SAPIENZA
UNIVERSITÀ DI ROMA

Data-driven Intrusion Detection for Ambient Intelligence

Department of Computer, Control and Management Engineering

Corso di Laurea Magistrale in Engineeging in Computer Science

Candidate

Luca Maiano
ID number 1531016

Thesis Advisor

Prof. Ioannis Chatzigiannakis

Co-Advisor

Dr. Aris Anagnostopoulos

Academic Year 2018/2019

Thesis defended on 25 October 2019
in front of a Board of Examiners composed by:

Prof. Daniele Nardi (chairman)
Prof. Ioannis Chatzigiannakis
Prof. Alessandro De Luca
Prof. Fabrizio D′Amore
Prof. Giuseppe Oriolo

Data-driven Intrusion Detection for Ambient Intelligence
Master thesis. Sapienza – University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Version: October 19, 2019

Author's email: lucamaiano@gmail.com

Dedicated to
me, my family and friends

Abstract

Billions of embedded processors are being attached to industrial equipment to enhance business processes and enable smart manufacturing. These embedded processors have enough processing capabilities to process sensor data to produce smart insights, and are designed to operate for months without the need of physical interventions. Despite the compelling features of Industrial Internet of Things (IIoT), applied at several industry-related open issues (e.g., integrated enterprise systems, optimized decision making, real-time business analytics), due to the lack of a physical flow of information (e.g., absence of switches and cable-based gateways), the security of such networks is impeding their rapid deployment. In this work we look into IPv6 based IIoT deployments, since it is the leading standard for interconnecting the wireless devices with the Internet and we propose a data-driven anomaly detection system that operates at the transport-layer of 6LoWPAN deployments. We present a comprehensive experimental evaluation carried out using both simulated and real-world experimentation facilities that demonstrates the accuracy of our system against well-known network attacks against 6LoWPAN networks.

Acknowledgments

I would like to express my sincere gratitude to Professor Ioannis Chatzigiannakis for guiding me in carrying on this research project.

My special thanks go to my parents, Gabriele and Antonella, for always supporting me during these years of study, advising me in the most difficult choices and spurring me to give the maximum in every situation without ever making me feel the lack of something.

I would like to thank my friend Tharika for always being present, in times of weakness as in times of satisfaction and helping me to face the best every day.

Thanks to Marta and Dottore (aka Simone), with whom I lived for the last few years and who have become part of my family, making home a pleasant place to stay.

Finally, I would like to thank my friend and colleague Lorenzo, with whom I had the pleasure of working during the last months, and to friends of a lifetime Luca, Davide, and Stefano.

Contents

1 Introduction **1**
1.1 Challenges . . . 1
1.2 Research Questions . . . 3
1.3 Outline . . . 3

2 Background and Theory **4**
2.1 Communication in Industrial Internet of Things . . . 4
2.1.1 IP-based Communication . . . 4
2.1.2 Security Threats . . . 6
2.1.3 Security Solutions . . . 8
2.2 Anomaly Detection for Industrial Internet of Things . . . 8
2.2.1 Anomaly-based Intrusion Detection Systems . . . 9
2.2.2 Machine Learning for Intelligent Intrusion Detection . . . 12
2.3 Time Series Analysis for Industrial Internet of Things . . . 15
2.3.1 Seasonability and Stationarity . . . 15
2.3.2 Modelling Time Series . . . 16

3 Materials and Method **17**
3.1 System Architecture . . . 18
3.2 Experimental Scenario . . . 18
3.3 Methods for Anomaly Detection . . . 20
3.3.1 K-Nearest Neighbors (KNN) . . . 20
3.3.2 Random Forest (RF) . . . 20
3.3.3 Deep Neural Network (DNN) . . . 20
3.3.4 Support Vector machines (SVM) . . . 21
3.3.5 K-Means . . . 21
3.4 Prediction Performance Measure . . . 23
3.5 Implementation Details . . . 23

4 Results **25**
4.1 Exploratory Analysis . . . 25
4.2 Features Extraction and Selection . . . 29
4.3 Experimental Scenarios . . . 31
4.4 Experimental Results . . . 32

5 Conclusions and Future Work **34**
5.1 Conclusions 34
5.2 Future Work 34

A Sample of the Final Dataset **35**

B Project Code and Repository **36**

Chapter 1

Introduction

We live in a connected world, constantly surrounded by an incredible information flow that is changing the way we explore and perceive the world, enabling the development of new and incredible technologies. The European Union Agency for Cybersecurity (ENISA) defines the Internet of Things (IoT) as a cyber-physical ecosystem of interconnected sensors and actuators, which enable intelligent decision making [15]. IoT offers us the opportunity to be more efficient in how we do things, saving us time, money and helping us defining new processes. IoT is tightly bound to cyber-physical systems and in this respect is an enabler of smart infrastructures, such as Industry 4.0 allowing for applications like smart grid, energy and transport [9]. Connectivity is a key component of industry 4.0, enabling manufacturers to create an integrated ecosystem designed to optimize manufacturing, distribution and the product-consumption lifecycle. With the development of 5G networks, IoT devices will be able to connect discrete point solutions and sensors to monitor entire processes, from R&D to the very end of the product lifecycle.

1.1 Challenges

As IoT infrastructures and services become integral parts of the industry landscape, a number of issues related to privacy and trust need to be addressed. Data confidentiality, authentication, access control within the Industrial Internet of Things (IIoT) network, privacy and trust among users and devices, and the enforcement of security and privacy policies are among these issues. However, the different standards and communication stacks involved in combination with the wide variety of embedded hardware components make traditional security countermeasures difficult to be directly applied in the IIoT domain. Moreover, the high number of interconnected devices arises scalability issues [41].

The vision of the IoT has led to a competitive market for stakeholders that, in the absence of common standards, marketed proprietary and application-specific solutions, including a variety of hardware platforms, operating systems, communications protocols and data management schemes. Current IoT deployments are, more often than not, privately run to serve a specific application, enforcing a tight association between the application, the network used by the application and the sensors that constitute that network [27, 22, 4]. These application-specific deployments have

limited scope in data, while information and knowledge sharing is achieved through custom tailored internet gateways. Clearly, such attempts are inherently non-scalable, exhibit low cost-efficiency, are non-adaptive and usually require tremendous efforts to integrate them with existing and well-established services.

In parallel, substantial standardization progress has been made by different bodies (e.g., IETF CoAP, RPL, 6LowPAN, 6TSCH, IEEE 802.15.4e, ETSI M2M, 3GPP MTC, oneM2M), providing technical solutions tailored to the resource-constrained embedded nodes, ranging from the lower to the upper OSI layers. Nevertheless, until now, no standard has managed to attract the vast majority of the stakeholders and dominate the domain. This comes as no surprise as in most cases embedded systems are indeed application-specific and mission-oriented and no single protocol stack can address all possible functional and non-functional requirements and specifications [13].

In particular, the key observation of IIoT is that the connectivity of business processes with external users has opened the Internet to the greater public by providing open standards, resulting in a usable and interoperable software infrastructure and well-understood methodologies for design, implementation, and evaluation. With this trend to continue, the IIoT network provides online, real-time access to and control over the state of real world objects and places. However, the development of applications exploiting this combined infrastructure is currently cumbersome and difficult since application-level protocols, software and development environments, as well as design and validation methodologies and interoperability among proprietary solutions are vastly different and lack integration. The developer currently has to bridge this gap manually and has to be an expert in both worlds.

However, these IIoT networks do not face security issues found in traditional wired or wireless networks and one of the biggest challenges is to satisfy security requirements under the special operating conditions of sensor networks. Wireless sensor networks are comprised of vast numbers of cheap devices that are deployed in an ad-hoc manner. The devices are more vulnerable to various attacks as their location is not known at design time and protection against tampering is very difficult due to their low cost. Therefore, it is easy to assume that the adversary can easily capture the devices and easily read the content of their memory, thus learning the cryptographic secrets and possibly modifying their behavior. In addition, the high node-to-human ratio makes it infeasible to even consider the presence of an online trusted server that monitors and maintains individual nodes constantly. Thus techniques of pre-distribution of keys are much less effective than in traditional networks. Furthermore, as sensor nodes are battery operated, security systems must reduce the energy consumption. Also, since sensor nodes have limited computing and storage capability, cryptographic algorithms and protocols that require intensive computation, communication, or storage are simply not applicable in sensor networks. It is too costly (in terms of computation) to authenticate using a public key and too costly (in terms of memory and computation) to store one-way chains of keys.

These constraints significantly increase the difficulty of securing IIoT networks and make them more vulnerable to security threats. Still, since building secure IIoT networks, the only viable solution is to combine different techniques for securing the system, i.e., develop secure routing schemes, secure aggregation, provide group key establishment methods, cryptographically encrypt messages, etc.; although each single level defense mechanism is highly vulnerable, the combination of multiple

attacking corners increases the overall achieved security.

1.2 Research Questions

In this study, we focus on the use of intrusion detection systems as an additional mechanism to further improve the security of IIoT networks. In contrast to the other approaches, intrusion detection can protect from both inside and outside adversaries. Compared to existing systems for IIoT networks, our approach is more plentiful in the sense that it addresses all the levels of the node stack and is more energy efficient. Specifically, given a set of traces collected from the network, we want to answer the following questions.

1. Does the whole network include at least one malicious node?
2. Does the whole network include at least one malicious node and also identify the type of attack?

In order to answer the questions above, we need to face the following problems too.

1. We want to characterize each node in the network if it is a malicious node.
2. We want to characterize each node in the network if it is affected by a malicious node.
3. We want to characterize each node in the network is a malicious node and also identify the type of attack.
4. We want to characterize each node in the network if it is affected by a malicious node and also identify the type of attack.

1.3 Outline

In the next chapters, the author will guide the reader through the methodological approach and theory that led to the development of this research project. Chapter 2 introduces background knowledge of IIoT security and communication solutions for intrusion detection systems through a detailed description of the actual state of the art. Chapter 3 describes the methodological approach adopted in this work. Here we describe all the processes that led us from data collection to the entire data cleaning and transformation phase that preceded the analysis. This section also explains the algorithms and evaluation metrics that we have used. Chapter 4 discusses the results obtained in our experiments. Finally, Chapter 5 draws conclusions and defines future works.

Chapter 2

Background and Theory

2.1 Communication in Industrial Internet of Things

As already explained, IIoT is integrating general purpose computing and its networking backbone, the Internet, with embedded computing and low-power, lossy wireless networks. The key goal of IIoT is to extend traditional Service-Oriented Architectures (SOA) that are predominantly used within industrial computing and networking infrastructures, into the embedded world, where digital representations of real-world entities offer services to access their physical state and to mash up these real-world services with traditional services and data to create novel applications. Yet, IIoT networks that are based on resource-constrained devices are exempted from these powerful abstractions due to the heavyweight nature of SOA technologies such as SOAP Web Services, TCP/IP, HTTP, and XML.

2.1.1 IP-based Communication

IIoT provides alternative lightweight services to integrate sensor data into existing service networks and their interaction with services running on servers or in clouds through the use of Internet standards. This allows the composition of higher and more complex industrial processes which directly incorporate services provided by resource-constrained devices. In particular, the 6LoWPAN working group [24] proposed a light-weight IPv6 adaptation layer for IIoT which allows efficient transmission and adaptation of IPv6 packets over IEEE 802.15.4 as a data-link and physical layer protocol. By communicating natively with IP, 6LoWPAN networks are connected to other networks simply using IP routers. The 6LoWPAN uses only UDP as TCP is considered to be too resource-demanding. IPv6LowPAN networks usually operate on the board by acting as stub networks, meaning that data entering the network is destined for one of the devices on the IPv6LowPAN network. There are two types of devices within a typical IPv6LowPAN network: routers that route data destined for another node on the network and hosts that represent the receiver devices of messages.

Routing is the ability to transfer a data packet from one device to another one, sometimes over multiple hops. Depending on what layer the routing mechanism is placed, two categories of routing are defined: mesh-under or route-over. In a mesh-under system, routing of data happens transparently, consequently, mesh-under

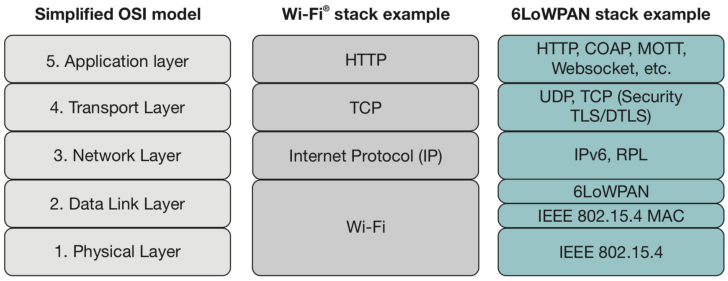


Figure 2.1. The OSI model, a Wi-Fi stack example and the 6LoWPAN stack [37].

networks are considered to be one IP subnet. In route-over networks the routing takes place at the IP level, thus each hop in such networks represents an IP router. The usage of IP routing provides the foundation to larger and more powerful and scalable networks since every router must implement all features supported by a normal IP router.

The IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) [51] is a standardized routing protocol primarily used in a route-over 6LoWPAN network. RPL creates a destination-oriented directed acyclic graph (DODAG) between the nodes in a 6LoWPAN. It supports unidirectional traffic towards a DODAG root and bidirectional traffic between 6LoWPAN devices and between devices and the DODAG root (typically the border router - 6BR). There may exist multiple global RPL instances for a single 6LoWPAN network, and a local RPL DODAG can be created among a set of nodes inside a global DODAG. Each node in a DODAG has a rank that indicates the position of a node relative to other nodes and with respect to the DODAG root. Ranks strictly decrease in the up direction towards the DODAG root and strictly increase from the DODAG root towards nodes. The RPL protocol provides new ICMPv6 control messages to exchange routing graph information. RPL supports two different routing modes; storing mode and non-storing mode. In storing mode, all devices in the 6LoWPAN network configured as routers maintain a routing table and a neighbor table. The routing table is used to look up routes to devices, and the neighbor table is used to keep track of a node's direct neighbors. In non-storing mode, the only device with a routing table is the edge router; hence, source routing is used. Source routing indicates that the packet includes the whole set of hops it needs to travel to reach the destination. Storing mode needs higher requirements on the devices acting as routers while using non-storing mode the overhead increases with the number of hops a packet needs to traverse to reach the destination.

Building upon 6LoWPAN, the IETF CoRE working group [23] follows the REST (REpresentational State Transfer) principles to define CoAP, the Constrained Application Protocol [19], which is a generic application-level protocol for information exchange. CoAP provides exactly the subset of HTTP methods (GET, PUT, POST, and DELETE) which is necessary to offer RESTful web services in an IIoT-compatible manner. In addition, CoAP provides optional transport reliability, normally a core functionality of TCP which is due to the resource constraints by nature not available in IIoT. At the same time, OASIS [36] proposes the MQTT protocol, the Message Queuing Telemetry Transport [18], which is a client-server,

publish/subscribe messaging transport protocol specifically designed for IIoT contexts where a small code footprint is required and/or network bandwidth is at a premium.

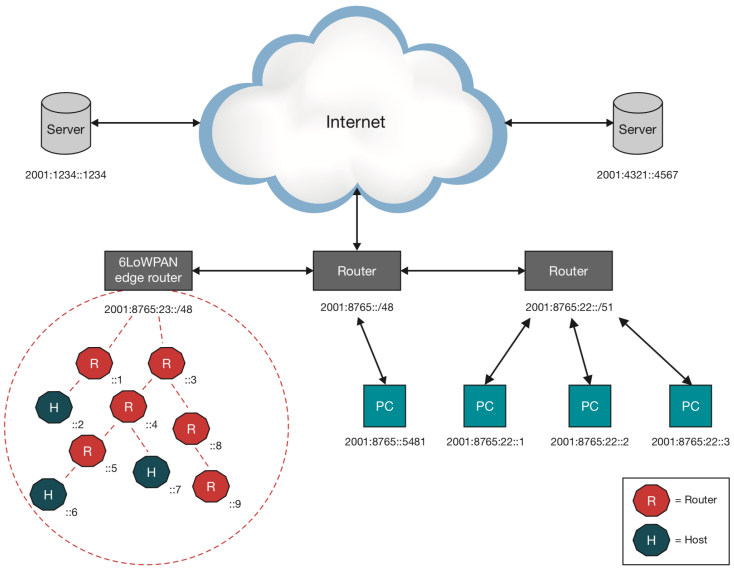


Figure 2.2. An example of an IPv6 network with a 6LoWPAN mesh network [37].

2.1.2 Security Threats

Unlike in typical stand-alone wireless network deployments, due to the global IP connectivity, the constrained devices in the 6LoWPAN networks are accessible from anywhere. Hence, they are exposed to threats both from the Internet and from within the wireless network. Morever, existing wireless networks are in most cases operated over a local or personal area, that is, the system is operated by humans that communicate with other humans or with some of the fixed components of the system. IIoT have a different purpose, in the sense that communications will not involve human interaction. This differentiates the challenges for securing the system and therefore the approaches for offering protection need to be reconsidered. There are many publications [34, 47, 26, 5, 8] that consider some of the most significant security problems. In this work we try to summarize all the existing threats and point out the major attacks against IIoT. We call an IIoT node a *normal node* if it operates based on the system specifications. Otherwise, it is a *malicious node* or an *adversary*.

Wireless Threats

The most basic threats are due to the nature of communication that takes place over a wireless channel. Wireless communication suffer from a number of vulnerabilities:

1. **Eavesdropping**. The most easy way is to overhear the information that a node transmits or receives and then analyze the captured data and extract sensitive information without interacting with the network. These attacks are also known as *passive attacks*.

2. **Data alteration**. An adversary can cause collisions of wireless transmissions and then try to modify the message exchanged between wireless parties. These attacks are also known as *active attacks*.

3. **Identity theft**. Untethered in nature, the adversary can impersonate a legitimate user and when the original user is inactive, transmit messages without being noticed.

Routing Threats

The simplicity of many routing protocols for IIoT networks makes them an easy target for attacks. Karlof and Wagner in [28] classify the routing attacks into the following categories:

1. **Spoofed, altered, or replayed routing information**. While sending the data, the information in transit may be altered, spoofed, replayed, or destroyed. Since sensor nodes usually have only short range transmission, an attacker with high processing power and larger communication range could attack several sensors simultaneously and modify the transmitted information.

2. **Selective forwarding**. In this kind of attack a malicious node may refuse to forward every messages it gets, acting as black hole or it can forward some messages to the wrong receiver and simply drop others.

3. **Sinkhole attacks**. In the Sinkhole attack, the goal of the attacker is to attract all the traffic. Especially, in the case of a flooding based protocol the malicious node may listen to requests for routes, and then reply to the requesting node with messages containing a bogus route with the shortest path to the requested destination.

4. **Sybil attacks**. In Sybil attack the compromised node presents itself with multiple nodes identity. This type of attack tries to degrade the usage and the efficiency of the distributed algorithms that are used. Sybil attack can be performed against distributed storage, routing, data aggregation, voting, fair resource allocation, and misbehavior detection [35].

5. **Wormholes**. Wormhole attack [12] is an attack in which the malicious node tunnels messages from one part of the network over a link, that doesn't exist normally, to another part of the network. The simplest form of the wormhole attack is to convince two nodes that they are neighbors. This attack would likely be used in combination with selective forwarding or eavesdropping.

6. ***HELLO* flood attacks**. This attack is based on the use by many protocols of broadcast *Hello* messages to announce themselves in the network. So, an attacker with greater range of transmission may send many *Hello* messages to a large number of nodes in a wide area of the network. These nodes are then convinced that the attacker is their neighbor. Consequently the network is left in a state of confusion.

7. **Acknowledgment**. Some IIoT network routing algorithms require link layer acknowledgments. A compromised node may exploit this by spoofing these acknowledgments, thus convincing the sender that a weak link is strong or a dead sensor is alive.

Denial of Service (DoS)

This class of attacks is not concerned with the information that is transmitted. Rather, the goal of the attacker is to exhaust the resources of the network and cause it not to function properly. Wood and Stankovic [52, 53] classify several forms of DoS attacks based on the layer that the attack uses. Some of these were already mentioned so we will not repeat them. At the physical layer the attacks take the form of jamming and tampering [34]. Jamming is done by interfering with the radio frequencies nodes are using. Tampering refers to the the physical altering or even damaging of the nodes. An attacker can damage and replace a node, for example, by stealing or replacing information or cryptographic keys. At the link layer the attacker can generate collisions and exhaustion may be caused by protocols that attempt retransmission repeatedly, even when triggered by an unusual and suspicious collision.

2.1.3 Security Solutions

Several attempts have been made in the past years towards preventing intrusions over IIoT networks based on alternative encryption and authentication techniques. Some solutions focus on the validation of the integrity of message exchanges in order to securely route information across the wireless medium [56, 46]. Others propose lightweight, distributed secure group communication primitives that operate on-top of the networking layer to protect data and to cope with potential compromises [14, 7], while others work at the application layer and propose energy-efficient encryption methods to guarantee the confidantiality of the information exchanged [10]. However, these approaches cannot completely prevent malicious users from intruding the network and tamper with the operation of the network [11]. For example, the above techniques may not be fully protected against compromised nodes which participate in the network and already have the shared cryptographic keys [55].

In the context of the IP-based IIoT solutions, consideration of TCP/IP security protocols is important as these protocols are designed to fit the IP network concept and technology. While a wide range of specialized as well as general-purpose key exchange and security solutions exist for the Internet domain, the 6LoWPAN and CoRE IETF working groups look into IKEv2/IPsec [30], TLS/SSL [40], DTLS [38], HIP [33], PANA [17], and EAP [1] as candidate solutions of IIoT, we focus on the discussion of in this study. Application layer solutions such as SSH [54] also exist, however, these are currently not considered.

2.2 Anomaly Detection for Industrial Internet of Things

In an attempt to improve the overall security of networks, a number of Intrusion detection systems (IDS) have been proposed [10]. Such IDS architectures are

classified into two basic categories depending on the *data collection mechanism*:

- **Host-based**. The IDS consults several type of log files (kernel, system, application, etc.) and compares the logs against an internal database of common signatures for known attacks.
- **Network-based**. The IDS is scanning network packets, auditing packet information, and logging any suspicious packets.

IDS architectures can be further classified based on the *detection technique*:

- **Signature-based**. IDS centers on finding an occurrence of predefined signatures or behavior that matches a previously known malicious action or indicates an intrusion.
- **Anomaly-based**. IDS checks for any behaviors that fall outside the predefined or accepted model of behavior.
- **Specification-based**. IDS defines a set of constrains that are indicative of a program's or protocol's correct operation.

Furthermore, *IDS architectures specific to wireless ad-hoc networks* are further divided into three categories:

- **Stand-alone**. Each node operates as a independent IDS and is responsible for detecting attacks only for itself. Such an IDS does not share any information or cooperate with other systems. This architecture implies that all the nodes of the network are capable of running an IDS.
- **Distributed and Cooperative**. All nodes are running their own IDS, but they also cooperate in order to create a global intrusion detection mechanism.
- **Hierarchical**. The network is divided into clusters with cluster-head nodes. These nodes are responsible for routing within the cluster and accept all the accusation messages from the other cluster-members indicating something malicious. Additionally, the cluster-head nodes may also detect attacks against the other cluster-head nodes of the network, as they constitute the backbone of the routing infrastructure.

Based on the above classifications of available methods, we continue by describing the basic philosophy of some characteristic IDS that are specifically designed for wireless sensor networks. Each IDS combines different methods in order to define the resulting architecture.

2.2.1 Anomaly-based Intrusion Detection Systems

Anomaly detection is a critical data analysis task that detects anomalous or inconsistent data from a given dataset. Researchers proposed several definitions of what an anomaly is, but a widely accepted definition is that of Hawkins: "*An anomaly is an observation which deviates so much from other observations as to arouse suspicions that is was generated by a different mechanism*" [25]. Notwithstanding the many techniques available, the following are the research challenges [2].

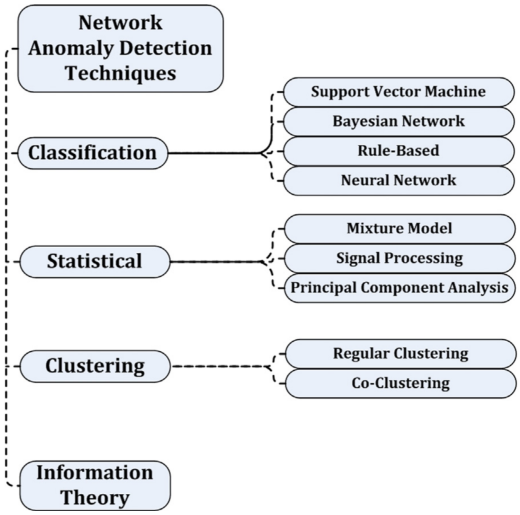


Figure 2.3. Taxonomy of network anomaly detection techniques [2].

1. A lack of universally applicable anomaly detection techniques.
2. Data contains noise, which tends to be an actual anomaly and, therefore, is difficult to segregate.
3. As normal behaviors are continually evolving and may not be normal forever, current intrusion detection techniques may not be useful in the future.

An essential aspect of anomaly detection is the nature of the anomaly. We can identify the following types of anomalies.

1. **Point anomaly**. A particular data instance deviates from the normal pattern of the dataset.
2. **Contextual anomaly**. When a data instance behaves anomalously in a particular context, it is termed a contextual or conditional anomaly.
3. **Collective anomaly**. A collection of similar data instances behave anomalously for the entire dataset, the group of data instances is termed a collective anomaly.

Designing an anomaly detection system, a fundamental aspect to decide is how anomalies are represented as output, which is usually one of the two following ways.

1. **Scoring-based**. This technique assigns an anomaly score to each data point. The scores are ranked and filtered according to some threshold.
2. **Binary/label-based**. Outputs are considered in a binary or multi-labeled fashion.

The comprehensive nature of anomaly detection problems led to the development of several techniques. Focusing on network attacks, a number of different approaches can be employed to detect irregularities.

Statistical Anomaly Detection

Exploits statistical laws to detect anomal behavior in networks.

- **Mixture Models**. Each element falls into one of the following two classes: (i) having a small probability of λ; or (ii) the majority of elements having the probability $1 - \lambda$. From a mixture model point of view, the two probability distributions which generate the data are called the majority (M) and anomalous (A) distributions, with an element x_i generated from either. When the generative distribution for the data is D, it can be represented as $D = (1 - \lambda)M + \lambda A$.
- **Principal Component Analysis (PCA)**. PCA is used to analyze high dimensional data. It uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. This transformation is defined in such a way that the first principal component has the largest possible variance, and each succeeding component, in turn, has the highest variance possible under the constraint that it is orthogonal to the preceding components. The resulting vectors (each being a linear combination of the variables and containing n observations) are an uncorrelated orthogonal basis set [49]. A PCA-based anomaly detector has the benefits of (i) being free from any assumption of statistical distribution, and (ii) being capable to reduce the dimension of the data without missing important information; therefore having minimal computational complexity and allowing support to real-time detections.

Information Theory

Information theory investigates the quantification, storage, and transmission of information. Supervised anomaly detection techniques require a training dataset followed by a test set to evaluate the performance of a model. In such cases, information-theoretic measures are used to determine whether a model is suitable for testing the new dataset. Applications involving information theory usually rely on the following knowledge [31].

- **Entropy**. Measures the uncertainty of a collection of data items. For a dataset D, in which each data item belongs to a class ($x \in C_D$), the entropy of D relative to the $|C_D|$ classification is defined as $H(D) = \sum_{X \in C_D} P(X) log \frac{1}{P(X)}$.
- **Conditional Entropy**. This is the entropy of D given that Y is the entropy probability distribution, i.e. $P(x|y)$.
- **Relative Entropy**. The entropy between two probability distributions $p(x)$ and $q(x)$ defined over the same $x \in C_D$.
- **Relative Conditional Entropy**. The entropy between two probability distributions $p(x|y)$ and $q(x|y)$ defined over the same $x \in C_D$ and $y \in C_Y$.
- **Information Gain**. The measure of the information gain of an attribute or feature A in a dataset D and is $Gain(D, A) = H(D) - \sum_{v \in values(A)} \frac{|D_v|}{|D|} H(D_V)$.

Classification-based

This technique relies on experts' extensive understanding of the properties of network attacks. This methodology is based on the analysis of normal traffic activity, which builds the knowledge base and recognizes activities that deviate from the baseline profile as anomalous. In this way, it is possible to detect intrusions that are completely new, assuming that they manifest extensive deviations from the normal outline.

Clustering-based

Clustering refers to unsupervised algorithms that do not require pre-labeled data to extract rules for grouping similar data examples. This type of anomaly detection, together with the classification-based approach, will be better explained in the next section.

2.2.2 Machine Learning for Intelligent Intrusion Detection

Machine Learning (ML) is a branch of Artificial Intelligence (AI) that provides systems with the ability to learn from data identifying patterns and making decisions with minimal human intervention. Machine learning algorithms build a mathematical model based on sample data (also known as training data) in order to make predictions or decisions without being explicitly programmed to perform some specific task. It is intensively applied in situations where a solution to some specific problem changes in time, like routing in a computer network or finding anomalies in a specific data observation.

Machine learning algorithms can be classified into four categories:

- **Supervised Learning**. Those kinds of algorithms learn features from labeled training data. The algorithms iteratively construct a model that learns how to provide the right output given a set of features.
- **Unsupervised Learning**. Some types of problems only provide inputs without desired targets. Unsupervised learning does not require labeled data and can investigate similarity among unlabeled data, and from there, it can learn to group, cluster, or organize the data in a way such that a human (or other intelligent algorithms) can come in and make sense of the newly organized data.
- **Semi-Supervised Learning**. This class of algorithms falls in between Supervised and Unsupervised methods. In many practical situations, the cost to label data is too high, since it requires skilled human experts to do that. Therefore, semi-supervised algorithms are explicitly designed to work in applications c characterized by the absence of labels in the majority of the observations but present in few.
- **Reinforcement Learning**. Inspired by the learning behaviors of human beings, in Reinforcement Learning, no specific outcomes are defined, and the agent learns from feedback after interacting with the environment. The agent performs some actions and makes decisions on the basis of the reward obtained.

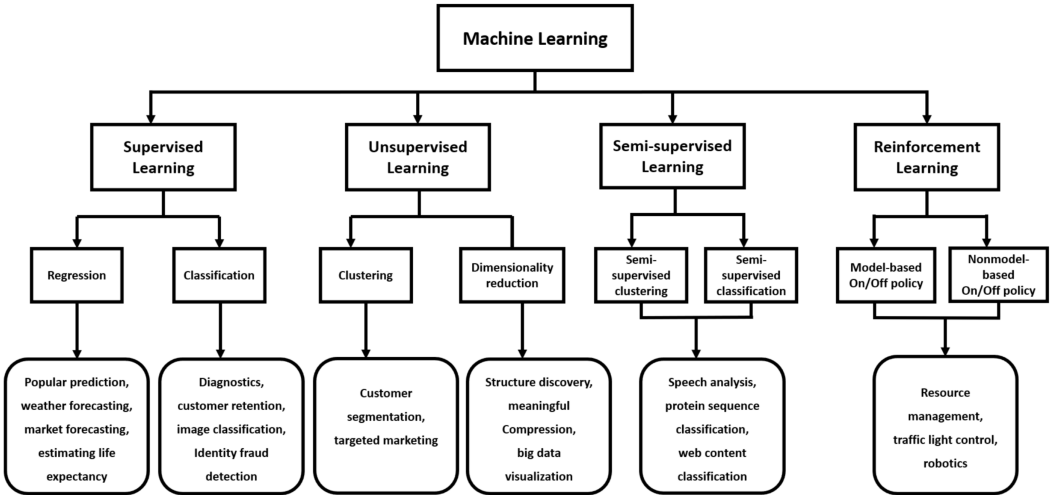


Figure 2.4. Types of Machine Learning Algorithms [16].

An agent can be rewarded for performing good actions or punishment for bad actions and use feedback criteria to maximize the long term rewards.

Deep Learning algorithms can further extend the classification above. Artificial Neural Networks (ANN) are composed of neurons connected through weighted connections. Deep Neural Networks make use of complex and wide layers of neurons to produce effective solutions that typically outperform classical Machine Learning algorithms. The strength of Deep Learning models is that they can extract additional features themselves based on the initial set of features provided to the algorithms, which enables the learning process to be more accurate.

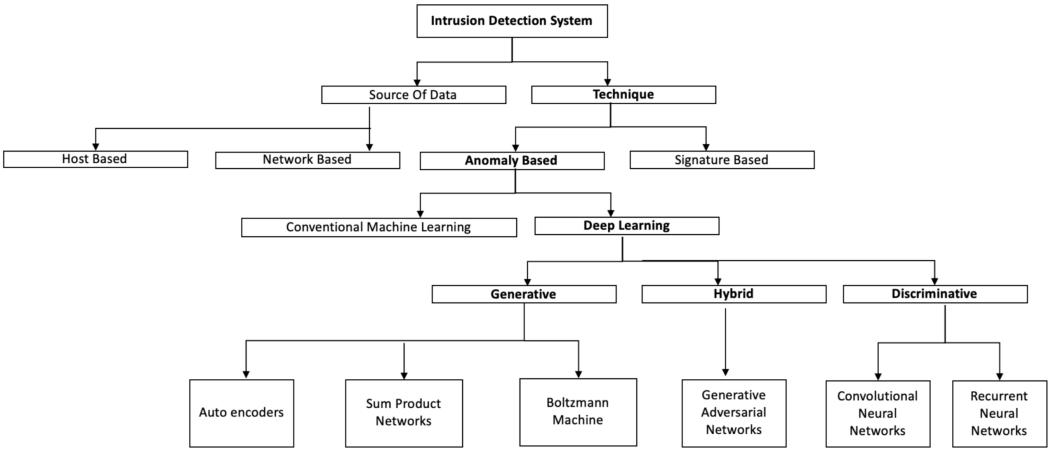


Figure 2.5. Classification of deep learning methods for Intrusion Detection [6].

Machine Learning Techniques for IIoT Security

The rise of machine learning applications has led to the vast adoption of several AI-based solutions. A number of techniques have been employed to detect anomalies

in IIoT environments:

- **Naive Bayes**. This is a probabilistic based on the Bayes theorem and is used for classification tasks. It uses conditional independence to approximate the solutions.
- **K-Nearest Neighbour**. The KNN algorithm assumes that similar things exist nearby. An object is classified through a mechanism of voting of the k nearest objects and is assigned to the most voted class. In KNN regression, the output is the average of the values of k nearest neighbors.
- **K-Means**. It aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. Starting from an initial group of randomly selected centroids, it performs an iterative optimization of the centroids adding new data points.
- **Random Forests and Decision Trees**. Decision trees use a tree data structure to navigate from observations about an item to conclusions about the item's target value. Tree models where the target variable can take a discrete set of values are called classification trees. Decision trees where the target variable can take continuous values are called regression trees.
- **Support Vector Machines**. An SVM training algorithm builds a model that, learning from training examples, assigns new examples to one class. The objective of the support vector machine algorithm is to find a hyperplane in an n-dimensional space that distinctly classifies the data points.
- **Recurrent Neural Networks**. Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs. Long Short-Term Memory (LSTM) is the most widely used RNN in IIoT anomaly detection applications.
- **Deep Learning**. As explained in the previous section, deep learning is extensively adopted to solve anomaly detection problems.

Limitations in Applying Machine Learning in IIoT Networks

Intelligent algorithms usually require large data sets to learn effectively, making it difficult to apply efficient and scalable solutions for low-power IIoT devices [16]. Two main challenges limit the spread of practical solutions.

1. *Processing power and energy*: direct employment of traditional machine learning solutions is not suitable in resource-constrained settings. Even more, efficient IIoT IDS require real-time data processing to detect attacks actively, while conventional ML methods are not designed to handle constant streams of data in real-time.
2. *Data management and analytics*: in an IIoT environment, data is constantly generated from different sources. Heterogeneous data coming from different devices also leads to problems in terms of efficient and unified generalization, requiring pre-processing and data cleaning steps.

2.3 Time Series Analysis for Industrial Internet of Things

Observing an IIoT device over time, we learn how to distinguish between its attacked and normal behavior. A time series is simply a sequence of data points ordered in time. In a time series, time is often the independent variable, and the goal is usually to make a forecast for the future. However, dealing with time observations, it is useful to analyze the forecast to locate stationarity or seasonability. Time series analysis involves generating models that best capture or describe an observed time series in order to understand the underlying features. This often means making assumptions about the form of the data and decomposing the time series into base components.

2.3.1 Seasonability and Stationarity

Seaonability implies periodic fluctuations in data observations over time. This behavior is typical of electricity consumption, which is usually high during the day and low during the night. Seasonability can be derived from examining the autocorrelation, i.e., the similarity between observations as a function of the time lag between them.

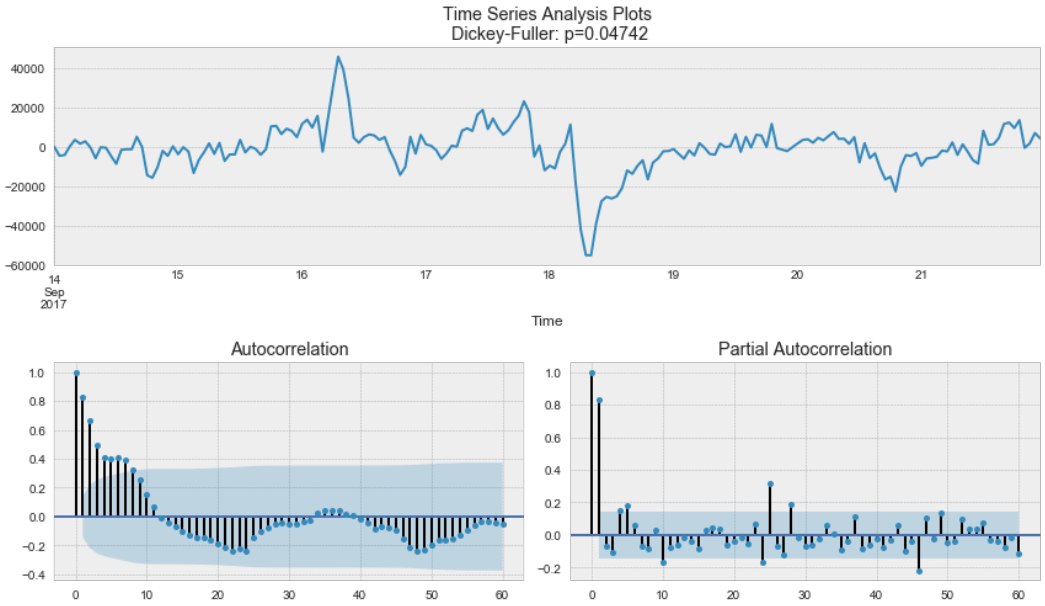


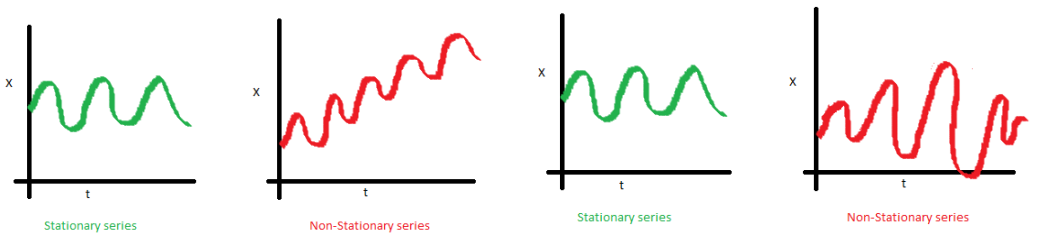
Figure 2.6. An example of time series analysis plot [42].

Stationarity refers to the presence of statistical properties that do not change over time The time series has constant mean and variance, allowing for periodical fluctuations. Stationarity is an important feature of a time series because it makes it easier to make predictions if we assume that the future statistical properties will not be different from the currently observed.

Dickey-Fuller is a statistical test that determines if a time series is stationary or not. It tests the null hypothesis that a unit root is present in an autoregressive model. If $p > 0$ then the process is *non-stationary*. If $p = 0$, the process is considered

to be stationary.

Transformations are used to stabilize the non-constant variance of a series. Common transformation methods include power transform, square root, and log transform. Most of the time, log transformation is used to flatten out non-stationary nodes back to a linear relationship.



(a) Mean increases over time.

(b) Variance varying spread of values over time.

Figure 2.7. Example of stationary vs non-stationary time series [42].

2.3.2 Modelling Time Series

Modeling a time series can be useful in making predictions or searching for trends. In this respect, a number of techniques have been developed:

- **Moving Average**. This model asserts that the subsequent observation is the mean of all past observations. A window can be applied to smooth the time series. The moving average can be used to identify interesting trends in the data. The longer the window, the smoother the trend will be.
- **Exponential Smoothing**. This model works similarly to moving average, but a different decreasing weight is assigned to each observation. This less importance is given to observations as we move further from the present. Each measurement is represented in the form $y = \alpha x_t + (1 - \alpha) y_{t-1}$, where $t > 0$, $0 \leq \alpha \leq 1$ and α represent the smoothing factor.
- **Double Exponential Smoothing**. Used when there is a trend in the time series. In that case, we can use this technique to smooth the trend factor of the observations. Similarly to exponential smoothing, observations can be represented as $y = \alpha x_t + (1-\alpha)(y_{t-1} + b_{t-1})$ and $b = \beta(y_t - y_{t-1}) + (1-\beta) b_{t-1}$ where $t > 0$, $0 \leq \alpha \leq 1$, $0 \leq \beta \leq 1$. and β represent the trend smoothing factor.

Chapter 3

Materials and Method

Existing security solutions cannot completely prevent malicious users from intruding the network and tamper with the operation of the network. In this work, we look into intrusion detection as a second line of defence to protect IoT once an intrusion is detected by activating certain actions to minimize damages, gather evidence for the prosecution, and even launch counter-attacks. We propose an *Anomaly-based* intrusion detection system (IDS) that operates at the transport-layer level on top of 6LoWPAN networks. Assuming an IPv6 enabled IoT, a solution that operates at transport-layer is generic enough to be applied also in combination with closed systems that implement proprietary hardware and custom network protocols. Our solution does not require any particular implementation of 6LoWPAN or RPL, or any additional encryption primitive.

Our IDS uses the ICMPv6 control messages defined within 6LoWPAN to *passively* collect high-level information for the state of the network, such as the round-trip-time, hop distance and packet loss. The IDS periodically transmits a sequence of ICMPv6 control messages to each node of the network based on the public IP addresses. Then it checks for any behaviours that fall outside the predefined, accepted model of behaviour.

In contrast to existing general purpose security systems, in IoT, it is simply infeasible to expect a network administrator to be aware of the full range of potentially relevant possibilities and be able to pull them together manually. For this reason, we completely avoid defining a series of hard-coded alarm limits associated with assumed "steady states" for the network, known to send a large number of false alarms. Instead, we use a combination of machine-learning and statistical analysis to address the specific problem of anomaly detection.

We assume that there is an initial period of time where the network administrator can collect sufficient data regarding the operation of the network. Then periodically we examine the operation of the network by collecting additional statistics and try to identify observations which differ significantly from the majority of the initial data.

Given a set of traces collected: a *network-level anomaly detector* characterizes if the network is affected by at least 1 malicious node; a *class-sensitive network-level anomaly detector* characterizes if the network is affected by at least 1 malicious node of a specific attack class; a *node-level anomaly detector* characterizes each

node of the network if it is a malicious node or not; and a *class-sensitive node-level anomaly detector* characterizes each node of the network if it is a malicious node of a specific attack class. The performance of an anomaly detector is characterized by the *precision*, that is the ability of the detector not to label as positive a sample that is negative and is measured by the ratio of correctly labelled as L to all items actually labelled as L; the *accuracy*, representing the set of labels detected that exactly match the corresponding set of actual labels; the *sensitivity*, the ability of the detector to find all the positive samples and is measured by the ratio of items correctly labelled as L to all items that belong to label L; and the *overhead*, the number of packets that constitute the trace.

3.1 System Architecture

The system that we propose is structured in three main components. *Figure 3.1* depicts our system architecture.

First, we collect heterogeneous data about industrial networks. This process generates log files that contain information of each RPL DODAG and ICMP packets for each node of the network. Log files are manipulated in order to extract relevant information and pre-processed for subsequent analysis. This module includes data exploration and visualization components that are useful for the analysis tasks.

Analysis components get in charge of feature selection for attack *detection* and *classification*. Both components implement the algorithms that will be described in *Section 3.3*.

Results obtained by classification and detection modules are finally exported in a *CSV* format for further analysis and checks.

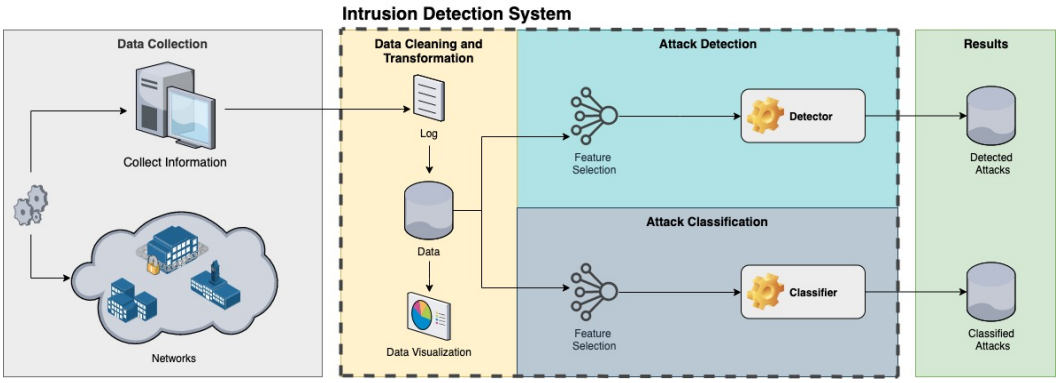


Figure 3.1. Final system architecture.

3.2 Experimental Scenario

The target of our experiments aim at providing an automated way of recognizing intrusions in a network. Given an RPL DODAG, we generate ICMP packets for each node of the network. *Figure 3.2* depicts our proposed methodology.

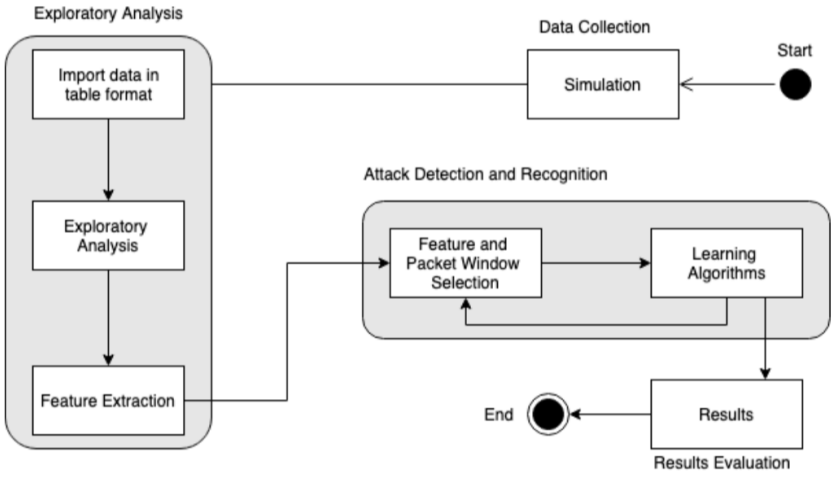


Figure 3.2. Methodology Flow Diagram.

The experiments are based on simulated networks generated by the Cooja tool, the standard Contiki network simulator. We conducted several experiments on different grid schemes (i.e., 3x3, 4x4 and 5x5 grids) as well as random topologies (of 9, 16, 25 nodes). For each node, we collected 200 ICMP packets (sending ping messages from the root node to each other node of the network in every 5 sec), the round trip time (RTT) of each of these messages and the distance of the node from the RPL DODAG root. We parse raw data collected from each experiment to extract relevant features in a table structure. This is helpful to gather statistical information from the dataset. From the analysis of this set of data, we expected to retrieve enough information to learn the usual behaviour of the network.

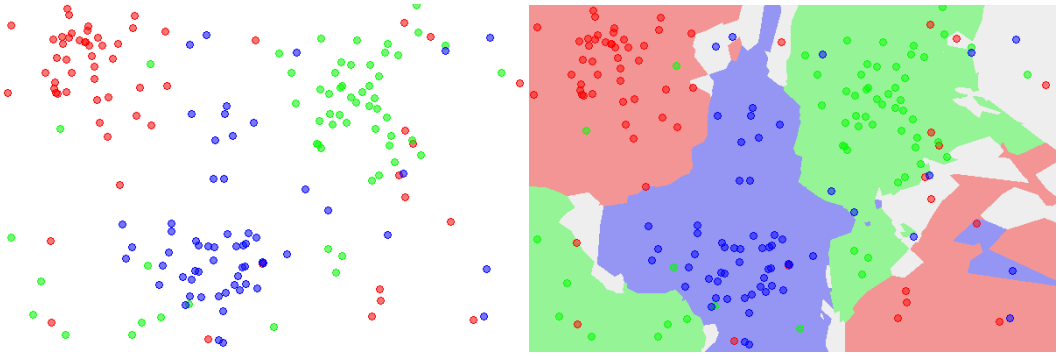
Given the generated network topologies, we substitute some of the nodes of the network with malicious nodes realizing either Black Hole or Grey Hole attacks. The number of malicious nodes is significantly fewer than the number of benign nodes. This is typical in anomaly-detection problems. Since all nodes in the experiment run RPL protocol, we expect that an attacked node will affect the transmissions of its neighbours. Even more, we realized that the malicious nodes usually have more or less important effects on the entire network. Thus all nodes belonging to an attacked network have been labelled as attacked.

After this first exploratory analysis of the data, we select a set of features based on the statistics of the ICMP packets collected over a non-overlapping window of ICMP packets, to train and evaluate the learning algorithms. IMCP packets window is a subset obtained dividing the 200 ICMP messages received by a node by a fixed number in the interval $[12, 24, 48, 100, 200]$ and extract features on that number of ICMP packets. Finally, we run the learning algorithms on the selected dataset and collect results.

3.3 Methods for Anomaly Detection

3.3.1 K-Nearest Neighbors (KNN)

KNN algorithm is a non-parametric method used for classification and regression [3]. The input consists of the k closest training examples in the feature space. An object is classified by a plurality vote of its neighbours, with the object being assigned to the class that is most common among its k nearest neighbours (k is a positive integer, typically small).



(a) Example dataset. **(b)** Example 5NN classification map.

Figure 3.3. Example KNN visualization [48].

3.3.2 Random Forest (RF)

Random Forest (RF) algorithms employ a technique known as bagging, whereby data instances are resampled multiple times to produce multiple training subsets from the training data. Decision trees are then created from each training subset until ensembles of trees have been created. Each tree then casts a unit vote for the outcome of an incoming data instance class label. RF is flexible, with constrained computational resources required.

3.3.3 Deep Neural Network (DNN)

Artificial neural networks (ANN) are computing systems that are inspired by the biological neural networks that constitute human brains. Each neuron performs some calculation and outputs a value that is then spread through all its outgoing connections as input into other units . Connections are characterized by weights that correspond to the importance of the link between two neurons. The computation performed by a unit is separated into two stages: the aggregation and the activation functions. The aggregation function calculates the sum of the inputs received by the unit through all its incoming connections. The resulting value is then fed into the activation function. Neurons are organized in levels. Layers between input and output layers are called the hidden layers. When a new input is given, information passes across layers until the output, where classification happens.

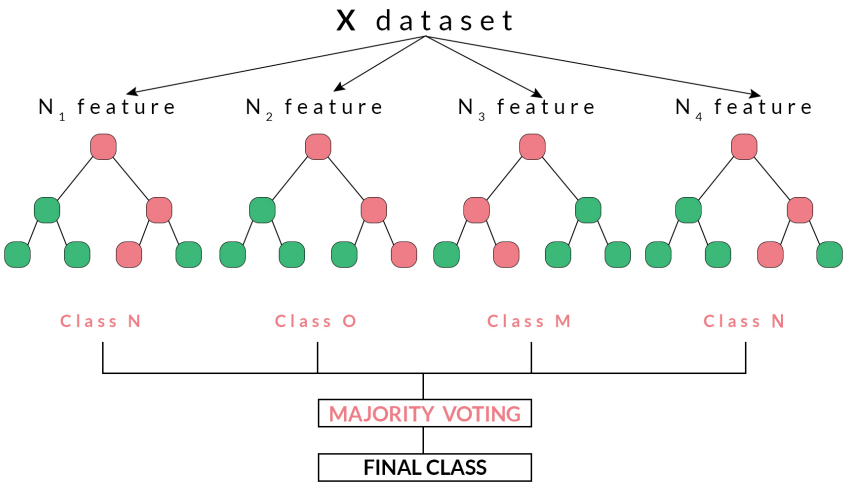


Figure 3.4. Random Forests example visualization [43].

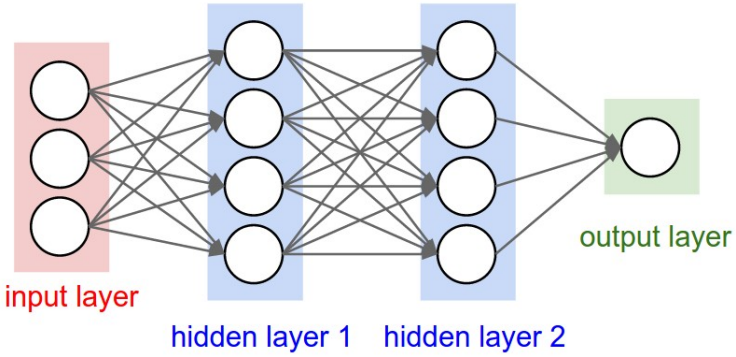


Figure 3.5. 3-layer Neural Network example [29].

3.3.4 Support Vector machines (SVM)

The basic idea of SVM for time-series approximation is mapping the data into a high-dimensional feature space by a nonlinear mapping and then performing a linear regression in the feature space [45]. The nonlinear mapping can be efficiently computed through a kernel function, without iterating over all the corresponding data points. Given the kernel function, the SVM learner tries to find a hyperplane that separates positive from negative data points and at the same time maximizes the separation (margin) between them. This method is known to be resilient to overfitting and to have good generalization performance, due to the max-margin criterion used during optimization. Furthermore, the SVM is guaranteed to converge to a global optimum due to the corresponding convex optimization formulation.

3.3.5 K-Means

K-means is an unsupervised algorithm used to compute (K) clusters in data. A point is considered to be in a particular cluster if it is closer to that cluster's centroid

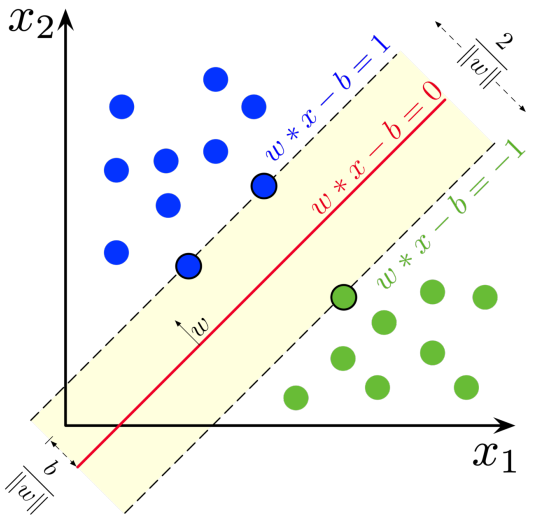


Figure 3.6. SVM example visualization [50].

than any other centroid. K-Means finds the best centroids by repeating over the following phases: (i) assign data points to clusters based on the current centroids; (ii) choose centroids based on the current assignment of data points to clusters [39].

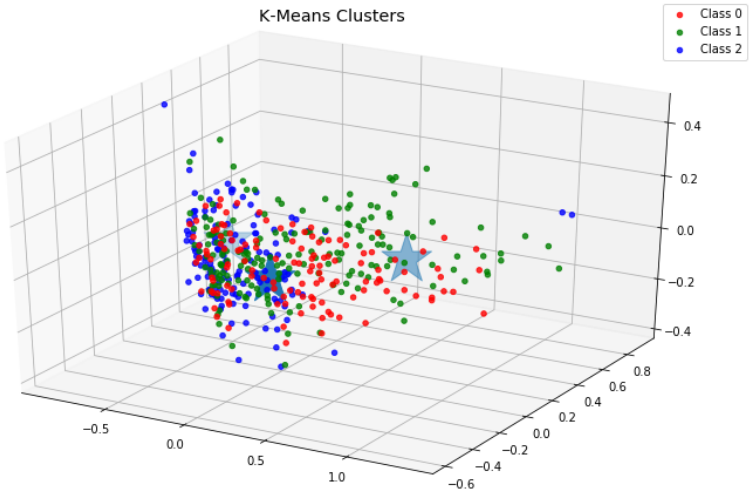


Figure 3.7. K-Means 3-classes clustering visualization from the dataset.

Formally, given a set of n unlabeled training examples $x_1, ..., x_n$, we want to group the data points into k clusters. Our aim is to predict k centroids and a label c_i indicating the cluster to which each data point belongs to. The k-means clustering algorithm iterates as follows:

1. Initialize cluster centroids $\mu_1, ..., \mu_k \in \mathbb{R}^n$
2. Repeat until convergence:
   (a) For every i:
   $$c_i = argmin_j ||x_1 - \mu_j||^2$$

(b) For every j

$$\mu_j = \frac{\sum 1\{c_i = j\} x_i}{\sum 1\{c_i = j\}}.$$

3.4 Prediction Performance Measure

The performance of the classifiers is based on calculating the precision, accuracy, recall and F1 Score for each class and then computing their unweighted mean. This approach does not take label imbalance into account, e.g., when 95% of items are labelled Normal and 5% are labelled Under-Attack, if all the items are labelled as Normal, the accuracy would be 95% but all items from label Under-Attack would be misclassified. For this reason we made sure that there is a good balance across the different labels considered during both training and validation. The four metrics used are defined as follows:

Precision (or Positive Predictive Value (PPV)) – represents the ratio of items correctly labelled as L to all items actually labelled as L, that is, the ratio $\frac{TP}{(TP+FP)}$, where TP is the number of true positives and FP is the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.

Accuracy (or Proportion Correct) – represents the set of labels predicted that exactly match the corresponding set of actual labels.

Recall (or Sensitivity or True Positive Rate or Probability of Detection (PD) or Detection Rate) – represents the ratio of items correctly labelled as L to all items that belong to label L, that is, the ratio $\frac{TP}{(TP+FN)}$, where TP is the number of true positives and FN is the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.

F1 score, or balanced F-score or F-measure represents a weighted average of the precision and recall, where 1 is the best score and 0 the worst score. The relative contribution of precision and recall to the F1 score are equal. The formula for the F1 score is: $\frac{2\times(precision\times recall)}{(precision+recall)}$.

AUC (or Area Under the ROC curve) – ROC is a probability curve and AUC represents degree or measure of separability. It tells how much model is capable of distinguishing between classes.

3.5 Implementation Details

The learning algorithms are implemented using Python 3.7.1 in combination with Keras 2.2.4, Pandas 0.24.1, Scikit 0.20.3 and Numpy 1.15.4. All the experiments have been performed on a Macbook Pro 2017 - 2,3 GHz Intel Core i5, 8 GB 2133 MHz LPDDR3. We release all the dataset and the code used for the experiments on a Github repository
(`https://github.com/ichatz/iot-netprofiler`).

All mentioned algorithms have been extensively used by the state of the art solutions in order to solve anomaly-detection problems. We used the following operating parameters:

1. **K-Nearest Neighbor** (KNN) - using $K = 3$.
2. **Random Forests Classifier** - building 100 estimators.
3. **Support Vector Machines** (SVM) - using a linear kernel.
4. **Deep Neural Network** Classifier - using 5 hidden layers of 32, 64, 96, 64, 32 neurons respectively. Dropout and L2 regularization are applied to reduce overfitting. We use the cross-entropy loss function to evaluate training and validation losses.
5. **K-Means** - with K equal to the number of classes. In this case, data are pre-processed using a PCA.

Chapter 4

Results

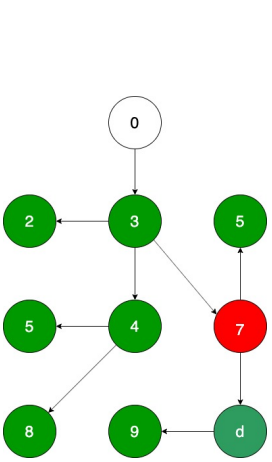
4.1 Exploratory Analysis

A first exploratory analysis has been conducted in order to have a better understanding of the parameters available. Notice that the RPL DODAG could have a different configuration each time that you run an RPL instance on the network, thus, even with the same topology, the network could have a different structure. Even the maximum distance of a node from the root could vary. The distribution of RTT at each node is more or less the same. Even more, by observing the distribution of a node at certain hop-distance from the root, we can usually understand if the node (or one of its neighbours) has been attacked. In fact, depending on which kind of attack has been performed, it's easy to notice that the distribution of a certain number of nodes is affected. In particular, we can distinguish between the following cases:

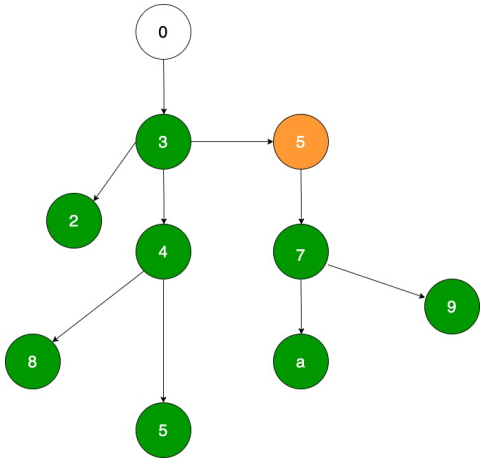
- if the network has been attacked with a Black Hole attack, then we can observe that at least a node becomes unavailable. Most of the times, if this node N was not a leaf of the DODAG, this attack could also affect child nodes of N. In such cases, other nodes could remain unavailable for the entire experiment, or could start again to communicate with the root though an alternative path;
- if a Gray Hole attack has been performed, we can still observe an effect on variance and latencies of some nodes.

RTTs can vary over time; thus, it could be useful to study them to find:

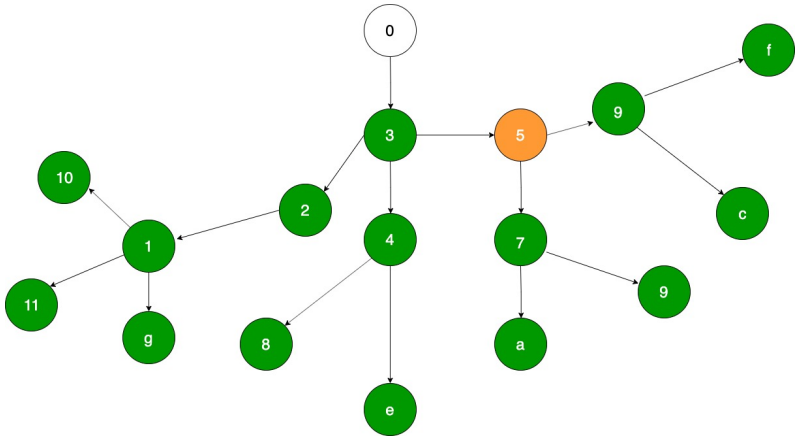
1. **Autocorrelation** that highlights the similarity between observations as a function of the time lag between them.
2. **Seasonability**, i.e. periodic fluctuations.
3. **Trends**.
4. **Stationarity** that describes variations of statistical properties over time. That information could help us to study trends in data.



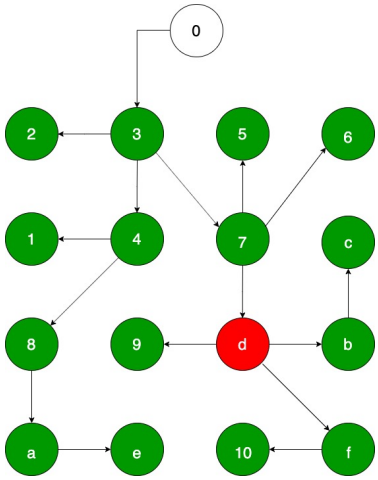
(a) Grid 3x3 topology.



(b) Random 9-nodes topology.

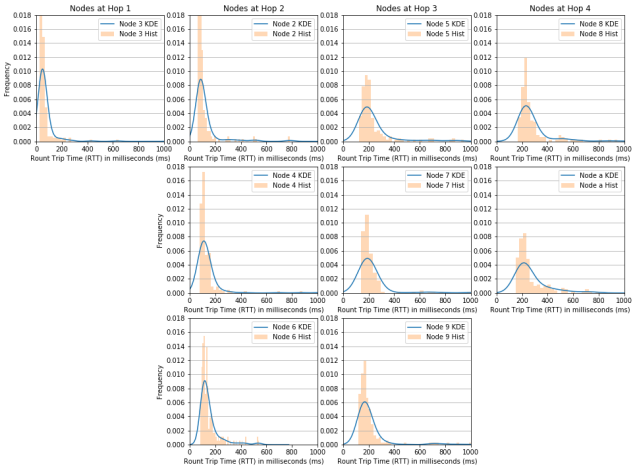


(c) Random 16-nodes topology.

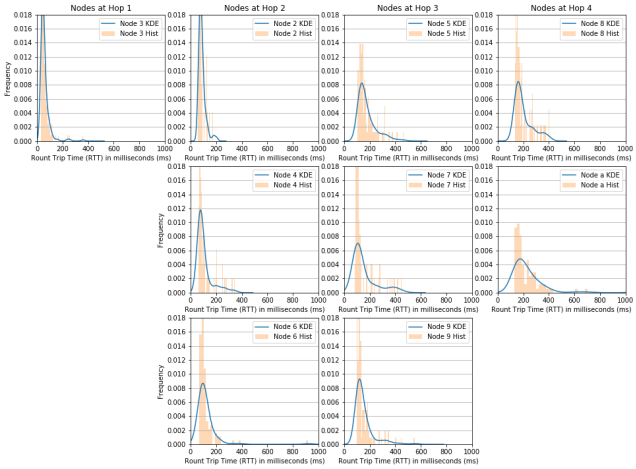


(d) Grid 4x4 topology.

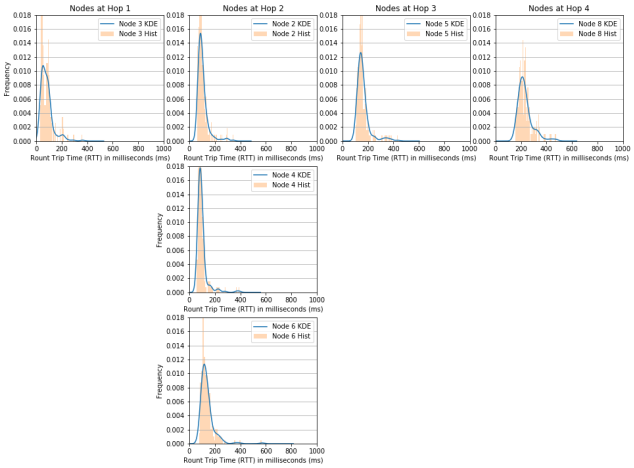
Figure 4.1. Network topologies used for our experiments. Red and orange nodes represent Black Hole and Grey Hole attacks respectively.



(a) Normal behaviour.



(b) Grey Hole Attack.



(c) Black Hole Attack.

Figure 4.2. RTT distribution of a 3x3 grid experiment.

Let us start discussing the autocorrelation that is also helpful to derive seasonability. Assuming that the distribution of each variable fits a Gaussian (bell curve) distribution, we can use the *Pearson's correlation coefficient* to summarize the correlation between the variables. Pearson's correlation coefficient is a number between −1 and 1 that describes a negative or positive correlation, respectively. A value of zero indicates no correlation. We can calculate the correlation for time-series observations with previous time steps, called lags. Because the correlation of the time series observations is calculated with values of the same series at previous times, this is called a serial correlation, or an autocorrelation. A plot of the autocorrelation of a time series by lag is called the AutoCorrelation Function, or the acronym ACF.

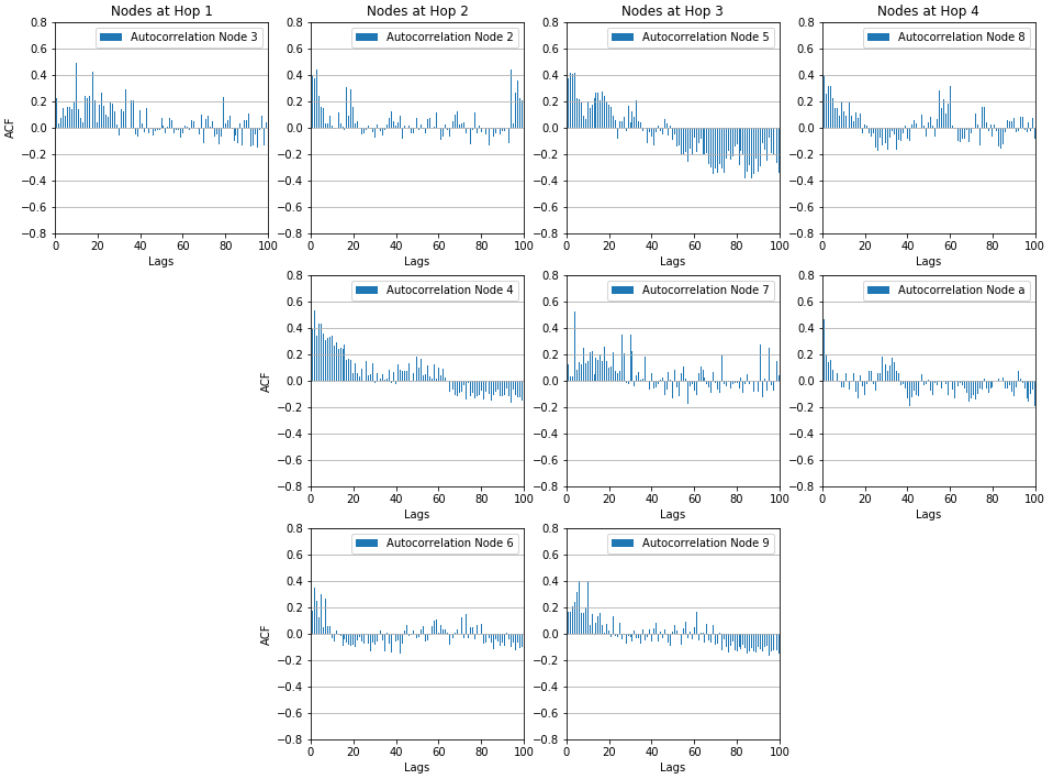


Figure 4.3. Autocorrelation of a 3x3 grid experiment.

The results obtained above do not show particular trends, suggesting that the RTT time series are stationary. In order to make strong assumptions about our data, we computed *Advanced Dickey-Fuller test*[20]. This is a statistical test called a unit root test. The intuition behind a unit root test is that it determines how strongly a time series is defined by a trend. It uses an autoregressive model and optimizes an information criterion across multiple different lag values. The null hypothesis of the test is that the time series can be represented by a unit root, that it is not stationary (has some time-dependent structure). The alternate hypothesis (rejecting the null hypothesis) is that the time series is stationary.

1. Null Hypothesis (H0): If failed to be rejected, it suggests the time series has a

unit root, meaning it is non-stationary. It has some time-dependent structure.

2. Alternate Hypothesis (H1): The null hypothesis is rejected; it suggests the time series does not have a unit root, meaning it is stationary. It does not have a time-dependent structure.

Results from Advanced Dickey-Fuller test confirms that the hypothesis that RTTs time series is stationary, at least for more than 79% of nodes. Starting from these results, a *log transform* has been used to flatten out non-stationary nodes back to a linear relationship. This could help learning algorithms; in fact, stationary time series can be easier to model.

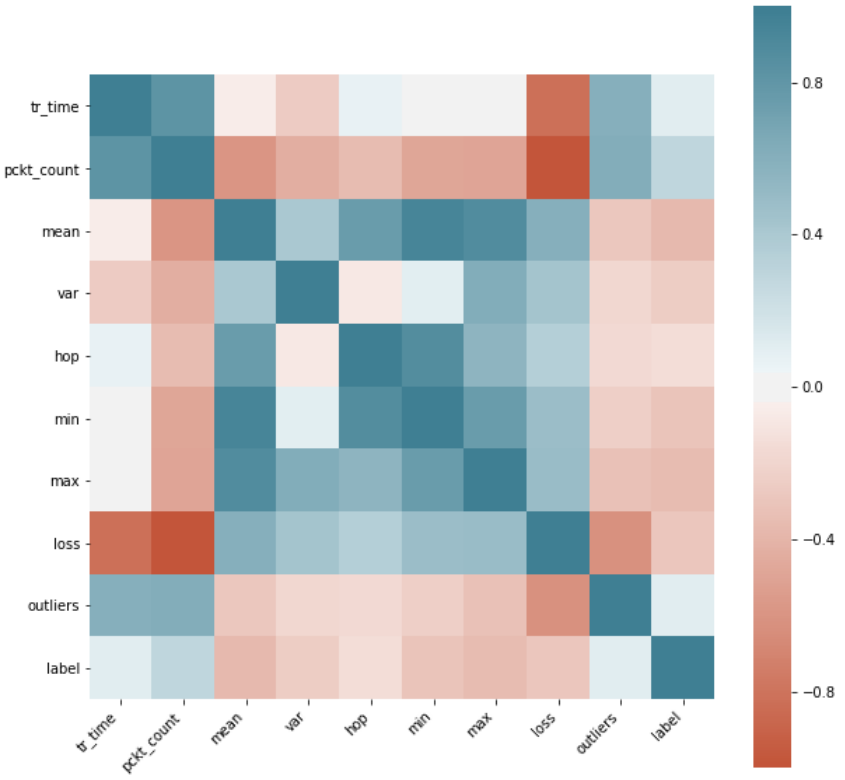
4.2 Features Extraction and Selection

From the results stated above, we consider the following features:

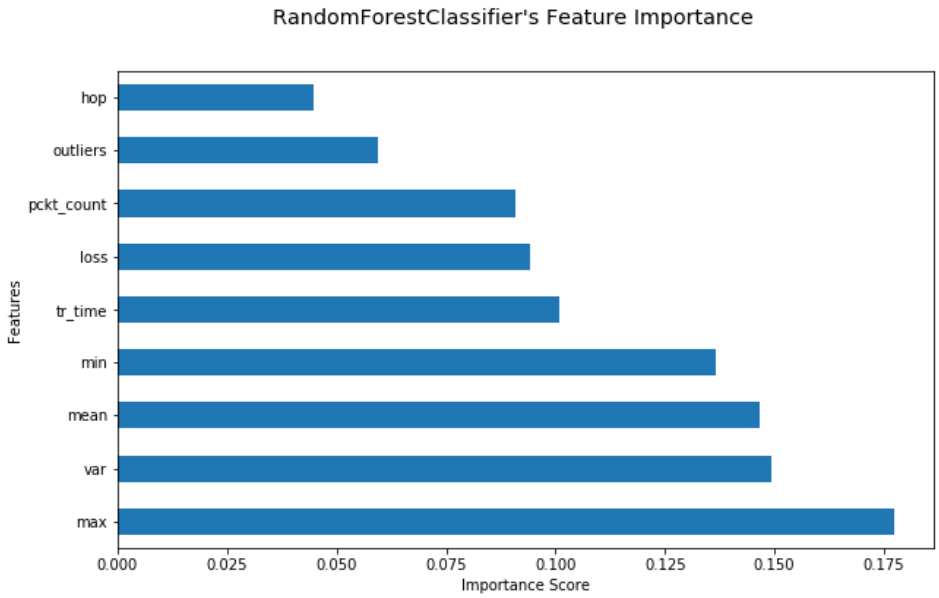
1. the id of the **experiment**;
2. **node** id;
3. **tr_time**, i.e. the sum of RTTs of a node during the experiment in a fixed window of packets;
4. **pckt_count** number of packets that have been received in a fixed window of packets;
5. **mean** of the RTT;
6. **var** of the RTT;
7. **hop** of the node during the experiment;
8. **min** value of the RTT;
9. the **max** value of the RTT;
10. **loss**, i.e. the number of lost packets in a fixed window;
11. the number of **outliers** in a fixed window.
12. the **label** indicating if a node belongs to an attacked network or not.

Data resulting from different networks could have different mean and variance due to their different network topologies. This could reduce the learning rate of machine learning and deep learning algorithms. Thus we apply *feature normalization*.

Some features could be more important than others, so if we select the right set of features, this could help to improve the results given by a learning classifier. In order to select the best set of features, we calculate the correlation matrix of features. We use the Pearson Coefficient that assigns a value within 1 (i.e. positive correlation) and -1 (negative correlation). Following the approach proposed by Furkan Yusuf Yavuz, Devrim ÃIJnal and Ensar Gul in [21], we use *Random Forest Classifier* to iteratively select the most relevant features. In fact, as they suggest, if feature importance is high, it dilutes the effect of the others and may cause overfitting, while less important could slow down (or even deviate) the learning process.



(a) Correlation Matrix.



(b) Random Forest Classifier Features Importance.

Figure 4.4. Features correlation and importance.

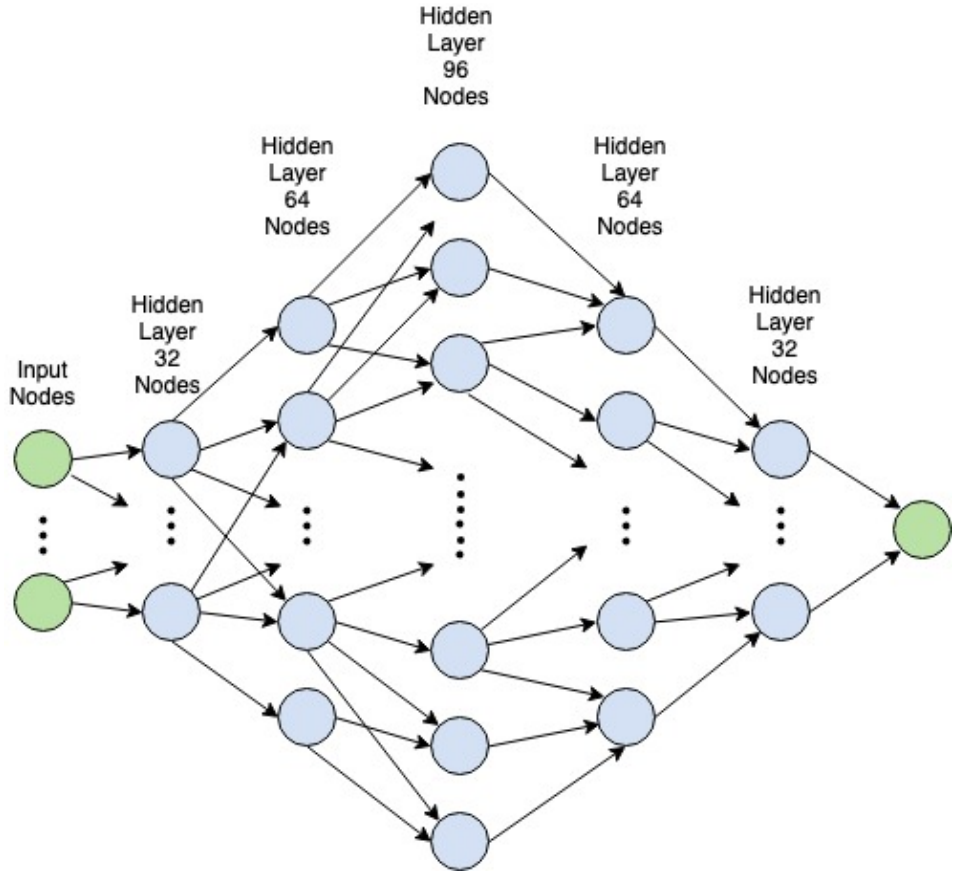


Figure 4.5. Neural Network Model used for attacks detection and classification.

4.3 Experimental Scenarios

We apply learning algorithms with the aim of detect attacks in an IoT network as accurately as possible. Particularly, we focus on supervised and unsupervised algorithms. With this analysis, we want to compare the effectiveness of the following algorithms:

1. **K-Nearest Neighbor** (KNN) - using $K = 3$.
2. **Random Forests Classifier** - building 100 estimators.
3. **Support Vector Machines** (SVM) - using a linear kernel.
4. **Deep Neural Network** Classifier - using 5 hidden layers of 32, 64, 96, 64, 32 neurons respectively. Dropout and L2 regularization are applied to reduce overfitting. We use the cross-entropy loss function to evaluate training and validation losses.
5. **K-Means** - with K equal to the number of classes. In this case, data are pre-processed using a PCA.

All mentioned algorithms have been extensively used by the state of the art solutions in order to solve anomaly-detection problems [16].

We take into consideration two different experimental scenarios. In the first one, we want to detect if a network has been attacked or not. In the second scenario, we try to recognize what kind of attack has been performed if any. In each scenario, we train the learning algorithms with the following approach:

1. Select a window of N ICMP packets for each node ($N \in [12, 24, 48, 100, 200]$).
2. Select a subset of features.
3. Split the dataset using 80% of the data for training and 20% for testing.
4. Train the algorithm using 5-fold cross-validation. Therefore, split the training set and iteratively evaluate the results on 20% of this set.
5. Finally, we test the algorithm on the test set.

The dataset that we generated comes from 43 independent experiments (i.e. networks) that we performed on Cooja. Therefore, we have data coming from 442 different nodes.

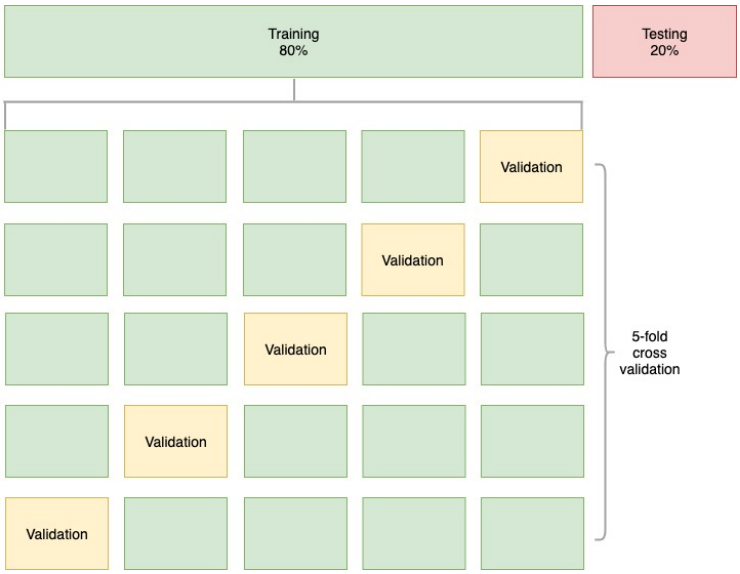


Figure 4.6. 5-fold partitioning of dataset for model training, validation and testing.

4.4 Experimental Results

Results suggest that we can accurately detect if an attack has been performed. Attacks are accurately detected by KNN, Random Forests and Deep Neural Network that achieve accuracy of 80%. Even more, thanks to the stationary property of the time series, we noticed that varying the size of the windows, the results remain almost the same. Therefore, we can detect an attack observing packets in a window of size 12 (corresponding to 1 minute). Results became more accurate, increasing

the size of the windows. Experiments suggest that selecting the entire set of features gives better results.

The most surprising result was given by a dataset containing features coming from experiments with any windows sizes. The dataset was obtained extracting the features of every single network with a window of randomly selected value from $[12, 24, 48, 100, 200]$. This variety of window sizes helps models to generalize. The figure below shows the results of this experiment for attack detection.

model	n_classes	accuracy	precision	recall	f1-score	auc roc
knn	2	0.803406	0.797810	0.794549	0.796010	0.794549
random forest	2	0.829721	0.829888	0.815789	0.820818	0.815789
svm	2	0.752322	0.755062	0.726316	0.731972	0.726316
neural network	2	0.823529	0.847324	0.795865	0.806430	0.795865
kmeans	2	0.375891	0.383400	0.382168	0.375668	0.382168

Figure 4.7. Attack Detection Results.

Finally, the attack classification obtained decent results from the randomly selected windows sizes dataset. KNN and Random Forests obtain 70% accuracy. If we fix the window size to a value in the interval defined before, performances decrease, suggesting that more data are needed in order to train suitable models.

model	n_classes	accuracy	precision	recall	f1-score	auc roc
knn	3	0.712963	0.698080	0.708799	0.701690	0.777039
random forest	3	0.750000	0.759638	0.727007	0.740593	0.791564
svm	3	0.616667	0.686075	0.503812	0.514424	0.628717
neural network	3	0.505556	0.168519	0.333333	0.223862	0.500000
kmeans	3	0.439585	0.433131	0.381860	0.376426	0.538705

Figure 4.8. Attack Recognition Results.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

Our study presents an approach that can detect routing attacks based on a simple analysis of ICMP packets. We introduce an Anomaly-based intrusion detection system that works at the transport-layer level on top of 6LoWPAN networks. Operating at transport-layer, our solution is generic enough to be also applied in combination with closed systems that implement proprietary hardware or custom network protocols and does not require any particular implementation of 6LoWPAN or RPL, or any additional encryption primitive. Our proposed attack detection model successfully detect packet-drop attacks (black-hole attack and grey-hole attack). Our methodology offers a simple and efficient solution to detect if such attacks have been performed. Experimental results suggest that even kind of attack could be detected with high accuracy if learning models are trained with a sufficiently wide range of data samples. The lack of open-source datasets and the difficulty in collecting this type of data surely makes it difficult to train learning algorithms efficiently.

5.2 Future Work

As future work, we intend to extend the dataset with additional attack types and scenarios, introducing a different rate of malicious and normal nodes as well as a larger number of nodes. Data extension can surely allow for the development of a stronger and more robust solution. We also plan to test more sophisticated Recurrent Neural Network architectures like LSTM, which could help to achieve better results on a larger dataset. LSTM has been successfully applied for both time series analysis and attack detection tasks [32]. Unlike standard feedforward neural networks, LSTM has feedback connections, making these networks particularly suited for classification and prediction tasks based on time series data, since there can be lags of unknown duration between important events in a time series.

Appendix A

Sample of the Final Dataset

The dataset used in this project is available on the GitHub repository. This appendix illustrates the structure of the dataset and shows some examples. We collect data using Cooja network simulator [44]. For each node in an experiment, we collect 200 ICMP messages. We simulate examples of normal-behavioral nodes and malicious nodes. The firmware used to simulate attacked nodes is available in the open-source GitHub repository of this project. We include both gray hole examples and black hole examples.

```
PING aaaa::212:7403:3:303(aaaa::212:7403:3:303) 56 data bytes
64 bytes from aaaa::212:7403:3:303: icmp_seq=1 ttl=63 time=59.7 ms
64 bytes from aaaa::212:7403:3:303: icmp_seq=3 ttl=63 time=433 ms
64 bytes from aaaa::212:7403:3:303: icmp_seq=4 ttl=63 time=148 ms
64 bytes from aaaa::212:7403:3:303: icmp_seq=5 ttl=63 time=65.9 ms
64 bytes from aaaa::212:7403:3:303: icmp_seq=6 ttl=63 time=65.0 ms
64 bytes from aaaa::212:7403:3:303: icmp_seq=7 ttl=63 time=105 ms
64 bytes from aaaa::212:7403:3:303: icmp_seq=8 ttl=63 time=132 ms
64 bytes from aaaa::212:7403:3:303: icmp_seq=9 ttl=63 time=70.8 ms
64 bytes from aaaa::212:7403:3:303: icmp_seq=10 ttl=63 time=92.7 ms
64 bytes from aaaa::212:7403:3:303: icmp_seq=11 ttl=63 time=383 ms
64 bytes from aaaa::212:7403:3:303: icmp_seq=12 ttl=63 time=117 ms
64 bytes from aaaa::212:7403:3:303: icmp_seq=13 ttl=63 time=91.4 ms
64 bytes from aaaa::212:7403:3:303: icmp_seq=14 ttl=63 time=117 ms
64 bytes from aaaa::212:7403:3:303: icmp_seq=15 ttl=63 time=386 ms
64 bytes from aaaa::212:7403:3:303: icmp_seq=16 ttl=63 time=68.3 ms
64 bytes from aaaa::212:7403:3:303: icmp_seq=17 ttl=63 time=49.7 ms
64 bytes from aaaa::212:7403:3:303: icmp_seq=18 ttl=63 time=155 ms
64 bytes from aaaa::212:7403:3:303: icmp_seq=19 ttl=63 time=162 ms
64 bytes from aaaa::212:7403:3:303: icmp_seq=20 ttl=63 time=151 ms
64 bytes from aaaa::212:7403:3:303: icmp_seq=21 ttl=63 time=363 ms
64 bytes from aaaa::212:7403:3:303: icmp_seq=23 ttl=63 time=138 ms
64 bytes from aaaa::212:7403:3:303: icmp_seq=24 ttl=63 time=85.8 ms
64 bytes from aaaa::212:7403:3:303: icmp_seq=25 ttl=63 time=78.8 ms
64 bytes from aaaa::212:7403:3:303: icmp_seq=26 ttl=63 time=97.3 ms
64 bytes from aaaa::212:7403:3:303: icmp_seq=27 ttl=63 time=231 ms
64 bytes from aaaa::212:7403:3:303: icmp_seq=28 ttl=63 time=444 ms
64 bytes from aaaa::212:7403:3:303: icmp_seq=29 ttl=63 time=365 ms
64 bytes from aaaa::212:7403:3:303: icmp_seq=30 ttl=63 time=405 ms
64 bytes from aaaa::212:7403:3:303: icmp_seq=31 ttl=63 time=202 ms
64 bytes from aaaa::212:7403:3:303: icmp_seq=32 ttl=63 time=121 ms
64 bytes from aaaa::212:7403:3:303: icmp_seq=33 ttl=63 time=88.4 ms
64 bytes from aaaa::212:7403:3:303: icmp_seq=34 ttl=63 time=62.5 ms
```

Figure A.1. An example of collected ICMP messages.

Section 4.2 discusses the process that we adopted for feature extraction and selection.

Appendix B

Project Code and Repository

The code of this research project is completely open-source. This section describes the content of the GitHub repository (`https://github.com/ichatz/iot-netprofiler`). First, we describe the logical structure of the API; then, we conclude explaining how to execute the code.

Figure B.1 illustrate the logical structure of the project API.

- 1_*Data_Exploration.ipynb* is a Jupyter notebook that performs the first data exploration analysis explained in *Section 4.1*
- 2_*Attack_Detection.ipynb* is a notebook performing the attack detection analysis explained in *Section 4.3*
- *data* folder stores all input and output data.
  - The input data should be added in the *experiments* subfolder.
- The *lib* folder contains the API functions used to implement all data manipulation, analysis, and visualization steps.

The code can be executed running Jupiter notebooks with the following command on terminal: *jupyter-lab*. Numpy, Pandas, Sklearn, Keras, and Matplotlib libraries are required. Refer to *Section 3.5* for more details.

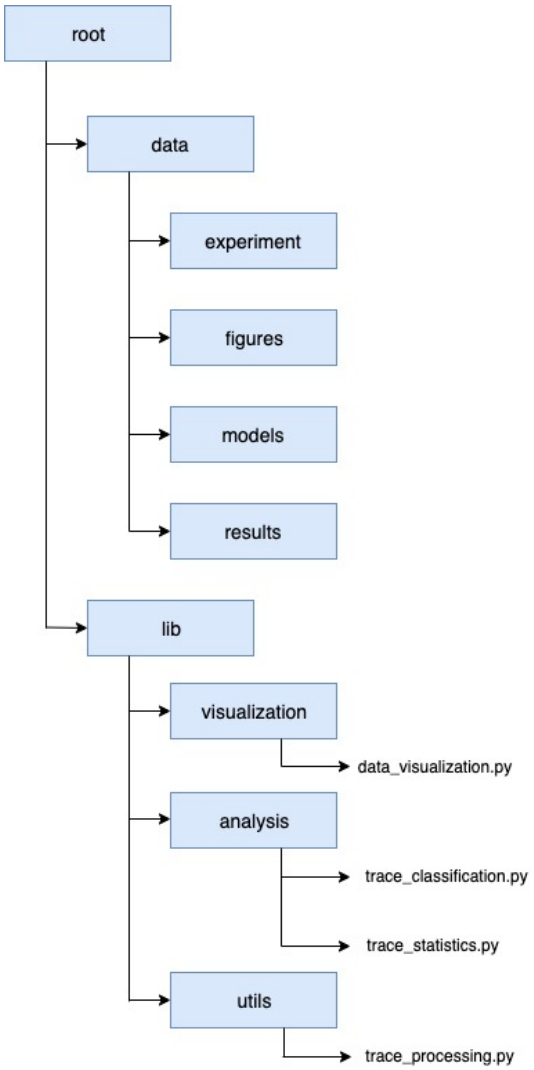


Figure B.1. Project API structure.

Bibliography

[1] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and Levkowetz, H. Extensible authentication protocol (EAP) (2005). Available from: https://tools.ietf.org/html/rfc4017.

[2] Ahmed, M., Mahmood, A., and Hu, J. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, **60** (2015), 19. doi:10.1016/j.jnca.2015.11.016.

[3] Altman, N. S. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, **46** (1992), 175âĂŞ185. Available from: https://www.tandfonline.com/doi/abs/10.1080/00031305.1992.10475879, arXiv:https://www.tandfonline.com/doi/abs/10.1080/00031305.1992.10475879, doi:doi:10.1080/00031305.1992.10475879.

[4] Baumgartner, T., Chatzigiannakis, I., Fekete, S. P., Fischer, S., Koninis, C., KrÃűller, A., KrÃijger, D., Mylonas, G., and Pfisterer, D. Distributed algorithm engineering for networks of tiny artifacts. *Computer Science Review*, **5** (2011), 85 . Available from: http://www.sciencedirect.com/science/article/pii/S1574013710000535, doi:https://doi.org/10.1016/j.cosrev.2010.09.006.

[5] Butun, I., Morgera, S. D., and Sankar, R. A survey of intrusion detection systems in wireless sensor networks. *IEEE communications surveys & tutorials*, **16** (2014), 266.

[6] Chalapathy, R. and Chawla, S. Deep learning for anomaly detection: A survey. (2019).

[7] Chatzigiannakis, I., Konstantinou, E., Liagkou, V., and Spirakis, P. Design, analysis and performance evaluation of group key establishment in wireless sensor networks. *Electronic Notes in Theoretical Computer Science*, **171** (2007), 17 . Proceedings of the Second Workshop on Cryptography for Ad-hoc Networks (WCAN 2006). Available from: http://www.sciencedirect.com/science/article/pii/S1571066107000655, doi:https://doi.org/10.1016/j.entcs.2006.11.007.

[8] Chatzigiannakis, I., Liagkou, V., and Spirakis, P. G. Brief announcement: Providing end-to-end secure communication in low-power wide area networks. In *International Symposium on Cyber Security Cryptography and Machine Learning*, pp. 101–104. Springer (2018).

[9] Chatzigiannakis, I., Mylonas, G., and Vitaletti, A. Urban pervasive applications: Challenges, scenarios and case studies. *Computer Science Review*, **5** (2011), 103 . Available from: http://www.sciencedirect.com/science/article/pii/S1574013710000444, doi:https://doi.org/10.1016/j.cosrev.2010.09.003.

[10] Chatzigiannakis, I., Pyrgelis, A., Spirakis, P. G., and Stamatiou, Y. C. Elliptic curve based zero knowledge proofs and their applicability on resource constrained devices. In *2011 IEEE Eighth International Conference on Mobile Ad-Hoc and Sensor Systems*, pp. 715–720 (2011). doi:10.1109/MASS.2011.77.

[11] Chatzigiannakis, I. and Strikos, A. A decentralized intrusion detection system for increasing security of wireless sensor networks. In *2007 IEEE Conference on Emerging Technologies and Factory Automation (EFTA 2007)*, pp. 1408–1411 (2007). doi:10.1109/EFTA.2007.4416949.

[12] chun Hu, Y., Perrig, A., and Johnson, D. B. Wormhole detection in wireless ad hoc networks. In *Ninth International Conference on Network protocol (ICNP)*, vol. 1 (2002).

[13] Dimitrios, V. G. D. G., A. and Ioannis, C. Employing internet of things technologies for building automation. In *In Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies & Factory Automation (ETFA 2012)*, pp. 1–8. IEEE (2012).

[14] Du, W., Deng, J., Han, Y. S., Varshney, P. K., Katz, J., and Khalili, A. A pairwise key predistribution scheme for wireless sensor networks. *ACM Transactions on Information and System Security (TISSEC)*, **8** (2005), 228.

[15] ENISA. Iot and smart infrastructures (2019). Available from: https://www.enisa.europa.eu/topics/iot-and-smart-infrastructures/iot.

[16] Fatima Hussain, S. A. H., Rasheed Hussain and Hossain, E. Machine learning in iot security: Current solutions and future challenges. *arXiv:1904.05735v1*, (2019). Available from: https://www.researchgate.net/publication/332368962_Machine_Learning_in_IoT_Security_Current_Solutions_and_Future_Challenges, arXiv:https://www.researchgate.net/profile/Rasheed_Hussain3/publication/332368962_Machine_Learning_in_IoT_Security_Current_Solutions_and_Future_Challenges/links/5cfe04bfa6fdccd1308f889a/Machine-Learning-in-IoT-Security-Current-Solutions-and-Future-Challenges.pdf.

[17] Forsberg, D., Ohba, Y., Patil, B., Tschofenig, H., and Yegin, A. Protocol for carrying authentication for network access (PANA) (2008). Available from: https://tools.ietf.org/html/rfc5191.

[18] Frank, B., Shelby, Z., Hartke, K., and Bormann, C. Message Queuing Telemetry Transport (MQTT) (2015). Available from: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html.

[19] Frank, B., Shelby, Z., Hartke, K., and Bormann, C. Constrained Application Protocol (CoAP) (2016). Available from: https://tools.ietf.org/html/rfc7252.

[20] Fuller, W. A. Introduction to statistical time series, 2nd edition. (1995). Available from: https://www.wiley.com/en-us/Introduction+to+Statistical+Time+Series%2C+2nd+Edition-p-9780471552390.

[21] Furkan Yusuf YAVUZ, D. Ã. G. Deep learning for detection of routing attacks in the internet of things. *International Journal of Computational Intelligence Systems*, **12** (2018), 39. Available from: https://www.researchgate.net/publication/328782359_Deep_Learning_for_Detection_of_Routing_Attacks_in_the_Internet_of_Things, arXiv:https://www.researchgate.net/profile/Devrim_Uenal/publication/328782359_Deep_Learning_for_Detection_of_Routing_Attacks_in_the_Internet_of_Things/links/5be27bce4585150b2ba45f6c/Deep-Learning-for-Detection-of-Routing-Attacks-in-the-Internet-of-Things.pdf, doi:10.2991/ijcis.2018.25905181.

[22] Georgitzikis, V., Akribopoulos, O., and Chatzigiannakis, I. Controlling physical objects via the internet using the arduino platform over 802.15. 4 networks. *IEEE Latin America Transactions*, **10** (2012), 1686.

[23] Group, I. C. W. Constrained restful environments (2019). Available from: https://tools.ietf.org/wg/core/.

[24] Group, I. L. W. Ipv6 over low power wpan (2013). Available from: http://tools.ietf.org/wg/6lowpan/.

[25] Hawkins. Identification of outliers (1980). doi:doi:10.1007/978-94-015-3994-4.

[26] Heer, T., Garcia-Morchon, O., Hummen, R., Keoh, S. L., Kumar, S. S., and Wehrle, K. Security challenges in the ip-based internet of things. *Wireless Personal Communications*, **61** (2011), 527.

[27] Karkazis, P., Trakadas, P., Zahariadis, T. B., Chatzigiannakis, I., Dohler, M., Vitaletti, A., Antoniou, A., Leligou, H., and Sarakis, L. Resource and service virtualisation in M2M and iot platforms. *IJIEI*, **3** (2015), 205. Available from: https://doi.org/10.1504/IJIEI.2015.069897, doi:10.1504/IJIEI.2015.069897.

[28] Karlof, C. and Wagner, D. Secure routing in wireless sensor networks: Attacks and countermeasures. In *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications, 2003.*, pp. 113–127. IEEE (2003).

[29] Karpathy, A. Convolutional neural networks (cnns / convnets) (2019). Available from: https://cs231n.github.io/convolutional-networks/.

[30] Kaufman, C. Internet key exchange (IKEv2) protocol (2005). Available from: https://tools.ietf.org/html/rfc4306.

[31] Lee, W. and Xiang, D. Information-theoretic measures for anomaly detection. *Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001*, (2001), 130. doi:doi:10.1109/SECPRI.2001.924294.

[32] Malhotra, L. S. G. A. P., Pankaj; Vig. Long short term memory networks for anomaly detection in time series. *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning âĂŤ ESANN*, (2015). Available from: https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2015-56.pdf.

[33] Moskowitz, R., Nikander, P., Jokela, P., and Henderson, T. Host identity protocol Version 2(HIPv2) (2015). Available from: https://tools.ietf.org/html/rfc7401.

[34] Mpitziopoulos, A., Gavalas, D., Konstantopoulos, C., and Pantziou, G. A survey on jamming attacks and countermeasures in wsns. *IEEE Communications Surveys & Tutorials*, **11** (2009), 42.

[35] Newsome, J., Shi, E., Song, D., and Perrig, A. The sybil attack in sensor networks: analysis & defenses. In *Third international symposium on information processing in sensor networks, 2004. IPSN 2004*, pp. 259–268. IEEE (2004).

[36] OASIS. Iot/m2m committee (2019). Available from: https://www.oasis-open.org/committees/tc_cat.php?cat=iot.

[37] Olsson, J. 6lowpan demystified. (2014). Available from: https://www.ti.com/lit/wp/swry013/swry013.pdf.

[38] Phelan, T. Datagram transport layer security (DTLS) over the datagram congestion control protocol (DCCP) (2008). Available from: https://tools.ietf.org/html/draft-ietf-dccp-dtls-06.

[39] Piech, C. K means (2013). Available from: http://stanford.edu/~cpiech/cs221/handouts/kmeans.html.

[40] Rescorla, E. The transport layer security (TLS) protocol version 1.3 (2018). Available from: https://tools.ietf.org/html/rfc8446.

[41] Sadeghi, A.-R., Wachsmann, C., and Waidner, M. Security and privacy challenges in industrial internet of things. In *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6. IEEE (2015).

[42] Sergeev, D. Open machine learning course (2018). Available from: https://link.medium.com/aNxXU5I9L0.

[43] Shagufta Tahsildar. Random forest algorithm in trading using python (2019). Available from: https://blog.quantinsti.com/random-forest-algorithm-in-python/.

[44] ThÃlbaudeau, B. An introduction to cooja (2019). Available from: https://github.com/contiki-os/contiki/wiki/An-Introduction-to-Cooja.

[45] Vapnik, V. The nature of statistical learning theory (2013).

[46] Velivasaki, T.-H. N., Karkazis, P., Zahariadis, T. V., Trakadas, P. T., and Capsalis, C. N. Trust-aware and link-reliable routing metric composition for wireless sensor networks. *Transactions on Emerging Telecommunications Technologies*, **25** (2014), 539. Available from: https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.2592, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/ett.2592, doi:10.1002/ett.2592.

[47] Wallgren, L., Raza, S., and Voigt, T. Routing attacks and countermeasures in the rpl-based internet of things. *International Journal of Distributed Sensor Networks*, **9** (2013), 794326.

[48] Wikipedia contributors. K-nearest neighbors algorithm — Wikipedia, the free encyclopedia (2019). [Online; accessed 13-October-2019]. Available from: https://en.wikipedia.org/w/index.php?title=K-nearest_neighbors_algorithm&oldid=920309707.

[49] Wikipedia contributors. Principal component analysis — Wikipedia, the free encyclopedia (2019). [Online; accessed 15-October-2019]. Available from: https://en.wikipedia.org/w/index.php?title=Principal_component_analysis&oldid=920334148.

[50] Wikipedia contributors. Support-vector machine — Wikipedia, the free encyclopedia (2019). [Online; accessed 13-October-2019]. Available from: https://en.wikipedia.org/w/index.php?title=Support-vector_machine&oldid=920756090.

[51] Winter, T., et al. IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) (2012). Available from: https://tools.ietf.org/html/rfc6550.

[52] Wood, A. D. and Stankovic, J. A. Denial of service in sensor networks. *computer*, **35** (2002), 54.

[53] Wood, A. D. and Stankovic, J. A. A taxonomy for denial-of-service attacks in wireless sensor networks. *Handbook of sensor networks: compact wireless and wired sensing systems*, (2004), 739.

[54] Ylonen, T. and Lonvick, C. The secure shell (SSH) protocol architecture (2006). Available from: https://tools.ietf.org/html/rfc4251.

[55] Zhang, Y., Lee, W., and Huang, Y.-A. Intrusion detection techniques for mobile wireless networks. *Wireless Networks*, **9** (2003), 545.

[56] Zhou, L. and Haas, Z. J. Securing ad hoc networks. *IEEE network*, **13** (1999), 24.