continuous data, assuming it follows a normal (Gaussian) distribution.

Multinomial Naïve Bayes: Ideal for discrete data like word counts in text classification (e.g., spam detection or sentiment analysis).

Bernoulli Naïve Bayes: Suitable for binary or Boolean features (e.g., whether a specific word is present or absent in an email).

Bayesian Belief Networks (BBNs): Also known as Bayesian networks or probabilistic graphical models, these use a directed acyclic graph to represent variables and their complex conditional dependencies, providing a robust way to model uncertainty. They are used in diagnostic systems and risk assessment.

Bayesian Linear Regression: This technique uses the Bayesian framework to estimate the parameters of a linear regression model, providing a probability distribution over the possible values rather than a single point estimate.

Bayesian Optimization: A sequential strategy used for finding the optimal configuration of hyperparameters in complex ML models. It builds a probabilistic model of the objective function and intelligently searches the space to find the best results efficiently.

Bayes Optimal Classifier: This is a theoretical model that is considered the best possible classifier, as it makes the most probable prediction by averaging over all possible hypotheses weighted by their posterior probabilities.

Core Concept: Bayesian Inference

The central idea behind these algorithms is Bayesian inference, the process of incrementally updating the probability of a hypothesis (our belief) as new evidence or data becomes available. This allows models to handle uncertainty gracefully and adapt their predictions over time, which is highly valuable in dynamic, real-world environments like medical diagnosis, financial analysis, and robotics

**K-Nearest neighbour(KNN):** The K-Nearest Neighbors (KNN) algorithm is a simple, non-parametric, supervised machine learning method used for both classification and regression tasks. It is often referred to as a "lazy learner" because it does not build a model during a dedicated training phase but instead memorizes the entire dataset and performs computations only when a prediction is requested.

The fundamental principle behind KNN is that similar data points are likely to be found near one another in a feature space. The algorithm follows a simple, step-by-step process to classify a new, unlabeled data point:

1. Choose the value of K: Select the number of neighbors (K) to consider. This is a crucial hyperparameter that needs to be tuned for optimal performance.
2. Calculate Distances: The algorithm calculates the distance (similarity) between the new query point and all other existing data points in the training set.
3. Find the K Nearest Neighbors: It sorts the calculated distances in ascending order and selects the top K data points with the smallest distances (the nearest neighbors).
4. Make a Prediction:
  1. For Classification: The new data point is assigned the class label that is most frequent among its K neighbors (a majority vote). It is recommended to choose an odd number for K in classification to

avoid ties.

2. For Regression: The algorithm predicts a continuous value by taking the average (mean) of the values of the K nearest neighbors.

#### Distance Metrics

The choice of distance metric is critical to the performance of the KNN algorithm, as it defines how "closeness" is measured.

**Euclidean Distance:** The most common metric for continuous data, representing the straight-line distance between two points.

**Manhattan Distance:** Also known as the "city block" or "taxicab" distance, it sums the absolute differences of the coordinates and is useful for high-dimensional data or grid-based paths.

**Minkowski Distance:** A generalization of both Euclidean ( $p=2$ ) and Manhattan ( $p=1$ ) distances, allowing flexibility via a parameter 'p'.

**Hamming Distance:** Used for categorical or binary data to count the number of positions where two vectors differ.

**Random Forest (RF):** A Random Forest (RF) is a powerful and versatile ensemble learning algorithm for both classification and regression tasks. It operates by constructing a large number of individual decision trees during training and merging their predictions to produce a single, highly accurate, and stable final result.

The strength of the random forest lies in its two primary sources of randomness, which ensure the individual trees are diverse and uncorrelated:

1. **Bagging (Bootstrap Aggregation):** The algorithm creates multiple random subsets of the original training data by sampling *with replacement* (meaning some data points may be repeated, and others left out). Each decision tree is trained independently on one of these unique subsets.
2. **Feature Randomness:** At each node of a tree during the splitting process, the algorithm considers only a random subset of the available features (rather than all features). This forces the trees to be diverse and prevents them from all using the same strong predictor, thereby reducing correlation among them.

The final prediction is determined by:

**Classification:** Taking a majority vote of the classes predicted by all individual trees.

**Regression:** Calculating the average of the numerical predictions from all trees.

**Association Rule algorithms:** Association rule algorithms are unsupervised machine learning methods used to uncover interesting "if-then" patterns and relationships between items within large datasets, most famously in market basket analysis. The primary goal is to find rules that reveal which items are frequently bought or occur together.

#### Core Concepts and Metrics

The strength and significance of the discovered rules are measured using three key metrics:

**Support:** This measures how frequently the entire item set (both the "if" part, or antecedent, and the "then" part, or consequent) appears in the dataset. A higher support value indicates a more popular or common pattern.

**Confidence:** This indicates the reliability of the rule, measuring the conditional probability that the consequent appears when the antecedent is present.

**Lift:** This measures the strength of the association compared to the random co-occurrence of the items.

A lift value greater than 1 indicates a strong positive association (items occur together more often than expected by chance).

A lift value equal to 1 implies the items are independent.

A lift value **less than 1** suggests a negative correlation (one item's presence reduces the likelihood of the other).

#### Popular Association Rule Algorithms

Several algorithms are used to generate these rules, with varying approaches to efficiency:

**Apriori Algorithm:** This is the foundational algorithm that uses a breadth-first search approach. It relies on the "Apriori property" that all subsets of a frequent itemset must also be frequent, which helps prune the search space. Its main drawback is that it requires multiple scans of the database, making it slow on very large datasets.

**FP-Growth (Frequent Pattern Growth):** This algorithm is more efficient than Apriori because it avoids the explicit generation of candidate itemsets. It uses a compact data structure called an FP-tree to represent transactions and recursively mines frequent patterns directly from the tree, requiring only two database scans.

**Eclat (Equivalence Class Transformation):** This algorithm uses a depth-first search strategy and represents data in a vertical format (each item linked to a list of transaction IDs). It efficiently finds frequent itemsets by computing the intersections of these transaction lists and uses less memory than Apriori for certain dataset types.

#### Common Applications

Association rule mining has wide-ranging real-world applications:

**Market Basket Analysis:** Retailers use it to identify products frequently bought together (e.g., placing bread and butter near each other to boost sales).

**Recommendation Systems:** E-commerce platforms (like Amazon's "Customers who bought this also bought") use these rules to suggest complementary products.

**Healthcare:** Physicians can use association rules to find links between symptoms and diseases, aiding in more accurate diagnoses.

**Fraud Detection:** Financial institutions analyze transaction data for unusual patterns or combinations of activities that might indicate fraudulent behavior.

**Ensemble learning (EL):** Ensemble learning is a machine learning paradigm where multiple individual models (called "base learners" or "weak learners") are strategically combined to achieve better predictive performance than could be obtained from any single model alone. This approach leverages the "wisdom of the crowd" to reduce common issues like overfitting, bias, and variance.

The fundamental idea is that while a single model might make specific errors, the errors of diverse models are less likely to be correlated. By aggregating their predictions (through voting for classification or averaging for regression), these errors can cancel each other out, resulting in a more robust, accurate, and generalized final model.

#### Main Types of Ensemble Methods

Ensemble learning techniques are generally categorized into three primary approaches:

##### **Bagging (Bootstrap Aggregating):**

**Method:** This is a parallel method that involves training multiple instances of the same base learning algorithm (e.g., decision trees) on different random subsets of the original training data, sampled *with replacement* (bootstrapping).

**Goal:** Primarily aims to reduce variance and prevent overfitting, which is particularly effective with high-variance models like decision trees.

**Example:** The Random Forest algorithm is the most prominent example, where many decision trees are built on random data and feature subsets, and their outputs are combined by a majority vote or averaging.

### **Boosting:**

**Method:** This is a sequential method that trains models one after another. Each new model is designed to correct the errors (or "residuals") made by the previous ones by giving higher weight or emphasis to the misclassified data points.

**Goal:** Primarily aims to reduce bias and convert many weak learners into a single strong learner.

**Examples:** Popular algorithms include AdaBoost, Gradient Boosting Machines (GBM), and optimized versions like XGBoost and LightGBM.

### **Stacking (Stacked Generalization):**

**Method:** This is a meta-learning approach that combines the predictions from multiple, often heterogeneous (different types of algorithms), base models. The predictions of these base models are used as input features to a second-level model, known as a meta-learner or combiner model, which learns the optimal way to combine them into a final prediction.

**Goal:** Aims to leverage the strengths of diverse models to further improve predictive accuracy and generalization.

**Why Use Ensemble Learning?**

**Improved Accuracy:** Combining multiple perspectives generally leads to a more accurate final prediction.

**Increased Robustness:** Ensembles are less sensitive to noise and outliers because individual errors are often averaged out.

**Reduced Overfitting:** Techniques like bagging explicitly help mitigate overfitting by introducing randomness in the training data and averaging predictions.

**K-Means clustering:** K-means clustering is an unsupervised machine learning algorithm that partitions a dataset into a predefined number of groups, known as clusters, where each data point belongs to the cluster with the nearest mean (centroid). It is a popular and efficient method for finding natural groupings in unlabeled data based on feature similarity.

The standard K-means algorithm operates through an iterative refinement process to minimize the sum of squared distances between data points and their assigned cluster centroids:

1. **Initialization:** The user specifies the number of clusters,  $K$ . Then,  $K$  initial cluster centers (centroids) are randomly selected from the data points or the data space.
2. **Assignment Step (Expectation):** Each data point in the dataset is assigned to the nearest centroid based on a distance metric, usually the Euclidean distance. This forms the initial clusters.
3. **Update Step (Maximization):** The centroids are recalculated by taking the mean (average) position of all data points assigned to that cluster.
4. **Repeat:** Steps 2 and 3 are repeated. The algorithm stops when the centroids no longer change significantly, cluster assignments remain the same, or a maximum number of iterations is reached, indicating convergence to a stable solution.

## Key Considerations

**Choosing K:** Determining the optimal value for K is a critical challenge. Common methods like the Elbow Method (plotting within-cluster sum of squares, WCSS, against K to find where the decrease in variation levels off) and Silhouette Analysis are used to estimate the best number of clusters.

**Initialization Sensitivity:** The final clusters can vary depending on the initial random placement of centroids. To mitigate this, the algorithm is often run multiple times with different starting conditions, and the result with the lowest WCSS is chosen. The K-means++ initialization method is also commonly used to select initial centers in a smarter way that speeds up convergence.

**Data Assumptions:** K-means assumes that clusters are typically spherical, equally sized, and have similar density. It may perform poorly on data with non-spherical shapes, varying densities, or significant outliers, which can skew the centroid positions.

## Applications

K-means clustering is widely used due to its simplicity, speed, and efficiency.

**Customer Segmentation:** Grouping customers based on purchasing behavior or demographics to tailor marketing strategies.

**Image Processing:** Used for image segmentation and compression by grouping similar pixels or colors.

**Anomaly Detection:** Identifying outliers or unusual data points that do not fit well into any cluster (e.g., fraudulent transactions).

**Document Clustering:** Organizing similar documents or articles based on content for information retrieval or topic modeling.

**Principal component analysis (PCA):** Principal Component Analysis (PCA) is a machine learning technique for dimensionality reduction that transforms a dataset with many variables into a smaller set of new variables called principal components. These components are linear combinations of the original features that capture the most variance, or information, in the data. By using PCA, you can simplify complex datasets, speed up model training, and improve performance by reducing noise and overcoming the "curse of dimensionality".

## How it works

**Identifies directions of high variance:** PCA finds the directions in the data where the points are most spread out, which represent the most important information.

**Creates new axes:** It creates a new set of perpendicular (orthogonal) axes, or principal components, that align with these directions of maximum variance.

**Projects data onto new axes:** The original data is then projected onto these new principal components, creating a lower-dimensional representation.

**Retains most information:** The first principal component captures the most variance, the second captures the next most, and so on. By keeping only the first few components, you can reduce dimensionality while retaining most of the original data's information.

## Why it's useful in machine learning

**Reduces noise:** By focusing on the components with the most variance, PCA can filter out less important variables that might be noise.

**Overcomes the "curse of dimensionality":** In high-dimensional data, a model can become complex and perform poorly. PCA simplifies the data, making it more manageable for algorithms.

**Improves model performance:** A model with fewer features can train faster and is less prone to issues like multicollinearity and overfitting.

Aids in visualization: It allows for the visualization of high-dimensional data by projecting it down to two or three dimensions.

Key considerations

Standardize data: PCA is sensitive to the scale of the features. It is best practice to standardize your data before applying PCA so that features with larger values don't unduly influence the results.

Uses numeric data: PCA only works with numeric, quantitative data.

Outlier impact: Outliers can have a disproportionate effect on PCA results, so it is often advisable to address them before running the analysis.

Reinforcement learning method: Reinforcement learning (RL) methods are a category of machine learning algorithms that train an agent to make sequential decisions by interacting with an environment to maximize a cumulative reward. The agent learns through trial and error, adapting its behavior (policy) based on feedback (rewards or penalties) without explicit programming for every scenario.

RL methods can be broadly categorized in several ways, primarily by how they approach the problem:

Model-Based vs. Model-Free Methods

This distinction refers to whether the agent builds an internal representation (or model) of the environment's dynamics.

Model-Free RL: The agent learns the optimal policy or value function directly from its interactions and experiences within the environment. It doesn't attempt to predict the next state or reward, but learns which actions are best through repeated trial and error.

*Examples:* Q-learning, SARSA, Policy Gradient methods.

Model-Based RL: The agent first learns or is given a model of the environment and then uses planning (like a "thought experiment") to determine the best actions. This can be more sample-efficient but is only practical when the environment can be accurately modeled.

*Examples:* Dynamic Programming, Adaptive Dynamic Programming (ADP).

Value-Based vs. Policy-Based Methods

This classification focuses on what the algorithm is learning.

Value-Based Methods: These algorithms aim to estimate the value (expected future reward) of being in a certain state or taking a specific action. The policy is then derived from these values by always choosing the action that yields the highest value.

*Examples:* Q-learning, SARSA, Deep Q-Networks (DQN).

Policy-Based Methods: These algorithms directly learn the optimal policy without necessarily using a value function. They adjust the policy (a mapping from states to actions) based on performance.

*Examples:* REINFORCE, Proximal Policy Optimization (PPO), Trust Region Policy Optimization (TRPO).

Actor-Critic Methods: These hybrid methods combine both value-based and policy-based approaches. The "actor" proposes actions (policy-based), and the "critic" evaluates those actions (value-based).

*Examples:* A2C, A3C, Deep Deterministic Policy Gradient (DDPG).

Some of the most prominent reinforcement learning algorithms include:

Q-Learning: A popular model-free, off-policy algorithm that learns the optimal quality (Q-value) of state-action pairs. It uses a Q-table or a neural network (in the case of DQN) to store and update expected rewards.

SARSA: A model-free, on-policy algorithm. Unlike Q-learning, which learns the value of the next *optimal* action, SARSA learns based on the *actual* next action taken by the current policy. This often makes it more conservative or "cautious" in its learning approach.

Deep Reinforcement Learning (DRL): The application of deep neural networks to reinforcement learning problems, allowing algorithms to handle complex, high-dimensional data like images or raw sensor inputs. DQN is a prime example of DRL.

Conclusion: The application area of IoT is quite diverse. Further developments are required in the IoT security. In this article the author has discussed various types of Machine learning techniques used in Internet of things (IoT) security solutions. This paper will help the researchers in their further studies and on finding new dimensions on Iot security solutions.

### References:

1. L. Xiao, Y. Li, G. Han, G. Liu, and W. Zhuang, "PHY-layer spoofing detection with reinforcement learning in wireless networks," *IEEE Trans. Vehicular Technology*, vol 65, no. 12, pp 10037-10047, Dec. 2016.
2. M. Abu Alsheikt, S. Lin, D. Niyato, and H. P. Tan, "Machine learning in wireless sensor networks: Algorithms, strategies, and applications," *IEEE Commun Surveys and Tutorials*, vol. 16, no. 4, pp. 1996-2018, Apr 2014
3. L. Xiao, C. Xie, T. Chen, and H. Dai, "A mobile offloading game against smart attacks," *IEEE Access*, vol 4 pp. 2281-2291, May 2016
4. L. Xiao, Y. Li, X. Huang and X. J. Du, "Cloud-based malware detection game for mobile devices with offloading", *IEEE Trans. Mobile Computing*, vol. 16, no. 10, pp. 2742-2750, Oct. 2017.
5. M. Ozay, I. Esnaola, F. T. Yarman Vural, S. R. Kulkarni, and H. V. Poor, "Machine learning methods for attack detection in the smart grid," *IEEE Trans. Neural Networks and Learning Systems*, vol. 27, no. 8, pp. 1773-1786 Mar. 2015.
6. J. W. Branch, C. Giannella, H. Szymanski, R. Wolff, and H. Kargupta. "In-network outlier detection in wireless sensor networks," *Knowledge and Information Systems*, vol. 34, no. 1, pp. 23-54, Jan. 2013.
7. F.A. Narudin, A. Feizollah. N. B. Anuar, and A. Gani, "Evaluation of machine learning classifiers for mobile malware detection", *Soft Computing*, vol. 20, no. 1, pp. 343-357, Jan. 2016
8. A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surveys and Tutorials*, vol. 11, no. 2. pp. 1153-1176, Oct 2015
9. R. V. Kulkarni and G. K. Venayagamoorthy, "Neural network based secure media access control protocol for wireless sensor networks," in *Proc. Int'l Joint Conf Neural Networks*, pp. 3437-3444, Atlanta, GA, Jun. 2009.
10. Z. Tan, A. Jamdagni, X. He, P. Nanda, and R. P. Liu, "A system for Denial-of-Service attack detection based on multivariate correlation analysis," *IEEE Trans. Parallel and Distributed Systems*, vol. 25, no. 2, pp. 447-456, May 2013.
11. L. Xian, Q. Yan, W. Lou, G. Chen, and Y. T. Hou, "Proximity-based security techniques for mobile users in wireless networks," *IEEE Trans. Information Forensics and Security*, vol 8, no. 12, pp. 2089-2100, Oct. 2013

12. Y Gwon, S. Dastango, C. Fossa, and H. Kung, "Competing mobile network game: Embracing anti-jamming and jamming strategies with reinforcement learning," in *Proc. IEEE Conf Commun and Network Security (CNS)*, pp. 28-36. National Harbor, MD, Oct. 2013
13. M. A. Aref, S K. Jayaweera, and S. Machuzak. "Multi-agent reinforcement learning based cognitive anti jamming, in *Proc IEEE Wireless Commun and Networking Con (WCNC)*, pp. 1-6. San Francisco, CA. Mar 2017
14. Y.Li, D.E.Quevedo, S. Dey, and L. Shi, "SINR-based DoS attack on remote state estimation: A game- theoretic approach" *IEEE Trans Control of Network Systems*, vol 4. no 3. pp. 632-642, Apr. 2016
15. G Han, L. Xiao, and H.V.Poor, "Two-dimensional anti-jamming communication based on deep reinforcement learning" in *IEEE Int'l Conf Acoustics, Speech and Signal Processing*. pp. 2087-2001, New Orleans, LA Mar 2017