Modeling and Verification of Lightweight Defense Strategies in IoT Security: A Discrete Event System Approach

Dipojjwal Ray

Modeling and Verification of Lightweight Defense Strategies in IoT Security: A Discrete Event System Approach

Thesis submitted in partial fulfilment
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science and Engineering

by

Dipojjwal Ray

Under the supervision of

Dr. Pinaki Mitra
&
Dr. Santosh Biswas

Department of Computer Science and Engineering
Indian Institute of Technology Guwahati

October 2024

This thesis is dedicated to my Mother, Father and my Wife

- whose blessings, inspiration, unconditional love and support made my path of success.

Declaration

I hereby certify that

a. The work contained in this thesis is original and has been done by myself and the general supervision of my supervisor.

b. The work has not been submitted to any other Institute for any degree or diploma.

c. Whenever I have used materials (data, theoretical analysis, results) from other sources, I have given due credit by citing them in the text of the thesis and giving their details in the references. Elaborate sentences used verbatim from published work have been clearly identified and quoted.

d. No part of this thesis can be considered plagiarism to the best of my knowledge and understanding and take complete responsibility if any complaint arises.

Date : ____/_____/_____ **Dipojjwal Ray**

Place: Guwahati, India

Department of Computer Science and Engineering
Indian Institute of Technology Guwahati
Guwahati - 781039, Assam, India

Dr. Pinaki Mitra
Associate Professor
Ph: +91-361-2582352
Email: pinaki@iitg.ac.in

Dr. Santosh Biswas
Professor
Ph: +91-9957561026
Email: santosh@iitbhilai.ac.in

Thesis Certificate

This is to certify that the thesis entitled **"Modeling and Verification of Lightweight Defense Strategies in IoT Security: A Discrete Event System Approach"** being submitted by **Dipojjwal Ray** to the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, is a record of bonafide research work carried out by him under my supervision and is worthy of consideration for the award of the degree of Doctor of Philosophy of the Institute.

To the best of my knowledge, no part of the work reported in this thesis has been presented for the award of any degree at any other institution.

Date : ____/____/_____ _____

Place: Guwahati, India **Dr. Pinaki Mitra**

Date : ____/____/_____ _____

Place: Bhilai, India **Dr. Santosh Biswas**

(Thesis Supervisors)

Acknowledgments

First and foremost, I express my sincerest gratitude and thanks to my supervisors Dr. Pinaki Mitra and Dr. Santosh Biswas. It has been my honour and pride to be their student. I wholeheartedly appreciate their contributions and ideas. It is because of their help, patience, belief towards me, their efforts and valuable suggestions that my Ph.D. experience has been joyous and worthwhile.

I would also like to express my thankfulness to Prof. Sukumar Nandi for his immense help and guidance which has helped me strive to be a better researcher. I must thank my respected doctoral committee members Prof. Jatindra Kumar Deka, Prof. Hemangee Kapoor and Dr. Aryabartta Sahu for their insightful suggestions and comments that have helped me improve the quality of my thesis work.

I would also like to express my gratitude and thankfulness to Prof. Purandar Bhaduri and Dr. Arnab Sarkar. Without their help and guidance, this journey would have been incomplete. I would also like to thank Dr. Chandan Karfa whose moral support and valuable ideas have helped me overcome tough times. I am also thankful to Dr. Sukanta Bhattacharya for his help every time and his friendly suggestions. I would like to thank heads of the Department of Computer Science and Engineering, Prof. Diganta Goswami and Prof. T. Venkatesh for allowing me use the department facilities and available resources. My first inspiration in the department was Dr. Vijaya Saradhi and I would like to thank him for his invaluable guidance. My research work has also been partially supported by ICPS, Department of Science & Technology (DST), Govt. of India for my involvement in the project entitled “Formal Methods for Modeling and Verification of Intrusion Detection System in Wireless Networks", and I would like to express my thankfulness to them.

Thanks to my adorable seniors who had made this journey fulfilling, who had been with me throughout their stay at IIT Guwahati, namely Dr. Awnish Kumar, Dr. Akash Anil, Dr. Vasudevan, Dr. Sandeep Vidyapu, Dr. Rakesh Pandey, Dr. Rajesh, Dr. Piyoosh, Dr. Surajit Das, Dr. Hema, Dr. Ranajit Senko, Dr. Saptarshi Pyne and Pradeep Sharma. I will also cherish the long tea time chats with my friends, Sameer, Bhargav, Deepankar, Subrata Tikadar, Subrata Nandi, Pradeep Bhale, Ujjwal Biswas, Arijit, Sumita, Partha Pati, Palash, Soumen, Mishra and Akshay. I have spent short yet sweet time with my juniors Nilotpal, Saurav and Soumya. Apart from my department friends, I have cherished long talks and support from Amritava, Saptarshi, Bandhan, Anirban, Ankan and Niloy. Special thanks to my childhood friends, Sanjeeban, Arko, Kunal and Debango who have all mentally supported me throughout. It goes without mentioning the help and assistance I received each time from office staffs, especially Ms. Gauri, Mr. Monojit and Nanu da, whom I would also express my thanks for.

The moments spent will be long cherished and will be ingrained in my memory for a long time. It has been a enjoyable yet tough time. Above all, without the blessings of Almighty and my parents, this journey would never have been possible. Their belief and perseverance has helped me reach this stage. All through the PhD journey, there needed to be decision making at every stage which has helped me grow in not only research but also in life. Lastly, I would like to thank my loving wife for the infinite favours and unwavering support, without whom this journey would have been incomplete. I am truly indebted to all of them.

Guwahati, August 2024 **Dipojjwal Ray**

Abstract

The Internet of Things (IoT) revolution has ushered huge technological benefits and has made future communication and human lives easier. However, the rapid proliferation of IoT introduces numerous security challenges. IoT systems have been shown vulnerable to device-level attacks. Also indubitably, there exists a multitude of network-level attacks that make IoT systems vulnerable due to lack of secure provisions in place. At the device-level, secure IoT devices can be heavily compromised to various side-channel attacks. There exists scan-based side-channel attacks for which the proposed countermeasures are either insufficient, or compromise on testability, or of high-overhead. At the network-level, IoT-specific protocols are prone to varied internal DDOS attacks at each layer. Given the resource-constrained environment, lightweight, accurate and malicious node identification schemes are highly demanding among attack mitigation techniques. For genuine reasons, Intrusion Detection Systems (IDS), a software or hardware component monitoring host or network threats, are widely used to secure IoT systems and deemed suitable for most of such detection or prevention scenarios. The two most popular IDS-based design techniques are Signature based IDS, which use known signatures, and Anomaly-based IDS that use statistical features. However, there exists no known signatures or features in attacks like RPL rank attack, RPL version number attack, 6LoWPAN based fragmentation attacks, CoAP request spoofing and CoAP response spoofing attacks, rendering Signature-based and Anomaly-based methods futile. Basically they generate lots of false positives since the IoT network traffic, operational under attack, cannot be differentiated from the normal traffic. This dissertation presents few novel attack mitigation and attack node location identification mechanisms for IoT security, utilizing controller and IDS implementations, while using various Discrete Event System (DES) based formalisms. DES models are designed for the IoT systems under normal and abnormal conditions. DES based formalisms ensure proofs of correctness and completeness which are preferable. DES security and Fault Detection and Diagnosis (FDD) theoretic properties in Finite State Automata are leveraged for the proofs.

The thesis work comprises four contribution chapters. In the first chapter, we show an Opacity preserving countermeasure using a novel hardware controller unit design that circumvents all state-of-the-art differential scan attacks including a proposed co-relation scan attack that is more effective on an IoT device having AES implementation. The controller uses a mask determination algorithm to selectively allow bit-flipped scan outputs. Our scheme incurs nominal overhead and maintains full testability. Opacity, a security notion in the DES community, is used to formally prove the security of our system modeled in DES. Diagnosability is a property used in classical DES approaches for performing FDD of complex control systems. For the subsequent network-level attack countermeasures, we

adopt this approach in our DES based IDS framework, since faults and attacks have similar properties. In the second chapter, we present a RPL version number attack detection and rank attacker detection as well as identification mechanism that utilizes an intelligent active probing technique and DES based IDS. Our proposed architecture is centralized with inputs from sensing at the RPL leaf nodes. Active probing helps create differentiating sequences of events in normal and attacker specific conditions. I-diagnosis framework of DES is extended to model rank attacker node behaviour incorporating the controllable events (probes). In the third chapter, we present a novel 6LoWPAN fragmentation attacker identification mechanism that utilizes fabricated fragments as active probes. Such attacks are low overhead with an eavesdropping attacker capable of exploiting nodes which are just a single hop away. Consequently, a decentralized DES based IDS is proposed. Decentralized I-diagnosis helps globally diagnose an attack node based on the response generated from the forwarded spoofed fragments. Global I-diagnosability is ensured from the local I-diagnosis. Lastly, we present a CoAP request spoofing attacker identification mechanism using crafted request and response messages. I-diagnosis framework is successfully adapted for detecting an attack node that has launched the request spoofing attack in IoT application layer. However, proposed technique is limiting while detecting an response spoofing attack node, since correctly crafted fragments need to be ensured. Consequently, we adapt and extend the I^2-diagnosis framework of DES to model the crafted fragment indicator event (active probe) along with an empowering event. All the countermeasures are experimentally analysed both in simulation and real testbed. Results show that our schemes are accurate and lightweight compared to existing approaches. Furthermore, we prove the correctness and completeness of our proposed mechanisms.

Table of Contents

List of Figures

Page

List of Tables

Page

List of Algorithms

List of Acronyms

Acronym	Expansion
6BR	6LoWPAN Border Router
6LoWPAN	IPv6 over Low-power Wireless Personal Area Networks
ACC	Accuracy
ACK	Acknowledgement
AES	Advanced Encryption Standard
API	Application Programmable Interface
ATPG	Advanced Test Pattern Generator
CIA	Confidentiality Integraity Availability
COAP	Constrained Appplication Protocol
CON	Confirmable Message
CONMO	Control Message Overhead
COSA	Co-relation Scan Attack
CSO	Current-State Opacity
CUT	Circuit Under Test
DAO	DODAG Advertisement Object
DDOS	Distributed Denial of Service
DES	Data Encryption Standard
DES	Discrete Event System
DFT	Design for Tesatbility
DIO	DODAG Information Object
DIS	DODAG Information Solicitation
DODAG	Destination Oriented Directed Acyclic Graphs
DOS	Denial of Service
DTLS	Datagram Transport Layer Security

DUT	Device Under Test
ECC	Energy Consumption
EU	Energy Utilization
FDD	Fault Detection and Diagnosis
FRAG1	Fragment Header
FRAGN	Fragment
GIS	Geographic Information System
GPS	Global Positioning System
HTTP	Hypertext Transfer Protocol
IC	Integrated Circuit
ICMPV6	Internet Control Message Protocol version 6
ICT	Internet and Communication Technology
IDS	Intrusion Detection System
IIoT	Industrial Internet of Things
IoT	Internet of Things
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
JTAG	Joint Test Action Group
LFSR	Linear Feedback Shift Register
LLN	Low-power and Lossy Networks
LR-WPAN	Low Rate Wireless Personal Area Networks
MAC	Message Authentication Code
MITM	Man-in-The-Middle
MKR	Mirror Key Register
MQTT	Message Queuing Telemetry Transport
MTU	Maximum Transmission Unit
MUX	Multiplexer
NON	Non Confirmable Message
NTA	Network Traffic Analysis
OCP	Objective Code Point

OF	Objective Function
PAN	Personal Area Network
PDR	Packet Delivery Ratio
PRNG	Pseudorandom Number Generator
QOS	Quality of Service
RAD	Rank Attack Detection
RAI	Rank Attack Identification
RAM	Random Access Memory
REST	REpresentational State Transfer
RFID	Radio Frequency Identification
ROM	Read Only Memory
RPL	IPv6 Routing Protocol
RSA	Rivest Shamir Adleman
RST	Reset Message
RTOS	Real Time Operating System
SCAL	Scalability
SDN	Software defined Networking
SIB	Segment Insertion Bit
SMQTT	Secure Message Queuing Telemetry Transport
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
THP	Throughput
TKL	Token Length
TLS	Transport Layer Security
TNR	True Negative Rate
TPM	Tamper-Proof Memory
TPR	True Positive Rate
UDP	User Datagram Protocol
VNAD	Version Number Attack Detection
WI-FI	Wireless Fidelity
WSN	Wireless Sensor Network

भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
Indian Institute of Technology Guwahati

List of Symbols

Symbol	Description
H	DES model
Σ	Set of events of the DES model H
Σ_o	Set of observable (unobservable) events of the DES model H
V	Set of model variables of the DES model H
$\Im$	Set of transitions of the DES model H
τ	A transition $\tau \in \Im$
Y	Set of states of the DES model H
Y_0	Set of initial states of the DES model H
σ	Event on which a transition is enabled
$L(H)$	Set of all traces *generated* in H
$Y_S(Y_{NS})$	Set of secret (non-secret) states of the DES model H
(I, I')	An S-box input pair
(P, P')	A plaintext input pair
$R(p)$	Round output value obtained from applying plaintext p
IHW_i	Vector of S-box input pairs generating Hamming weight i
$Mask(P, x)$	A mask computation function
$FE(FD)$	Flip Enable (Disable) event

CHAPTER

1

Introduction

The idea behind Kevin Ashton's 1999 promotional slogan, 'Internet of Things', has evolved into a revolutionary technical paradigm today, more than twenty years down the line [1]. With a plethora of technologies, Ashton's vision of everyday goods, home appliances and mobile devices being interconnected via the Internet, industrial machineries communicating seamlessly aiding production, and deployed sensors helping in setting up smart cities, has now come to practice. The traditional Internet has emerged as the Internet of Things (IoT) and has touched every nook and corner of our globe, easing human living incredibly [2, 3].

The IoT is more often described as a system of interconnected heterogeneous entities. The term entity applies to uniquely addressable 'things' (or smart objects), people and the environment. Therefore, IoT vision can be broadly understood as service provisioning for humans and things by enabling their communication over the Internet. The IoT hinges upon a conglomerate of different technologies starting from radio frequency identification (RFID) and wireless sensor devices to Bluetooth, Wireless Fidelity (Wi-Fi), Zigbee, Low-rate Wireless Personal Area Networks (LR-WPANs) and advanced ones like Big-data, Blockchains, and Cloud Computing [4]. Nonetheless, IoT encompasses a large heterogeneity in devices which communicate as per the various standardized protocols and primarily caters to harvesting and exchanging data for collection and analysis without the need for human intervention. Furthermore, IoT has been widely acclaimed as one of the cornerstones of the 21st century Information and Communication Technology (ICT) development.

By enabling smooth connectivity, IoT has benefited people technologically, economically

and on a societal level. It has brought in a dramatic change to communities with 'things' ranging from implantable medical devices, merchandise tags, smart thermostats to drones and automobiles with sensors built-in. IoT is helping us live more comfortably, automating tasks such as turning lights on, adjusting thermostats and even turning off equipment when unnecessary, thereby saving time and energy. Smart devices like wearables, alarms, camera systems, smart home appliances offer various functionalities that improve the quality of life [1]. Physical boundaries have dissolved with IoT that has made us stay connected even from far away places, eventually becoming an indispensable component in our lives.

1.1 IoT Applications

IoT offers a wide array of applications that has boosted productivity and made us more informed with weather forecasts, traffic tracking, etc. IoT applications such as monitoring security systems, detecting fire or calling for help in emergencies has improved the overall safety of mankind. A few examples of IoT applications include:

- Smart home: IoT devices (such as smart locks, baby monitors, and fire detectors) can be deployed at home and communicate with each other wirelessly. These devices can also be accessed remotely through a home gateway [5].

- Smart healthcare: IoT devices can be used to collect, transmit, and store patient data, such as heart rate. This data can then be sent to a hospital server for diagnosis and tracking [6].

- Smart transportation: Smart vehicles can communicate with each other (vehicle-to-vehicle), with external infrastructure (vehicle-to-infrastructure), and with pedestrians (vehicle-to-pedestrian) over wireless networks. This allows smart vehicles to detect traffic conditions, manage their speed, and exchange data to improve safety and efficiency [5].

- Smart agriculture: IoT sensors can be used to remotely monitor and control temperature, humidity, irrigation, soil moisture, and micro-climate conditions in agricultural settings. This can help to increase production and quality while reducing costs.

- Smart industry and Mission-Critical Systems: Industrial IoT (IIoT) uses machine-to-machine technology to automate manufacturing processes with minimal human intervention. IIoT aims to improve production efficiency and reliability while reducing costs and quality issues.

- Smart retail: IoT sensors can be attached to retail products to track their status and location. This information can be used to develop smart shopping systems that provide improved services to customers and attract new customers.

- Smart grid: Smart grids use IoT devices to measure, monitor, and manage electricity consumption. This enables more efficient and reliable electricity management, reduces energy costs, and improves grid reliability [7].

1.2 IoT Technologies and Protocols

The Internet of Things (IoT) is embedded with sensors, software, and other technologies for the purpose of connecting and exchanging data with other devices and systems over the internet. A number of technologies facilitate and enable the IoT, including:

- IPv6: It is the next generation of the internet protocol (IP), the underlying protocol that powers the internet. IPv6 was developed to address the limitations of the current IP protocol, IPv4. One of the main limitations of IPv4 is that it has a limited address space, which means that there are not enough IPv4 addresses to accommodate all of the devices that are being connected to the internet. IPv6 solves this problem by providing a much larger address space. IPv6 also has a number of other features that make it well-suited for the IoT. For example, IPv6 supports auto-configuration, which means that IoT devices can automatically configure themselves with IP addresses when they are connected to the network. This is important for the IoT because it can be difficult or impossible to manually configure IP addresses on large numbers of IoT devices.

- RFID: It is a technology that uses radio waves to identify and track tags attached to objects. RFID tags can be used to track the movement of goods, people, and animals, and to collect data from sensors [8]. RFID tags are made up of two main components:

a microchip and an antenna. The microchip stores the tag's unique identifier and other data. The antenna transmits and receives radio signals. RFID readers are used to read data from RFID tags. RFID readers emit radio waves that power the tag's microchip and cause it to transmit its data. The RFID reader then receives the tag's data and decodes it. RFID is a widely used technology in the IoT. For example, RFID tags are used to track the movement of goods in supply chains, to track the location of patients in hospitals, and to track the movement of animals in farms.

- WSN: They are a network of small, low-power devices that can sense and collect data from the environment. WSNs are often used in IoT applications to monitor environmental conditions, such as temperature, humidity, and air quality. WSN nodes are typically equipped with sensors, a microcontroller, and a radio transceiver. The sensors collect data from the environment, the microcontroller processes the data, and the radio transceiver transmits the data to other nodes in the network. WSNs are well-suited for IoT applications because they are low-cost, energy-efficient, and scalable. WSNs can be deployed in a variety of environments, including remote and hazardous areas.
- IEEE 802.15.4: It is a protocol standard that specifies the physical layer and media access control (MAC) requirements for low-power wireless personal area networks (PANs). It is designed for low-speed, low-cost communication between devices. The IEEE 802.15.4 standard can be used with the IPv6-based Low Power Wireless Personal Area Network (6LoWPAN) protocol to build wireless embedded networks for the Internet of Things (IoT) [4]. The basic communication range is 10 metres with a transfer rate of 250 Kbit/s. The higher layers of the protocol stack are not defined in the standard.

In addition to the core technologies, other enabling technologies for the IoT include:

- Cloud computing: Cloud computing provides a scalable and cost-effective platform for storing, processing, and analyzing data from IoT devices.
- Global Positioning Systems (GPS): GPS can be used to track the location of IoT devices, which is essential for many applications, such as asset tracking and vehicle management.

- Service-oriented architectures (SOAs): SOAs provide a flexible and scalable way to develop and integrate IoT applications.
- Geographic information systems (GIS): GIS can be used to visualize and analyze data from IoT devices, which can help businesses and organizations to make better decisions.
- Cellular devices (3G/4G/5G): Cellular networks provide a reliable and ubiquitous way to connect IoT devices to the internet.

The Internet is essential for the operation of IoT systems. Therefore, a TCP/IP protocol stack similar to the one used for the Internet is used for IoT systems. Some of the standard protocols defined for IoT ecosystems include:

- RPL: Routing Protocol for Low-Power and Lossy Networks (RPL) is a distance-vector routing protocol designed for low-power and lossy networks (LLNs) [9]. LLNs are mostly IoT networks that consist of devices with limited resources, such as power, bandwidth, and processing power. RPL is designed to be scalable, efficient, and reliable, even in challenging network conditions.
- 6LoWPAN Protocol: 6LoWPAN stands for IPv6 Low-Power Wireless Personal Area Network. It is a protocol that allows IPv6 packets to be transmitted over low-power wireless networks, such as IEEE 802.15.4. 6LoWPAN is designed to meet the requirements of low-power consumption devices and weak computing capabilities nodes and sensors [10, 11]. It works by compressing IPv6 packets to make them smaller and easier to transmit over low-power wireless networks. 6LoWPAN also provides a number of other features such as header compression, fragmentation, neighbour discovery that makes it suitable for IoT applications.
- Constrained Application Protocol (CoAP): CoAP is a specialized web transfer protocol for use with resource-constrained devices [12]. It is designed to be lightweight and efficient, and to support the same basic operations as HTTP, such as GET, POST, PUT, and DELETE. CoAP is well-suited for IoT applications because it can operate over low-bandwidth and unreliable networks, and it has a small memory footprint. It is based on a subset of the HTTP protocol, making it interoperable with HTTP.

CoAP runs over UDP, which reduces bandwidth requirements and supports multicast and unicast communication.

- Message Queue Telemetry Transport (MQTT): MQTT is another lightweight application layer protocol designed for machine-to-machine (M2M) communication. It runs over TCP and is based on the publish/subscribe model. MQTT is designed to minimize bandwidth and power consumption, making it ideal for IoT applications.

The basic architecture of the IoT ecosystem can be broadly categorized into device-level (or perception layer), network-level and application-level. The categorization of the IoT ecosystem into device-level, network-level, and application-level provides a useful framework for understanding the different components and layers involved in IoT implementations. We here give an overview of each level:

- **Device-Level:** At the device level, IoT focuses on the physical objects or things that are equipped with sensors, actuators, and connectivity capabilities. These devices collect data from the surrounding environment through sensors and interact with the physical world through actuators. They can include various types of devices such as sensors, wearables, industrial machinery, vehicles, and consumer devices. Device-level considerations involve hardware design, sensor integration, power management, and firmware development.

- **Network-Level:** The network-level in IoT refers to the communication infrastructure that enables connectivity and data exchange between devices. This layer encompasses the protocols, connectivity technologies, and network architecture required to establish reliable and secure connections. Wireless technologies like Wi-Fi, Bluetooth, Zigbee, cellular networks (3G, 4G, 5G), and low-power wide-area networks (LPWAN) play a crucial role in connecting devices. The network-level also involves considerations such as data transmission, bandwidth, latency, scalability, and network protocols to ensure seamless connectivity and efficient data transfer.

- **Application-Level:** The application-level in IoT encompasses the software, protocols and services that utilize the data collected from devices to deliver specific functionalities and value-added services. This layer involves data processing, analytics, and application

development. IoT applications can range from simple data visualization and monitoring dashboards to complex analytics platforms that leverage machine learning and artificial intelligence to derive insights and enable intelligent decision-making. Application-level considerations involve data storage, processing capabilities, protocol security, user interfaces, and integration with other systems.

1.3 Context

As correctly hinted by Moore's law, the number of transistors in a given IC has increased by at least three-folds thereby reducing costs and catalyzing the miniaturization of integrated circuits (ICs) to their tiniest possible orders [13]. With the ubiquitous growth of IoT, the future is shaping towards quantification and digitalization of the physical world. If we go by Gartner's prediction, the number of IoT devices is expected to globally reach 75 billion by 2030 [14]. But with such growth, enormous challenges cloud IoT as well. An exponential increase in the number of IoT devices indicates a proportional scale up of the possible vulnerabilities.

The three levels of the IoT architecture are interconnected and interdependent. Each of these levels are susceptible to unique array of attacks as demonstrated in multiple studies. Device-level threats address the security attacks made to IoT devices and sensors. Network-level attacks concern the vulnerabilities of the IoT communication protocols. Application-level threats comprise of attacks pertaining to end-user applications and softwares.

1.3.1 Device-level attacks

Device-level attacks have their primary motif to compromise the hardware elements of a device, with a focus on its ports, memory, power source, and other components. These attacks have the potential to disrupt the device's operation and modify its behavior. They are typically categorized as physical attacks, firmware attacks, and encryption attacks.

Physical attacks necessitate physical access to the target devices and encompass various tactics, including:

- Radio Frequency Interference or Jamming: This involves creating and transmitting disruptive signals over radio frequencies, mainly causing Denial of Service (DoS)

attacks.

- Physical Penetration: Adversaries exploit exposed device interfaces like JTAG to gain direct access to memory, sensitive keys, passwords, configuration data, and other critical parameters [15].
- Physical Damage or Tampering: Attackers inflict physical harm on the device to disrupt its service availability.

Firmware attacks, on the other hand, revolve around manipulating the software installed on IoT devices, leading to:

- Backdoors: Attackers modify device firmware to insert code that grants them remote access when the device connects to the network [16, 17].
- Unencrypted Information: Attackers reverse engineer firmware to uncover encrypted data like passwords, API keys, and public-key certificates. This information enables them to intercept device communication and access sensitive user data [17, 18].
- Malicious Firmware: In addition to creating backdoors, attackers may modify firmware to launch various attacks, including Distributed Denial of Service (DDoS) attacks.

Encryption attacks specifically target encryption mechanisms utilized within IoT networks [19], including:

- Side Channel Attack: Attackers eavesdrop on side channel emissions to bypass device encryption by intercepting encryption keys during device access.
- Cryptanalysis Attacks: These attacks focus on breaking the encryption scheme by analyzing ciphertext or plaintext. Examples include Known-plaintext attacks, Chosen-plaintext attacks, Chosen Cipher-text attacks, and Ciphertext-only attacks.
- Ciphertext Only Attack: This approach allows attackers to access ciphertext and determine the corresponding plaintext.
- Known Plaintext Attack: Attackers exploit knowledge of plaintext and partial ciphertext to decrypt the remaining ciphertext.

1.3.2 Network-level attacks

Network-based attacks within IoT ecosystems can be executed remotely from anywhere in the world, without the need for physical access to the devices. These attacks encompass:

- Passive Traffic Analysis: Adversaries passively monitor network activity to intercept sensitive user data, collecting information by observing network traffic [20]. They can also disrupt traffic flow, a technique known as Man-in-the-Middle (MITM) attacks.

- Spoofing: Attackers impersonate legitimate devices to make malicious entities appear as genuine ones, facilitating the launch of attacks [21].

- Rank or Sinkhole: A compromised device lures network traffic by broadcasting fake routing updates. These attacks can be exploited to launch additional attacks [22].

- Denial of Service (DoS) and Distributed Denial of Service (DDoS): These are the most prevalent attacks in IoT networks, capable of reducing, interrupting, or completely disabling network communications [23, 24]. Ranging from simple jamming to sophisticated assaults, they can be initiated remotely and are challenging to detect until the network or service becomes unavailable.

- Sleep Deprivation: In IoT networks, devices can enter power-saving modes to conserve energy. This attack aims to prevent devices from entering these modes by continually sending traffic, depleting their battery resources.

- Man-in-the-Middle (MiTM) Attack: This involves an attacker establishing independent connections with victims B and C, deceiving them into believing they are communicating directly [25]. The attacker intercepts messages between B and C, rerouting them in the process.

- Eavesdropping Attack: Also known as sniffing or snooping cyberattacks, this exploit takes advantage of unsecured network communication to intercept data transmitted over the network. Attackers can implement this type of attack by leveraging weak network connections and using sniffer software on a server or connected client computer to capture network data.

1.3.3 Application-level attacks

- Man-in-the-Middle (MiTM) attacks pose a significant threat to the security of the application layer protocols: (a) CoAP Protocol: Datagram TLS (DTLS) was introduced to provide confidentiality, authentication, and integrity for CoAP. However, the limitations of DTLS can be considered security weaknesses in the CoAP protocol. (b) MQTT Protocol: To secure data transfer using MQTT, the Secure Socket Layer (SSL) was introduced, employing asymmetric cryptographic techniques for data encryption and decryption. Despite this, SSL remains vulnerable to MiTM attacks.

- Lack of Standardization: Secure MQTT (SMQTT), an extension of MQTT, aims to enhance data transfer security. SMQTT relies on secret keys for encryption and decryption, but key generation and encryption algorithms lack standardization.

- DDoS Attacks: DDoS attacks target IoT networks by exploiting protocol behaviors, overwhelming target servers with traffic to prevent processing legitimate requests. These attacks infiltrate the application layer and flood web servers with HTTP requests, typically at lower rates to evade detection. Examples include DNS service-based attacks and HTTP flooding.

- Software Attacks: Software attacks involve invading network software programs to compromise network devices [17, 26]. Attackers exploit software vulnerabilities and may impersonate or manipulate legitimate users to gain access to IoT systems. The absence of robust user authentication has led to notable IoT attacks like Bashlite and Mirai.

Securing IoT hence requires securing each of the levels. Moreover, most IoT platforms need to satisfy resource-constrained requirements of devices, like sensors and actuators, and use low-energy communication technologies. The primary objective therefore lies in managing the complexity of the interconnecting infrastructure of these devices without making any compromise to security. As IoT devices are becoming more accessible and affordable with various zero-day attacks on the rise, the need for security and reliability is therefore naturally demanding [27]. Securing industrial infrastructures and mission critical systems become more challenging since malicious entities can gain access to personal and/or

private information about people, vehicles, and systems. Moreover, life saving and emergency infrastructures may face serious breakdown. Notwithstanding the growing research to secure these levels, securing IoT remains a daunting task due to several vulnerabilities and challenges that are typically posed.

1.3.4 IoT Security Challenges

There lies three primary reasons for the vulnerability of IoT devices to cyber attacks:

- Heterogeneity of devices: The wide variety of IoT devices having different shapes and sizes, with different operating systems, hardware capabilities, and security features, makes it difficult to implement security mechanisms that are fit for all aspects. For example, a security measure that is effective for a smart thermostat may not be effective for a security camera.

- Limited computational resources: IoT devices often have limited computational power, memory, and battery life, which restricts their ability to run complex security measures [28]. For example, a security measure that requires encryption or decryption may be too computationally expensive for an IoT device with limited resources. It is infeasible for IoT devices to execute computationally heavy and delay-prone security tasks.

- Difficulty in updating software: It can be difficult or impossible to update the software on IoT devices or apply patches, especially those that are deployed in remote or inaccessible locations [29].

Although ongoing research provides meaningful insights in improving the security of IoT systems, they are also faced with several intrinsic challenges that require attention and efforts:

- High requirements of emerging technologies: Some emerging technologies and approaches, such as blockchain, homomorphic encryption, searchable encryption, and machine learning algorithms, require high processing and storage capabilities. This makes it challenging to trade-off between security and performance in IoT infrastructure.

- Dynamic and heterogeneous environment: IoT environments are dynamic, heterogeneous, and large-scale. This requires adaptive trust models to enable devices to recognize trustworthy nodes. Fog computing can be used to achieve different security requirements in IoT environments by providing real-time and latency-sensitive services. However, it is challenging to ensure that all joining fog nodes are trusted, as fog nodes do not have any information about each other. Additionally, it is important to select trustworthy fog nodes, as users often have multiple fog nodes available to cooperate for guaranteeing IoT services.

- Scalability of centralized SDN architecture: The centralized SDN architecture cannot deal with a large number of IoT devices. Additionally, SDN-based solutions are not efficient in highly dynamic IoT environments, such as vehicular networks. Therefore, it is necessary to enforce scalability in SDN networks.

- Privacy and security of data transmission: IoT devices generate and exchange a massive amount of data, including sensitive data. Blockchain technology can efficiently address the scalability issue due to its distributed architecture. However, the blockchain does not ensure the privacy of transactions and is prone to data leakage. Additionally, fog nodes in fog computing-based architectures are responsible for forwarding data to the cloud. If fog nodes are not trustworthy or compromised by an adversary, they can disclose personal information.

- Key management in scalable IoT environments: Data transmission can be secured using encryption techniques. Encryption of transmitted data prevents intruders from revealing the content of messages. This approach can be applied when the communication parties share encryption/decryption keys. In symmetric encryption (i.e., block ciphers, stream ciphers, and hash functions), the key must be pre-distributed or securely communicated. However, key management, including distribution, agreement, update, and revocation, remains a significant challenge in scalable IoT environments.

To address the challenges discussed in the previous subsection, it is therefore necessary to develop security solutions that are tailored to the specific needs of IoT ecosystems. Solutions must also be lightweight and efficient, designed such that it is easy to deploy and maintain. Additionally, it is also important to develop mechanisms for securely updating

the software on IoT devices, even those that are deployed in remote or inaccessible locations. To this end, Intrusion Detection Systems (IDS) have been one of the most efficient and valued security solutions for the detection of malicious network behaviour [30, 31].

1.3.5 Intrusion Detection System (IDS)

An IDS is a software application or hardware appliance that monitors network traffic and alerts the system administrator if suspicious activity is found. Designing an IDS requires a data collection module, an analysis module for processing of the collected data, and a reporting mechanism to notify network administrators.

IDSs can be classified into several categories based on their deployment, detection methods, architectures, and deployment time. Based on the nature of their deployment, there are three main types of IDSs:

- Host-based IDS: This type of IDS is installed on individual devices, such as computers and servers.
- Network-based IDS: This type of IDS is installed on a network and monitors all traffic flowing through the network.
- Hybrid IDS: This type of IDS combines the features of host-based and network-based IDSs.

IDSs can also be classified based on their detection methods as:

- Signature-based IDS (or, specification-based): This type of IDS uses a database of known attack signatures to identify malicious traffic.
- Anomaly-based IDS: This type of IDS monitors network traffic and identifies anomalies, which are deviations from normal behavior.
- Hybrid IDS: This type of IDS combines the features of signature-based and anomaly-based IDSs.

IDSs can be classified based on their deployment architecture as:

- Centralized IDS: This type of IDS is deployed on a single server and monitors all traffic flowing through the network.

- Distributed IDS: This type of IDS is deployed on multiple servers and monitors different segments of the network.

Finally, IDSs can be classified into two main categories based on their deployment time:

- Real-time IDS: The IDS detects attacks in real time as they are happening.
- Offline IDS: The IDS analyzes historical network traffic data to detect attacks that may have occurred in the past.

Proposed security solutions range from user-level solutions, protocol modifications, cryptographic solutions, machine learning. Still, given the context of IoT, chosen security standards also suffer from some demerits. For example, machine learning and deep learning based solutions sometimes require extensive training and DL model fitting time, and are greedy in terms of computation and storage [32, 33]. Cryptography based techniques are highly resource exhaustive while using heavy encryption algorithms [34, 35]. Authentication based techniques and using digital certificates also incur significantly high resource overhead due to authentication, may need protocol modifications and face issues with certificate management [36, 37]. Solutions that are protocol based require modification of the protocol policies [38, 39, 40]. User level solution methods sometimes need proprietary hardware support too [41]. IDS based solutions are less heavy and are naturally chosen for intrusion detection purposes that generate alert on attack detection or identification. However, traditional IDSs are not well-suited for IoT environments. This is because IoT environments are highly dynamic and heterogeneous, with a wide range of devices and protocols. Traditional IDSs need to adapt to the unique security challenges of IoT devices, such as their limited resources and their susceptibility to physical attacks.

A security solution needs to restore the compromised CIA security requirement namely, confidentiality, integrity and availability. Furthermore, standard security implementations of IoT can be overwhelmed by attacks since a single malicious bot node can be equipped enough to disrupt a whole network infrastructure. Despite significant research advancements in IoT security, given the growth of the IoT industry, it is still nascent. There exist several attacks that pose serious threats to IoT devices and networks, either without a solution or having further scope of an improved mitigation or detection methodology. More research is

therefore essential to strengthen IoT security; *this dissertation aims to close the growing gap by proposing solutions for resource-constrained environments.*

1.4 Motivation

IoT devices popularly consist of scan chains for testing purposes. As a result enhanced observability and controllability of the internal register contents of the device is induced. There exists a class of intrusive (non-invasive) device-level attacks on cryptographic IoT devices that makes use of this testability-induced vulnerability to leak out confidential contents. In a scan-based side-channel attack, the secret key values can be easily differentiated from the non-secret ones, inflicting loss. On the contrary, there exist a class of non-intrusive network-level attacks and application-level attacks where the attack traffic resembles the normal traffic and cannot be differentiated, making attack detection challenging. Two characteristic properties of these attacks that make them subtle are:

- They are low overhead attacks; they either make use of the protocol-induced update mechanism in which regular control packets are fabricated to misuse the feature or require just a single fragment or application packet to be launched, while other network traffic statistical properties are left intact. So an anomaly is unnoticeable on occurrence of such attacks.

- The sequence of fragment-level or packet-level communication traffic across attack and normal scenarios are indifferentiable. So development of a signature pattern or anomaly pattern is not only difficult but also high-false positives may ensue.

These attacks either create differentiable characteristics that make it challenging to preserve a secret, or adversely affect the network or device performance in spite of generating attack behaviour that is indistinguishable from the normal making detection challenging. This is our first motivation; proposing a countermeasure for such attacks overcoming the discussed challenges.

Existing scan chain protection methods employ encryption methods or obfuscation techniques which either hamper full testability or are of high overhead. Moreover, there exists a class of recently proposed scan-based attacks with no solutions proposed in the

literature. Existing solutions to mitigate the discussed network-level and application-level attacks mostly use encryption techniques, machine learning based approaches, protocol based approach, all of which have their inherent limitations like high overhead, change in protocol policies, extensive training time in an IoT environment. IDS based solutions are more preferable in this regard. But given a resource constrained environment, they do not all comply with all the desired parameters like QoS, response time, scalability, mobility that a solution needs to guarantee. Hence, applicability of such solutions are limited. Therefore designing more resource friendly and energy-efficient techniques suitable for the IoT ecosystem forms the second motivation of our work.

Signature-based IDS works by using a database which stores knowledge sets of attack patterns (or signatures) and compares them with monitored network (or system) activity. On finding a match, it generates an alert to signal malicious activity detection. Though signature based IDS do not generate excessive false alarms and prove effective in attack detection, yet it suffices to detect known attacks only. Hence such IDS require constant updation with new attack signatures. Moreover, there exist many such IDS that use fixed signatures in their design, making them not capable enough to detect attack variants. It is noteworthy here that IoT attacks with newer attack surfaces are cropping up on a per-day basis. On the other hand, anomaly based IDS have attack detection ability without having acquired knowledge of attack behaviour specifications. It produces information which helps define signatures for detection by misuse detectors. However it generates increased false alarms due to the unpredictability induced by networks and users. Though anomaly based IDS use statistical learning approaches, characterizing a normal behaviour pattern requires them to use training sets of network events extensively, which is not an easy task. The above mentioned attacks in IoT ecosystem, namely differential scan attack, co-relation scan attack (COSA), RPL rank attack, RPL version attack, 6LoWPAN fragmentation attacks, CoAP request and response spoofing attacks evade detection by such IDS or are not accurate enough, leaving scope of improvement. Moreover, mostly proposed IDS schemes resort to attack detection only. Attack node identification helps minimize chances of launching fresh attacks. This is our third motivation; proposing near accurate IDS solutions that also help identify attack node location for the discussed non-intrusive network-level and application-level attack vulnerabilities.

DES is a suitably abstracted modeling paradigm where complex system dynamics can be naturally modeled as a set of discrete states and event driven transitions occurring asynchronously over time. DES has found much success in aspects of failure detection and diagnosis (FDD) and security. It involves modeling the system behavior under normal and abnormal (fault or attack) conditions. Opacity is a security notion coined in the DES community and can be used to perform security analysis of systems. A state estimator is constructed that helps perform security analysis of attack mitigation techniques or generate alerts on attack detection or identification. Formal verification of IDS approaches is also lacking in IoT literature. As a third motivation, intelligent techniques using DES are used to formally verify security and prove correctness and completeness of our proposed strategies. Evaluation of proposed approach and comparison is also performed to show the applicability of our work. We next discuss the preliminaries of DES.

1.5 Preliminaries of Discrete Event System (DES)

1.5.1 DES Security

There is a strict rise in security and privacy concerns today with the emergence of cyber-physical systems, shared online services and shared infrastructures. Security attacks can be classified broadly into two categories based on the nature of intrusion: one that depends on potentiality of the intruder and the other that does not. The second category, Information flow, sometimes includes a passive attacker who observes information through some information leakage path. Various different information flow properties like anonymity, secrecy, privacy, and non-interference have been studied.

Opacity is a security notion based on the property of Information flow [42], [43] . The opacity problem was first studied in the context of labeled transition system and then adapted to Petri Nets and finite automata models. It has also been studied in the probabilistic setting. Opacity enforcing mechanisms has been studied in the light of supervisory control, and has also been applied to different security problems of location based services, coverage analysis of mobile agent trajectory, ship information systems and pseudo-random generators.

Opacity in DES modeled in the framework of non-deterministic finite state automata is defined as a property over system states or executions where the truth value of a predicate

is to be kept secret from a malicious adversary (or multiple adversaries) who observes the system through some projection map, let P. An opaque system is such that the attacker is never able to infer the predicate truth. That is, after having observed an execution of the system, the intruder cannot determine if the state of the system belongs to the secret state. In *current-state opacity* the secret is defined as a subset of system states. Given a secret, a system is current-state opaque if the malicious observer (or intruder) is not able to determine if the current system state is a member of the secret states. Therefore, current-state opacity requires that any estimate of the intruder about system's current state lies outside the secret states. In other words, given an observer projection mapping on the system events or states, whenever the system enters a secret state, there exists another equivalent state that belongs to the non-secret states.

1.5.2 Failure Diagnosis and Diagnosability (FDD) of DES and IDS

The problems of opacity and diagnosability are related. Whereas opacity requires information to be hidden from an external malicious agent such that intended secret information is protected, diagnosbaility requires the diagnoser to be provided with enough information.

A fault in a system results in an undesirable deviation of one or more of its components from their intended or normal functioning. The deviation is sometimes within tolerable limits. In case the deviation is critical, then there is system breakdown or failure. Fault diagnosis comprises of three objectives, fault detection, isolation and identification. Fault diagnosis techniques generally rely upon system being modeled as per their normal behaviour and also as per their faulty behaviour. Also, depending on the types of faults, they can be partitioned into failure modes. Faults are typically used as an additional input (event) for system modeling purposes. In discrete event systems, faults can be treated as permanent, intermittent or incipient depending on their nature. The diagnosis problem is essentially concerned with determination of fault type (a particular unobservable or unmeasurable event) from observed sequence of events depending on the model under consideration [44], [45]. Thus fault diagnosis needs to construct a diagnoser and is directly related to state observability, which requires building an observer automaton. The diagnoser helps to determine if the system under operation is normal, faulty or uncertain.

Classical DES theory has been largely adopted in systems for Fault Detection and

Diagnosis (FDD) [46, 44, 45]. Motivated from fault diagnosis, DES based IDSs have been successfully used in network attack detection [47, 48]. The characteristic similarities of network attacks and faults in DES literature is what motivates its usage. The basic idea is to develop a model for the normal functioning of the network and another for attack (fault) behavior. Additionally, multiple fault types in DES literature are diagnosed by developing exclusive fault DES models corresponding to each fault type. Each fault type leads to unique deviations from the normal behavior. Analogously, we augment traditional DES based IDS with attack types in our work here. It may be noted that an attack type corresponds to behavior of the network under the influence of a particular attacker. Attack type DES models corresponding to the location of the attacker are modeled. In DES based IDS a DES diagnoser is used as our IDS engine. It is a state estimator automaton which is constructed from the knowledge of normal and attack type DES models. The diagnoser observes system event traces and gives a decision on the system condition being normal or under attack by generating alerts. To summarize, by using DES based IDS, and given all possible attack instances, it can be ascertained if an attack can always be exclusively identified, correctly and completely.

1.6 Research Questions

This dissertation work was carried out with the objective of providing feasible answers to these following questions:

- Are there attacks where a cryptographic IoT device leaks secret since secret values of internal register contents differ from the non-secret values making the device vulnerable?
- What are some of the IoT attacks where attack behaviour is same as normal and can not be differentiated?
- Can the above attacks be detected using certain mechanisms that create differences between normal and attack behaviour or help secure values resemble the non-secure values in a cryptographic IoT device? Can such a controller using such mechanisms be synthesised or IDS be designed using Discrete Event System (DES)?

- Will DES notions of diagnosability or opacity notions hold true in such solution techniques?

- Are the solutions of low overhead and cost-effective? How good are the performance results compared to state-of-the-art solutions?

1.7 Contributions

Throughout this dissertation, various DES paradigms are adapted for the design and verification of lightweight strategies for mitigation of IoT attacks. Though the paradigms vary, still the fundamental DES philosophies are almost same: 1) designing a normal DES model for the system normal behaviour 2) designing an attack DES model or a flipped DES model for the system abnormal behaviour 3) designing a state estimator using the normal model and attack or flipped model. The state estimator observes system values to determine the system dynamics to be of normal or abnormal behaviour. Additionally, using DES based framework ensures formal verifiability, which means it can be ascertained if all of the possible attack cases are detectable, thereby making the schemes robust.

We discuss scan-chain based side-channel attack, RPL version and RPL rank attacks, 6LoWPAN fragmentation attacks and CoAP request/response spoofing attacks. We briefly describe the related literature that prevent/detect such attacks and any issue/shortcoming of them. We then demonstrate how our scheme uses effective mechanisms and lightweight strategies leveraging DES frameworks for mitigation, detection and identification of these attacks without requiring to modify protocols standards, encryption, installing proprietary hardware or extensive training. The contributory chapters are summarized sequentially in the following subsections.

1.7.1 Scan-based Side-channel Attack

Testing of manufacturing defects has seen a surge recently with growing demands in micron and sub-micron IoT devices in the semiconductor industry. Among the techniques prevalent in the industry, scan-based DfT is the most popular one. The internal scan chain comprising of serially connected flip-flops (scan cells) is made more observable and controllable, which effectively facilitates testing. While in test mode, a designated tester provides a relevant

input to stimulate the internal nodes and observes corresponding output responses. Again, attackers look out for various means to gain access to the internal scan elements. This hands them an opportunity to leak sensitive information embedded within a cryptographic IC of an IoT device. Side-channel attacks can be precisely classified to be scan-based if the above inherent vulnerability is made use of.

Universal standard cipher, AES, was shown to be vulnerable to scan-based attacks [49] after it was already reported to be effective on a DES cipher implementation [50]. Switching a crypto-chip to test mode after having run the chip in normal mode for a few cycles was shown to be effective in the retrieval of the secret embedded key. A differential analysis technique applied on a set of related plaintext input and cipher output pairs helped infer the secret information. This could be realized due to observing the internal scan cell contents corresponding to controlled input pairs applied beforehand. Most of these discussed attacks exploit only the unique Hamming weights corresponding to one round output Hamming weight distribution by applying all possible input pairs with a fixed input difference targeting a particular AES input byte. Subsequently, this style of attack was extended to public key cryptosystems such as elliptic curve cryptography (ECC) [51], Rivest-Shamir-Adleman (RSA) [52] circuits, and in lattice-based public key cryptosystems [53].

These attacks are limited in the way that only the unique hamming weights are the prime target of the attackers. Having established a correct mapping between words and scan cells and algorithmically determining the correspondence between scan cells and bytes, differential scan attack will be easy to launch. Non-unique hamming weight input pairs are also susceptible to attacks with a much trivial complexity. We present a new class of scan attack, *Co-relation scan attack (COSA)*, that can work on non-unique hamming weights which removes the constraints of the attacker to look for only unique hamming distances. A comprehensive attack analysis is performed on all possible hamming weights by exploiting the scan functionality of AES crypto-system. We also show the theoretical limits of the proposed attack along with extensive simulation results on AES for validation.

Numerous countermeasures have been proposed to prevent scan-based side-channel attacks in the recent past. In [54, 49], a mode reset countermeasure was proposed to flush all sensitive data contained in a scan chain during a switch from normal to test mode. Still, it remains vulnerable to test-mode-only attack [55, 56, 57]. A concept of Mirror Key

Registers (MKRs) was introduced by the authors in [49], where the actual key is isolated in the test mode, and the testing is performed using a user-provided key. But, it does not provide support for online testing with the actual key. In [58, 59, 60, 61, 62], the ordering of the sub-chains in multiple scan chain designs was manipulated by obfuscation with the help of Segment Insertion Bits (SIB), which limits the attacker from observing the actual states [63]. Since scan-based attacks do not depend on the ordering of the scan chain, hence the access to the complete set of states induces a vulnerability. Advanced DFT structures, such as X-masking [64], and de-compressors [65], which were once considered secure against scan-based attacks [66], have also been shown to be insecure against advanced differential scan attacks [67, 68]. In [69], the obfuscation techniques discussed, both static and dynamic, draw inspiration from the lock-and-key scheme. However, a delay in the test process is imbibed from the user authentication process, which requires the test key to be loaded. Also, the dynamic obfuscation is vulnerable to [70]. The above countermeasures mainly protect the secret key of a cipher embedded inside the crypto-chip against scan attacks. There is a parallel stream of research work aimed at protecting the hardware IP against reverse engineering, thus restricting counterfeiting or overproduction of chips [71]. A dynamic obfuscation based scan design, proposed in [72] and [73], protects the IP by restricting scan access using dynamic scan obfuscation. This technique obfuscates scan data dynamically, using a protected obfuscation key generated by an LFSR. These obfuscation keys are XORed with the scan data using XOR gates inserted between the scan cells in the scan chain. Such designs are resistant to existing scan-based attacks due to dynamic obfuscation, though recently, the technique in [72] has been shown to be susceptible to ScanSAT [74]. In the scan protection technique proposed in [75], key gates are inserted in the scan chain, which is controlled by a static secret key stored in a tamper-proof memory (TPM), when a user applies the same secret key as a test key. Testing must be performed with an obfuscated pattern, where the test patterns are obfuscated using the same static secret key. If the test key applied does not match the secret key, the key gates are controlled by dynamic keys generated by a pseudo random number generator (PRNG), obfuscating the scan data dynamically in each clock cycle. The technique is vulnerable to the DynUnlock attack [76]. The defense mechanism proposed in [77] is an enhanced version of [78, 73], where the combinational logic of the circuit is additionally protected

with functional obfuscation. In functional obfuscation, additional authentication is required to gain access to the correct functionality of the circuit via an additional functional key. Although the proposed technique [77] provides additional security against scan attacks, while not compromising on testability, their hardware overhead is large. A cryptographic hash function is shown to modify the test response in [79], though the cost of implementation is dependent on a hash module that is made available inside the chip. In [80], the proposed countermeasure performs an encryption of the test data, which again requires additional modules for encryption/decryption, essentially incurring significant overhead in area, delay, and power.

We emphasize on the security of our proposed countermeasure in the context of Discrete Event Systems (DES) [81, 82]. There exist areas where security threats can be mitigated by ensuring that the *Opacity* property holds true for a system modeled as DES. Moreover, the opaque property is verifiable, hence the correctness of a defense model can be formally proved by constructing an estimator automaton.

The existing literature on defense techniques relies on partial scan design, data obfuscation, and scan chain encryption. Though these countermeasures are effective in the prevention of traditional scan attacks and signature attacks, they are ineffective as far as COSA [83] [84] are concerned.

Our major contributions in this work are outlined as follows:

1. A novel design of a security scheme using a hardware controller unit is presented. We show to circumvent differential scan attacks by allowing selective bit-flipped outputs, deterministically, using pre-computed mask values, without hampering testability.

2. A mask determination algorithm is presented that helps compute secret states and determines the controller action across a setpoint.

3. Security of our scheme is proved by theoretically modeling the cryptosystem in the framework of Discrete Event Systems and analyzing using the notion of *Opacity*. Moreover, the entropy of the secret key is preserved. The proof of correctness is guaranteed by constructing a state estimator automaton.

4. A theoretical analysis of the ineffectiveness of attacks launched from exploiting the Hamming weight distribution obtained at the output is performed. A case study

performed shows our approach is resilient to such attacks at a nominal extra overhead of 1.78%.

1.7.2 RPL Version Number Attack and RPL Rank Attack

To enable efficient and reliable communication, IETF has standardized IPv6 Routing protocol for Low Power and Lossy Networks (RPL) [9]. The design of RPL is tailored for low-power IoT devices. RPL maintains loop-free Destination Oriented Directed Acyclic Graphs (DODAG). A DODAG is created and maintained using control messages, primarily, DODAG Information Object (DIO) for upward paths, DODAG Advertisement Object (DAO) for downward paths and DODAG Information Solicitation (DIS) for node joining. RPL ensures cost-optimized topologies by ordering participating nodes on the basis of an integer cost function, *rank*. Individual rank of a node determines its position in the DODAG, relative to a 6BR sink node (root). Also, a single *version* number prevalent in the DODAG is maintained in DIO for consistency. Though RPL provisions various mechanisms and is secure enough from external attackers, yet the resource constrained nature of IoT devices, the typical characteristics of IoT networks such as lossy links, lacunas in infrastructure, dynamic topology, etc., can render IoT-RPL susceptible to internal attacks [85, 11, 86, 87].

Various internal attacks have been shown in the literature, of late, that make illicit use of RPL. Puppet attacks [88], advanced vampire attacks [89] make use of forged source routes, while attacks like energy depletion attack [90] and vampire attacks [91] drain resources by repeatedly sending useless data packets. Sybil attacks [92] and spam DIS attacks [93] have been shown to make use of DIS messages with counterfeit identities, essentially causing denial of service. DIO suppression attacks eavesdrop DIO messages for replaying it repeatedly in fixed intervals [94]. Out of the various DIO-specific attacks explored, proposed rank and version attacks continue to be of paramount importance since they are of low overhead and are realizable using DIO only. To launch such attacks, the rank and version number fields of a DIO message are fabricated causing formation of loops, sub-optimal routes, traffic redirection and network partitioning. Significant path delay is incurred since a large number of control messages are exchanged in the DODAG, resulting in energy depletion of the constrained nodes and disruption of network services. Moreover, rank attacks may be combined with other cross-layer attacks like selective forwarding attacks to alleviate the

damage caused.

Proposed methods for securing IoT networks against RPL rank and version attacks have their own typical limitations[95, 96, 97, 98, 99, 100, 101, 102, 103, 32, 104]. Cryptography-based mitigation schemes are resource exhaustive and computationally heavy, especially in a network of resource constrained devices. Machine learning-based approaches require investment of extensive training time, as per the system under consideration. Protocol-based approaches require modifying the protocol policies. IDS based approaches do not suffer from these above limitations, but the implementation of these schemes for rank attack detection is challenging since attack behaviour resembles normal behaviour. Hence, use of signature-based IDS and anomaly-based IDS schemes in the context of IoT attacks generate a large number of false positives. Furthermore, there exists many variants of rank attacks which present complex characteristics to evade detection capabilities of IDS. Formal verification of IDS schemes are also lacking.

This chapter presents an intelligent probing based scheme for the detection of rank and version attacks that also identifies location of malicious nodes. The probing mechanism helps differentiate the normal and attack behaviour. Our scheme incorporates a *centralized* I-Discrete Event System (DES) based IDS [47, 105, 48] and a set of agents with event monitoring enabled, that make use of probe packets [106, 107], judiciously. System failures and network attacks involve analogous behavioural deviations from the normal system functioning, which motivates the use of DES based IDS. Deploying our IDS does not require a change in protocol policies, encryption, extensive training time or any need for proprietary hardware support. Using our IDS also helps ensure a formally verifiable proof of correctness of our approach. As opposed to Opacity, the property of DES Diagnosability holds if an abnormal network behaviour can be differentiated from normal network behaviour. Since we require to identify a pure attack behaviour, we restrict ourselves to FDD theory of DES. But traditional DES diagnosability cannot help detect the attack node behaviour. An indicator event is required such that diagnosis is possible only in those states where such an event follows the attack event. I-diagnosability offers a relaxed definition of Diagnosability and is applicable to partial diagnosis problems. Probe packets that are sent correspond to indicator events. Attack detection is tested in only those paths where an indicator event follows the attack. The major contributions in this chapter are enumerated as follows:

- We propose a novel rank attacker identification scheme that also detects version attacks in IoT-RPL. Our scheme makes use of an intelligent active probing technique that helps create a deviation of attack traffic and normal traffic [106]. Our proposed scheme is centralized and uses an I-DES based IDS.

- We extend the power of traditional I-DES based IDS framework with attack type modeling for attacker identification.

- We prove the correctness and completeness of our approach by enumerating all the attack cases.

- The performance of our scheme is tested through simulations and real testbed. The experimental results highlight the applicability of our approach. Comparison of our scheme to state-of-the-art countermeasures shows our approach is energy-efficient with less packet overhead. The proposed solution is scalable, has minimum false positives and achieves more than 99% accuracy in identifying the malicious nodes.

1.7.3 6LoWPAN Fragmentation Attack

Recently, a significant portion in the network usage of Internet of Things (IoT) [5, 108, 109] in healthcare to home automation, industrial control systems to agriculture and smart cities, mostly employ 6LoWPAN, an IETF-standardized adaptation layer, for IPv6 based communication [110]. With the surge in the number of resource constrained devices constituting the IoT, the need for a huge address space as well as IP-connectivity over low power and lossy networks (LLN), including Wireless Sensor Networks (WSN), are facilitated using 6LoWPAN [111, 10]. Fragmentation is therefore essential at this layer, since IEEE 802.15.4 limits the frame size to 127 bytes and hence does not permit transmission of IPv6 packets with MTU 1280 bytes. Consequently, the adaptation layer is proposed to forward, buffer and process the fragments of the transmitted packets.

Malicious nodes make use of the challenges due to the proposed implementation and exploit the fragmentation and reassembly procedures to launch various Denial-of-Service (DoS) attacks. Lack of mechanisms to verify authenticity of the sender and the fragment helps mount spoofing attacks. A malicious entity that is overhearing a communication requires just a mere fragment to illegitimately occupy the buffer of a resource constrained

node, or to disrupt the integrity of the packet by slipping in duplicate fragments. In both the cases, since 6LoWPAN does not have means to verify fragment ingenuity, the buffer is freed and the packet needs to be resent. The security aspect of availability is at stake due to the trumped-up buffer reservation. The spurious occupancy also exhausts huge memory and time, since the fragments need to be kept in the constrained memory of the nodes till timeout. Moreover, the impact of the attack is unbound if the attacker drops fragments and replaces them with fragments containing spoofed content, since 6LoWPAN processes out-of-order fragments. Thus the services of the receiver node are blocked as fresh fragments wait on its buffer while the network performance as a whole is hindered.

Approaches to secure 6LoWPAN from fragmentation attacks have been mostly cryptography based [112, 113, 37], which are resource unfriendly, since they generate packet overhead. Thus, energy efficient schemes to save the resources of the constrained battery-powered low power devices are required. Moreover, the countermeasure techniques against IoT attacks have attempted attack detection mostly and not isolation, which leaves ample opportunities for a malicious entity to launch fresh attacks. With the advancements in adversarial machine learning techniques, complex attack characteristics and distributed attacks are also severely threatening. In this chapter, we investigate the fragment duplication attack from the topology perspective and by analysing the possible attack space. The *centralized* I-DES framework discussed in the earlier chapter cannot be used successfully here. Rather, we present here an intrusion detection scheme that is based on *decentralized* I-Discrete Event System (DES) framework. Decentralized I-diagnosis helps globally diagnose an attack node based on the response generated from the forwarded spoofed fragments. Global I-diagnosability is ensured from the local I-diagnosis. Also, due to the analogous behaviours posed between a system fault and a malfunctioning network node, DES based IDS have been in use of late. We also overcome the plausible issues mentioned above. Broadly, our contributions in this work can be summarised as follows:

- We propose a novel fragment duplication attacker identification scheme in 6LoWPAN. Our scheme makes use of an intelligent active probing technique that helps create a deviation of attack traffic and normal traffic [106]. Our proposed scheme is decentralized and uses I-DES based IDS.

- We extend the power of traditional decentralized DES based IDS with attack type

modeling for attacker identification.

- We prove the correctness and completeness of our approach by enumerating all the attack cases.

- The performance of our scheme is tested through simulations and real testbed. The experimental results highlight the applicability of our approach. Results show our approach is more energy-efficient and has less response time. The proposed solution has minimum false positives and achieves more than 99% accuracy in identifying the malicious nodes.

1.7.4 CoAP Request and Response Spoofing Attack

The Internet of Things (IoT) technology has revolutionised the outlook of connected devices and IP-connected smart objects. A significant portion of such interconnected heterogeneous devices are being extensively deployed to perform mission-critical tasks in areas such as the health sector, energy management, industrial process control systems, etc. [4] For uninterrupted services over the internet, a reliable end-to-end communication is demanding for the family of constrained devices. The IoT protocol stack has been designed and adopted with an aim of achieving such a standard [10].

IETF has standardized CoAP as an Application layer web transfer protocol to provision for usage of Internet services in M2M applications constituting resource-constrained devices across lossy, low-power networks [114]. Specially designed for provisioning web interoperability, applications that make use of constrained sensing or actuating devices, having limited power, memory and processing capabilities, render them fragile to various external threats and DoS attacks. CoAP uses UDP as the transport protocol, which is unreliable and is devoid of handshaking mechanism between client and server. CoAP is susceptible to various attacks like Cross-Protocol attacks, Amplification attacks, Man-in-The-Middle (MiTM), etc. [115] Amongst these, IP address spoofed DoS attacks can be launched at ease and is the focus of our work here. Moreover, it can help mount stealthier attacks when used in combination, such as amplification attacks. Availability of devices and accessibility to services can be immensely compromised by a malicious endpoint that is exercising *read* and *write* access.

Research works reported in the literature have proposed numerous countermeasures to effectively mitigate DoS attacks in general. Mostly, the adopted approaches either employ host-based, router-based or hybrid techniques [116]. Adaptive solutions employing frequency based approaches have been successfully used to detect mixed-rate IP spoofed DDoS attacks [117]. Off-path response spoofing attacks in TCP and DNS have been analyzed thoroughly in the literature [118, 119]. A countermeasure based on source-port randomization has been proposed for an off-path attack on TCP [120]. In [121], researchers launch an off-path attack by analyzing the remote server access support in CoAP and a request spoofing vulnerability. Request spoofing is further shown to be mitigated using machine learning based approaches.

The proposed schemes, except a few, do not address mitigation techniques for request spoofing attack in 6LoWPAN applications using CoAP. None of the reported works have attempted identification of malicious or misconfigured endpoints in the past. Owing to the similarity between diagnosis of faults and detection of network attacks in DES, we employ here an I^2-DES based IDS (Induced I-DES) [122, 48]. The I-diagnosability framework discussed in the earlier chapters is limiting and cannot be used successfully for detection and identification. Active probing based indicator events do not suffice since the generated probe responses are same in number across normal and attack scenarios. Hence, an empowering event is required here, justifying the use of traditional I^2-DES framework.

Enumerated below are our contributions in this work:

1. We design an IDS scheme using I^2-DES that identifies the attacker when an IP spoofed DoS attack is launched in 6LoWPAN applications employing CoAP without DTLS support.

2. Malicious network behaviour is detected from comprehensive analysis of network event dynamics. Attacker is successfully identified by constructing a I^2-DES diagnoser, which serves as our IDS engine. Proof of correctness can be guaranteed using formal verification.

3. The results demonstrate an energy efficient and scalable solution at considerably lesser response times compared to related state-of-art solutions.

1.8 Organization of the Thesis

This dissertation aims to provide lightweight DES-based countermeasures for mitigation of IoT attacks that either have differentiable or they have indistinguishable normal and abnormal characteristics. The considered IoT attacks can be categorized as a device-level or a network-level or an application-level attack, namely scan-chain based side-channel attack, RPL version and RPL rank attacks, 6LoWPAN fragmentation attacks and CoAP request/response spoofing attacks. An overview of our contributions is shown using Figure 1.1.

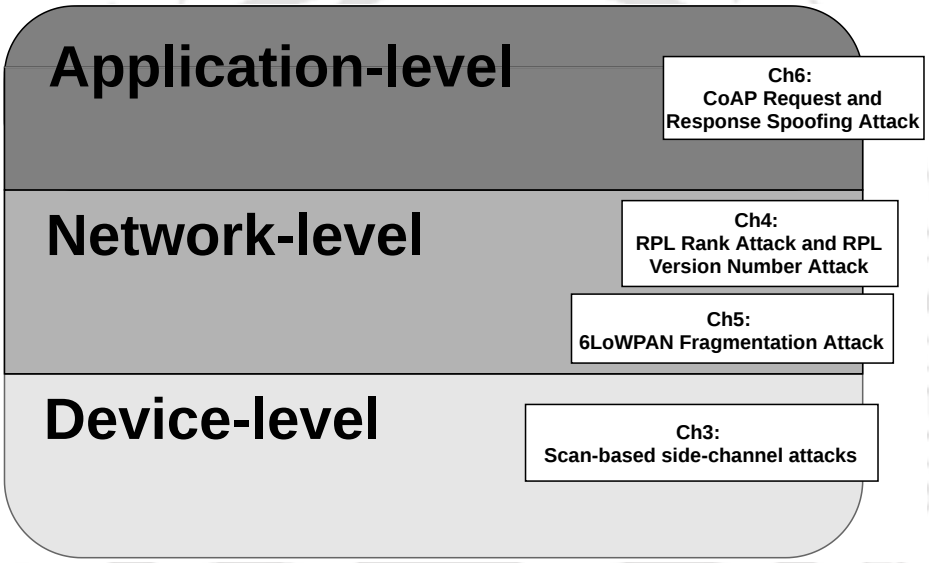


Figure 1.1: Overview of contributions in the IoT architecture

In the device-level attack, scan-based side-channel attack on cryptographic IoT devices, the secret contents are leaked by an adversary who gets hold of the device. The secret key values are differentiable from the non-secret values at the scan out pin that aids the attacker motif. A proposed solution idea needs a mechanism such that secret and non-secret values are indistinguishable while incurring minimal overhead and retaining full testability. On the contrary, in case of each of the network-level IoT attacks and application-level IoT attacks, the IoT network behaviour during attack is non-differentiable from the normal network behaviour. A signature or anomaly profile is naturally lacking. Generation of signature pattern or anomaly observation results in increased false positives. Consequently, depending on the attack, a suitable mechanism and solution design needs to be chosen such

that attack behaviour can be made out from the normal, accurately with minimal overhead. This has motivated the study of applicability of the main idea of the dissertation, i.e., "using DES paradigm to develop lightweight strategies" for different levels of attack in the IoT ecosystem. Each attack mitigation technique requires a different DES paradigm for our solution strategies. The characteristic features of attack, the solution architecture and the detection philosophy are significantly different in each of the chapters. The chapter wise thesis organization is given as follows:

- **Chapter 2:** In this chapter, we propose a DES opacity-based low overhead controller framework that thwarts most of the scan-based side-channel attacks on a cryptographic IoT device, namely differential scan attack, co-relation scan attack and signature attacks.
- **Chapter 3:** In this chapter, an I-Diagnosability based DES framework is proposed. This framework is used to propose an active I-DES-based centralized IDS scheme that uses an intelligent probing technique to not only detect RPL version and RPL rank attacks but also identifies the rank attack node location.
- **Chapter 4:** In this chapter, an active I-Diagnosability based decentralized IDS scheme is proposed that detects 6LoWPAN fragmentation attacks using fabricated probe fragments and also identifies the malicious node.
- **Chapter 5:** In this contribution chapter, a novel I^2-Diagnosability based DES framework is proposed. An active I^2-DES based IDS scheme is presented that uses empowering events apart from request or response probe packets to detect and identify attack node location when a CoAP request/response spoofing attack is launched.
- **Chapter 6:** The concluding chapter highlights the findings of our study and summarizes the thesis contributions. Future scope of research is also discussed.

CHAPTER

2

Mitigation of Differential Scan Attacks

Testing of manufacturing defects has seen a surge recently with growing demands in micron and sub-micron Internet of Things (IoT) devices in the semiconductor industry. Among the techniques prevalent in the industry, scan-based Design-for-Testability (DfT) is the most popular one. The internal scan chain comprising of serially connected flip-flops (scan cells) is made more observable and controllable, which effectively facilitates testing. A designated tester may provide a relevant input to stimulate the internal nodes during test mode, and observe the corresponding output responses. Again, attackers look out for various means to gain access to the internal scan elements. This hands them an opportunity to leak sensitive information embedded within an IoT device containing a cryptographic IC. Side-channel attacks can be precisely classified to be scan-based if the above inherent vulnerability is made use of.

Universal standard cipher, Advanced Encryption Standard (AES), was shown to be vulnerable to scan-based attacks [49] after it was already reported to be effective on a Data Encryption Standard (DES) cipher implementation [50]. Switching a crypto-chip to test mode after having run the chip in normal mode for a few cycles was shown to be effective in the retrieval of the secret embedded key. A differential analysis technique applied on a set of related plaintext input and cipher output pairs helped infer the secret information. This could be realized due to observing the internal scan cell contents corresponding to controlled input pairs applied beforehand. Most of these discussed attacks exploit only the unique Hamming weights corresponding to one round output Hamming weight distribution

by applying all possible input pairs with a fixed input difference targeting a particular AES input byte. Subsequently, this style of attack was extended to public key cryptosystems such as elliptic curve cryptography (ECC) [51], Rivest-Shamir-Adleman (RSA) [52] circuits, and in lattice-based public key cryptosystems [53].

Numerous countermeasures have been proposed to prevent scan-based side-channel attacks in the recent past. In [54, 49], a mode reset countermeasure was proposed to flush all sensitive data contained in a scan chain during a switch from normal to test mode. Still, it remains vulnerable to test-mode-only attack [55, 56, 57]. A concept of Mirror Key Registers (MKRs) was introduced by the authors in [49], where the actual key is isolated in the test mode, and the testing is performed using a user-provided key. But, it does not provide support for online testing with the actual key. In [58, 59, 60, 61, 62], the ordering of the sub-chains in multiple scan chain designs was manipulated by obfuscation with the help of Segment Insertion Bits (SIB), which limits the attacker from observing the actual states [63]. Since scan-based attacks do not depend on the ordering of the scan chain, hence the access to the complete set of states induces a vulnerability. Advanced DFT structures, such as X-masking [64], and de-compressors [65], which were once considered secure against scan-based attacks [66], have also been shown to be insecure against advanced differential scan attacks [67, 68]. In [69], the obfuscation techniques discussed, both static and dynamic, draw inspiration from the lock-and-key scheme. However, a delay in the test process is imbibed from the user authentication process, which requires the test key to be loaded. Also, the dynamic obfuscation is vulnerable to [70]. The above countermeasures mainly protect the secret key of a cipher embedded inside the crypto-chip against scan attacks. There is a parallel stream of research work aimed at protecting the hardware IP against reverse engineering, thus restricting counterfeiting or overproduction of chips [71]. A dynamic obfuscation based scan design, proposed in [72] and [73], protects the IP by restricting scan access using dynamic scan obfuscation. This technique obfuscates scan data dynamically, using a protected obfuscation key generated by an LFSR. These obfuscation keys are XORed with the scan data using XOR gates inserted between the scan cells in the scan chain. Such designs are resistant to existing scan-based attacks due to dynamic obfuscation, though recently, the technique in [72] has been shown to be susceptible to ScanSAT [74]. In the scan protection technique proposed in [75], key gates are inserted

in the scan chain, which is controlled by a static secret key stored in a tamper-proof memory (TPM), when a user applies the same secret key as a test key. Testing must be performed with an obfuscated pattern, where the test patterns are obfuscated using the same static secret key. If the test key applied does not match the secret key, the key gates are controlled by dynamic keys generated by a pseudo random number generator (PRNG), obfuscating the scan data dynamically in each clock cycle. The technique is vulnerable to the DynUnlock attack [76]. The defense mechanism proposed in [77] is an enhanced version of [78, 73], where the combinational logic of the circuit is additionally protected with functional obfuscation. In functional obfuscation, additional authentication is required to gain access to the correct functionality of the circuit via an additional functional key. Although the proposed technique [77] provides additional security against scan attacks, while not compromising on testability, their hardware overhead is large. A cryptographic hash function is shown to modify the test response in [79], though the cost of implementation is dependent on a hash module that is made available inside the chip. In [80], the proposed countermeasure performs an encryption of the test data, which again requires additional modules for encryption/decryption, essentially incurring significant overhead in area, delay, and power.

We emphasize on the security of our proposed countermeasure in the context of Discrete Event Systems (DES) [81, 82]. DES broadly refers to the class of systems whose dynamics can naturally be modeled as a set of discrete events occurring asynchronously over time. Over the years, DES has been successfully applied to the theory of Fault Detection and Diagnosis (FDD). It involves modeling the system behavior under normal and failure conditions. In recent years, the DES community has highlighted security properties pertaining to confidentiality, integrity, and availability of data. *Opacity* is a security notion that is concerned with the confidentiality of a system secret. A system can be shown to be opaque if a well-defined secret (expressed through states) can be kept hidden from an external adversary. Therefore, there exist areas where security threats can be mitigated by ensuring that the opacity property holds true for a system modeled as DES. Moreover, the opaque property is verifiable, hence the correctness of a defense model can be formally proved by constructing an estimator automaton.

The existing literature on defense techniques relies on partial scan design, data ob-

fuscation, and scan chain encryption. Though these countermeasures are effective in the prevention of traditional scan attacks and signature attacks, they are ineffective as far as co-relation scan attacks [83] [84] are concerned.

Our major contributions in this work are outlined as follows:

1. A novel design of a security scheme using a hardware controller unit is presented. We show to circumvent differential scan attacks by allowing selective bit-flipped outputs, deterministically, using pre-computed mask values, without hampering testability.

2. A mask determination algorithm is presented that helps compute secret states and determines the controller action across a setpoint.

3. Security of our scheme is proved by theoretically modeling the cryptosystem in the framework of Discrete Event Systems and analyzing using the notion of *Opacity*. Moreover, the entropy of the secret key is preserved. The proof of correctness is guaranteed by constructing a state estimator automaton.

4. A theoretical analysis of the ineffectiveness of attacks launched from exploiting the Hamming weight distribution obtained at the output is performed. A case study performed shows our approach is resilient to such attacks at a nominal extra overhead of 1.78%.

The rest of the chapter is organized as follows: Section 2 presents the background and prerequisites. The design of our proposed scheme is presented in Section 3. Section 4 demonstrates the proof of security. Experimental results and performance analysis are summarised using a case study in Section 5. We compare our work and discuss the relevance of our countermeasure in Section 6. We finally conclude with Section 7.

2.1 Background

In this section, we briefly discuss AES and scan attacks, followed by the preliminaries of DES and Opacity.

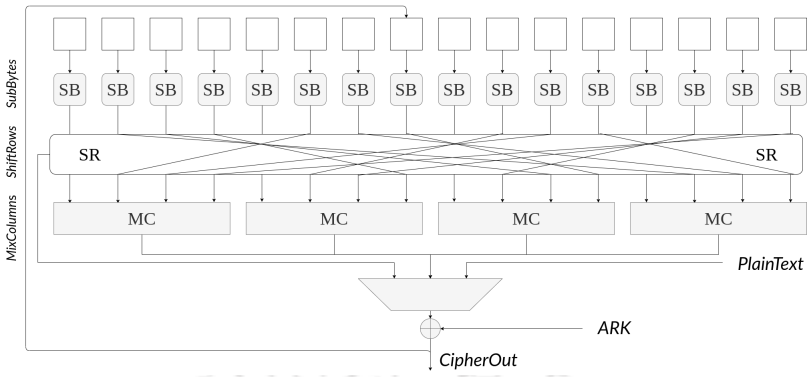


Figure 2.1: AES round operation

2.1.1 AES

AES is a symmetric block cipher that encrypts fixed size data blocks of 128 bits using key sizes of 128, 192, or 256 bits consisting of 10, 12, and 14 round operations, respectively. Each round operation consists of the below mentioned operations:

1. *SubBytes:* This constitutes of a non-linear substitution operation taking 8-bit input and producing 8-bit output.
2. *ShiftRows:* It produces a byte-wise permutation of the state.
3. *MixColumns:* It is a 4 byte mixing operation.
4. *AddRoundKey:* It XORs the round key with the state.

A key schedule in AES-128 generates the round key for each round for all of the 10 rounds. In addition to this, prior to the first round, the plaintext is XORed with the user-specified key as shown in Figure 2.1, and during the last round, the MixColumns operation is not used.

To start with, a 128 bit plaintext input is arranged in the form of a 4×4 matrix, where each byte can be represented as $a_{i,j}$ $(0 \leq i, j \leq 3)$. Initially in the key whitening phase, a bitwise xor operation is performed on plaintext input with $RK0$ showing as follows:

$$b_{i,j} = a_{i,j} \oplus RK0_{i,j} \tag{2.1}$$

SubBytes is the transformation that is based on the S-box. In this non-linear operation,

S-box substitutes each byte of the plaintext data arranged as a state matrix, $b_{i,j}$, with a byte $c_{i,j}$ as obtained from the lookup table as follows:

$$c_{i,j} = S - box(b_{i,j}) \tag{2.2}$$

The third step constitutes the ShiftRow operation where the bytes in each row of the 4×4 matrix are left shifted, in the following cyclic manner.

$$(d_{0,0}, d_{0,1}, d_{0,2}, d_{0,3}) = (c_{0,0}, c_{0,1}, c_{0,2}, c_{0,3})$$

$$(d_{1,0}, d_{1,1}, d_{1,2}, d_{1,3}) = (c_{1,1}, c_{1,2}, c_{1,3}, c_{1,0})$$

$$(d_{2,0}, d_{2,1}, d_{2,2}, d_{2,3}) = (c_{2,2}, c_{2,3}, c_{2,0}, c_{2,1})$$

$$(d_{3,0}, d_{3,1}, d_{3,2}, d_{3,3}) = (c_{3,3}, c_{3,0}, c_{3,1}, c_{3,2}) \tag{2.3}$$

MixColumn is a column-wise operation where every 4 bytes in each column obtained after the ShiftRow are multiplied by a polynomial of the form $m(x) =' 03'x3 +' 01'x2 +' 01'x +' 02'$, and 4 bytes are generated in corresponding output column as follows:

$$(e_{0,j}, e_{1,j}, e_{2,j}, e_{3,j}) = (d_{0,j}, d_{1,j}, d_{2,j}, d_{3,j}) \otimes m(x) \tag{2.4}$$

Finally, before the intermediate register R is loaded, a xor operation is performed using round key $RK1$ on the MixColumn output as follows:

$$f_{i,j} = e_{i,j} \oplus RK1_{i,j} \tag{2.5}$$

2.1.2 Differential Scan Attacks

Fundamental Scan Attack

In fundamental (differential) scan attack [49], intermediate round outputs stored in round registers are scanned out and analysed to find the key byte. The attacker applies pairs of plaintexts, having a difference in the least significant bit (LSB) of a byte, and derives a Hamming weight from the difference in observed outputs. Then the attacker searches for a Hamming weight that corresponds to a unique input pair. For example, the Hamming

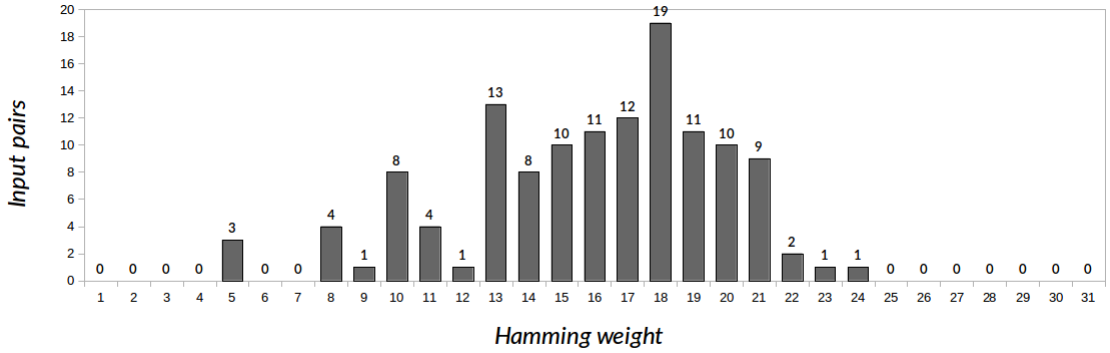


Figure 2.2: Hamming weight distribution (original) corresponding to $0x01$ plaintext difference

weight 9, as shown in Fig. 2.2, corresponds to one input pair. Now, if the attacker observes Hamming weight 9, he can infer the key by XORing the plaintext byte with the corresponding input byte.

Signature Attack

In signature attack [123, 124], an ordered tuple of plaintext input pairs with 1 bit difference, $I = \{I_1, \ldots, I_{128}\}$ is used to generate an ordered tuple of Hamming weights, $HW_i = \{HW_1, \ldots, HW_{128}\}$ with key K_i, $1 \leq i \leq 256$. The HW_i is generated for all possible keys K_i using all plaintext input pairs in I, which is the signature of that corresponding key stored in the signature table. The attacker uses the same plaintext input pairs to generate the signature in the target device that gives $HW' = \{HW'_1, \ldots, HW'_{128}\}$. To compare HW' with all HW_i, he needs to match Hamming weight by Hamming weight from left to right. If all the 128 Hamming weights match, he then recovers the key.

Proposed Co-relation scan attack (COSA)

The designer can easily overcome the existing attack by not allowing unique hamming weights to the attacker. Attackers aim is to increase the attack surface and launch a more lethal form of attack. In this respect, one natural choice for the attacker here is to look for a possible attack using non-unique hamming weights.

In the proposed attack, the attacker can use a simulated version of the DUT which the attacker has a complete control over. The attack consists of three steps :

1. Determining S-box input pair in offline phase: The attacker needs to determine the set

$I = \{(I_0, I_0 \oplus \delta), (I_1, I_1 \oplus \delta), ..., (I_{n-1}, I_{n-1} \oplus \delta)\}$ of the S-box input pairs for which the output vectors after one round of AES have a hamming weight Δ between them. The values of Δ and δ are chosen by the attacker, where $0 \leq \delta \leq 255$ and $0 \leq n \leq 19$. We will only require 19 pairs because in case of 1 bit difference, there are 19 input pairs possible for hamming weight of 18.

2. Determining input plaintext pairs in online phase: The attacker needs to determine the set $P = \{(P_0, P_0 \oplus \delta), (P_1, P_1 \oplus \delta), ..., (P_{n-1}, P_{n-1} \oplus \delta)\}$ of the plaintext pairs for which the output vectors after one round of AES have a hamming weight Δ between them.

3. Determining the key byte value : The attacker has to determine the key byte value using the sets P and I.

ALGORITHM 1: S-box Input Pairs

input : $\Delta \rightarrow$ Hamming Weight, $\delta \rightarrow$ Input difference

output : A set I of the S-box input pairs

```
1 I = ∅
2 for p ← 0 to 255 do
3     p' ← p ⊕ δ
4     o ← AES_0(p)
5     o' ← AES_0(p')
6     if Hamming weight between o and o' is Δ then
7         I ← I ∪ (p, p')
8 return I
```

Determining the S-box Input Pairs (Offline)

To determine all of the S-box input pairs generating a hamming weight of Δ between the output vectors, the attacker needs to try every possible plaintext pairs of the form $(p, p \oplus \delta)$. The attacker would also have to use a modified version of AES, in which the plaintext is not XORed with the key before the first round. Algorithm 1 illustrates this. The algorithm iteratively generates all possible pairs of the form $(p, p \oplus \delta)$ where $0 \leq p \leq 255$. The algorithm uses the method $AES_0()$ to obtain the one round output of the modified version of AES. It then checks for the hamming weight between the output vectors if it is equal to Δ or not. In case it is, the algorithm then adds the pair to the set I, otherwise the

pair is discarded and the algorithm continues until all possible pairs have been exhausted. This algorithm will add both the pairs $(p, p \oplus \delta)$ and $(p \oplus \delta, p)$ to the set I. The attacker can either discard the second pair after the algorithm has terminated or define the set I such that it is a set of unordered pairs.

Determining the Plaintext Pairs (Online)

Algorithm 2 illustrates the process of determining the required plaintext pairs. In addition to the values of hamming weight Δ and input difference δ the algorithm also needs to know the position of the byte which the attacker wishes to determine. This is because the plaintext pairs will change based on the value of the key byte. In Algorithm 2, we have used the method $AES_0()$ representing DUT with first round operation. It returns the round output after one round of AES.

The algorithm generates all pairs of the form $(p, p \oplus \delta)$ where $0 \leq p \leq 255$. The two plaintexts $pp1$ and $pp2$ are produced by setting the i^{th} byte in them as p and $p \oplus \delta$, respectively. The algorithm sets the other bytes to zero. It then checks whether the hamming weight between the output vectors produced by one round of AES on the two plaintexts is Δ or not. If it is, then the algorithm adds the pair $(p, p \oplus \delta)$ to the set P, otherwise it is discarded and the algorithm moves on to the next pair. As in Algorithm 1, this algorithm too adds both the pairs $(p, p \oplus \delta)$ and $(p \oplus \delta, p)$ to the set P, and one of these pairs is discarded.

ALGORITHM 2: Plaintext Pairs

input : $\Delta \rightarrow$ Hamming Weight, $\delta \rightarrow$ Input difference, $i \rightarrow$ Key byte position
output : A set P of the plaintext pairs

```
1  P = ∅
2  for p ← 0 to 255 do
3      p' ← p ⊕ δ
4      pp1 ← i^th byte set to p, rest all zeroes
5      pp2 ← i^th byte set to p', rest all zeroes
6      o ← AESOneRound(plaintext_1)
7      o' ← AESOneRound(plaintext_2)
8      if Hamming weight between o and o' is Δ then
9          P ← P ∪ (p, p')
10 return P
```

Determining a Valid Mapping (Offline)

Algorithm 3 iterates through all 256 possible key byte values and checks if XORing them

with the elements of set P produces a permutation of the set I. The possible key byte value is denoted by K_G. The value of K_G is XORed with each element of the set P, and the algorithm next searches for the resultant value in the set I. If the resultant value is not present in I, then that value of K_G is not a valid key byte. However, if all values generated by XORing K_G with the elements of set P are present in the set I, then that value of K_G is a possible key byte value and the algorithm prints it. The worst case time complexity of this algorithm is $O(nlogn)$.

ALGORITHM 3: Key Guessing Algorithm

input : No Input

output : Possible key bytes

```
1 for K_G ← 0 to 255 do
2     valid = true
3     for p ∈ P do
4         s ← p ⊕ K_G
5         if s ∉ I then
6             valid = false
7             break
8     if valid then
9         print(K_G)
```

2.1.3 Preliminaries of Discrete Event System and *Opacity* in Security

A Discrete Event System (DES) is characterized by discrete state space. The occurrence of a system event causes a transition from one state to another. System dynamics, where the state space can be described as a discrete set, $\{0, 1, 2, ...\}$, and events occur at discrete points in time, can be naturally modeled as a DES. Systems such as circuit networks, digital controllers, etc., are discrete event systems by nature. Hybrid systems like motors, power systems, etc., can also be discretized at some level of abstraction for modeling purposes. Therefore, non-DES systems or analog systems, by abstraction, based on the usage of some logic variables, can be modeled as DES. In the context of this work, we restrict ourselves to the framework of Finite State Automata, while discussing the DES modeling. The symbols used, along with their meanings, are listed in Table 2.1.

Let Σ represent an alphabet of symbols (also called elements or events), and Σ^* denotes the set of all finite-length strings composed of symbols from Σ, including the empty string ϵ

Table 2.1: Notations

Symbol	Definition
H	DES model
Σ	Set of events of the DES model H
$\Sigma_o(\Sigma_{uo})$	Set of observable (unobservable) events of the DES model H
V	Set of model variables of the DES model H
$\Im$	Set of transitions of the DES model H
τ	A transition $\tau \in \Im$
Y	Set of states of the DES model H
Y_0	Set of initial states of the DES model H
σ	Event on which a transition is enabled
$L(H)$	Set of all traces *generated* in H
$Y_S(Y_{NS})$	Set of secret (non-secret) states of the DES model H
(I, I')	An S-box input pair
(P, P')	A plaintext input pair
$R(p)$	Round output value obtained from applying plaintext p
δ	A plaintext input difference
IHW_i	Vector of S-box input pairs generating Hamming weight i
HW_i	Vector of plaintext pairs generating Hamming weight i
IDT	Hamming distribution table of S-box input pairs
DT	Hamming distribution table of plaintext pairs
$Mask(P, x)$	A mask computation function
Δ	A hexadecimal mask value
RK_i	i^{th} round key
$FE(FD)$	Flip Enable (Disable) event

(of length zero). A language $L \subseteq \Sigma^*$ can be described as some subset of all finite-length strings from Σ^*.

Our DES model is represented as a 5-tuple $H = (V, Y, \Sigma, \Im, Y_0)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is the finite set of model variables with each v_i assuming values from its respective domain set, $Dom(v_i)$, Y is the finite set of states, Σ is the event set, $\Im$ represents the set of transitions, and Y_0 is the set of initial states of the automaton, so $Y_0 \in 2^Y$. A state $y \in Y$ is of the form $\{v_1, v_2, \ldots, v_k\}, k \leq n$, with each associated variable assuming values from their respective domain sets. A transition $\tau \in \Im$, represented as an ordered pair $[y, \sigma, y^+]$ from y ($initial(\tau)$), the initial state of the transition, to the final state of the transition, y^+ ($final(\tau)$), occurs because of an input symbol $\sigma \in \Sigma$.

Now the language generated in DES H is represented as $L(H) = \{s \in \Sigma^* | \exists (y, y^+) \in Y : [y, s, y^+] \in \Im\}$. A *trace* of a DES model H is a string of events in H and is denoted as $\{\tau_1, \tau_2, \tau_3, \ldots\}$, where $initial(\tau_1)$ is an initial state in Y_0 and by the consecution property

$initial(\tau_{i+1}) = final(\tau_i)$, for $i \geq 1$ holds. Considering that only a subset of the events can be observed and monitored by the attacker, we assume that Σ can be partitioned into two disjoint subsets sets, Σ_o and Σ_{uo}, where Σ_o signifies the set of observable events and Σ_{uo} represents the set of unobservable events. We take a natural projection mapping, P_H, on the system H which is defined as $P_H : \Sigma^* \rightarrow \Sigma_o^*$. It is used to map any trace executed in the system to the sequence of observable transitions associated with it. The projection operation is defined recursively as: (a) $P_H(\epsilon) = \epsilon$, (b) $P_H(\sigma) = \sigma$ if $\sigma \in \Sigma_o$, (c) $P_H(\sigma) = \epsilon$ if $\sigma \in \Sigma_{uo}$, (d) $P_H(s \cdot \sigma) = P_H(s)P_H(\sigma), s \in \Sigma^*, \sigma \in \Sigma$, where ϵ represents the empty trace. Furthermore, the state set can also be partitioned based on a defined secret as a set of *secret states*, Y_S and a set of *non-secret* states, Y_{NS}. Y_S refers to states that contain secret information. All other states belong to Y_{NS}. So, $Y_S \cup Y_{NS} = Y$ and $Y_S \cap Y_{NS} = \Phi$.

Opacity is a security notion used in the DES community [43]. Given a system behavior represented as a non-deterministic automaton, opacity is a system property that can be verified to hold true using model-checking algorithms. It will be used to verify the security of our proposed defense model against scan attacks.

Definition 1 (*Current-state opacity (CSO)*) [125, 42]: Given a DES system model $H = (V, Y, \Im, \Sigma, Y_0)$, a projection P_H, a set of secret states $Y_S \subseteq Y$, and a set of non-secret states $Y_{NS} \subseteq Y$, $Y_S \neq Y_{NS}$, H is current-state opaque (or, (Y_S, P_H, 0)-opaque), if, $\forall t \in L(H), \exists s \in \{L(H) \setminus \{t\}\}$, such that, $final(t) \in Y_S \Rightarrow [final(s) \in Y_{NS}$ and $P_H(t) = P_H(s)]$.

The above definition means that if we are given any finite trace in the language of H, and such a trace ends in a secret state, the system is CSO if and only if there exists another different trace which is observably equivalent to the first trace, but terminates in a non-secret state. It implies that if such two traces exist, then the current state of the system cannot be estimated with certainty if it belongs to a subset of secret states. With the knowledge of H from its observations, if the attacker is not able to infer the current state of the system, the secret can be said to be opaque. In this chapter, we restrict to CSO, where the secret is defined in terms of the current state of the system.

Verifier Construction : Verifying current-state opacity involves the construction of the current-state estimator (CSE). Given a system modeled as a DES H and an observed sequence of input symbols in its language, CSE converts H into an equivalent deterministic

estimator automaton, H_o, which consists of estimates of the system state when H is simulated on any trace in $L(H)$ [46, 43]. The estimator automaton is defined here as a 5-tuple $H_o = (V_o, Y_o, \Im_o, \Sigma_o, Y_{0,o})$. Let $Y_{S,o}$ be the set of all states whose current estimate contains at least one secret state from Y_S, then $Y_{S,o}$ is the set of secret state estimates of the estimator automaton. Similarly, let $Y_{NS,o}$ be the set of all states whose current estimate contains at least one non-secret state $\in Y_{NS}$, then, $Y_{NS,o}$ is the set of non-secret state estimates of the estimator. If none of the estimator states reveal the current system estimate to purely contain states from Y_S, then the system H can be said to be opaque to an observer who observes through some projection mapping P_H.

ALGORITHM 4: Key Recovery Procedure

Input : IDT, PP

/* IDT is the Hamming distribution table of S-box input pair vectors for δ difference, $IHW_i, i \in \{1, 2, \ldots, 31\}$ and PP is the set of plaintext pairs $\{(P_1, P'_1), (P_2, P'_2), \ldots, (P_{128}, P'_{128})\}$; */

Output : key

/* key is one byte of Round Key, RK_0 */

1 $CP \leftarrow CP \cup \{(P_j, P'_j) | j \in \{1, 2, \ldots, 128\}\}$; /* Plaintext pairs are added inductively to set of chosen plaintext pairs, CP */

2 $R(P) \leftarrow AES(P)$; /* $AES()$ computes the AES first round output and $(P, P') \in CP$ */

3 $R(P') \leftarrow AES(P')$;

4 $IHW_i \leftarrow HWT(R(P) \oplus R(P'))$;/* $HWT()$ computes the Hamming weight */

5 $CI \leftarrow CI \cup \{(I, I') | (I, I') \in IHW_i\}$;/* S-box input pairs are added inductively to set of chosen plaintext pairs, CI */

6 $key \leftarrow CP \oplus CI$;

2.2 Proposed Defense Scheme

2.2.1 Threat model

Assuming that the attacker obtains control of the crypto-chip running the AES implementation, a general procedure for launching differential scan attacks, typically, is shown in Algorithm 4. The attacker applies chosen plaintext pairs and computes the Hamming weights between response pairs after one AES round, depending on some fixed value of

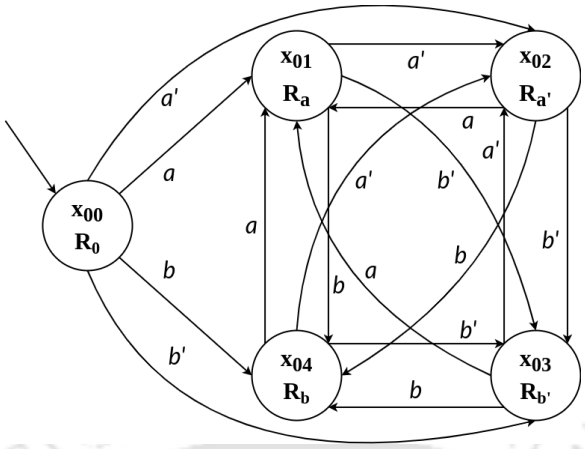


Figure 2.3: Proposed DES attack model G with 2 plaintext pairs

difference, δ. The Hamming distribution table for δ, IDT, is checked for the Hamming weight that is obtained. Round key value, RK_0, can be lastly recovered from solving a system of equations involving the chosen S-box input pair set, CI, and chosen-plaintext pair set, CP. Identification of the correct Hamming weight is, therefore, a primer to retrieve a correct key byte. We consider the following set of assumptions from the perspective of the attacker:

1. For the sake of simplicity, we assume that the attacker applies only a 0x01 difference between plaintext pairs.

2. As already mentioned, the attacker has the crypto-chip running the AES implementation at hand.

3. The attacker is not able to perform invasive attacks like de-packaging of the chip for probing the internal signals.

2.2.2 State based Attack Model

DES representation of the attack model is demonstrated using an example automaton as shown in Fig. 2.3. In this modeling, we consider two differential plaintext pairs and their corresponding round register outputs. The plaintexts are represented using the transitions, while the states represent the round outputs in our automaton G. Each state is represented using a variable v. Here, the domain of the variable v, $Dom(v) = \{R_0, R_a, R_{a'}, R_b, R_{b'}\}$.

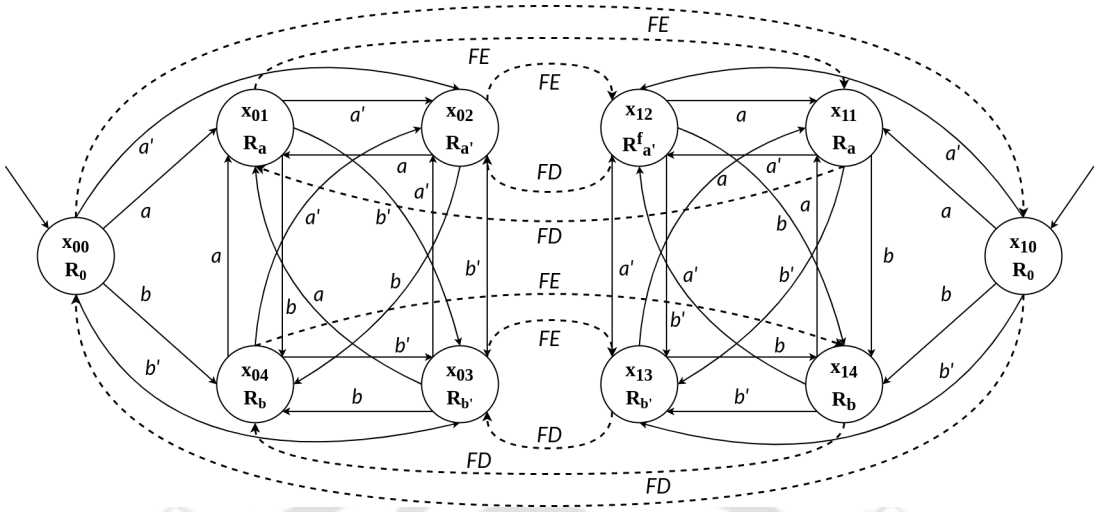


Figure 2.4: Proposed defense model H with 2 plaintext pairs

Thus, each state assumes values from $Dom(v)$ and is symbolic of 128-bit values of the intermediate round register output. Similarly, in the case of transitions, the input symbols are test vectors, each of length 16 bytes. The state set of the DES G consists of $Y_t = \{x_{00}, x_{01}, x_{02}, x_{03}, x_{04}\}$. We assume that the correspondence between round register cells to scan cells has been correctly established by the attacker. One having access to the scan chain can observe its contents by performing n shift out operations at SO pin, where n represents the number of cells in the round register. The series of above mentioned scan out operations are implicitly considered to be a part of each of the transitions that correspond to test vectors being applied at the chip primary inputs. We consider that input test vectors form the alphabet set $\Sigma = \{a, a', b, b'\}$. Amongst these (a, a') and (b, b') are the differential plaintext pairs. Self transitions have been omitted for simplicity purposes.

2.2.3 State based defense model

The proposed DES defense model H is shown in Fig. 2.4 and consists of two submachines, one normal machine and another flipped submachine. The attack model, G, shown in Fig. 2.3, is extended by incorporating Flip Enable and Flip Disable events. States of the flipped submachine are appended to the state set of G. Therefore, $Y = \{x_{00}, x_{01}, x_{02}, x_{03}, x_{04}, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}\}$. Since each state corresponds to a round register output, a flipped machine consists of states where a subset of the round register output

bits may be flipped (i.e., 0 if 1 and vice versa). The input test vectors are observable and belong to the alphabet set $\Sigma_o = \{a, a', b, b'\}$, while $\Sigma_{uo} = \{FE, FD\}$ consists of the unobservable events. Events in Σ_{uo} are associated with transitions from the normal model to states in the flipped model and vice versa. While FE is the event associated with the Flip Enable signal set to HIGH, FD corresponds to the Flip Enable signal being set to LOW. A transition with FE event that takes a state y in the normal model to a state y^+ in the flipped model is therefore represented as $[y, FE, y^+]$. Such a transition is unobservable in the eye of the attacker, who observes H through a projection P_H. Suppose a state, x, is reached by applying a test vector at the primary input. Then, the transitions containing the events FE and FD, at state x, i.e., may be enabled and disabled depending on x. This is maintained by the controller throughout the system operation time. States in the normal machine are of the form x_{0j} and states belonging to the flipped model are represented as x_{1j}. This is shown in the Fig. 2.4.

From the modeling perspective, it is assumed that the following properties hold in our DES model H:

1. Any state in H is reachable from an initial state.
2. H is *alive* with respect to observable events. This means there exists an observable transition defined at each state $y \in H$.
3. H contains a secret information defined within the states. We designate such states as *secret states*.
4. Attacker is an observer having extensive knowledge of H and also the controller but only observes H through a projection P_H.

In our example, we have designated the state x_{02} as the secret state. All other states, namely, $\{x_{00}, x_{01}, x_{03}, x_{04}, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}\}$ are the non-secret states. The set of states, $\{x_{00}, x_{01}, x_{02}, x_{03}, x_{04}\}$, belong to the normal machine, while the set of states, $\{x_{10}, x_{11}, x_{12}, x_{13}, x_{14}\}$, belong to the flipped machine. Suppose x_{0j} and x_{1j} have the same round register value. Then they are said to be equivalent states and may be expressed as $x_{0j}Ex_{1j}$. Therefore, $x_{00}Ex_{10}$, $x_{01}Ex_{11}$, $x_{03}Ex_{13}$ and $x_{04}Ex_{14}$, since in each case the round output values of the pairs of states are the same. But, $x_{02}Ex_{12}$ does not hold true since

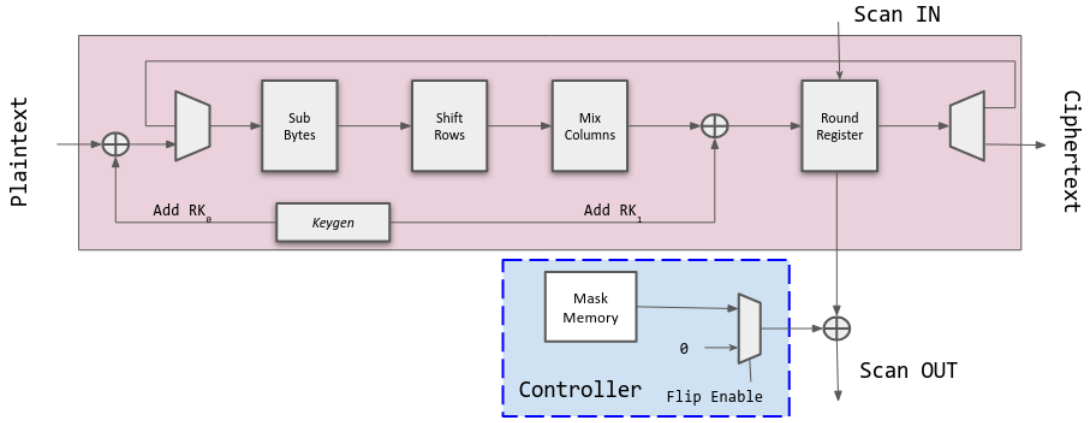


Figure 2.5: Architecture

$R_{a'} \neq R^f_{a'}$, i.e., the round register output at state x_{02} differs from round register output at state x_{12}.

2.2.4 Architecture

The architecture of our proposed scheme is shown in Fig. 4.4. The controller is placed at the scan out pin. After the key whitening phase, the input to the S-box is also fed to the controller. A tamper-proof memory (TPM) is used to store the masks computed using Algorithm 5, which is a novel mask determination algorithm, discussed in the next subsection. The controller controls the application of the mask, where the input to the S-box for the first round determines the mask value for its round output. For each byte of S-box input, a 32-bit mask value can be retrieved from the mask memory. For a 16-byte input, 16 different 32-bit masks are combined to generate a 128-bit mask. Due to the MixColumns operation of AES, a difference in one byte affects 4 bytes of a round output, which justifies using 32-bit mask values. If the Flip Enable signal is SET, then the 128-bit mask is selected at the MUX output and added (XORed) with the round output, bit by bit. When the Flip Enable signal is RESET, 0 is selected instead of the mask value. An overview of the Flip Enable signal is discussed below:

Flip Enable signal

The Flip Enable signal is SET by the controller and is associated with the FE event. Such an event is taken in the system model when the controller enables it, depending on the

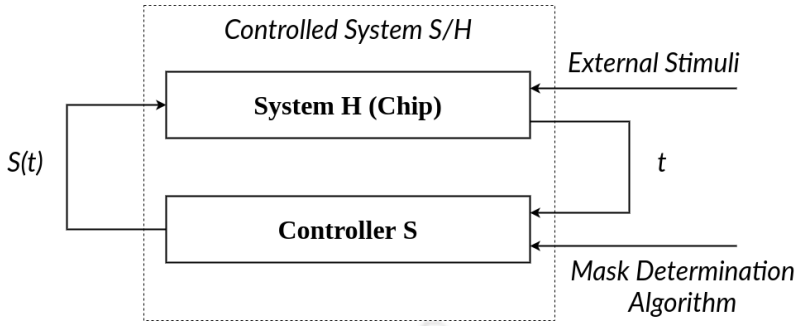


Figure 2.6: Block diagram of the controlled system S/H

plaintext input. The controller sets the SELECT signal for the 2x1 MUX. Accordingly, the Flip Disable event is associated with the RESET of the MUX signal. The MUX output is connected to the controller output pin. As a result, the pre-computed mask value or a 0 value is selected, respectively. The FE events in our system model shown in Fig. 2.4 are associated with transitions that are unobservable. The controller is minimally restrictive. Depending on the state visited, it enables a minimum set of transitions consisting of the events FE or FD.

A block diagram of the feedback network of the controlled system S/H is shown in Fig. 2.6. The controller measures t, the plaintext input, which is the current variable of the monitored system, and uses a mask determination algorithm to pre-compute mask values that help manipulate the system behavior across a set point. The intended behavior of the system is to prevent leakage of the secret key. Here, $S(t)$ denotes enabling or disabling of certain FE, FD events of the system model from observing t. When the system state makes a transition to a secret state, the pre-computed mask value from the specification is selected at the controller output. The external stimuli indicates accessing the chip by an external agent, which might be an adversary or a tester. Before going into the actual algorithm, an outline of the mask determination algorithm and the concept of the secret state is discussed as follows:

Mask determination algorithm

It is the specification provided to the controller. Since the differential scan attacks target the particular nature of the hamming weight obtained at output when differential plaintexts are applied, a modified Hamming weight distribution is chosen which we also refer to as an

intended Hamming weight distribution. The algorithm uses an original and an intended Hamming weight distribution. Given these distributions, a mask value Δ is computed for a plaintext pair (p, p') so as to make $(R(p) \oplus R(p') \oplus \Delta)$ map to a Hamming weight in the intended distribution which is different from the original one. The intended distribution is meant to resist differential scan attacks.

Secret States

The set of states designated as the secret states are the ones whose contents are sensitive, since such a state includes information about the secret key that needs to be prevented from leakage. These states are therefore meant to be kept secret from an adversary. If the contents of such states are disclosed, then it aids in the retrieval of the key. The specification is provided by the mask determination algorithm to the controller so that system behaves such as to keep the information hidden.

2.2.5 Mask Determination Algorithm

Algorithm 5 computes MDT and a mask value corresponding to a plaintext pair. $Mask((P, x))$ is a function that takes plaintext, P, as input and returns a hexadecimal number such that $R(P)$ and $R(P')$ differ in x bits. Only one plaintext pair retains its actual position while others are mapped to a different Hamming weight such that the distribution is uniformly spread. The difference between their original and modified Hamming weight is the hexadecimal mask value, Δ.

2.2.6 Discussion

The Hamming weight distribution is modified such that it is resistant to differential scan attacks. The attacker may even guess the correct key by selecting the one with the maximum frequency from all possible key distributions. To mitigate such a scenario, we modify our Hamming weight distribution with the objective to make the distribution uniform. Such uniformity means there would be no Hamming weight containing a single input pair, as is the case with Hamming weights of 9, 12, 23, and 24 in the actual Hamming weight distribution. Moreover, any correct key value frequency which is greater than incorrect key repetitions will be prone to attacks. Except for a few, many of the keys in

a particular Hamming weight distribution are repeated twice, and the rest appear just once. We propose an extra layer of security by devising our transform function such that a correct key after the transformation occurs just once. On top of that, since our algorithm distributes the key uniformly over the distribution, the chances of guessing the correct key are minimized. Any key hypothesis value of 10 or more, for one key byte, would suffice to prevent the attack. For 16 bytes of the keys, this occurs because the attacker lands with a huge complexity for key hypothesis while guessing the keys which can be computed as $10^{16} = (10^3)^5 \times 10 = (2^{10})^5 \times 10 = 2^{53} \times 10/8 > 2^{53}$. There exists $\binom{147}{19}$ possible ways of transforming the Hamming weight distribution. This is the number of possible non-integral solutions to $X_1 + X_2 + \cdots + X_{20} = 128$, where 128 is the total number of Hamming weights in the distribution, X indicates the number of plaintext pairs mapped to a specific Hamming weight, and index of X ranges from 1 to 20, since the spread lies between Hamming weights 5 to 24.

ALGORITHM 5: Computing a modified HW distribution and Mask value

Input : DT, Range (HW_a, HW_b)
Output : MDT, (P, P') and $Mask(P, x)$

```
/* DT and MDT are the original and modified distribution table;          */
Uni(HW) = {9, 12, 23, 24};
for All HW_i in HW and Size(HW_i) > 2 do
    Move one (P, P') in P_i from DT[HW_i] to MDT[HW_i];
for All HW_i in HW do
    for All (P, P') in P_i do
        HWC ← (i + 1); /* HWC is a Hamming weight counter               */
        Set ltCount to 1 , rtCount to 2;
        if HWC = i then
            if ltCount = rtCount then
                HWC ← i + ltCount;
                Increment ltCount;
            else
                HWC ← i + rtCount;
                Increment rtCount;
        j = HWC;
        Move (P, P') from DT[HW_i] to MDT[HW_j];
        Mask(P, j − i);
        if ltCount = rtCount then
            HWC ← i + ltCount;
            Increment ltCount;
        else
            HWC ← i + rtCount;
            Increment rtCount;
```

2.2.7 Complexity Analysis

Attack complexity

In case of any number of bit flips, to successfully launch the co-relation scan attack at a targeted hamming weight, HW_i, consisting of n S-Box input pairs, a correct mapping can be established at a complexity of $\binom{128}{n} \times n! \times n$ for recovering 1 key byte value. In a modified distribution with 11 input pairs per Hamming weight on an average, the complexity is $\binom{128}{11} \times 11! \times 11 \times 16$, which is a huge number.

Model Complexity

The actual discrete event system model implementation consists of $(256 + 1) \times 2$ states, that is $(2^8 + 1) \times 2 \approx 2^9$ states. Since a one-bit change in plaintext affects 32 bits (1 word) at the output, mask values can range between 0 to $(2^{32} - 1)$. Because of 256 plaintexts, FE and FD, the alphabet size is $256 + 2 = 258$. The size of transition set is $(256 \times 256 + 1 \times 256 + 2 \times 256) \times 2 = (259 \times 256) \times 2 \approx 2^{17}$. The number of secret states are 115 out of 256 possible round register output states.

Since there could be 2^{32} mask values, the probability of guessing an output is $1/(2^{32})$. Let us suppose that the information that k outputs are not flipped is known to the attacker. Among 128 possible output pairs, then the probability of guessing a correct distribution of non-flipped outputs is $\binom{128}{k} \cdot (1/(2^{32}))^{128-k}$. If k is not known, the probability of identifying the correct distribution is $1/(2^{32})^{128} \approx 1/2^{4000}$. 257 different states including a start state belong to each of the normal as well as the flipped submachine in our proposed defense model H, explained in Section 2.2.3.

2.2.8 Testability

Testability is not compromised due to masking. This is because there exist non-secret states that the designer knows and through which it is possible to observe and test faults. In our model automaton, the secret states are minimally chosen, i.e., a maximum of 128 states out of 256. So, at least half of the states can always be utilized to test the faults directly. Full testing can also be performed, including secret states, as we can have pre-computed mask values for each input with the prior knowledge of the AES key. Test patterns can be

modified by adding this mask to the test output. Functional testing can be performed using modified test patterns, by ATPG or traditional testing procedures without modifying the test interface. Furthermore, the test time is also not much compromised. Faults in the cipher module can be tested through the non-secret states only. The controller uses additional logic, the correctness of which can be tested using the secret states afterward. Also, plaintexts can be applied exhaustively, in any order, independent of any input difference. Since the designated secret states and non-secret states for our proposed technique do not depend on the order of applied plaintexts, thus, application of any plaintext would essentially maintain testability.

2.3 Security Proof

2.3.1 Verifying current-state opacity

The verification is shown here in a simplified manner using the estimator automaton, H_o, obtained from our DES defense model, H. Given an observable string of input symbols, the problem of verifying current-state opacity can be reduced to checking if the current state estimate belongs solely to a subset of secret states.

Given an initial set of states, Y_0, a set of secret states, Y_S, a set of non-secret states, Y_{NS}, an adversary projection mapping P_H and $\Sigma_o = \{a, b, a', b'\}$, the estimator automaton H_o is constructed as shown in Fig. 2.7, where $V = V_o$. Here, $Y_o = \{Z_0, Z_1, Z_2, Z_3, Z_4\}$ are the state estimates after having observed a sequence of transitions in H. $Y_{0,o} = \{Z_0\} = \{x_{00}, x_{10}\} = \{R_0\}$ is the initial state estimate which is actually the unobservable reach [1] of the states in the initial state set of H. x_{00} and x_{01} belong to the set of non-secret states and are pairwise equivalent. Given a string of input symbols, the current-state estimate obtained by simulating H_o on ω is $Z_3 = \{\{x_{02}\}, \{x_{12}\}\}$, where $\omega \in \Im_{y_o}$ and $\omega = a'$. Since the estimate set consists of $\{\{R_{a'}\}, \{R^f_{a'}\}\}$, the attacker cannot ascertain if the actual state belongs to a secret state or a non-secret state, since $x_{02} \in Y_S$ and $x_{12} \in Y_{NS}$.

The controller *deterministically* enables or disables the flip enabling transitions [126, 127]. In our state based model shown in Fig. 2.4, $R_{a'}$ is the round output value corresponding to a secret state. Since this state needs to be kept confidential from the attacker, the controller

[1] Unobservable reach of a state T is a set of states which are reachable from T using unobservable transitions

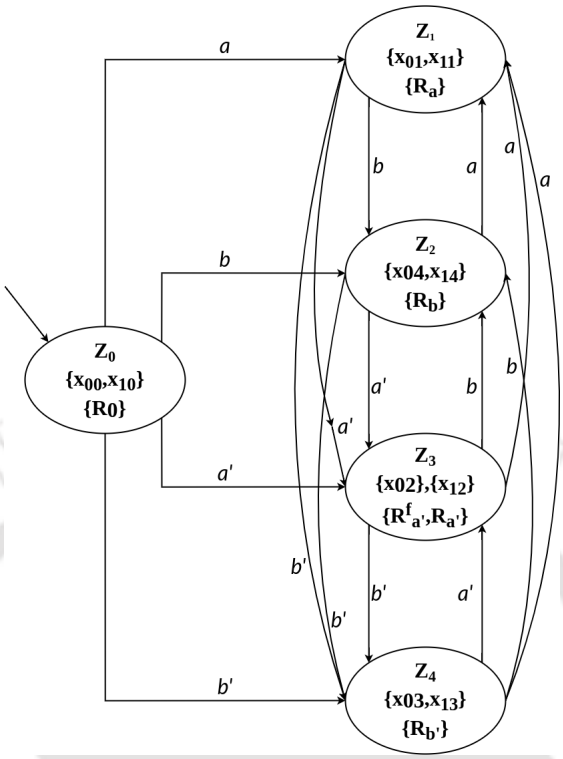


Figure 2.7: Estimator automaton for DES H

enables the flip enabling transition on reaching x_{02}, such that the attacker is never able to determine the actual contents of the intermediate register. The attacker essentially would get a round output value of $R^f_{a'}$ from its observations of the system. The system is, therefore, CSO as there exists a non-secret state, x_{12}, which is not equivalent and yet reached via an equivalent trace.

2.3.2 Why differential scan attacks will fail

We aim to prove here that a target Hamming weight is opaque. If this can be proved, then it would imply that if plaintexts are incorrectly mapped to the original Hamming weight distribution, then the key cannot be retrieved. The attack procedure shown in Algorithm 4 needs to compute the actual S-box inputs from the modified distribution, correctly, to retrieve the key.

Let us consider the Hamming weight distribution of plaintext pairs when mapped to round outputs, T_{HW}. The plaintext pairs that map to a specific Hamming weight, w, are added inductively to the set $T_{HW}(w)$. Now, the difference in outputs corresponding to

(p_1, p_1') is $R(p_1) \oplus R(p_1') \oplus Mask_{p_1} \neq R(p_1) \oplus R(p_1')$. Suppose DES H is CSO on having applied plaintext input p_1 or plaintext input p_1'. In our DES model, the system was CSO on reaching state x_{02}, where $R^f(a) = (R(a) \oplus Mask_a)$. Hence, $R^f(a)$ was opaque. Now, since any one of the states reached via plaintext inputs p_1 or p_1 can belong to the set of secret states, hence, either $R^f(p_1)(\neq R(p_1))$ or $R^f(p_1')(\neq R(p_1'))$ is obtained. Given any of these may occur, it implies $R^f(p_1) \oplus R^(p_1')$ is opaque to the adversary, since either $R^f(p_1)$ or $R^f(p_1')$ is opaque. Therefore, $D^f_{p_1}$ is opaque, where $R^f(p_1) \oplus R^f(p_1') = D_{p_1}$. Consequently, $HW(D^f_{p_1})$ is opaque to the adversary.

Suppose, there exists plaintext inputs p_1, p_1', p_2, p_2', p_3 and p_3', which when applied in any given order, generate round outputs $R(p_1)$, $R(p_1')$, $R(p_2)$, $R(p_2')$, $R(p_3)$ and $R(p_3')$, respectively. Now, $T_{HW}(5)$ consists of plaintext pairs such that the Hamming weight of their observed differences is all equal to 5. Considering that R_{p_1}, R_{p_2} and R_{p_3} belong to Y_S and the corresponding mask values are $Mask_{p_1}$, $Mask_{p_2}$ and $Mask_{p_3}$, respectively, then $HW(D^f_{p_1})$, $HW(D^f_{p_2})$ and $HW(D^f_{p_3})$ are opaque and all these Hamming weight values are equal to 5. Nonetheless, the system is $T_{HW}(5)$ opaque (there exist actual Hamming weights 16, 17, and 18 when computed originally without the masks), since the attacker is unable to infer the correct S-box input pair mappings corresponding to Hamming weight $T_{HW}(5)$.

2.3.3 Security considerations for generalized input differences

The security of the countermeasure is dependent on the mask values. 256 possible mask values are necessary for pre-computation in our scheme using our mask determination algorithm, depending on the intended modification of the Hamming weight distribution. Given any input difference δ applied between the plaintext pairs, 256 mask values will be required. A 0x01 difference is considered while computing a secure Hamming distribution for the same. In case of any general instances of the Hamming distribution, a different mask determination algorithm can instead be devised, with some logical modifications, for computing an updated set of 256 mask values. Depending on the distribution of the Hamming weight for a specific input difference in one byte of input, any different transformation function then needs to be used, such that 256 mask values are computed out of 2^{32} possible, considering that the mask values are of 32 bits. The modified Hamming weight distribution essentially requires the mask values to be applied only on different occasions, for any given

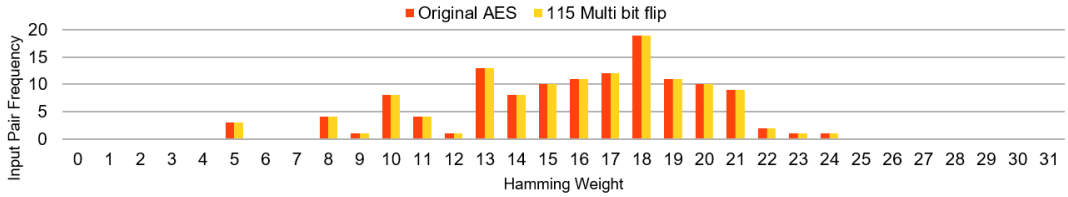


Figure 2.8: Hamming weight distributions

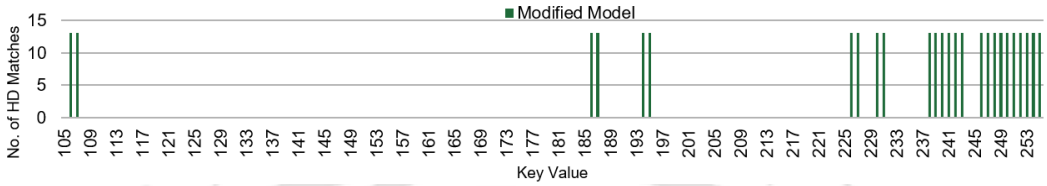


Figure 2.9: Signature Matches

input difference. This is because the original distributions specific to input differences vary from one another. Consequently, more secret states can be used based on the occasion at hand. There is no extra overhead due to such change since no changes are needed to be made to the hardware. The proof of the countermeasure can be guaranteed in the same manner as proposed in the manuscript. The number of secret states will increase on such pretexts.

2.4 Case Study

Our defense procedure is tested against simulated attacks, written in C language, on a 64-bit x86-64 Intel Core i7-6700 CPU at 2.6GHz, consisting of 8 virtual processors, 16 GB of RAM and running Ubuntu 18.04 OS. We test our defense while using δ = 0x01. Mentor Graphics Leonardo Spectrum Level 3 (Version 2018a.2) is used to evaluate the cost of implementation in terms of the extra number of gates, using the library fse0a_d_generic_core_ff1p1vm40c.

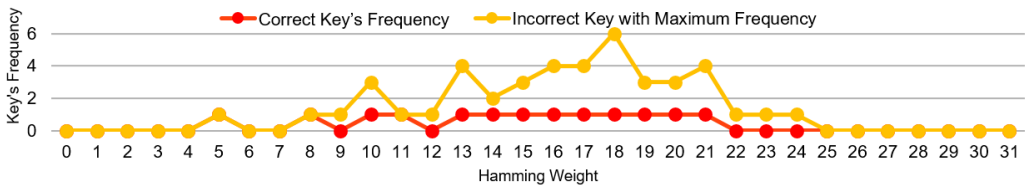


Figure 2.10: Correct and incorrect key distribution in co-relation scan attack

Table 2.2: RTL Components Summary for AES Implementation With and Without the Proposed Countermeasure

Component	Input	Size	Quantity		
			Original AES	Secure AES	Difference
Adders	2	4 bit	5	5	0
XORs	3	32 bit	4	4	0
	2	33 bit	2	2	0
	4	34 bit	2	2	0
	2	8 bit	166	160	-6
	4	9 bit	12	0	-12
	5	10 bit	4	32	28
	9	11 bit	16	16	0
Registers		128 bit	3	3	0
		32 bit	10	10	0
		8 bit	64	64	0
		4 bit	5	5	0
		1 bit	6	6	0
RAM		1K bit	1	1	0
Muxes	2	32 bit	8	8	0
	2	8 bit	32	48	16
	2	4 bit	2	2	0
	3	4 bit	1	1	0
	2	3 bit	1	1	0
	2	1 bit	2	2	0
	3	1 bit	1	1	0
LUT		256x8	2	2	0
Block RAM		256x8	39	39	0
Area Overhead					1.78%

2.4.1 Experimental Results

In our scheme, the DES model has 115 secret states. However, for experimental purposes, we have considered 128 secret states, with none of the plaintext pairs retaining their original Hamming weight. A 32-bit mask is used that requires a memory of 32×256 bits. The proposed defense is implemented on an iterative implementation of AES [128] similar to [69], in Verilog using Xilinx Spartan-7 xc7s100fgga676-2. The extra logic required for our implementation is given in Table 2.2. The area overhead is 1.78% in terms of equivalent gate counts. The simulation is done with Xilinx Vivado in the system configuration mentioned in the introduction of this section to evaluate our results.

2.4.2 Performance analysis

Our scheme is $(Y_S, P_H, 0)$ - opaque since 115 plaintext pairs are shifted. The analysis of our proposed countermeasure to scan-based attacks is discussed below:

Differential scan attack

In the modified distribution, all the plaintext pairs corresponding to an unique Hamming weight belong to another non-unique Hamming weight. So the attacker is unable to find an unique plaintext to XOR with a S-box input pair, for example, 0xE2 in the case of a 9 Hamming weight. Fig. 2.8 shows that all the unique Hamming weights, 9, 12, 23, and 24, no more contain actual unique plaintext pairs.

Signature attack

Here, all 128 Hamming weights need to match to recover the key. Since the Hamming weights are shifted, a change in the Hamming weight of at least one of the input pair changes the signature. As a result a match is not found with any of the signature HW_i. Fig. 2.9 shows the Hamming weight matching results with actual signature.

Co-relation scan attack

Because of the modified distribution, which contains only one correct plaintext pair in each Hamming weight, XORing the S-box inputs with the plaintexts gives incorrect keys with one correct key. Fig. 2.10 shows the frequency of incorrect keys is always greater than the correct key's frequency. Hence, the co-relation scan attacks fail.

2.5 Comparison with other works

A comparison of our proposed mask determination algorithm based countermeasure against some existing state-of-the-art defense schemes is presented in Table 2.3. The schemes SOSD-64, SOSD-64, DOSD-64, DOSD-128 [69] and HBSD [79] are used to evaluate our area overhead, security attack vulnerabilities, and impacts on testability. The iterative implementation of SOSD and DOSD has been considered while comparing. SOSD and DOSD refer to scan designs that perform static and dynamic obfuscation of scan data as

Table 2.3: Comparison of Different Designs

Design	Area Overhead (%)	Security Vulnerability	Testability
SOSD-64	1.52	TMOSA, Reverse Engineering Attacks	64 clocks before testing
SOSD-128	2.81		128 clocks before testing
DOSD-64	2.08	Reverse Engineering Attacks	64 clocks before testing
DOSD-128	3.91		128 clocks before testing
HBSD	2.42		Nil
DOSC	4.6	SAT attack	
DOSC + Functional Obfuscation	5.7		
Encrypt Flip-flop	5 (approx)	DynUnlock	
Proposed Scheme	1.78	None	Nil

reported in [69], where 64 or 128 scan cells may be controlled. SOSD-64 and SOSD-128 are susceptible to Test Mode Only Signature Attack (TMOSA). Moreover, all of the compared schemes have been shown to be vulnerable to reverse engineering attacks [129]. In terms of testability, the CUT is testable with no significant impact except that a fault in the additional hardware circuitry is rendered non-testable. We show that our scheme is secure, and there is a significant improvement on the area overhead measure with no compromise made to testability compared to other schemes. In [69, 79], authors show a detailed comparison of their works to other countermeasures in the literature.

Static and dynamic obfuscation techniques as discussed in [75, 77] protect the Intellectual Property (IP). The technique discussed in [75] suffers from DynUnlock attack [76]. The preliminary version of [77] is [73], which is shown to be vulnerable to ScanSAT attack [74]. Hence, the security of [77] still remains doubtful, as shown in ScanSAT [74]. The LFSR design can be reverse-engineered and the key update frequency p can be obtained by repeatedly applying the same test vector again and again [129]. As the LFSR's design can be recovered using reverse engineering, the LFSR seed can be obtained with the information of LFSR key sequences. Therefore, DOSC in [77] may be vulnerable. However, oracle access to design, if available, may render DOSC + Functional Obfuscation susceptible [74, 76].

We do not resort to IP protection and leave the scan chain as it is. Since we do not perform logic locking, such attacks are not effective against our countermeasure. Thus, our proposed countermeasure is resilient to reverse engineering and SAT attacks, with a lower area overhead of 1.78% compared to DOSC and DOSC + Functional Obfuscation in [77], which have area overheads of 4.6% and 5.7%, respectively, for the AES Core. Encrypt Flip-flop [75] reports an area overhead of approximately 5% on average and ranges from 1% to as high as 40% across various ISCAS'89 and ITC'99 benchmarks. As a result, it makes

our proposed countermeasure a preferable choice for the designer.

Our proposed scheme is not vulnerable to reverse engineering attacks since the pre-computed mask value considers performing multiple bit flips in one round output. This mask value is stored in tamper-resistant memory. Our proposed approach does not employ any additional circuitry involving logic gates except an XOR gate at the scan output. Therefore, reverse engineering the gate-level netlist design will not be beneficial in recovering the secret key. Attackers having the capability of creating stuck-at-faults by obtaining control of the flip-enable signal may be successful in launching various fault-injection or probing attacks. However, in real life, the chances of launching such attacks are very low since injecting faults through a probing attack or by using laser beam techniques is difficult and not easy to implement. Moreover, the flip-enable signal, or rather the MUX could be replaced in our design by directly feeding the mask memory to the XOR operation at the scan out pin. In that case, we need to use 256 secret states instead, yet have no impact on testability and with negligible overhead. We have employed the MUX and Flip-enable so that the controller is minimally restrictive.

2.6 Conclusion and Future Directions

In this chapter, we have proposed a simple yet effective countermeasure against scan-based attacks on IoT devices running an AES implementation. A security scheme using a controller hardware unit is presented that uses a mask determination algorithm to pre-compute mask values. The cryptocircuit implementation has been modeled in the framework of Discrete Event Systems. Our scheme ensures security without hampering testability. Using a proof of security, we show the resilience of the transformed Hamming weight distribution to differential scan attacks as well as attacks based on key frequencies.

To guarantee the proof of our countermeasure, the security analysis in this work using *Opacity* considers a particular instance of Hamming weight distribution (for example, when a plaintext difference $0x01$ is applied). In our future work, our objective would be to analyze the security aspects of our countermeasure irrespective of any given instance of a Hamming distribution. We would also further investigate our defense scheme in the presence of advanced DfT structures as well as on other ciphers.

In the following chapter, we look into RPL attacks, namely, rank and version attacks,

a class of network-level IoT attacks launched by an internal attacker using irregular RPL DIO control packets. Unoptimized network performance, resource exhaustion, network partitioning and DoS occur as a result of such attacks. These RPL attacks exploit the topology induced vulnerability of IoT-RPL networks. The network behaviour under the influence of such attacks pose no change from the normal. A mechanism is therefore necessary to create differentiating network characteristics for diagnosing these attacks. Moreover, directly incorporating the network packet information in states and transitions will effect in the state-space explosion problem. Also, a solution needs to be energy-efficient, scalable and accurate for successful implementation in IoT networks. Furthermore, detection of attack is not enough on this occasion since with the advent of botnet attacks and mobility, a malicious node might easily launch fresh attacks. Intrusion detection systems are naturally chosen for securing networks. Yet since signatures or known anomalies are naturally lacking in such attacks, signature or anomaly based IDS have limited success and generate lots of false postives. We analyze the attack from topology perspective to devise a novel countermeasure scheme, to not only detect rank and version attack but also identify a rank attacker node. Our proposed scheme utilises a centralised I-DES based IDS and a set of agents deployed at the leaf levels and is presented in the next chapter. Our proposed I-Diagnosability DES framework is adopted and extended to identify an attacker and uses extended finite state automata formalisms for the modeling and design of our IDS. Faults and network attacks create analogous deviations from the normal, which makes the classical Fault Detection and Diagnosis (FDD) theory naturally applicable to network attack detection. Moreover, our intelligent probing mechanism ensue differentiable attack characteristics due to generated responses. Employing I-DES based IDS does not require extensive training, protocol modifications, or encryption, etc. Furthermore, this being a software countermeasure that runs in a host, it does not require any upgradation or patching. The proposed scheme is validated in simulation and on a testbed with sufficiently large number of nodes. Results obtained show effectiveness of our technique in most aspects than state-of-the-art schemes.

CHAPTER

3

Mitigation of RPL-based Attacks

The Internet of Things (IoT) system is witnessing a rapid evolution, due to the ever increasing number of connected smart and pervasive devices [5]. Consisting of a multitude of connected heterogeneous objects, which we rather call as *things*, the IP-connected IoT is spread over diverse domains like smart cities, autonomous vehicles, industrial cyber-physical systems, smart homes, e-health sector, etc [108, 109]. IoT networks are typically Low power and Lossy Networks (LLN), comprising mostly of embedded sensors and actuators. Not only do such networks require to uniquely address billions of these connected devices, but also support embedded technologies for sensing and gathering data from the environment. With the mighty responsibilities in hand, IoT-connected resource constrained devices suffer from major operational challenges like constrained processing capabilities, inadequate memory and limited power. Hence, IoT remains vulnerable to a wide array of attacks because of insecure LLNs, device limitations, varying technologies, etc.

To enable efficient and reliable communication, IETF has standardized IPv6 Routing protocol for Low Power and Lossy Networks (RPL) [9]. The design of RPL is tailored for low-power IoT devices. RPL maintains loop-free Destination Oriented Directed Acyclic Graphs (DODAG). A DODAG is created and maintained using control messages, primarily, DODAG Information Object (DIO) for upward paths, DODAG Advertisement Object (DAO) for downward paths and DODAG Information Solicitation (DIS) for node joining. RPL ensures cost-optimized topologies by ordering participating nodes on the basis of an integer cost function, *rank*. Individual rank of a node determines its position in the DODAG, relative to a 6BR sink node (root). Also, a single *version* number prevalent in the DODAG

is maintained in DIO for consistency. Though RPL provisions various mechanisms and is secure enough from external attackers, yet the resource constrained nature of IoT devices, the typical characteristics of IoT networks such as lossy links, lacunas in infrastructure, dynamic topology, etc., can render IoT-RPL susceptible to internal attacks [85, 11, 86, 87].

Various internal attacks have been shown in the literature, of late, that make illicit use of RPL. Puppet attacks [88], advanced vampire attacks [89] make use of forged source routes, while attacks like energy depletion attack [90] and vampire attacks [91] drain resources by repeatedly sending useless data packets. Sybil attacks [92] and spam DIS attacks [93] have been shown to make use of DIS messages with counterfeit identities, essentially causing denial of service. DIO suppression attacks eavesdrop DIO messages for replaying it repeatedly in fixed intervals [94]. Out of the various DIO-specific attacks explored, proposed rank and version attacks continue to be of paramount importance since they are of low overhead and are realizable using DIO only. To launch such attacks, the rank and version number fields of a DIO message are fabricated causing formation of loops, sub-optimal routes, traffic redirection and network partitioning. Significant path delay is incurred since a large number of control messages are exchanged in the DODAG, resulting in energy depletion of the constrained nodes and disruption of network services. Moreover, rank attacks may be combined with other cross-layer attacks like selective forwarding attacks to alleviate the damage caused.

Proposed methods for securing IoT networks against RPL rank and version attacks have their own typical limitations[95, 96, 97, 98, 99, 100, 101, 102, 103, 32, 104]. Cryptography-based mitigation schemes are resource exhaustive and computationally heavy, especially in a network of resource constrained devices. Machine learning-based approaches require investment of extensive training time, as per the system under consideration. Protocol-based approaches require modifying the protocol policies. IDS based approaches do not suffer from these above limitations, but the implementation of these schemes for rank attack detection is challenging since attack behaviour resembles and normal behaviour. Hence, use of signature-based IDS and anomaly-based IDS schemes in the context of IoT attacks generate a large number of false positives. Furthermore, there exists many variants of rank attacks which present complex characteristics to evade detection capabilities of IDS. Formal verification of IDS schemes are also lacking.

This chapter presents an intelligent probing based scheme for the detection of rank and version attacks that also identifies location of malicious nodes. The probing mechanism helps differentiate the normal and attack behaviour. Our scheme incorporates I (Indicator)-Discrete Event System (DES) based IDS [47, 105, 48] and a set of agents with event monitoring enabled, that make use of probe packets [106, 107], judiciously. System failures and network attacks involve analogous behavioural deviations from the normal system functioning, which motivates the use of I-DES based IDS. Deploying our IDS does not require a change in protocol policies, encryption, extensive training time or any need for proprietary hardware support. Using our IDS also helps ensure a formally verifiable proof of correctness of our approach. The major contributions in this chapter are enumerated as follows:

- We propose a novel rank attacker identification scheme that also detects version attacks in IoT-RPL. Our scheme makes use of an intelligent active probing technique that helps create a deviation of attack traffic and normal traffic [106]. Our proposed scheme is centralized and uses a I-DES based IDS.
- We extend the power of traditional I-DES based IDS with attack type modeling for attacker identification.
- We prove the correctness and completeness of our approach by enumerating all the attack cases.
- The performance of our scheme is tested through simulations and real testbed. The experimental results highlight the applicability of our approach. Comparison of our scheme to state-of-the-art countermeasures shows our approach is energy-efficient with less packet overhead. The proposed solution is scalable, has minimum false positives and achieves more than 99% accuracy in identifying the malicious nodes.

The rest of this chapter is organized as follows: We discuss the related works and motivation in Section II. Section III is background. The design of our proposed scheme using a DES based IDS is presented in Section IV. Experimental results are summarised in Section V, highlighting the performance of our scheme, and we finally conclude with Section VI.

3.1 Related Work

We here discuss the various schemes proposed in the literature. The existing methods either employ mitigation techniques [130, 95, 96] using cryptographic solutions [35, 97, 98], acknowledgement based schemes [36], trust based methods [99, 100, 101], recent machine learning approaches [102, 103, 32, 104], or IDS based approaches [131, 132] using specifications and mathematical (statistical) methods to make DODAG secure. One of the primary works, VeRA [35], suggested the use of one-way hash functions generated by RPL root, where each of the nodes authenticate neighbours' rank by repeated usage of the function. TRAIL [36] improved upon VeRA by abstaining from a fully cryptographic technique. Newer attack vectors are also identified. Their proposed approach detects and mitigates topological inconsistencies in the network by checking for upward routes. They make use of encryption chain authentication as opposed to MAC authentication, thus ensuring backward secrecy. Their scheme lacks in scalability and requires maintaining state information. Nikravan et.al [97] utilise an identity based offline-online signature. Their solution is scalable compared to VeRA and TRAIL, requiring the size of signature to be independent of the network size. The above approaches to mitigate rank attacks however are resource exhaustive or computationally heavy.

Trust based methods have also been largely used in this direction [100, 101, 133]. They mostly resort to reputation score calculations and trust values for attack detection. SecTrust-RPL [99], a time-based trust-aware routing protocol, used a trust based principle that computes reliability, gained from message exchanges. They also validate their approach, however, it required each node to be run in promiscuous mode for sniffing packets. Later, a dynamic hierarchical trust model is proposed in DCTM-RPL [134]. Secure communication is shown to have been achieved by building up trust above a threshold value in their approach. Among the various protocols proposed [96, 40], a secure protocol, SRPL-RP [38], mitigates rank and version number attacks. It uses a timestamp threshold to validate a legitimate sender node. Though their approach improves upon overhead and average energy consumption, energy is wasted in the absence of any attacker. Furthermore it may be noted that protocol based approaches modify the protocol policies. There has also been significant contributions, of late, that use machine learning based methods. Specifically, deep learning based [33, 32] and artificial neural network based approaches [104] have been

applied to detect rank and other routing attacks. However, it is worth mentioning that such approaches require investment of extensive training time and further improvements in their accuracy can be achieved by better dataset.

Usage of IDS has received considerable attention over the years in the security research community. Network based Intrusion Detection Systems (NIDS) have been largely employed to secure the IoT network against attacks [135, 136, 132, 131]. NIDS for IoT are mostly signature-based (or, knowledge-based), anomaly-based, specification-based or hybrid [137, 138, 139]. SVELTE [132], one of the notably important proposal has shown the use of real-time intrusion detection in IoT. A specification-based IDS with hybrid placement that detects blackhole, sinkhole and selective forwarding attack, SVELTE used a mix of both signature based and anomaly based methods. While an IDS module runs on the root node, the firewall and response model runs on every node, which is again resource intensive. Some of the other limitations of the scheme are false detection and the lack of DIO synchronisation. Recently, FORCE [138], a specification based IDS that exploits the parent-child relationship is proposed and performs better than SVELTE in terms of detection rates and energy consumption. A version attack detection scheme using temporal logic based IDS [140] is shown, but a comparison of their scheme is lacking. A few works [141, 142, 130] improve upon SVELTE in terms of false positives. Version attack is mitigated and attacker is identified using trust-based distributed IDS [143] and also by distributed monitoring mechanisms [144]. A sink-based IDS is proposed in [139], but the scheme suffers from high computational overhead and average power consumption.

Few approaches in the literature have performed malicious node identification and isolation [145, 130, 143, 144]. In IoT networks, control packets are exchanged in the RPL for maintenance and a rank update legitimacy cannot be directly verified, since they are not differentiable across normal and spoofed conditions. An increased rank may be advertised due to various genuine reasons like a node gone off or not running, node services interrupted, etc. Moreover, variations of rank attacks lack direct anomalies or known signatures. In this regard, signature-based and anomaly-based IDS approaches in turn result in an increased number of false positives when generating relevant signatures or statistics. We overcome the discussed shortcomings by developing an energy-efficient and formally verifiable probing based scheme. Probing helps differentiate the attack characteristics from the normal network

characteristics. Analyzing the topological changes due to rank attacks aid our development of probing techniques for malicious node identification. We not only detect but also identify the location of the malicious node with enhanced precision, lower false positives and lower detection time. Our scheme is centralized and uses an I-DES based IDS, correctness of which can be formally verified. I-DES based IDS are accurate and generate minimal false positives [47, 105, 48, 146]. Moreover, using our proposed IDS does not require a change in protocol policies, extensive training time, encryption or a need for proprietary hardware support.

3.2 Background

In this section, we discuss the preliminaries of RPL protocol, DODAG creation and RPL attacks, namely, increased rank and version attacks, in particular.

3.2.1 RPL Protocol

RPL is inspired from distance-vector routing protocol, source routing protocol and DAG. It is the de-facto routing protocol that operates on top of IEEE 802.15.4 MAC layer while supporting multipoint-to-point traffic using upward routes, point-to-multipoint traffic using downward routes and a combination of the above routes to facilitate multipoint-to-multipoint traffic. Independent downward routes and upward routes are established in DODAG. Depending on the mode of operation, downward routes may be optionally supported. RPL supports three node types, namely, (i) Low Power and Lossy Border Routers (LBRs) which acts rather as gateway between LLNs and the Internet, (ii) Routers which can forward as well as generate traffic and (iii) Hosts that can generate but not forward traffic. Nodes are organized in the form of DODAG tree with a provision for parallel execution of multiple RPL instances, as shown in Figure 3.1. An RPL instance is uniquely characterized using RPL Instance ID and a DODAG using DODAG ID. The DODAG root is a special kind of node that acts as an LBR or a destination sink. The root determines and maintains the DODAG configuration parameters and starts disseminating DIOs [147].

In RPL, an ICMPv6 control message can be any one of these following types: (i) DODAG Information Solicitation (DIS) (ii) DODAG Information Object (DIO) (iii) DODAG Advertisement Object (DAO) (iv) DODAG Advertisement Object Acknowledgement (DAO-

ACK).

Rank is an integer value assigned to each node in the DODAG. All the nodes conforming to the inclusion policy in the DODAG instance are ordered on the basis of these values as per an instance defined metric. They are a measure of the position of the node relative to the sink node. A higher rank value pertaining to a node means it is more distant from the sink compared to another node with a lower rank value. Objective Functions (OF) are used for topology optimization depending on a set of goals that need to be met, such as link quality, hop count, energy consumption, etc. OF is used by the RPL to select the best routing path. Instances use OFs to determine the rank. The OF determines metrics that are included in the DIO messages. OF is realized using Objective Code Point (OCP) included in the DIO configuration options.

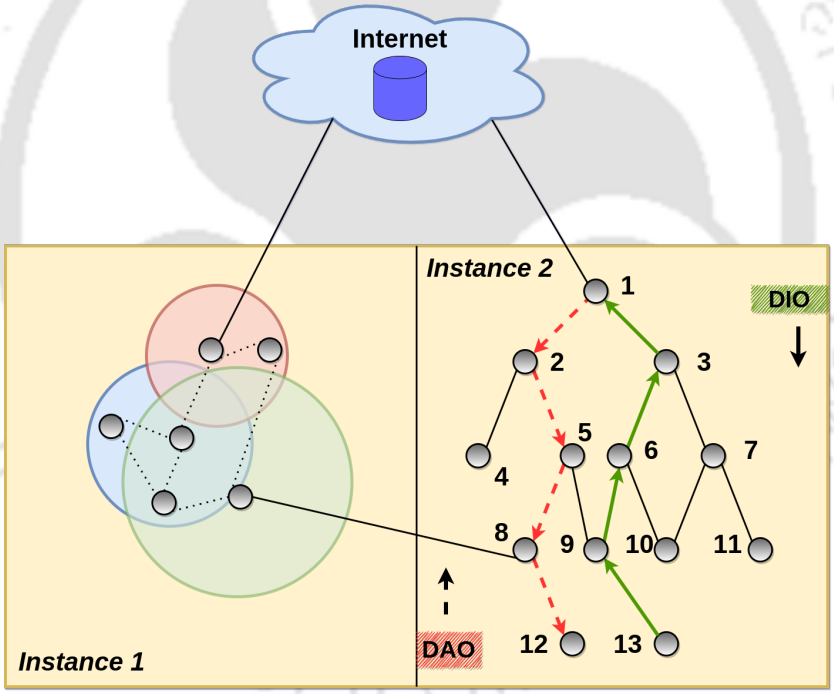


Figure 3.1: RPL DODAG

DODAG Creation and Maintenance

Creation and maintenance of an RPL DODAG is done using the DODAG control messages. When a DODAG is built, the root link local multicasts DIO messages for building upward paths. The rank value, objective code point and node ID are included in the DIO messages

[148]. DIO messages are periodically disseminated downwards, where the period is decided by the Trickle algorithm [149]. From the received DIO messages from neighbours, each node has the decision on selection of its parent set among its neighbours. Among its parent set, it selects a preferred parent from the best advertised rank value. Thus, when a node forwards a message to the DODAG root, the preferred parent is chosen by default. The received DIO message is then updated at the node and forwarded to its neighbours. On completion of DIO message exchanges till the leaf node, the upward route is created upto the DODAG root, consisting of preferred parents from each node. A node uses DIS broadcast messages to join a DODAG. DAO messages are used by the nodes for building downward paths.

3.2.2 Rank and version number attack

Alteration/Spoofing attacks in RPL have been widely investigated. Rank and version attacks in RPL are identified as misappropriation or alteration attacks where the ranking scheme is exploited, indirectly, making false advertisements using DODAG control packets [86, 95].

Version number attack: RPL incorporates versioning in DODAG to prevent loop formation and to ensure updated topologies. A malicious node makes use of the version number field to attract descendant nodes. False version number updation in the DIO advertisements practically actuate a DODAG tree rebuilding operation affecting the network performance, indirectly. As a result, energy exhaustion, loop formation, increased overheads ensue. Moreover, it provides avenues for launching more serious combined forms of attack.

Increased rank attack: One or more node(s) may misbehave in the network by increasing the rank values. We here restrict ourselves to the case where the network has a single misbehaving node. The malfunctioning node suddenly multicasts a DODAG Information Object (DIO) message to its neighbor nodes with an incremented rank value. The neighbor nodes, then, does the same, recursively, till the network upward routes are updated. Hence, there is a huge burst in control packet traffic in the network. The nodes being resource constrained illicitly face exhaustion of their battery. As a result of this type of attack, the network may even include loops that may not be mitigated using local repair mechanisms in RPL. Otherwise, the node simply joins at a lower rank in the network (i.e., more distant from the DODAG root) and such behavior may be primarily intended to starve a targeted node by disrupting communication.

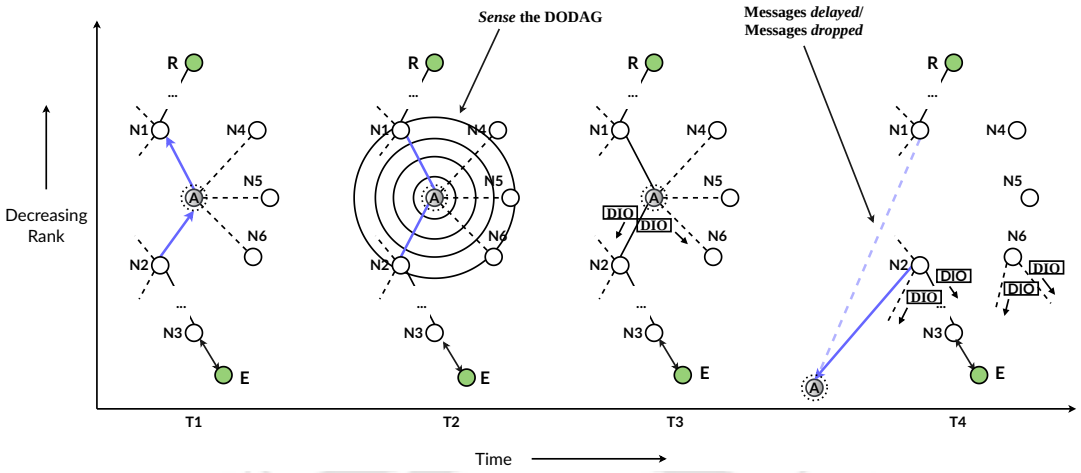


Figure 3.2: Rank Attack Timeline

3.2.3 Increased rank attack timeline

The increased rank attack timeline is shown in Figure 3.2. The time-slots $T1$ through $T4$ are briefly explained. [$T1$:] R is the root of the RPL DODAG while other nodes are numbered $\{A, N1, \ldots, N6\}$. Node, A is rendered vulnerable. [$T2$:] The vulnerable node probes rank values of the neighbouring nodes. [$T3$:] On having chosen a rank value, A now multicasts DIO messages with its updated rank value. [$T4$:] DIO messages are exchanged till leaf nodes update the upward route. The DODAG topology is modified at A.

3.2.4 Intrusion Detection Systems

Intrusion Detection Systems (IDS) are identified as one of the basic tools that are employed to protect networks and data. An alert is raised to the system administrator if any suspicious activity is detected by the IDS. An IDS can be software or hardware that are built to monitor and analyze the network packets that are sniffed or the events that occur in the host machine. Designing an IDS requires considering the processing ability and memory capacity of the nodes where they may be deployed. The primary components of an IDS are sensors that collect data, and an IDS engine that analyzes the collected data and reports to a network administrator for suitable actions. IDS are classified in the literature depending on the source of the data being monitored, depending upon the strategy it takes, and also depending on the monitoring techniques. Source of monitoring the data classifies IDSs into NIDS, HIDS and Hybrid. Based upon the strategy of detection, IDSs are classified

as signature-based, anomaly-based, specification-based, and hybrid. Depending on the monitoring technique, IDS are classified into active and passive monitoring which are further subdivided into centralized, decentralized and hybrid monitoring techniques.

3.3 Proposed Rank Attacker Identification Scheme

Here, we present the different aspects of our proposed scheme for RPL attack detection and identification. We introduce I-DES based IDSs followed by an overview of the detection methodology using our proposed IDS. We then discuss the employed techniques and algorithms to identify the attacker. The construction of normal, attack models and DES diagnoser that are indispensable for attacker identification are described next. Proof of correctness and completeness is presented subsequently. We assume that an attacker is unable to differentiate probe packets from normal packets and hence responses to them.

3.3.1 I-DES based IDS

Classical DES theory has been largely adopted in systems for Fault Detection and Diagnosis (FDD) [46, 44, 45]. Motivated from fault diagnosis, DES based IDSs have been successfully used in network attack detection [47, 48]. The characteristic similarities of network attacks and faults in DES literature is what motivates its usage. The basic idea is to develop a model for the normal functioning of the network and another for attack (fault) behavior. Additionally, multiple fault types in DES literature are diagnosed by developing exclusive fault DES models corresponding to each fault type. Each fault type leads to unique deviations from the normal behavior. Analogously, we augment traditional I (Indicator)-DES based IDS with attack types in our work here. It may be noted that an attack type corresponds to behavior of the network under the influence of a particular attacker. Attack type I-DES models corresponding to the location of the attacker are modeled. In I-DES based IDS an I-diagnoser is used as our IDS engine. It is a state estimator automaton which is constructed from the knowledge of normal and attack type DES models. The I-diagnoser observes system event traces and gives a decision on the system condition being normal or under attack by generating alerts. Attackers are identified only through the states that lie on the path after an indicator event has occurred. To summarize, by using I-DES based IDS, and given all possible attack instances, it can be ascertained if an attack can always be

exclusively identified, correctly and completely.

3.3.2 Overview of proposed attacker identification procedure

The primary research challenges in detection of rank attacks are as follows: **(i)** Nodes with rank values lower than the malfunctioning node, including the 6BR root, remain unaware of the inconsistency created in any subtree **(ii)** Normal scenario cannot be differentiated from the attack scenario by monitoring network traffic or topological changes. Sensing of network events at the leaf level using agents helps overcome the first challenge, while an intelligent probing technique helps overcome the second challenge discussed above. Active probe packets generate distinguishable packet sequences between normal and attack scenario. The system we consider consists of an IoT network of resource constrained devices using RPL. We use a centralised IDS, functioning at the network layer, working in a distributed manner with the help of agent nodes. An example of a DODAG with our IDS and agents deployed is demonstrated using Figure 3.3. The 6BR root (n_R) is software controlled and IDS handles communication for this node. The set of agents, $\mathcal{T} = \{n_1, n_2, \ldots, n_t\}$, with event monitoring enabled are deployed at the leaves. Henceforth, the IDS node is designated as n_R. The notations used are listed in Table 4.1.

Table 3.1: NOTATIONS

Notation	Meaning
$DIORQP$	Rank Update Packet
$DIOINMP$	DIO Rank Update Intimation Packet
$DIOvINMP$	DIO Version Update Intimation Packet
$PRQDP$	Probe Request Data Packet
$PRSDP$	Probe Response Data Packet
$PRSDP^*$	Delayed Probe Response Data Packet
PR_TO	Probe Timeout Event
$URDES$	Unreachable Destination Message

Components in the IDS: The block diagram of our proposed IDS with the basic components is shown in Figure 3.4 and are discussed here as follows:

- **Packet Sniffer:** It captures control and data packets in the network while working in promiscuous mode. Relevant packets are sniffed and others are dropped. It then forwards the sniffed packets to the "RQST_RSP_HANDLER()" component.

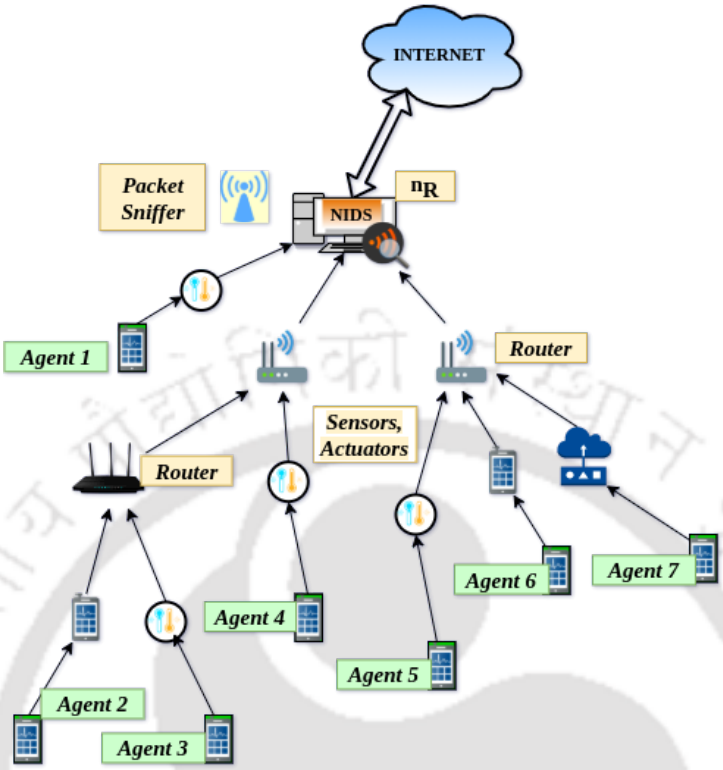


Figure 3.3: IoT network DODAG representation with IDS and agents deployed

- **RQST_RSP_HANDLER():** Its prime responsibility is to extract vital information from the control or data packets like source client's IP address, MAC address, Transaction identifier, etc. It also makes note of rank and version value attributes and generates the events *DIOINMP, DIOvINMP, URDES, PRQDP, PRSDP, PR_TO, PRSDP**. The generated events are passed to the DES diagnoser. The working procedure of this handler is described in Section 3.3.5.

- ***I*-DES Diagnoser:** This component diganoses the attacker node and is implemented as a software module. Given the knowledge of the DES model specifications pertaining to normal and attack type conditions, the diagnoser can be constructed. RQST_RSP_HANDLER() passes information regarding network events to the diagnoser. Based on the event parameters that are shared, the diagnoser generates an alert on attack detection or identification of malicious nodes. The usage and construction of the diagnoser is described in Section 5.2.6.

Attack detection and identification is sequentially carried out in phases. Version attack

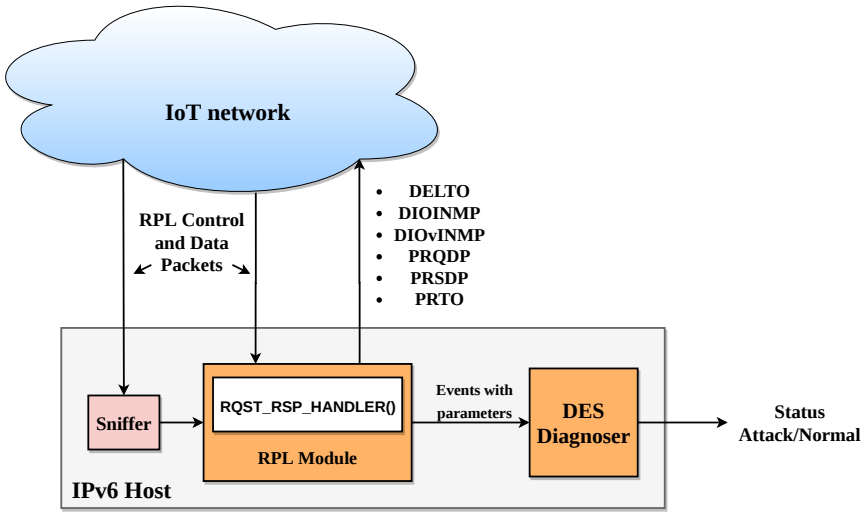


Figure 3.4: Architecture of proposed IDS

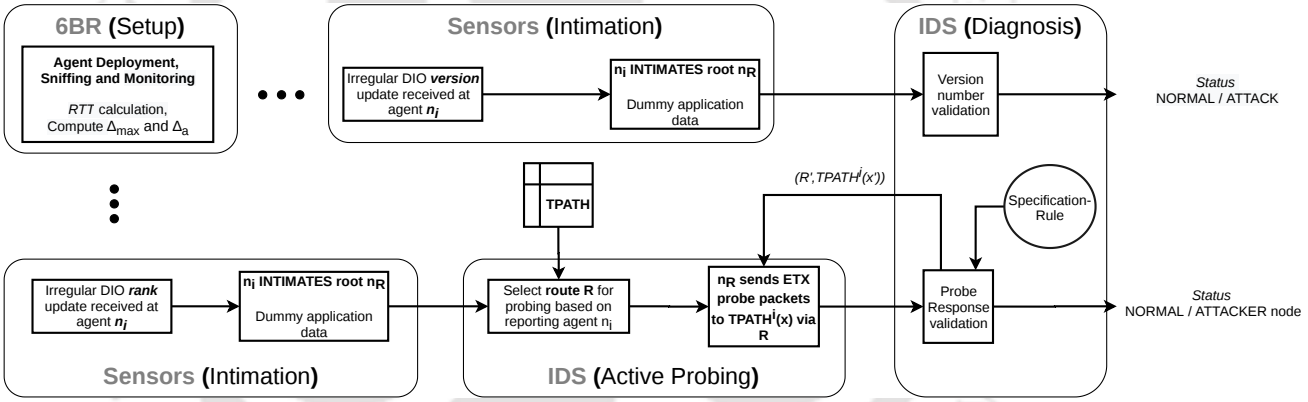


Figure 3.5: Workflow of proposed scheme

detection phases are **setup**, **intimation** and **diagnosis**, whereas, rank attack detection consists of **setup**, **intimation**, **active probing** and **diagnosis**. The working principle of our proposed scheme is demonstrated next. The flow of our scheme is shown using Figure 3.5. Prior to attack, network traffic is monitored and data is logged to setup the IDS as shown in the initial module. This forms the setup phase. IDS performs all the normal functionalities besides gathering and analysing the sniffed data in this phase. Considering there are t agents deployed, t tables (TPATH) are maintained and updated during this phase. Each table consists of round-trip time (RTT) values and information of the intermediate nodes between n_R and an agent. The table elements are ordered on rank values. After the IDS is setup, suppose an irregular DIO is received by an agent, n_j, where $n_j \in \mathcal{T}$

and $1 \leq j \leq t$. It then intimates this information as obfuscated application data to n_R after a random delay. This is the intimation phase. In case a version inconsistency is intimated, the diagnoser (IDS engine) validates the report and declares the status to be normal or a version attack, which is the diagnosis phase. On the other hand, on receipt of an irregular rank update intimation from an agent n_j, a j^{th} table is chosen. Subsequently, the RQST_RSP_HANDLER() on behalf of n_R sends ICMPv6 request packets to the nodes in this table, one by one, to probe for topological inconsistencies in the DODAG. This forms the active probing phase. An acknowledgement (ACK) response is generated for a probe request packet when received at a destination node. Now, a probe ACK response may not be received at all at n_R, genuinely, if any node has gone off, or if a link is broken, or a loop is present, and falsely if an attacker is present. So a missing ACK probe response cannot be directly marked as a suspicious activity. We hence characterize the received responses based on RTT values. RTT for a destination that is probed is computed and compared with RTT computed before intimation. Depending upon the learnt characteristics of RTT values from the sequence of probe packets sent, further probing is continued or a decision is taken by the diagnoser. The latter validates the probe responses against the DES model specifications provided at the start corresponding to normal as well as attacker specific behavior. Our normal and attack modeling capture the characteristic differences. The RTT values computed using the probing technique for a parent and child pair pose unique characteristics that help differentiate a normal and attack scenario. Moreover, the RTT characteristics for the sequence of nodes probed in the chosen table, i.e., j^{th} here, are differentiable in case of a specific attack node. The phases in our detection procedure are now sequentially demonstrated.

3.3.3 IDS Setup

This phase consists of administrator intervention for parameter setup. Traffic is monitored, relevant data is collected and parameters are measured for Network Traffic Analysis (NTA) purposes. Regular monitoring and sniffing yield to our detection procedure by maintaining tables and computing essential parameters, respectively. An array of table pointers, **TPATH**, is used for storing the intermediate node information. $TPATH^2$ in the example DODAG of Figure 4.3 is shown in Table 4.2. An element of the array, $TPATH^j$, stores the IP, MAC,

RANK and RTT values of the intermediate nodes along the path connecting the IDS, n_R to an agent n_j. $\langle TPATH^j \rangle_{SIZE}$ represents the size of $TPATH^j$, i.e., the number of nodes along the path $\overline{n_R n_j}$, excluding the root node. Values such as maximum RTT and maximum round-trip delay for 1-hop are computed and continuously updated. Variables Δ_{max} and Δ_a hold the maximum delay and admissible delay values, respectively. Sniffers deployed at n_R capture the traffic of underlying network as demonstrated in the Figure 3.3. The sniffing component retrieves general information from the packets communicated. The retrieved information from the control and data packets consist of DODAG ID, packet type (i.e., DIO, DAO, DIS, DAO-ACK, application), sender IP, destination IP address, and forwarding path information. Rank and version number values are also looked into and stored when necessary. The agent intimation phase is demonstrated in the following subsection.

Table 3.2: Table for $TPATH^2$

Node	Link-local IPv6 address	MAC address	Rank	RTT
B	fe80::2ca:3fff:fed6:8d56	00:ca:3f:d6:8d:56	1	1.23s
C	fe80::3340:70ff:fedf:71f1	31:40:70:df:71:f1	2	4.15s
D	fe80::f6eb:3fff:fe92:3cd2	f4:eb:3f:92:3c:d2	3	5.7s
E	fe80::6fbf:35ff:fec6:1ffd	6d:bf:35:c6:1f:fd	4	7.68s
n_2	fe80::a68d:bcff:fe6c:89d4	a4:8d:bc:6c:89:d4	5	9.84s

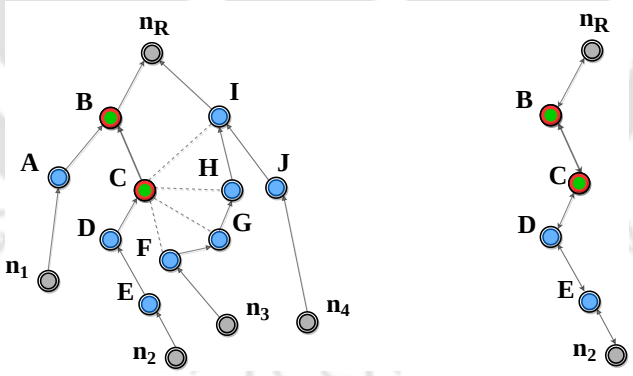


Figure 3.6: A DODAG instance (left) and path $TPATH^2$ (right). **IDS nodes** are denoted as *gray* circles, **non-attack nodes** are denoted in *blue* circles, **suspected attack nodes** are denoted in *red green* circles

3.3.4 Intimation

Our scheme consists of pieces of software, which are small programs, as agents for reporting any suspicious activity to the IDS, n_R. Based on their reports, version and rank attacks

are detected by the IDS using DES implemented at the root. The agents are event driven and perform minimally at leaf level in the monitored RPL-IoT network. They have no extra duties other than sensing suspicious activity and reporting. On receipt of an irregular DIO, the piggybacked information is obfuscated and reported to n_R. To prevent an attacker from profiling, the agents send the intimation packet with a random delay. Function of this component is explained using Algorithm 6. On receipt of a DIO packet $DIORQP$ with an increased version number, an agent node n_j reports an intimation packet to n_R. If the DIO is a trickle timer update, with an used version number and incremented rank value, the DIO is marked suspicious. A DIO is also marked suspicious if update is trickle timer inconsistent with an increased rank value. Information regarding such DIO receipts are also reported to n_R. Active probing and diagnosis phases are demonstrated through the RQST_RSP_Handler and I-DES diagnoser, respectively, in the following subsections.

ALGORITHM 6: Agent Intimation Procedure

Local Variables: **rank**, *currVerNum*
Input: Received DIO packet $DIORQP$
Output: Intimate received DIO packets $DIOINMP$, $DIOvINMP$

```
if (ipd(DIS) = ips(DIORQP)) and (macd(DIS) = macs(DIORQP)) then
    if verNo(DIORQP) > currVerNum then
        Send DIO receipt intimation DIOvINMP to n_R;
    if DIORQP is Trickle Inconsistent then
        if rank(DIORQP) > rank then
            Send DIO receipt intimation DIOINMP to n_R;
    else if DIORQP is Trickle Consistent then
        if verNo(DIORQP) = currVerNum and rank(DIORQP) > rank then
            Send DIO receipt intimation DIOINMP to n_R;
```

3.3.5 RQST_RSP_HANDLER()

The working our algorithm is described as follows. The input it takes are:

- DIO intimation packets that are reported from agents on receipt of irregular DIO packets.
- Probe request packets from the buffer that are yet to be sent (this becomes possible as RQST_RSP_HANDLER() is part of the modified RPL).
- Probe response packets.

ALGORITHM 7: RQST_RSP_HANDLER()

Data: $c1$, ver, $rcvd$, $flag = FALSE$, Δ_{max}, Δ_a, $lastSend$, rtd, j, rch
Input: DIO intimation packets, Probe response packets, $TEST\_FLAG$
Output: Events: $PRQDP$, $DIOINMP$, $DIOvINMP$, $PRSDP$, PR_TO, $PRSDP^*$, $URDES$

```
while Δ_max and Δ_a are not NULL do
    if Version update is reported then
        while (TEST_FLAG == 1) do
            Generate event DIOvINMP;
    if Rank update is reported then
        Generate event DIOINMP;
        j ← {i|n_i.IP == DIOINMP_IPS};
        Generate event PRQDP;
        Send ICMPv6 probe packet to TPATH^j[0] via stored downward route R;
        Start clock timer c1();
        lastSend = 0;
    if Received packet is a probe response then
        rtd ← TPATH^j[lastSend].RTT;
        Increment lastSend;
        if (c1() ≤ rtd + Δ_a) then
            Generate event PRSDP;
            rch ← lastSend − 1;
            Stop clock timer c1();
            Generate event PRQDP;
            Send ICMPv6 probe packet to TPATH^j[lastSend] via stored downward route R;
            Start clock timer c1();
        else if (c1() > rtd + Δ_a) then
            if (flag == FALSE) then
                Generate event PRSDP*;
                Stop clock timer c1();
            else if (flag == TRUE) then
                Generate event PRSDP*;
                Stop clock timer c1();
                Generate event PRQDP;
                Send ICMPv6 probe packet to TPATH^j[rch] via DAO advertised downward route R';
                Start clock timer c1();
                flag = FALSE;
    if (c1() > Δ_max) AND (No response packet is received) then
        Generate event PR_TO;
        Stop clock timer c1();
        Increment lastSend;
        if (flag == TRUE) then
            Generate event PRQDP;
            Send ICMPv6 probe packet to TPATH^j[lastSend] via DAO advertised downward route R';
            Start clock timer c1();
        else if (flag == FALSE) AND (lastSend < TPATH^j_SIZE) then
            Generate event PRQDP;
            Send ICMPv6 probe packet to TPATH^j[lastSend] via stored downward route R;
            Start clock timer c1();
        else if (flag == FALSE) AND (lastSend == TPATH^j_SIZE) then
            Generate event PRQDP;
            Send ICMPv6 probe packet to TPATH^j[lastSend] via DAO advertised downward route R';
            Start clock timer c1();
            flag = TRUE;
```

- TEST_FLAG indicates when to detect and identify the attack by sending probe packets to intended nodes.

If the values Δ_{max} and Δ_a, have been computed, the diagnoser sets TEST_FLAG = 1 (Line 1). The two values are pre-computed during non-attack condition in the RPL instance in use as discussed in Section 4.3.2. The handler outputs events, namely, $PRQDP$, $PRSDP$, $DIOINMP$, $DIOvINMP$, PR_TO, $PRSDP^*$, $URDES$, which are all passed to the DES diagnoser. The model variables used are $c1$, $flag$, Δ_{max}, Δ_a, j, $lastSend$, rtd and rch. They are shared among the handler and the I-DES diagnoser. When the TEST_FLAG is set by the diagnoser, it means that the attack detection and identification phase can be started. The algorithm is now explained step-wise. The I-DES diagnoser gets executed and remains so till the DODAG remains operational. Diagnoser sets the TEST_FLAG = 1 which is its initial transition.

If a version update intimation is reported, it checks if TEST_FLAG = 1 (Line 3). The event $DIOvINMP$ is sent to the diagnoser (Line 4). Diagnoser sets TEST_FLAG = 0 until a decision on the version inconsistency is made. If an irregular rank update is intimated, the event $DIOINMP$ is passed to the diagnoser (Line 8). Model variable j stores the index of the $TPATH$ array used. The variable is shared with the diagnoser (Line 9). $PRQDP$ event is passed to the diagnoser and a probe packet is sent to the node at 1-hop distance from the root in the table $TPATH^j$ (Line 11). $TPATH^j$ stores a saved route R for agent node n_j. $lastSend$ stores the index of the node in $TPATH^j$ to which the last probe request packet is sent. A clock timer is started to maintain a record of the transmission time of the packet that can be uniquely identified using a transaction identifier value, $transid$.

The module described through lines 15 to 37 is taken on receipt of a probe response packet. Variable rtd is set to the round-trip delay of the node to which the probe packet was last sent. The variable $lastSend$ is incremented (Line 16). The total response time it takes for a particular node is computed using the clock variable, $c1$ and is compared against a pre-computed RTT (old). We use Δ_a to characterise the admissible delay while awaiting a probe response. In case a response packet is not received at n_R after a Δ_a time period beyond the expected RTT, we consider it as delayed response. If $c1$ does not exceed $rtd+\Delta_a$, the generated event $PRSDP$ is passed to the diagnoser (Line 18). The variable rch is set to

point to the last node whose packet is received before delay timeout occurs (Line 19). The clock timer is then stopped and another request packet is sent to a subsequent node (Line 21). Consequently, the event $PRQDP$ is passed to the diagnoser. Clock timer is restarted to count the RTT via the stored route (Line 23). If $c1$ exceeds $rtd + \Delta_a$, then a $flag$ variable is checked (Line 25). It is set equal to FALSE during the algorithm initialization. In case $flag$ = FALSE and TEST_FLAG = 1, a delayed response received event $PRSDP^*$ is passed to the diagnoser which sets it to 0 (Line 27). The clock timer is stopped. On the other hand, if $flag$ is TRUE and a probe response packet is received from some node, suppose x, beyond $rtd + \Delta_a$, then the event $PRSDP^*$ is generated and passed to the diagnoser and clock timer stopped (Line 32). A request packet is sent via current downward route R' to node x, clock timer is restarted and $flag$ is set to FALSE (Lines 33-36).

The module described through lines 40 to 59 checks if $c1$ counts beyond a maximum probe timeout period and no response packet is received at n_R. We use Δ_{max} to characterise the maximum delay after next probe request is made. Consequently, a probe timeout event generated here is PR_TO which is passed to the DES diagnoser while the clock timer is stopped and $lastSend$ is incremented by 1 (Line 42). Three conditions over the variables $flag$ and $lastSend$ are checked if they are met. If $flag$ is determined to hold TRUE, then event $PRQDP$ is passed to the diagnoser and an ICMPv6 probe packet is sent to $TPATH^j[lastSend]$ via a current downward route R' and clock timer $c1$ is started (Lines 45-47). On the other hand, if $flag$ is found to be false while $lastSend$ is less than the size of $TPATH^j$, then event $PRQDP$ is passed to the diagnoser and an ICMPv6 probe packet is sent to $TPATH^j[lastSend]$ via the stored downward route R and clock timer $c1$ is started (Lines 50-52). If $flag$ is found to be false while $lastSend$ equals the size of $TPATH^j$, then event $PRQDP$ is passed to the diagnoser and an ICMPv6 probe packet is sent to $TPATH^j[lastSend]$ via a current downward route R', clock timer $c1$ is started and variable $flag$ is set to TRUE (Lines 55-58).

3.3.6 I-DES Model and I-Diagnoser

3.3.7 Basics of Discrete Event Systems

This subsection presents the prerequisites of our proposed DES framework. Using the knowledge and demonstration of this section, we later show that the framework can be

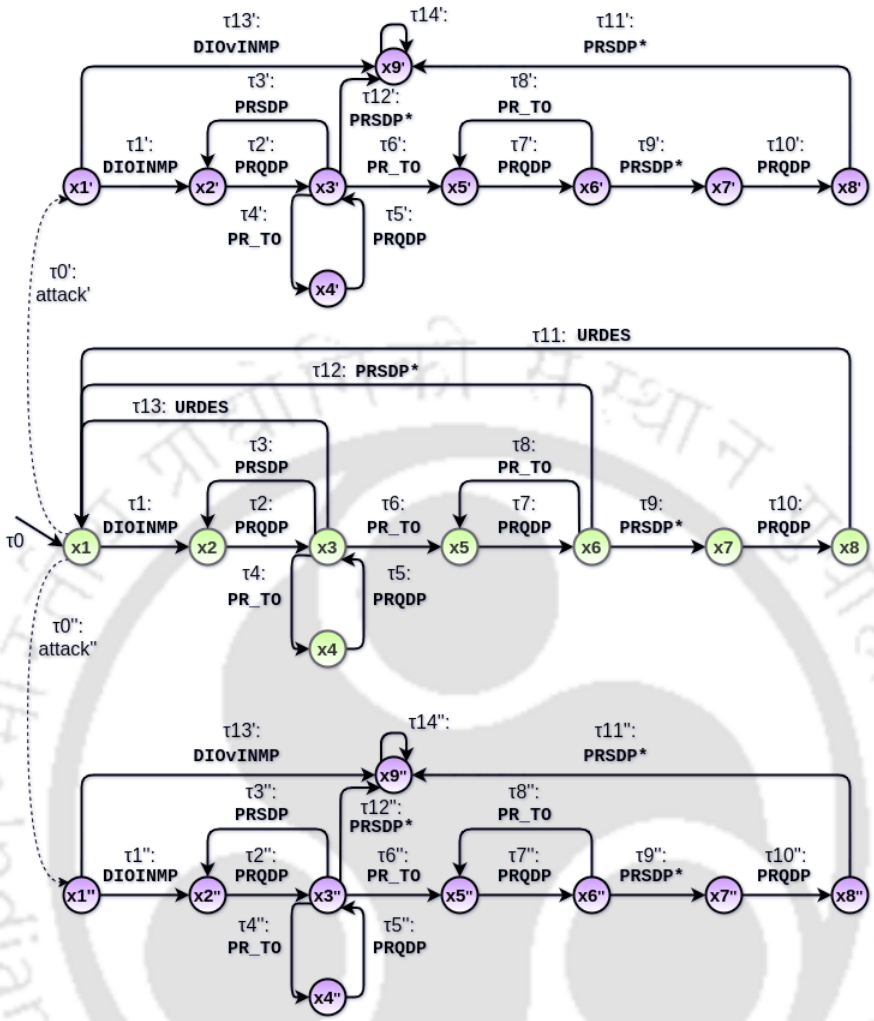


Figure 3.7: DES model H

used to diagnose attacks in wireless sensor networks containing resource constrained nodes [150, 146, 151].

I-DES Model

The I-DES model H is formally defined as a 6-tuple $H = \langle X, X_0, \Sigma, V, C, \Im \rangle$ [46, 152, 153, 154, 45]. Here, X is the set of states and is finite, $X_0 \subseteq X$ is the set of initial states, Σ is the finite set of events, V is the finite set of model variables, C is the finite set of clock variables and $\Im$ is the finite set of transitions. Elements of the set of model variables assume values from their respective domain sets. Suppose if $V = \{v_1, v_2, \ldots, v_n\}$ is the set of model variables (for some finite value of n) where each element v_i takes some values from its

Table 3.3: LIST OF SYMBOLS

Symbol	Definition
H	DES model
Σ	Set of events of the DES model H
Σ_m	Set of measurable events of the DES model H
Σ_{um}	Set of unmeasurable events of the DES model H
V	Set of model variables of the DES model H
$\Im$	Set of transitions of the DES model H
τ	A transition $\tau \in \Im$
Y	Set of states of the DES model H
Y_0	Set of initial states of the DES model H
σ	Event on which a transition is enabled
$check(V)$	Condition(s) on a subset of model variables, V
$assign(V)$	Assignment(s) on a subset of model variables, V
$L(H)$	Set of all traces *generated* in H
A_i	i^{th} attacker
Y_N	Set of normal states of the DES model H
Y_F	Set of faulty states of the DES model H for fault type F
Y_{A_i}	Set of attacker states of the DES model H for attacker A_i
σ_{A_i}	Event corresponding to attack launched by attacker A_i
O	Diagnoser of DES H
Z	Set of states of the diagnoser, O, also called O-states
Z_0	Set of initial nodes of the diagnoser, O
A	Set of transitions of the diagnoser, O, also called O-transitions

domain set Dom_i. The domain of each of the clock variables is the set of non-negative reals, R. A transition $\tau \in \Im$ is defined as a 7-tuple $\langle x, x^+, \sigma, \phi(V), \Phi(C), Reset(C), Assign(V)\rangle$, where x, x^+ are the source state and destination state of transition τ respectively. Due to the occurrence of the event $\sigma \in \Sigma$, the transition τ is enabled. $\phi(V)$ is defined as a boolean conjunction of equalities over some subset of the model variables, V, and which needs to hold true overall for a transition to be taken. $\Phi(C)$ is an invariant condition over some subset of the clock variables C. $Reset(C)$ is a subset of clock variables to be reset and $Assign(V)$ is a subset of model variables along with an assignment of values from their corresponding domains. Some of the fields in the tuple representing a transition maybe be denoted by "-". For example, if "-" is used for $\phi(V)$ or $Assign(V)$, then it would mean that no condition needs to be met (i.e., the condition is implicitly TRUE) or NO assignment is required respectively.

The I-DES modeling of the IoT-RPL network is demonstrated here. The principle of detection and identification by the diagnoser is discussed. We later show that an attacker,

if present, is correctly located in the DODAG.

Assumptions in the normal condition: After receiving an intimation from an agent n_j, a node is sent probe request packet along $TPATH^j$. Subsequent probes are then sent depending upon the measured RTT. During the normal condition, two cases can arise here. (i) While awaiting a probe RSP packet, destination unreachable message is received. (ii) After the rank update intimation is received, if a RSP packet is received after the delay timeout period, for a probe packet sent via current DAO advertised downward route. Both of these cases can occur due to a local repair operation and has been modeled as a normal DES.

Assumptions in the attack condition: In the presence of an attacker advertising illegitimate rank or version values, inconsistencies occur in the upward and downward routes. As a result, two cases can arise here as well. (i) Version inconsistency is intimated by agent node. (ii) A probe request packet sent to a child node of the attacker node along $TPATH^j$ (considering the reporting agent node to be n_j) responses with delay. Given an attack behavior due to node A, the above cases are modeled as attacker A type I-DES model. Since attacker can be located at multiple positions in the DODAG, there are multiple attacker type models. The diagnoser is constructed from the I-DES models. In both of these cases, since the diagnosability condition (Definition 6, Appendix B) is satisfied each time because there are no uncertain states (Definition 11, Appendix C), an attacker location is identified. The attack as well as the attacker type behavior are different from the normal or other attacker type behavior, respectively.

We consider the model of a networked system consisting of resource constrained IoT nodes arranged in a RPL DODAG. The notations used are listed in Table 3.3 and definitions of the various DES terminologies can be seen in Appendix A. The I-DES model which has been used to represent the *Probe Request Response* sequence during normal and rank or version attack conditions is drawn using Figure 3.7. The various components of the I-DES model $H = \langle X, X_0, \Gamma, V, C, \Sigma \rangle$ for the *Request Response* sequence after an irregular DIO intimation is received are discussed.

The state set X with initial set of states X_0 ($X_0 \subseteq X$) symbolise the control states of the RQST_RSP_HANDLER() component of the IDS. The normal I-DES model states and attacker type model states together constitute the state set $X = \{x1, x2, \ldots, x8, x1',$

$x2'$, ..., $x9'$, $x1''$, $x2''$, ..., $x9''$}. In our model, the set of model variables, V = {*ips, ipd, transid, j, flag, lastSend, rtd, ver, rch, {ips*$_1$*, ips*$_2$*, ..., ips*$_t$*}*}. The model variables correspond to program and data variables that are internal to the IDS. Certain program variables are designated as the clock variables, C which are absolute values of clock timer that can is SET and RESET using commands. In real-time applications, timing constraints are expressed by satisfying the conditions on the clock variables. We use a single clock variable in the set of clock variables, i.e., $C = \{c1\}$. Event set Σ contains the packet communication events. In our model, the set of events, Σ = {*DIOINMP, DIOvINMP, URDES, PRQDP, PRSDP, PR_ TO, PRSDP***, attack', attack''*}. A transition is enabled if the conditions are satisfied and is said to be taken on the occurrence of the associated event. The transitions set Γ consists of transitions {$\tau0$, $\tau1$, ..., $\tau13$, $\tau1'$, $\tau2'$, ..., $\tau14'$, $\tau1''$, $\tau2''$, ..., $\tau14''$}.

Considering that there is one attack node among n nodes, i.e., $\{A_1, A_2, \ldots, A_n\}$, in the IoT network, the state set, X, can be partitioned into disjoint sets X_N, X_{A_1}, X_{A_2}, ..., X_{A_n}, where, X_N represents the set of states belonging to the normal behavior of the network, while states of the form X_{A_i}, $1 \leq i \leq n$, $i \in \mathcal{N}$, represent the behavior of the network if A_i is the attack node. For simplicity, we model using 2 nodes, A_1 and A_2, among which one is an attack node, hence $X = X_N \cup X_{A_1} \cup X_{A_2}$. In Figure 3.7, the non-primed states are the states when the system behaves normally while the single and double primed states represent the system under attack by the nodes A_1 and A_2, respectively. The events of the system is disjoint union of measurable events and unmeasurable events Σ_m and Σ_{um}.

I-DES behavior under normal circumstances

The behavior of H under normal circumstances is shown in Figure 3.7. The system, when functioning normally, is represented using the states {$x1$, $x2$, ..., $x8$} and the transitions {$\tau0$, $\tau1$, ..., $\tau13$}. The initial state of X_0 is $x1$. We next discuss the transitions in normal condition as follows:

- $\tau0$, the initial transition leads to the initial state $x1$ as shown in Figure 3.7. It is assumed while modeling that the constant timeout values, Δ_{max} and Δ_a, have been computed and then $\tau0$ takes place. There is no explicit event that triggers $\tau0$. Occurrence of $\tau0$ implies that the DES model is invoked when the timeout values

Table 3.4: TRANSITIONS $\Im$ IN H CORRESPONDING TO NETWORK PACKET FRAMES

$Event(\sigma)$	Transition	$\phi(V)$	Assign(V)	$\phi(C)$	Reset(C)
DIOINMP	$\langle x1,x2\rangle,\langle x1',x2'\rangle,\langle x1'',x2''\rangle$	$ips_j \equiv DIOINMP_{IPS}$	-	-	-
		$ipd \equiv DIOINMP_{IPD}$	-	-	-
DIOvINMP	$\langle x1',x9'\rangle,\langle x1'',x9''\rangle$	$ips_j \equiv DIOvINMP_{IPS}$	-	-	-
		$ipd \equiv DIOvINMP_{IPD}$	-	-	-
		$ver < DIOvINMP_{VERNUM}$	-	-	-
PRQDP	$\langle x2,x3\rangle,\langle x2',x3'\rangle,\langle x2'',x3''\rangle$	-	$ips \leftarrow PRQDP_{IPS}$	-	-
	$\langle x4,x3\rangle,\langle x4',x3'\rangle,\langle x4'',x3''\rangle$	-	$ipd \leftarrow PRQDP_{IPD}$	-	-
		-	$transid \leftarrow PRQDP_{TRANSID}$	-	-
		-	$TEST\_FLAG \leftarrow 0$	-	-
		-	$lastSend \leftarrow lastSend + 1$	-	$c1 \leftarrow 0$
PRQDP	$\langle x7,x8\rangle,\langle x7',x8'\rangle,\langle x7'',x8''\rangle$	-	$ips \leftarrow PRQDP_{IPS}$	-	-
		-	$ipd \leftarrow PRQDP_{IPD}$	-	-
		-	$transid \leftarrow PRQDP_{TRANSID}$	-	-
		-	$TEST\_FLAG \leftarrow 0$	-	-
		-	$flag \equiv FALSE$	-	$c1 \leftarrow 0$
PRQDP	$\langle x5,x6\rangle,\langle x5',x6'\rangle,\langle x5'',x6''\rangle$	-	$ips \leftarrow PRQDP_{IPS}$	-	-
		-	$ipd \leftarrow PRQDP_{IPD}$	-	-
		-	$transid \leftarrow PRQDP_{TRANSID}$	-	-
		-	$flag \equiv TRUE$	-	$c1 \leftarrow 0$
PRSDP	$\langle x3,x2\rangle,\langle x3',x2'\rangle,\langle x3'',x2''\rangle$	$ips \equiv PRSDP_{IPD}$	-	-	-
		$ipd \equiv PRSDP_{IPS}$	$TEST\_FLAG \leftarrow 1$	-	-
		$transid \equiv PRSDP_{TRANSID}$	$rch \leftarrow PRSDP_{IPS}$	$c1 < ipd.RTT + \Delta_a$	-
PR_TO	$\langle x6,x5\rangle,\langle x6',x5'\rangle,\langle x6'',x5''\rangle$	-	-	$c1 \geq \Delta_{max}$	-
PR_TO	$\langle x3,x4\rangle,\langle x3',x4'\rangle,\langle x3'',x4''\rangle$	$lastSend < M^j$	$TEST\_FLAG \leftarrow 1$	$c1 \geq \Delta_{max}$	-
PR_TO	$\langle x3,x5\rangle,\langle x3',x5'\rangle,\langle x3'',x5''\rangle$	$lastSend = M^j$	$TEST\_FLAG \leftarrow 1$	$c1 \geq \Delta_{max}$	-
PRSDP*	$\langle x6,x7\rangle,\langle x6',x7'\rangle,\langle x6'',x7''\rangle$	$ips \equiv PRSDP^*_{IPD}$	-	-	-
		$ipd \equiv PRSDP^*_{IPS}$	-	-	-
		$transid \equiv PRSDP^*_{TRANSID}$	-	-	-
		$rch! = NULL$	-	-	-
		$flag \equiv TRUE$	-	$c1 \geq ipd.RTT + \Delta_a$	-
PRSDP*	$\langle x6,x1\rangle$	$ips \equiv PRSDP^*_{IPD}$	-	-	-
		$ipd \equiv PRSDP^*_{IPS}$	-	-	-
		$transid \equiv PRSDP^*_{TRANSID}$	-	-	-
		$rch = NULL$	-	-	-
		$flag \equiv TRUE$	-	$c1 \geq ipd.RTT + \Delta_a$	-
PRSDP*	$\langle x3',x9'\rangle,\langle x3'',x9''\rangle$	$ips \equiv PRSDP^*_{IPD}$	-	-	-
	$\langle x8',x9'\rangle,\langle x8'',x9''\rangle$	$ipd \equiv PRSDP^*_{IPS}$	-	-	-
		$transid \equiv PRSDP^*_{TRANSID}$	-	-	-
		$rch \equiv nip'$	-	-	-
		$flag \equiv FALSE$	$TEST\_FLAG \leftarrow 1$	$c1 \geq ipd.RTT + \Delta_a$	-
URDES	$\langle x3,x1\rangle,\langle x8,x1\rangle$	$ips \equiv URDES_{IPD}$	-	-	-
		$transid \equiv URDES_{TRANSID}$	$TEST\_FLAG \leftarrow 1$	-	-
attack'	$\langle x1,x1'\rangle$	-	-	-	-

are both not NULL. Table 5.2 shows $initial(\tau 0) = --$, i.e., there are no initial states and $final(\tau 0) = x1$. $\sigma = TRUE$ means that transition $\tau 0$ is always enabled and $x1$ is automatically reached at the start of the model. $check(V) = --$ implies that no condition over the model variables are checked and the condition is always satisfiable for the transition. Value 1 is assigned to variable TEST_FLAG as implied by $Assign(V) = \{TEST\_FLAG \leftarrow 1\}$, which in turn means that the detection of rank attacker can be started.

- $\tau 1 : (x1 \rightarrow x2)$ - $DIOINMP$: Since we model the rank attack scenario, the focus remains on DIO updates across the DODAG. So when the model is started and the current state is at $x1$, inconsistent DIO reports are looked into and is modeled using the transition $\tau 1$. Here, $initial(\tau 1) = x1$ and $final(\tau 1) = x2$. $\sigma = DIOINMP$ implies that transition $\tau 1$ is enabled when RQST_RSP_HANDLER() generates

event $DIOINMP$ (i.e., after an inconsistent rank update is reported from an agent). $check(V) = \{ips_j = DIOINMP_{IPS}, ipd = DIOINMP_{IPD}\}$ and $Assign(V) = --$. The parameters that validate a DIO packet intimation from an agent are source and destination IP. It is checked if the parameters equal the value stored in the model variables, ips_j and ipd, both of which are initialized to hold the IP address of agent node n_j and n_R, respectively, at the model start.

- $\tau 2$: $(x2 \rightarrow x3)$ - $PRQDP$: At state $x2$, the transition $\tau 2$ implies that a probe request ICMPv6 packet is sent. $\sigma = PRQDP$ implies that $\tau 2$ is enabled when the RQST_RSP_HANDLER() generates the event $PRQDP$ (i.e., after a RQST packet is sent). $check(V) = --$ meaning that no condition need to be satisfied and $Assign(V)$ $= \{ips \leftarrow PRQDP_{IPS}$, $ipd \leftarrow PRQDP_{IPD}$, $transid \leftarrow PRQDP_{TRANSID}$, TEST_ FLAG $\leftarrow 0$, $lastSend \leftarrow lastSend+1\}$. The parameters that uniquely identify a probe RQST packet are source IP, destination IP and a transaction identifier. Consequently, all the parameters that correspond to the RQST packet that is sent are stored in the model variables, ips, ipd and $transid$. TEST_FLAG is set to 0 such that no new probe packets are to be sent until a decision on normal or rank attacker can be ascertained. The model variable $lastSend$ is incremented, keeping a note of the number of probe packets that are sent. The destination IP of the probe request packet, i.e., $PRQDP_{IPD}$, is the first IP address that is looked up in the table $TPATH^j$. The clock variable $c1$ is RESET to make note of the transmission time of the sent RQST packet.

- $\tau 3$: $(x3 \leftarrow x2)$ - $PRSDP$: At state $x3$, the transition $\tau 3$ implies that a probe RSP packet has arrived from a node for some sent RQST packet. Here, $initial(\tau 3) = x3$ and $final(\tau 3) = x2$. $\sigma = PRSDP$ corresponds to enabling transition $\tau 3$ after the RQST_RSP_HANDLER() generates the event $PRSDP$ implying that a probe RSP packet has arrived and the condition on the model variables in $check(V)$ are satisfied. $check(V) = \{ips = PRSDP_{IPD}, ipd = PRSDP_{IPS}, transid = PRSDP_{TRANSID}\}$. The conditions over the model variables, ips, ipd and $transid$, ensure that the RSP packet is a response to the probe request packet sent in $\tau 2$. $Assign(V) = \{TEST\_FLAG \leftarrow 1, rch \leftarrow PRSDP_{IPS}\}$. TEST_FLAG is set to 1 meaning that rank attacker detection can be started. The model variable rch holds the IP address

of the latest node that responses to the probe packet before the delay timeout period is over, which again is ensured if the condition over $c1$, $\Phi(c1) = \{c1 < ipd.RTT + \Delta_a\}$, is satisfied.

- $\tau 4$: $(x3 \rightarrow x4)$ - PR_TO : At state $x3$, the transition $\tau 4$ corresponds to probe timeout period being reached while waiting for a probe RSP packet for a probe RQST packet sent. $\sigma = PR\_TO$ implies that the transition $\tau 4$ is enabled when the RQST_RSP_HANDLER() generates the event PR_TO. $check(V) = \{lastSend < M^j\}$, $Assign(V) = \{TEST\_FLAG \leftarrow 1\}$ and $\Phi(c1) = \{c1 \geq \Delta_{max}\}$. The condition over the model variable *lastSend* ensures that the number of probes sent is lesser than the size of $TPATH^j$. TEST_FLAG is set to 1 meaning that rank attacker detection can be started. The condition over $c1$ ensures that it exceeds the probe timeout period.

- $\tau 11$ $(x8 \rightarrow x1)$ - $URDES$: At state $x8$, the transition $\tau 11$ implies that a destination unreachable message is received in response to a probe packet sent from n_R to the last reachable node along the current DAO advertised downward route. It rules out the presence of any loop created. $\sigma = URDES$ implies that the transition is enabled when the RQST_RSP_HANDLER() generates the event $URDES$. $check(V) = \{ips = URDES_{IPD}, transid = URDES_{TRANSID}\}$. The condition check on the model variables *ips* and *transid* are used to ensure that the destination unreachability packet is a reply to the probe request packet sent in $\tau 10$. $Assign(V)$ makes $TEST\_FLAG = 1$ which means that the attack detection phase can restart, i.e., RQST_RSP_HANDLER() can again receive inconsistent DIO version or rank updates from agents.

I-DES behavior under attack circumstances

The I-DES model under rank or version attack condition launched by attacker A_1 is shown using the states in $X_{A_1} = \{x1', x2', \ldots, x9'\}$ and transitions, $\{\tau 1', \tau 2', \ldots, \tau 14'\}$. Similarly for attacker type A_2, states and transitions are represented using double prime notation, $X_{A_2} = \{x1'', x2'', \ldots, x9''\}$ and transitions, $\{\tau 1'', \tau 2'', \ldots, \tau 14''\}$ as shown in Figure 3.7. The DES model behavior under different attackers are mostly identical except a few transitions that differentiate them which are discussed.

- At state $x1$, the system reaches an attacker type state $x1'$ or $x1''$ following an unmeasurable attack transition (Definition 1, Appendix A)$\tau 0'$ or $\tau 0''$, respectively.

- $\tau 11'$ ($x8' \rightarrow x9'$) - $PRSDP^*$: At state $x8'$, the transition $\tau 11'$ corresponds to probe RSP packet that is received beyond the maximum 1-hop delay, i.e., $ipd.RTT + \Delta_a$ for a sent probe request packet. $\sigma = PRSDP^*$ implies that the transition is enabled when the RQST_RSP_HANDLER() generates the event $PRSDP^*$. $check(V) = \{ips = PRSDP^*_{IPD}, ipd = PRSDP^*_{IPS}, transid = PRSDP^*_{TRANSID}, flag = FALSE, rch = nip'\}$. The conditions over the model variables, ips, ipd and $transid$, ensure that the RSP packet is a response to the probe request packet sent in $\tau 10'$. The condition over variable $flag$ ensures that it is set to FALSE. The model variable rch holds the IP address of the last node that replied to the probe packet before the delay timeout period was over. $\tau 11'$ ensures that rch holds the IP address of attacker node A_1. A probe response beyond the delay period for probe packet meant for a node with IP address stored in rch via the currently advertised DAO route R' is a rank attack. $Assign(V) = \{TEST\_FLAG \leftarrow 1\}$. TEST_FLAG is set to 1 meaning that rank attacker detection can be started. $\Phi(c1) = \{c1 \geq ipd.RTT + \Delta_a\}$ means that $c1$ exceeds the delay timeout period.

- $\tau 13'$ ($x1' \rightarrow x9'$) - $DIOvINMP$: At state $x1'$, the transition $\tau 13'$ corresponds to the receipt of DIO version inconsistent intimation from an agent leaf node. $\sigma = DIOvINMP$ implies that the transition is enabled when the RQST_RSP_HANDLER() generates the event $DIOvINMP$. $check(V) = \{ver < DIOvINMP_{VERNUM}, ips_j = DIOvINMP_{IPS}, ipd = DIOvINMP_{IPD}\}$ and $Assign(V) = --$. The parameters that validate a DIO packet intimation from an agent are source and destination IP. It is checked if the parameters equal the value stored in the model variables, ips_j and ipd, both of which are initialized to hold the IP address of agent node n_j and n_R, respectively, at the model start. The model variable ver stores the latest version number advertised. The condition over ver ensures that it is lesser than the DIO version number reported by the source agent node. It may be noted that a DIO broadcast in the DODAG with a version number higher than already advertised by the DODAG root is a version number attack.

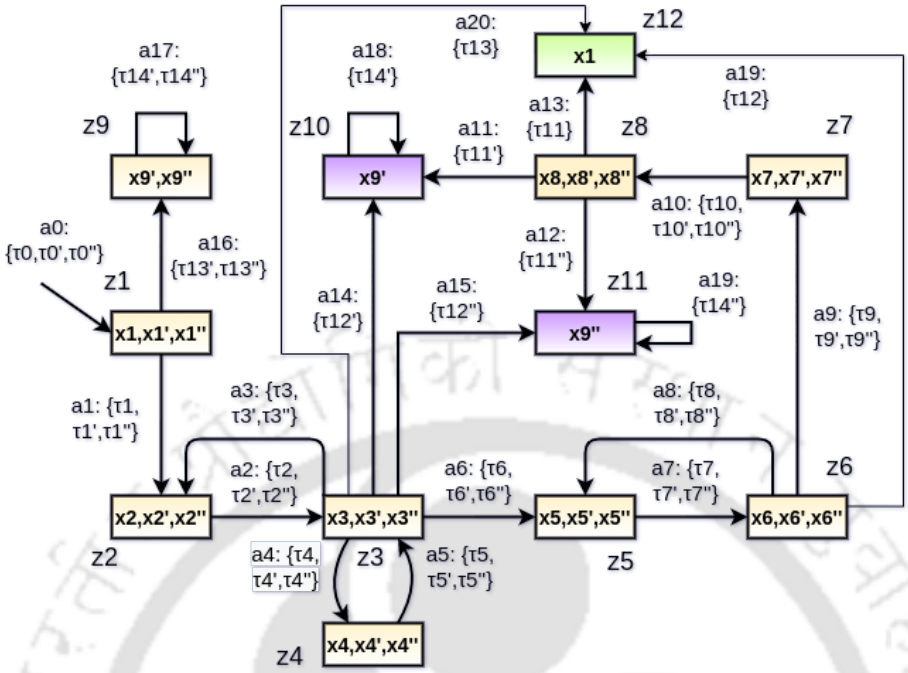


Figure 3.8: Diagnoser O for DES model H

I-Diagnoser

A key property relating to fault diagnosis in I-DES, I-diagnosability [44, 45], is discussed here. I-DES Diagnosability is a property related to event diagnosis where the earlier occurrence of certain events (faults) of interest are diagnosed depending on the occurrence of indicator transitions (Definition 13, Appendix C). An I-diagnoser, constructed from I-DES models, tracks the system behavior and gives a decision on the diagnosis of monitored events. Details of the diagnoser construction procedure and definitions pertaining to diagnosability are highlighted in Appendix B. Now, a fault is I-diagnosable in finite time, if the I-diagnosability condition is met (F_i-I-Diagnosability property is satisfied). Since fault occurrence and attack events exhibit identical deviations from the normal behaviour and in both scenarios they are unmeasurable, taking place without the knowledge of the system administrator, hence fault diagnosis has been successfully applicable in attack detection and attack type identification too. A lemma on the I-diagnosability property states that lack of attack A_i-indeterminate cycles (Definition 14, Appendix C) having an embedded indicator transition guarantees I-diagnosability. It means that the diagnoser is able to give a decision in finite time on the occurrence of the event diagnosed. Satisfaction of the I-diagnosability property, considering

the limitations in measurement, ensures efficient attack detection as well as diagnosis of the attacker type [150].

The I-DES diagnoser is basically an observer automaton. Given a measurable trace executed on the model, the diagnoser gives an estimate of membership of the current system state in the model among normal or any attacker type state from H. An alert is generated when it can be ascertained that the current state belongs to an attacker type. It is also notified in case it belongs to a set of attacker types.

Figure 4.7 shows the constructed diagnoser for our I-DES model H, considered in Figure 3.7. The working mechanism of our diagnoser is summarised here by showing one or more executions of sequences of measured events (transitions) as follows:

1. The initial state of the model H, $x1$, and states $x1'$ and $x1''$ reachable via unmeasurable attack transitions, $\tau 0'$ and $\tau 0''$, form the initial state of the diagnoser, $z1$.

2. Let $\Im_{z1_1} = \{\tau 1, \tau 1', \tau 1''\}$, i.e., the outgoing transitions from model states $\{x1, x1', x1''\} \in z1$. All the transitions in $\Im_{z1_1}$ are equivalent (Definition 2, Appendix A) and hence cannot be further subdivided and hence justifies O-transition $a1$. The O-state corresponding to the transition $a1$ is $z2 = \{x2, x2', x2''\}$.

3. Let $\Im_{z1_2} = \{\tau 13', \tau 13''\}$, i.e., the outgoing transitions from model states $\{x1', x1''\} \in z1$. All the transitions in $\Im_{z1_2}$ are equivalent and hence cannot be partitioned further and hence justifies O-transition $a16$. The O-state corresponding to the transition $a16$ is $z9 = \{x9', x9''\}$. Since, $z9$ consists exclusively of attacker type states only, it is an attack-certain O-state.

4. Let $\Im_{z2} = \{\tau 2, \tau 2', \tau 2''\}$, i.e., the outgoing transitions from model states $\{x2, x2', x2''\} \in z2$. All of outgoing transitions in $\Im_{z2}$ are measurement equivalent belonging to one measurement equivalence class of transitions, hence cannot be further partitioned. Therefore, it justifies O-transition $a2$. The O-state corresponding to the transition $a2$ is $z3 = \{x3, x3', x3''\}$. In a similar manner, the diagnoser states $\{z4, z5, z6, z7\}$ can be constructed using the corresponding O-transitions $\{a4, a6, a7, a9\}$. The principle can be safely extended.

5. From the definition, we can compute the attacker$_i$-certain O-states and the Normal-certain O-states. In our example, when $i = 1$ the attacker$_1$-certain O-state may be

computed as $z10 = \{x9'\}$ since it exclusively consists of states only belonging to attacker 1. Similarly, attacker$_2$-certain O-state may be computed as $z11 = \{x9''\}$ and the normal-certain O-state can be computed as $z12 = \{x1\}$.

3.3.8 An example of rank attacker node identification using DES Diagnoser

Suppose the following events occur in the DODAG chronologically due to packets received or sent from the DODAG root: $DIOINMP$, $PRQDP$, $PRSDP$, $PRQDP$, $PRSDP$, $PRQDP$, $PRSDP^*$.

The diagnoser starts from the O-state $z1$ and on occurrence of the $DIOINMP$ event, the diagnoser moves to O-state $z2$ via O-transition $a1$. The transition $a1$ might have been taken by the diagnoser due to the occurrence of any of the H-transitions (Definition 4, Appendix A), $\tau 1$, $\tau 1'$ or $\tau 1''$. Since the transitions $\tau 1$, $\tau 1'$ and $\tau 1''$ are measurement equivalent, it cannot be certainly said at this point if an attack has occurred. A probe request data packet is sent due to which the event $PRQDP$ occurs and the diagnoser moves to O-state $z3$ via O-transition $a2$. Now, the response to the probe is received and the event $PRSDP$ passed to diagnoser and O-state $z2$ is reached via $a3$. The O-states are then revisited due to the events $PRQDP$, $PRSDP$ and $PRQDP$ and the diagnoser reaches the O-state $z3$. Eventually, when the $PRSDP^*$ event occurs, suppose the diagnoser moves from O-state $z3$ to O-state $z10 = \{x9'\}$ via O-transition $a14$ due to the model transition $\tau 12'$. Since the O-state $z10$ reached by the diagnoser is an Attacker$_1$-certain O-state, it is ascertained that the system is under attack condition due to attacker node 1. Moreover, since there are no A_i-indeterminate cycles [46, 45], along all paths of the DES diagnoser, an unique malicious node i, when present, can be identified correctly. On each such occasion when the diagnoser reaches an Attacker$_i$-certain state due to an event trace, an alert is generated.

3.3.9 Correctness and Completeness

DES modeling aids in formalizing a system to check correctness and completeness [44]. We demonstrate correctness and completeness of our proposed IDS here, by taking into consideration all possible cases of rank attack. For each case considered, we show that

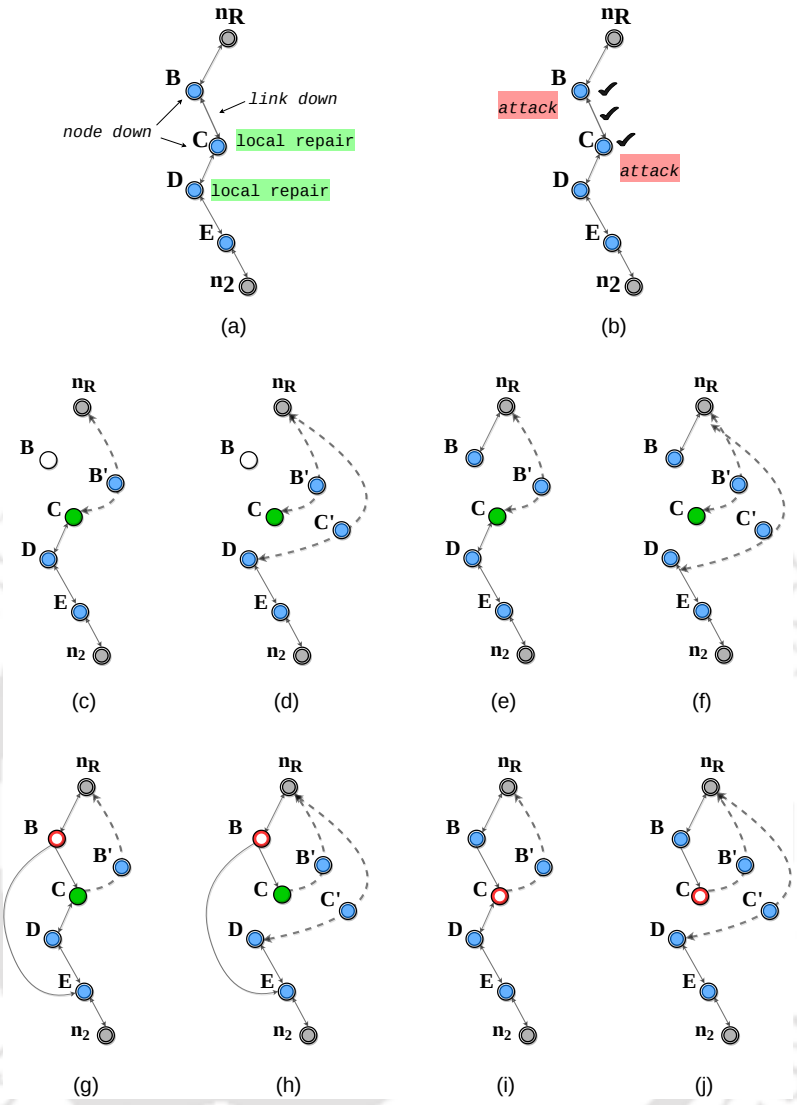


Figure 3.9: Normal and attack configurations

attacker node is correctly identified. We use the DODAG instance shown in Figure 4.3 for our proof, where n_R is the 6BR root and the set of agents $\mathcal{T} = \{n_1, n_2, n_3, n_4\}$. B and C are the two suspected rank attack nodes and can be related to nodes A_1 and A_2 used in our I-DES model. Since there are no A_i-indeterminate cycles in the diagnoser O, therefore the I-diagnosability condition is satisfied. This means that location of an attacker A_i in the DODAG, having launched a rank or version attack, is always diagnosable. We show using analysis that B or C is correctly identified as attack node when the corresponding attacker-certain state is reached in the diagnoser.

We now prove the completeness by justifying why all attack cases can be detected from

the traces in H. An irregular increased rank advertisement can be classified as a normal network condition if a local repair operation is undertaken, otherwise can be classified as an attack. As shown in Figure 3.9(a), we assume that nodes C and D undertake local repair operations due to the parent node being down, or link with the parent goes off or as part of loop avoidance. On the other hand, as shown in Figure 3.9(b), an attack might have been launched by node B or C. Though, the effects of attack mimics the normal scenario, however, there lies unique inconsistencies in the resulting topologies which can be made out from the probe response characteristics of nodes. We discuss the normal cases here first.

Case I: Node C undertakes local repair due to parent node B being down.

As shown in Figure 3.9(c) and 3.9(d), node C chooses alternate parent node B' for upward routing. Depending on the newly advertised rank, a successor node may conform to the update by not changing its preferred parent or may choose a better route instead. It may be noted that since B is down, any upward or downward path between the pairs (n_R, B) and (B, C) cease to exist. Our proposed procedure utilises the above facts. Firstly, n_2 reports the DIO update to n_R. On receipt of such intimation, the diagnoser moves from state $z1$ to $z2$. Now, a probe RQST packet $PRQDP$ is sent to node B via stored downward route $TPATH^2$ while the diagnoser reaches state $z3$. Since no response packets are received, event PR_TO is generated and the diagnoser consequently reaches state $z4$. Next, a probe request packet $PRQDP$ is sent to C via $TPATH^2$ with the diagnoser reaching state $z3$. Again, no RSP packet is received before Δ_{max} since the request packet itself is not delivered via B. This behavior is repeated for the subsequent probe request packets sent to D and E with the diagnoser reaching state $z4$. Now n_2 is sent the probe request packet and Δ_{max} is again exceeded while waiting for a response. The diagnoser reaches state $z5$ this time, since all the nodes in $TPATH^2$ are probed. Now, a probe packet is sent to the first unreachable node via a currently advertised downward path. Since B is down, no routing information is updated for node B. Since C had chosen a path via B', a downward path from the root exists. On a request packet $PRQDP$ being sent to C via B', the diagnoser reaches $z6$. As route through B' is longer, so delay is incurred while receiving the response. As a result, the delay timeout is exceeded. Consequently the diagnoser moves to state $z12$, since no node was reachable without delay prior to C which is a normal-certain O-state. So, a normal condition of local repair in the DODAG is correctly identified.

Case II: Node C undertakes local repair due to link (B, C) going down.

As in the situation discussed in Case I, the sequences of events are similar, except the fact that response from node B arrives before $RTT(B) + \Delta_a$. So, when the diagnoser moves to state $z2$, the model variable rch is set. Therefore, at state $z6$, when a delay timeout occurs, the diagnoser reaches state $z7$ instead of $z1$. A probe request packet is then sent to node B via node C along the current DAO advertised route. The diagnoser accordingly moves to state $z8$. A destination unreachable message is then received by n_R, and the diagnoser moves to normal-certain O-state $z12$ and it is ascertained that situation is normal, since a local repair operation was initiated as shown in Figure 3.9(e) and 3.9(f).

An attack launched by an attacker can be of the two following types: (i) The attacker illegitimately chooses a parent node that has higher rank, but does not lie in $TPATH^2$ (ii) the attacker illegitimately chooses a parent node that has higher rank, and is a successor node in $TPATH^2$. Type (i) is discussed as case III and type (ii) is discussed as Case IV.

Case III: Node C undertakes local repair due to loop detection while forwarding to B.

While forwarding packet upwards, suppose C detects a loop and initiates a local repair while forwarding through alternate parent node B'. Now, node B might be a direct attacker that chooses a successor node as its parent, fueling a loop creation. In that case, B must be a node in the subtree at C. As in the situation discussed in the normal scenario, B and C are probed. B responds before delay timeout occurs while C is unreachable. All the nodes successor to C are also unreachable. Consequently, the diagnoser node reaches state $z5$ after a probe timeout occurs while a probe packet is sent to the last node n_2. Now, a delay timeout occurs when a probe packet is sent to C via the current downward route. The diagnoser reaches $z8$ following the event $PRQDP$. The only difference arises when node B is sent a RQST packet via B, and a delayed response is received. The event $PRSDP^*$ is generated and the diagnoser reaches state $z10$ depending on the value of the variable rch, which is the IP address of B, the last node that replies without delay. It is therefore ascertained that B is an attack node here since it lies in the subtree of node C. As shown in Figure 3.9(g) and 3.9(h), the red line indicates that the attacker has chosen E as its parent. If a $URDES$ packet is received, the diagnoser again moves to normal-certain O-state $z12$, which is the case shown using the green line indicating the choice of B.

Case IV: Node C is an attack node that does not advertise DAO

In this case, C chooses a different parent in spite of an existing better parent for upward route. This situation is shown using the Figures 3.9(i) and 3.9(j). While probing nodes in $TPATH^2$, nodes B and C, both reply to the probe packets and the diagnoser reaches state $z3$ when a $PRQDP$ packet is sent to D. Now, if a delay timeout occurs while awaiting the response, the diagnoser reaches state $z11$ depending on the value of the variable rch which holds the IP address of C. Consequently, it can be ascertained that the attacker node is C and the diagnoser correctly detects the attack since $z11$ is an A_2-certain node.

So, all the possible cases of attack by specific attacker nodes are analyzed. The I-diagnoser correctly reports the network condition by identifying the corresponding attacker type states, for each case.

3.3.10 Overhead analysis

The extra communication overhead is added in our detection scheme due to probe requests and generated responses. The overhead is minimum when only 2 probe requests are sufficient to identify the malicious node. Such a scenario occurs if a probe request packet is sent to a node which responses in time and another probe packet sent subsequently to the child of this node is acknowledged beyond the admissible delay. We now discuss the scenario when maximum overhead is incurred in our solution. Suppose probe request packets are sent sequentially to nodes in $TPATH^j$. Now, the node with the lowest rank responses to the probe request in time. For, the subsequent probe requests sent, responses are not generated. Based on the DAO messages received after the IDS is setup, nodes with missing acknowledgements are sent probe requests through alternate routes. Only the node farthest from the root responses after an admissible delay. Hence, assuming that the height of the tree is equal to the number of nodes in the RPL, n, then a total of $(1+2(n-2)+1) \approx O(n)$ probe requests will be required here (1 for node with lowest rank, $2(n-2)$ for subsequent $(n-2)$ nodes that are probed twice and 1 for confirmation). Considering a balanced tree of n nodes, depth $= \log_k n$, for a branching factor k. In such cases, the number of probes that will be required in the worst case is $2\log n \approx O(\log n)$.

3.4 Experiments, results, and discussion

Three experiments are executed in Contiki Cooja [155] and one in a real testbed at FIT IoT-LAB [156]. Cooja is a network simulator explicitly developed to cater for IoT networks while the simulator builds on C base libraries of sensors and RFID chips, the FIT IoT-LAB is an open testbed and comprises of 117 mobile robots and 2728 low-power sensor nodes that are made available for conducting experiments in the heterogeneous environment (e.g., standardized protocol, OS, topologies, and hardware). Having unique hardware and node capabilities, interconnected locations are installed across France in FIT IoT-LAB and made available for experiments via a web portal. We used three different types of topology, as shown in Figure 4.9. In topology 1, the IoT nodes are distributed very densely, while a sparse distribution is used in topology 2. In topology 3, nodes are distributed in a mixed fashion. Furthermore, the hop count is more in topology 2 as compared to topology 1. We consider a OF0 implementation with hop count (HC) metric. The simulation or experimental parameters of Contiki Cooja and FIT IoT-LAB are presented in Table 3.5. To examine the performance of our proposed solution, three scenarios are designed as part of the experimental setup, namely, the non-rank attack scenario, increased rank attack scenario, and the increased rank attack scenario with the proposed solution, comprehensive analysis of which are demonstrated below.

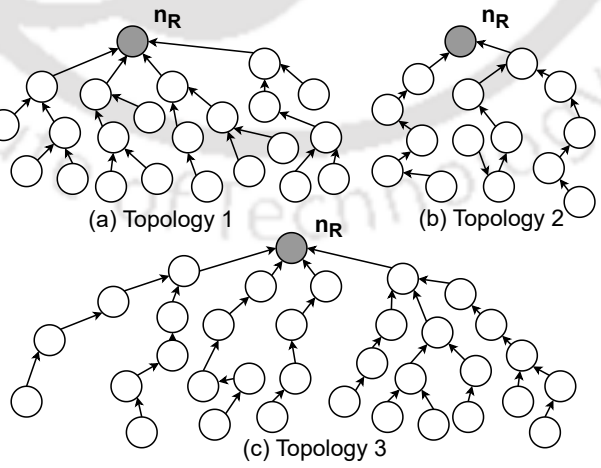


Figure 3.10: Topology considered for testbed and simulation experiments

Table 3.5: Contiki Cooja and FIT IoT-LAB experimental parameters

Parameter name	Value
Operating system	Contiki 3.0, Contiki 4.5
Simulator	Cooja
Testbed	FIT IoT-LAB, Grenoble
Network size	8, 16, 32, 64 nodes
Radio Environment	UDGM
Node Type	Tmote Sky , IoT-Lab A8
Routing Protocol	RPL
MAC/adaptation layer	ContikiMAC/6LoWPAN
Transmitter output power	(dBm) 0 to -25
Receiver sensitivity	(dBm) -94
Radio frequency	2.4 GHz
Attack Modeled	Rank and version number attack
Simulation Duration	Variable

3.4.1 Experiment 1: Non-rank attack scenario

All the external and internal nodes demand the IoT services (i.e., temperature and humidity) using the *Sky-Websense* server. The experiment has been executed on 8, 16, 32, and 64 nodes. The flow of IoT network packets and their behavioural changes are noted. Figure 3.11a shows an RPL DODAG with 16 nodes. The node having Node ID 65 is the 6BR root running our IDS. Nodes with IDs 16, 13, 30, 52 and 62 are the 5 agents deployed as leaves and behave like regular nodes. *Wireshark* and *power trace* tool are used during simulations for network traffic analysis. In the testbed setup, we have used A8-type nodes utilizing various topologies with Grenoble areas. A8 is a TI SITARA AM3505 (Arm Cortex A8) combined with STM32 microcontroller and a radio interface. It is one of the powerful IoT-LAB node which allows running RIOT, Contiki, and FreeRTOS. The adopted parameters during the testbed experiments are specified in Table 3.5. Figure 3.11a shows the DODAG topology in a non-attack scenario. Throughput, energy usage of the network, and the average power consumption on a per-node basis with their respective run times are shown in Figures 3.12 (a) and (b), analysed using 64 nodes in Contiki Cooja and FIT IoT-LAB, respectively. Our analysis shows average throughput within 86.45% to 94.89%, average network energy usage ranging from 27854 mJ to 33648 mJ, and average power consumption lying within 1.2 mW to 1.46 mW in this scenario. The values are moderately good because during the non-rank attack scenarios, RPL control messages, Objective Function (OF), and Rank computation

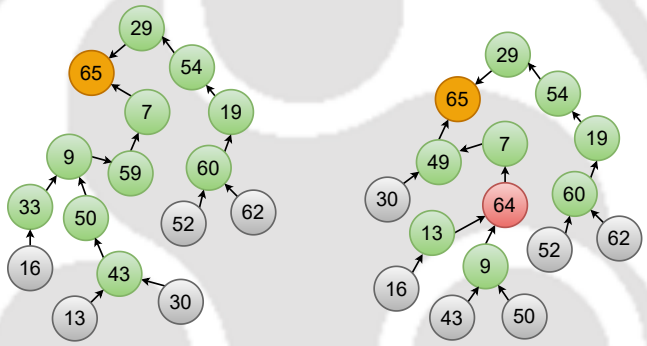
module are executed correctly with the required number of RPL control messages.

Table 3.6: Energy, Node Power, Throughput, and Packet Delivery Ratio for IoT ecosystem (During attack and after solution implementation in Contiki Cooja)

IoT Scenario	Energy (mJ)				Node Power (mW)				Throughput (Kbps)				Packet Delivery Ratio (%)			
During	8N	16N	32N	64N	8N	16N	32N	64N	8N	16N	32N	64N	8N	16N	32N	64N
attack	86615	10216	14425	17098	0.490	60.52	1.351	1.692	0.573	0.596	0.574	0.556	89.17	88.63	86.69	84.61
After solution	8N	16N	32N	64N	8N	16N	32N	64N	8N	16N	32N	64N	8N	16N	32N	64N
implementation	8261.4	8898.6	12129.5	16229.5	0.31	0.49	0.92	1.36	0.662	0.661	0.657	0.654	98.76	98.65	98.42	98.34

Table 3.7: Energy, Node Power, Throughput, and Packet Delivery Ratio for IoT ecosystem (During attack and after the solution implementation in FIT IoT-Lab)

IoT Scenario	Energy (mJ)				Node Power (mW)				Throughput (Kbps)				Packet Delivery Ratio (%)			
During	8N	16N	32N	64N	8N	16N	32N	64N	8N	16N	32N	64N	8N	16N	32N	64N
attack	9527.5	12259.2	17454.3	20176.6	0.59	0.74	1.54	1.81	0.463	0.504	0.487	0.478	80.31	78.11	73.12	73.92
After solution	8N	16N	32N	64N	8N	16N	32N	64N	8N	16N	32N	64N	8N	16N	32N	64N
implementation	7269.7	7919.2	10552.2	13307.8	0.27	0.49	0.83	1.19	0.559	0.543	0.572	0.552	91.58	90.88	88.86	87.12

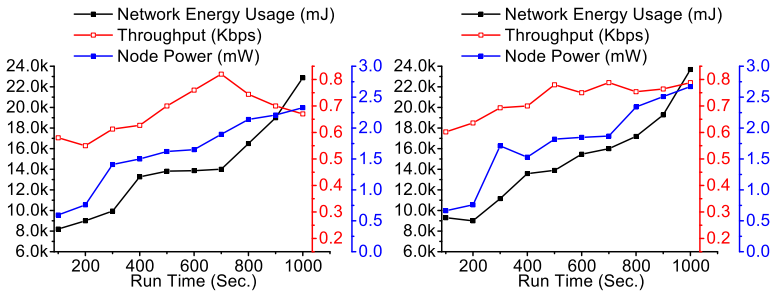


(a) Without Malicious node (b) With Malicious node

Figure 3.11: DODAG of the IoT ecosystem

3.4.2 Experiment 2: Increased rank and version number attack scenario

An increased rank attack is performed with 8, 16, 32, and 64 IoT nodes. The attack nodes, incorporated during our experiments, generate malicious RPL control messages and create falsified non-optimal routes. The IoT network behavioural changes are examined with different malicious nodes while varying node density. Figure 3.11b shows IDS node at root with ID 65 and the node ID 64 is the malicious node. Among the remaining nodes, nodes with ID 30, 16, 43, 50, 52 and 62 are the agents deployed that perform sensing at the leaf levels. Traffic generated from the attack is analysed using `collect view modules` for analysis purposes in simulation. Consequently, we use `Sysstat` [157] and `iperf` tool [158] for real testbed analysis. We additionally perceive the average power consumption per node,



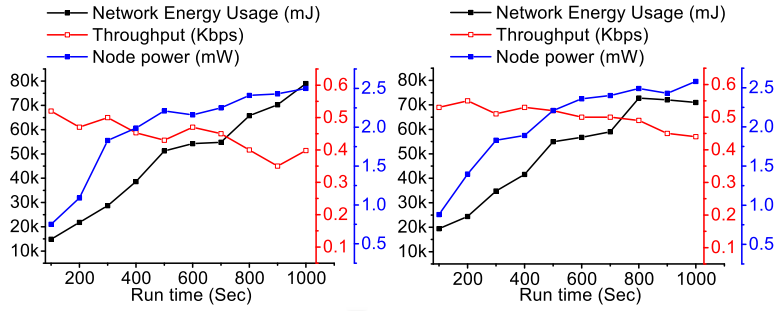
(a) Contiki Cooja (b) FIT-IoT-LAB

Figure 3.12: Average Energy, Throughput, Node Power over run time (nodes=64) (without malicious node)

and the energy usage of the complete RPL DODAG. Figure 3.13 (a) exhibits a considerable increase in the complete network's average energy usage and power consumption per node, i.e., 28.8% to 35.7% and 31.7% to 43.3%, respectively, in Contiki Cooja simulations. Figure 3.13 (b) shows similar outcomes in FIT IoT-LAB, i.e., 38.7% to 43.9% average energy usage and 36.5% to 52.4% power consumption per node. In both, the throughput graph can be seen to be going down significantly. The average throughput value is reduced and ranges from 37.3% to 43.5% in the attack scenario, both in simulation and real testbed. All experiments show huge network energy and node power consumption with reduced throughput because of a massive number of RPL control messages, malicious OF for routing, and unknown loop formations due to attack. During attack, the performance metrics that significantly affect RPL performance are listed in Tables 3.6 and 3.7 for Contiki Cooja and FIT IoT-Lab, respectively. The findings also demonstrate that a rise in the number of IoT nodes results in a significant increase in the amount of malicious RPL control messages, which consumes additional network energy due to node power and consumption. In addition, network performance and packet delivery ratio is shown to suffer and produce inferior outcomes.

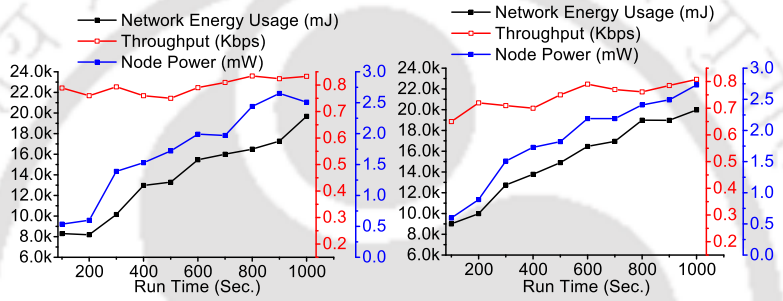
3.4.3 Experiment 3: Attack scenario with proposed solution

Experiment 2 is executed with the proposed solution, both in simulation and real testbed. The performance of our proposed solution is illustrated in Figure 3.14. Both during simulation and in real testbed, we have considered 8, 16, 32, and 64 IoT nodes, while the experiments are run for 1000 sec. We consider the values Δ_{max} and Δ_a to be 13 seconds



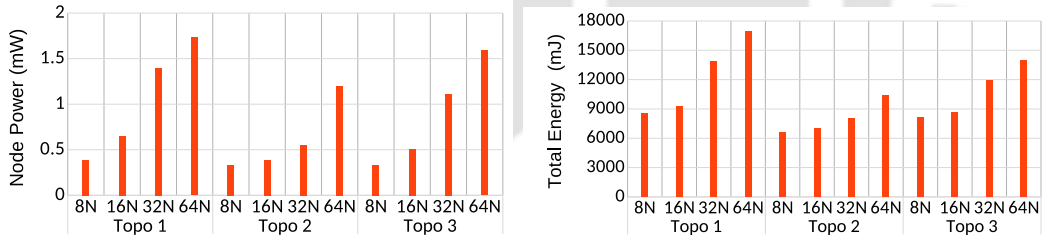
(a) Contiki Cooja (b) FIT IoT-Lab

Figure 3.13: Average Energy, Throughput, Node Power over run time (nodes=64) (with malicious node)



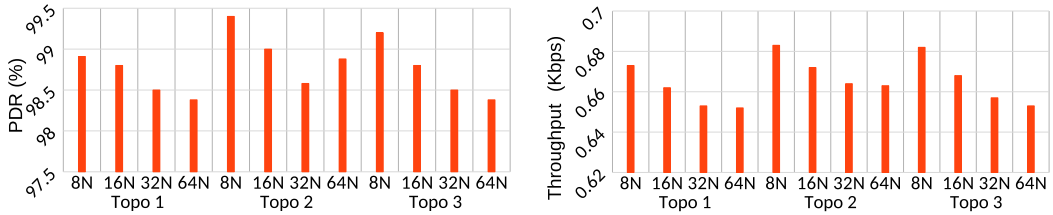
(a) Contiki Cooja (b) FIT-IoT-Lab

Figure 3.14: Average Energy, Throughput, Node Power over run time (nodes=64) (after solution implementation)



(a) Node Power comparison across topologies (b) Total energy comparison across topologies

Figure 3.15: Power and Energy for 50 *min* network execution with proposed solution



(a) PDR comparison across topologies (b) Throughput comparison across topologies

Figure 3.16: PDR and Throughput for 50 *min* network execution with proposed solution

and 3.8 seconds, respectively (discussed in Section 4.3.2). The trickle timer is of 10 seconds duration. Each experiment is conducted by varying the number of nodes, i.e., from 8 to 64 nodes and hop counts. The performance analysis of all the experiments is based on various metrics like True Positive Rate (TPR) (also known as *sensitivity*), True Negative Rate (TNR) (also known as *specificity*), Accuracy (ACC), Energy usage (EU), Throughput, Packet delivery ratio (PDR), and scalability. The performance analysis metrics are defined as follows:

- *True Positive Rate (TPR)* is the ratio of accurately identified attacker nodes to all of the attacker nodes and is estimated by:

$$TPR = \frac{p}{p+q} \tag{3.1}$$

- *True Negative Rate (TNR)* is the ratio of wrongly identified genuine nodes to all of the genuine nodes and is estimated by

$$TNR = \frac{r}{r+s} \tag{3.2}$$

  where, *p=Attacker nodes identified accurately q=Attacker nodes not identified correctly r=Genuine nodes identified accurately s=Genuine nodes not identified correctly.*

- *Accuracy (ACC)*: It calculates the overall rates of attacker nodes identification and false alarms. This result signifies the success rate of the proposed approach; it is estimated by

$$ACC = \frac{p+r}{p+q+r+s} \tag{3.3}$$

- *Energy Usage (EU)*: The amount of energy utilized for the proposed solution throughout its execution.

During the execution of our proposed approach, we consider three topologies, as shown in Figure 4.9. Figures 3.15(a) and 3.15(b) illustrate the node power consumption per node and network energy consumption after our solution is implemented for 50 minutes across various topologies and varying IoT nodes. The findings suggest that our proposed solution has a higher average total energy usage and node power consumption per node in topology

1 in comparison with other topologies and standard RPL with rank and version number attacks in place. When compared to the other possible topologies for this work, topology 2 has a lower average overall energy use and node power per node. The amount of energy consumed is proportional to the density of the individual nodes and DODAG configuration.

Figures 3.16(a) and 3.16(b) compare the proposed work's packet delivery ratio and throughput across three topologies with varying IoT nodes. As per the results, our proposed security approach has the lowest throughput (0.652 Kbps) and packet delivery ratio (98.4%) in topology 1 as compared to others. The performance of the suggested technique demonstrates promise in topologies 2 and 3, respectively. Topologies 2 and 3 have throughput of 0.664 Kbps, and 0.653 Kbps and packet delivery ratios of 98.55%, and 98.38%, respectively. Topology 1 has lower results than RPL with rank and version attacks due to packet loss and retransmission.

Figures 3.14(a) and 3.14(b) show the performance analysis of our proposed solution during simulation and in real testbed, respectively. A reduction in network energy usage and node power by 24.9% to 33.6% and 22.6% to 41%, respectively, can be noted. Throughput graph can be seen to significantly progressing upwards. The average throughput value was improved by 32.9% to 36.7% on the implementation of our solution in the IoT ecosystem. Tables 3.6 and 3.7 present the performance analysis during the recursive execution of our proposed solution across the various possible topologies involving the attack node. Based on the outcome, it can be noticed that different topologies take an unique amount of network energy and node power; it also varies with the number of nodes. It can be further observed that our solution requires minimum amount of network energy and node power. This is not only because we use only one centralized IDS node in our approach, but also because rank and version attacks are detected and identified accurately in lesser time.

3.4.4 Comparison with the existing works

This subsection presents the comparative analysis of the proposed rank and version number attack detection approach with state-of-the-art solutions. Experiments are fairly repeated multiple times to create tight confidence intervals. In general, we compare our real-time testbed results obtained across the different topologies to the simulation results. We observe that both executions provide reliable results (approximately 10% - 30% over/under estimated

experimental results). A comparison of our scheme is shown through Table 4.7 and graphs provided in Figures 3.17, 3.18, 3.19, 3.20 and 3.21. To measure the performance metrics, we use `collect view modules`, `Sysstat`, and `iperf tool`. Ten different performance metrics: Energy Usage (EU), Node Power, Throughput (THP), PDR, Control Message Overhead (CONMO), TPR, TNR, ACC, RAD for rank attack detection accuracy, VNAD for version number attack detection accuracy, RAI for rank attack node identification accuracy) and Scalability (SCAL) are considered. State-of-the-art methods [99, 159, 160] consume enormous energy, node power, and control message overhead. Hence they are not as suitable for a constrained IoT ecosystem. Figure 3.18 shows that our proposed approach takes 13759mJ, 12962mJ, and 14872mJ total energy with the 3 respective topologies. The state-of-the-art methods [161, 139, 97, 99, 38, 162, 160] consume more node power, energy, and have higher control message overhead, as shown in Figures 3.17, 3.18, and Table 4.7, respectively.

Basically, for comparison, we judiciously consider metrics that are maximum common with the state-of-the-art schemes. Further we consider those approaches that have maximum reported QoS metrics. We consider the derived parameters from the reported parameters, wherever required. Though DETONAR [145] achieves full accuracy in attack node identification, but achieves 80% in case of version attacks. Version attack detection accuracy using our proposed scheme fares better than DETONAR. Also, our approach is scalable while DETONAR is applicable to small networks only. The packet overhead (CONMO) in DETONAR is also significantly higher than our proposed scheme. InDReS [130] considers the QoS metrics but does not report the false positives, false negatives or accuracy of their algorithmic procedure. The results show that our proposed approach achieves comparatively better results overall with performance parameters, as shown in Figures 3.19, 3.20, and 3.21. The accuracy of our proposed approach is calculated based on TPR and TNR values shown in Figure 3.21, while a comparison of results is shown in Table 4.7.

3.4.5 Discussion

In our scheme, attack is detected and the attacker, that launches the attack, is identified at the same time. Accurate identification of node implies that attack is also detected accurately. Conversely, attack is detected implies some node is identified as an attack node.

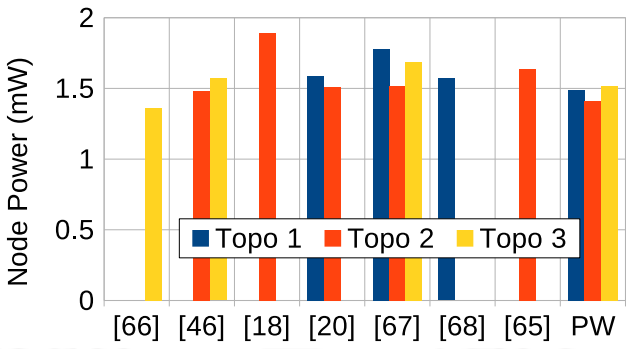


Figure 3.17: Node Power comparison with related works

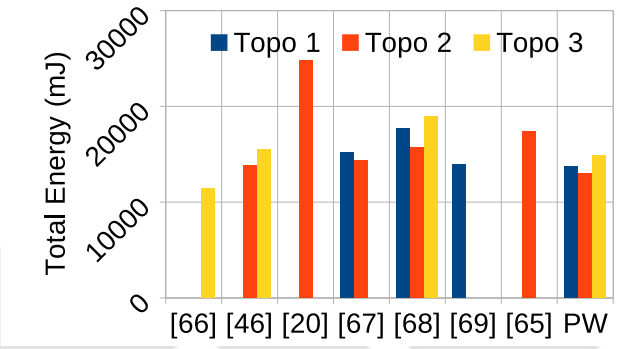


Figure 3.18: Energy comparison with related works

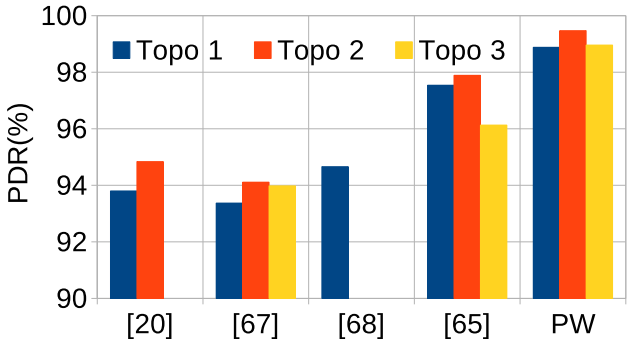


Figure 3.19: PDR comparison with related works

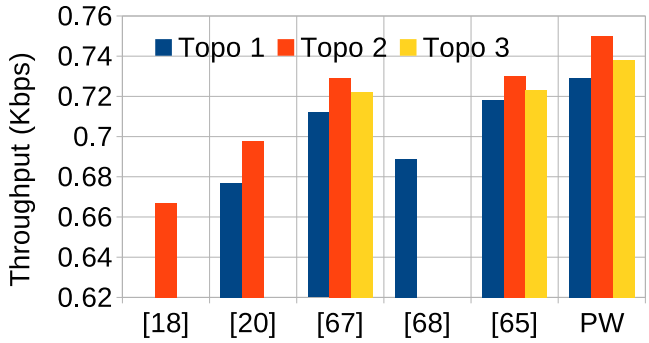


Figure 3.20: Throughput comparison with related works

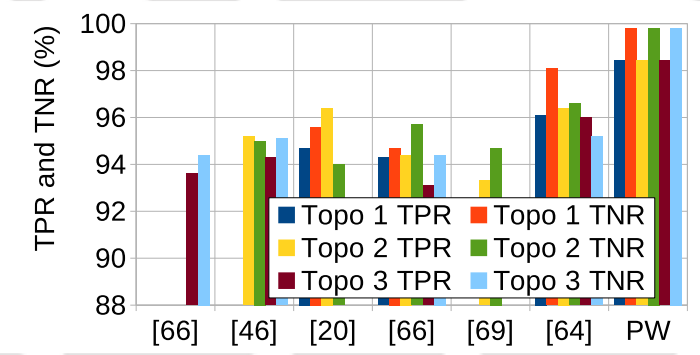


Figure 3.21: TPR and TNR comparison with related works

Table 3.8: Comparison of the proposed scheme with the closely related works

References	Ⓢ/Ⓣ EN	EU (mJ)	POWCPN (mW)	THP	PDR (%)	CONMO (in Pkt.)	TPR (%)	TNR (%)	ACC (%) RAD	ACC (%) VNAD	RAI ACC (%)	SCAL
A. Le et al. (2011) [161]	Ⓢ	11479	1.36	N/A	N/A	N/A	93.50	94.41	94.32	N/A	94.32	✗
S. Usman et al. (2018) [139]	Ⓢ	15479	1.48	N/A	N/A	N/A	94.75	94.70	95.20	N/A	95.20	✗
M. Nikravan et al. (2018) [97]	N/A	13938	1.89	0.667	N/A	N/A	N/A	N/A	90.11	90.11	N/A	✓
D. Airehrour et al. (2019) [99]	Ⓣ	22580	1.52	0.738	93.97	5045	94.70	95.62	94.89	N/A	N/A	✗
ZA. Almusaylim et al. (2020) [163]	Ⓢ	18953	1.69	0.717	93.45	1095	94.46	95.12	94.82	98.30	94.82	✓
S. Sharma et al. (2020) [162]	Ⓢ	13890	1.57	0.694	94.13	1012	N/A	N/A	N/A	N/A	N/A	✓
R. Sahay et al. (2020) [164]	Ⓢ	17385	N/A	N/A	N/A	2068	93.3	94.12	94.50	N/A	94.50	✗
S. Nayak et al. (2021) [159]	Ⓢ, Ⓣ	N/A	N/A	N/A	N/A	N/A	93.45	93.60	93.58	70.4	N/A	✗
S. Ibrahim et al. (2022) [160]	Ⓢ	18839	1.62	0.718	97.98	950	N/A	N/A	99.01	99.00	N/A	✓
A. Mayzaud et al. (2017) [144]	Ⓢ	N/A	N/A	N/A	N/A	N/A	97.28	N/A	98.53	98.53	N/A	✓
A. Zeeshan et. al (2017)[143]	Ⓢ	N/A	N/A	N/A	N/A	N/A	95.00	89.00	N/A	92.00	N/A	✓
A. Andrea et. al (2021) [145]	Ⓣ	N/A	N/A	N/A	N/A	15430	N/A	N/A	100	80.00	100	✗
M. Surendar et. al (2016) [130]	Ⓢ	12492	N/A	0.949	95.41	750	N/A	N/A	N/A	N/A	N/A	✓
Proposed solution	Ⓢ, Ⓣ	14872	1.41	0.743	**99.34**	**680**	**98.43**	**99.73**	99.1	99.1	**99.1**	✓

Ⓢ: Simulation, Ⓣ: Testbed, POWCPN: POWer Consumption Per Node, THP: Throughput, PDR: Packet delivery ratio, CONMO: CONtrol Message Overhead
SCAL: Scalability, ACC: Accuracy, RAD: Rank Attack detection, VAD: Version Attack detection, RAI: Rank attack Identification, N/A: Not available

A detection accuracy of 99.1% for our proposed solution, as shown in Table 4.7, means identification accuracy is also 99%. Our proposed design is inspired from intrusion detection using probing techniques that have been successfully applied to wired and wireless network security solutions [165, 166, 48]. The applicability of our approach in the IoT context has been shown through 6LoWPAN fragmentation [151] and CoAP request/response spoofing attack detection [146].

ICMPv6 probe request packets are sent with random payload. But, the receipt of an acknowledgement and the time of receipt of the acknowledgement only matter. Since the payload information is not of our interest, alteration of packets does not affect the detection procedure. A probe response transition is taken only if it is received from the same node to whom the probe was sent. Hence, spoofing will not help the attack motive. Probe packets may be communicated concurrently via different downward paths. To avoid self-identification, the attack node reports truly. Communication lags due to the underlying RPL-IoT network conditions will uniformly affect every node along a path in the DODAG. Response delay is an attack characteristic in our detection procedure. If a malicious node delays a packet, then it is identified more easily. If an attack node holds the packet for indefinitely long and does not forward it, then such a case is also an attack behavior. So delay or not responding does not deter the detection process. Furthermore, a DIO multicast simultaneously affects in route updation and inference of suspicious activity by multiple leaf agents. Hence, due to multiple leaf agents present, if any agent misses reporting, it does not hinder our identification mechanism. The case of malfunctioning leaf agents, if compromised, is not explicitly dealt with in this chapter.

Studies in the literature have analyzed variants of rank attacks. In Le et.al [167], the authors propose few variants. Their impact on the DODAG topology from the perspective of end-to-end delay and packet delivery ratio is highlighted. Their work shows that there exists unique threats to RPL that evade regular detection techniques. This is so because such attacks do not consider changing the advertised rank value; rather, they create un-optimized paths, silently. These types of attacks pose a different nature to the traditional threats making it complex enough to be defended, for example, the blackhole attacks that add delay to transmissions. They have specifically considered four types of rank attack variations, namely, 1) Permanently and updates about the rank change to its neighbors, 2)

Non-permanently (flipping between its choices between normal and abnormal) and updates about the rank change to its neighbors, 3) Permanently and does not update about the rank change to its neighbors and 4) Non-permanently and does not update about the rank change to its neighbors.

We show to detect version attacks apart from rank attack identification. As per knowledge, this is the first-of-a-kind attempt to mitigate RPL routing attacks using finite state automata based DES based IDS. State-of-the-art attacks that produce the same consequences as the increased rank attacks, can be also detected and malicious nodes can be uniquely located using our procedure, as it is. Attacks such as the worst parent attack, neighbour attack, etc., that result in similar consequences as covered in our rank attack procedure will also be detected in our scheme. Though we explicitly do not model these attacks, yet a class of worst parent attack with update, i.e., the worst parent choice is passed on to child nodes, falsely, is one of the attack cases that we consider. Hence, such an attack will be detected. Also, a class of neighbour attacks where the advertised parent node is out of range of the DIO recipient, and the attack results in a post-attack topology as dealt in our scheme will also be detected. Further, cross-layer attacks that use increased rank attacks are also detected using our scheme. RPL analysis on the packet exchange dynamics due to other attacks is thereby necessary. Decreased rank attacks, sinkhole attacks and blackhole attacks are also DIO specific attacks launched in a similar manner, i.e., a lower rank value is falsely advertised in DIO to attract nodes. The effects of these attacks are analogous to increased rank attack. DES based IDS can be extended to detect other attacks by adding relevant states and transitions for control or data packet communication behavior in the monitored RPL-IoT. As it is, decreased rank attack or sinkhole that are manifested towards increased rank attack can also be detected using our scheme with minimum customisation, even when combined with selective forwarding attack. This would require careful but minimum modifications to be made to our algorithm and extending our model for detection. Other forms of attacks, which directly map to an increased rank attack scenario, will also be detected using our scheme with minor changes. Moreover, one advantage of using DES based IDS is that false positives are minimal. A non-zero false positive in our experiments can be related to reasons such as, packet loss if considered as a missing acknowledgement response and network lags beyond the estimated values of Δ_{max} and Δ_a.The current solution can be

further improved with generation of optimized sequences of probes for more early detection and also thereby reducing complexity. The placement of the agents can be improved such that the overhead is further reduced.

3.5 Conclusion

A novel RPL rank attacker identification scheme that also detects version attack is presented. Our proposed scheme is centralized and uses an intelligent probing technique and DES based IDS. We augment traditional DES based IDS such that attacker type is also diagnosed. Using our scheme location of attack node is identified accurately. Active probe packets are used judiciously to capture a deviation of attack behavior from the normal behavior which is normally lacking. A DES diagnoser serves as our IDS engine that generates an alert when an attack node is identified. The correctness and completeness of our approach is also proved.

The performance analysis of our proposed scheme in simulation and real testbed considers both attack and non-attack behavior patterns, with a sufficiently large number of IoT devices. The average energy usage and accuracy of our proposed approach are $14872mJ$ and 99.1%, respectively. The observed results show our approach is energy-efficient with lowest packet overhead than existing works. It is scalable, achieves minimum false positives, and higher accuracy with lower detection time.

In the following chapter, we delve into IoT adaptation layer fragmentation attacks which are also low overhead network-level attacks launched by an eavesdropping attacker node. Buffer fragmentation and reassembly procedures are easily exploited since 6LoWPAN lacks mechanisms to verify authenticity of the sender and the fragment ingenuity. Due to such attacks, services of the receiver node are blocked as fresh fragments wait on its buffer while the network performance as a whole is hindered. As opposed to the RPL attacks that have network-wide effects, 6LoWPAN fragmentation attacks make nodes at 1-hop distance vulnerable, exhausting memory and resources. Hence, centralised IDS schemes are not suitable to deal with such attacks. We propose a 6LoWPAN fragmentation attack detection and attacker identification scheme utilising decentralised I-DES based IDS and active probing mechanism in the next chapter. Duplicate fragments with random payloads are used on part of probes. Our proposed decentralised I-Diagnosability DES framework is adopted

and extended to identify an attack node, with local I-diagnosers acting independently. Our proposed mechanism is successfully validated in simulation and on a testbed with varying topologies and nodes. The results illustrate energy-efficiency, high accuracy, least false positives and quicker response times compared to other schemes.

CHAPTER

4

Mitigation of 6LoWPAN Fragmentation Attacks

Recently, a significant portion in the network usage of Internet of Things (IoT) [5, 108, 109] in healthcare to home automation, industrial control systems to agriculture and smart cities, mostly employ 6LoWPAN, an IETF-standardized adaptation layer, for IPv6 based communication [110]. With the surge in the number of resource constrained devices constituting the IoT, the need for a huge address space as well as IP-connectivity over low power and lossy networks (LLN), including Wireless Sensor Networks (WSN), are facilitated using 6LoWPAN [111, 10]. Fragmentation is therefore essential at this layer, since IEEE 802.15.4 limits the frame size to 127 bytes and hence does not permit transmission of IPv6 packets with MTU 1280 bytes. Consequently, the adaptation layer is proposed to forward, buffer and process the fragments of the transmitted packets.

Malicious nodes make use of the challenges due to the proposed implementation and exploit the fragmentation and reassembly procedures to launch various Denial-of-Service (DoS) attacks. Lack of mechanisms to verify authenticity of the sender and the fragment helps mount spoofing attacks. A malicious entity that is overhearing a communication requires just a mere fragment to illegitimately occupy the buffer of a resource constrained node, or to disrupt the integrity of the packet by slipping in duplicate fragments. In both the cases, since 6LoWPAN does not have means to verify fragment ingenuity, the buffer is freed and the packet needs to be resent [110, 37, 34, 112]. The security aspect of availability is at stake due to the trumped-up buffer reservation. The spurious occupancy also exhausts huge memory and time, since the fragments need to be kept in the constrained memory of

the nodes till timeout. Moreover, the impact of the attack is unbound if the attacker drops fragments and replaces them with fragments containing spoofed content, since 6LoWPAN processes out-of-order fragments. Thus the services of the receiver node are blocked as fresh fragments wait on its buffer while the network performance as a whole is hindered.

Approaches to secure 6LoWPAN from fragmentation attacks have been mostly cryptography based [112, 113, 37], which are resource unfriendly. Since such attacks are launched by replaying a fragment at pre-computed intervals, such approaches incur significant overhead. Moreover, the proposed countermeasures to tackle IoT-6LoWPAN attacks have attempted attack detection mostly. Also, buffer quarantine strategies that perform logical isolation only, leaves ample opportunities for a malicious entity to launch fresh attacks. A node may be isolated and not included in the packet forwarding path but may still start a communication or launch an attack since it still has ability to send fragments. Furthermore, with the advancements in adversarial machine learning techniques, complex attack characteristics and distributed attacks are also severely threatening. Therefore, energy efficient schemes that save resources of the constrained battery-powered low power devices are primarily necessary. We investigate the fragment duplication attack from the topology perspective and by analysing the possible attack space. To this regard, in this chapter an intrusion detection scheme is presented that identifies a 6LoWPAN attacker node accurately, and is based on the decentralized I-diagnosability framework of Discrete Event System (DES). Due to the analogous behaviours posed between a system fault and a malfunctioning network node, DES based IDSs are often chosen of late [150]. We also overcome the plausible issues that make the current solutions unsuitable. We perform node elimination apart from identification which is more powerful than mitigation based approaches because the network remains secured from further attacks by the same malicious node. Broadly, our contributions in this work can be summarised as follows:

- We propose a novel fragment duplication attacker identification scheme in 6LoWPAN. Our scheme makes use of an intelligent active probing technique that helps create a deviation of attack traffic and normal traffic [106]. Our proposed scheme is decentralized that incorporates decision making capabilities distributed over the set of local I-DES based IDSs each have local I-diagnosers performing independently and in a parallel manner. An alert is generated to the system administrator when any of the

local I-diagnosers reach a conclusion. Global diagnosis is ascertained when any local diagnoser is successful [169].

- We extend the power of traditional I-DES based IDS with attack type modeling for identification of malicious node.
- We perform node elimination which is more powerful than mitigation based approaches. Kill switches are employed to kill a particular node when identified.
- We prove the correctness and completeness of our approach by enumerating all the attack cases.
- The performance of our scheme is tested through simulations and real testbed. The experimental results highlight the applicability of our approach. Results show our approach is energy-efficient and has less response time. Our proposed solution has minimum false positives and achieves more than 99.8% accuracy in identifying the malicious nodes.

The rest of the chapter is organized as follows. Section II presents the preliminaries of 6LoWPAN. Section III presents the related work. We demonstrate our proposed scheme in Section IV. Experimental results and performance analysis of our scheme are presented in Section V. We conclude in Section VI.

4.1 Preliminaries

4.1.1 6LoWPAN Fragmentation mechanism

Owing to the limited path MTU size of the 802.15.4 links, 6LoWPAN needs to do fragmentation of IPv6 packets. 6LoWPAN splits an IP packet into multiple fragments so as to fit in a 802.15.4 MAC frame. In conventional IP fragmentation, all the fragments contain the header information. 6LoWPAN provisions functionalities for fragmentation and reassembly of IPv6 packets. In 6LoWPAN, each fragmented packet consists of fragment headers for carrying information to facilitate in-place reassembly. Two different types of fragment headers are defined, FRAG1, for the first fragment and FRAGN for all the subsequent fragments. They consist of the following subfields: (1) Dispatch type or Dispatch value bit

pattern (5 bits) is 11000 for the initial fragment (FRAG1) and 11100 for all subsequent ones (FRAGNs). It helps differentiate FRAG1 from FRAGNs. (2) The datagram size (11 bits) encodes the size of the original IPv6 datagram, with header and payload. (3) The datagram tag field of 16 bits is used to uniquely identify all fragments that belong to the same IPv6 packet. The above discussed fields constitute FRAG1. The initial fragment header is therefore 32 bits (or, 4 bytes). FRAGN contains an extra field, the datagram offset (8 bits) to determine the relative position of the fragments in the IPv6 packet. This value is incremented by 8 bytes for each fragment. This is crucial during reassembly to process out of order fragment sequence. In conventional IP fragmentation, all the fragments contain the header information, while in 6LoWPAN fragmentation only FRAG1 carries the compressed IPv6 Header information. End-to-end routing information is therefore present in FRAG1 only. The compressed IPv6 header can range between 2 to 20 bytes depending on different scenarios, while the IPv6 header is originally 40 bytes long. As shown in Figure 4.1, the compressed IPv6 header is added to the first fragment of the packet [168].

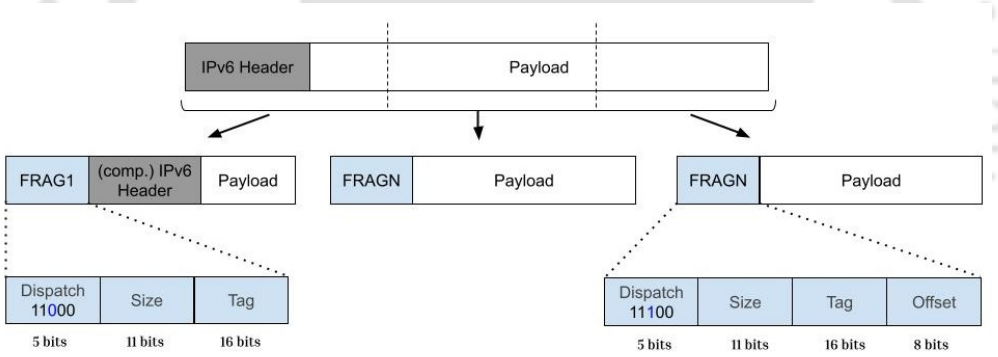


Figure 4.1: Fragmentation Process Overview

4.1.2 Fragment Duplication Attack

Various IP fragmentation attacks have been long discussed in the literature. Analogously, fragmentation attacks have been identified in 6LoWPAN and are further classified at the design-level based on their attack procedure. Fragment duplication attacks belong to a class of attacks that use duplicate fragments to deny a successful IPv6 datagram transmission between two communicating pair of nodes. Suppose that a sender S wants to communicate a datagram $\mathcal{D}$ to the receiver node R, and in the process, a fragment header,

$FRAG1(S, R, 5) \in \mathcal{D}$ is sent to R. Independent on the routing scheme in use, the fragment header, if acknowledged, contains the end-to-end routing information and ensures a buffer reservation at R for the ensuing fragments from S. A malicious node $\mathcal{A}$ that is overhearing this communication just needs a single fragment to carry out the attack. Considering that the fragments belonging to $\mathcal{D}$ are n in number, i.e., $FRAGN(S, R, i) \in \mathcal{D}$, $1 \leq i \leq n$, $\mathcal{A}$ need not disclose its identity and yet replay a fragment in $\mathcal{D}$ before the buffer reservation period gets over at the reassembling node. This is possible because the spoofed fragment, containing curated data, conforms to fragment header of packet $\mathcal{D}$.

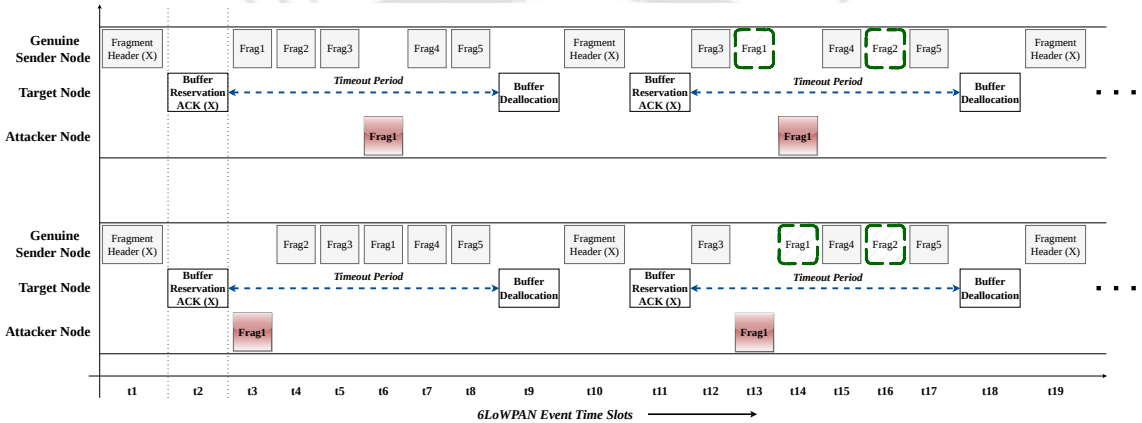


Figure 4.2: Attack Scenario for Fragment Duplication Attack

Furthermore, 6LoWPAN layer has no way of divining if the fragment came from S, since it lacks means to verify the authenticity of the sender. Moreover, the receiver buffer is not able to ascertain as to which of the redundant fragment is the original. Confirming this would require support from the upper layers, which it avoids and discards the packet. A malicious node at an intermediate hop has heightened capabilities of launching more severe forms of the attack. It can drop selective fragments and replace them with illegitimate ones such that the receiver cannot determine originality of the fragments. Possible attack timelines are shown in Figure 4.2. Here, the green dotted squares indicate dropped fragment while the red coloured squares indicate a spoofed fragment from an attacker node. In a resource constrained environment, application services can be fiercely hit due to a fragment duplication attack. Compromised are node energy and resources that face chances of exhaustion. Moreover, all secure connection setups are disrupted causing a DoS attack. Throughput of the network goes significantly down and the network services are compromised, partially or totally.

4.2 Related Work

Various schemes to protect the 6LoWPAN against fragmentation attacks have been suggested in the literature. One of the very first study addressing this issue is proposed in [113]. The authors present a security threat analysis from the point of view of IP fragmentation and Replay attacks. The proposed protection mechanism makes use of Timestamp and Nonce options to be added to packets at the adaptation layer. Fragmented packets between a pair of nodes, if unidirectional, is added with Timestamp option whereas Nonce option is added in case of bidirectional transfer. In the corresponding manner the newly formed packet formats are redefined. The work does not report of measures to be taken if the fields themselves may be spoofed as well to launch 6LoWPAN attacks.

A content chaining scheme is proposed in [112]. The scheme uses cryptographic techniques to bind the packet content to the fragment header. The output of a cryptographic hash function is appended to the end of the previous fragment in their scheme. The receiver node in turn recomputes and verifies these hash values, which helps it to identify the legitimate ones from the duplicate ones. However, since such attacks are launched by replaying a fragment at pre-computed intervals, such approach incurs significant overhead. Also, there is a lack of confidentiality in this scheme since encryption is not performed. Moreover, processing of out-of-order fragments is challenging.

In SecuPAN [34], authors show to mitigate a wide range of fragmentation attacks including the buffer reservation attack in 6LoWPAN. Spoofing attack mitigation is tackled using cryptographic address generation scheme whereas they propose using a nonce field in packet fragment header as a protection against replay attacks. Fragment fabrication vulnerability is handled using Message Authentication Code (MAC) in their scheme.

Nikravan et. al [37] propose a scheme that uses per-fragment Offline-Online Signcryption between communicating nodes to counter fragment duplication attack. The scheme is capable of processing out of order fragments and is suitable for resource constrained environment as they do not require point multiplication operation and certificate based verification of public key for authentication. However, the scheme is computationally costly with a packet overhead of almost 1.2 times the original packet. The scheme generates more fragments due to an increased payload.

4.3 Proposed Defense Scheme

In this section we demonstrate our defense scheme that wends off 6LoWPAN duplication attack. I-DES based IDS was introduced in the previous chapter (see details in Section 3.3.1). We start with an overview of the detection methodology using our proposed IDS scheme here. We then discuss the employed techniques and algorithms to identify the attacker. The construction of normal, attack models and DES diagnoser that are indispensable for attacker identification are described next. Proof of correctness and completeness is presented subsequently.

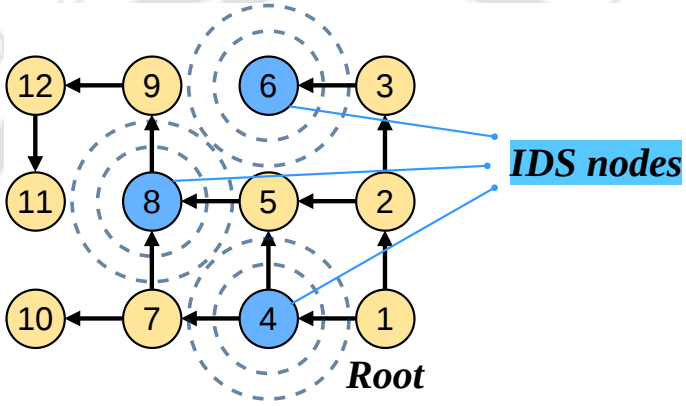


Figure 4.3: An example of 6LoWPAN deployment

4.3.1 Design Overview

The primary research challenges in the detection of fragment duplication attack are (i) resource constrained nodes (ii) normal node cannot be distinguished from attack node due to lack of sender authenticity verification in 6LoWPAN. To overcome the discussed challenges, we propose a decentralized approach with a set of IDS $\hat{\mathcal{I}}$ distributed across the monitored 6LoWPAN. Each of the IDS $\mathcal{I}_i \in \hat{\mathcal{I}}$ work in a similar manner and perform diagnosis independent of each other. Furthermore, we use intelligent probe datagram with overlapping fragments. Active probing helps generate unique responses such that an attack node can be differentiated from a normal node. A demo example of our IDS setup in an arrangement of RPL-6LoWPAN nodes is shown in Figure 4.3. The normal network nodes are coloured yellow while deployed IDS nodes are designated in blue. The node numbered 1 is the RPL root node. We show 3 IDS nodes for the sake of understanding only, while in

actual setup the arrangement may vary. The frame notations we use henceforth are listed in Table 4.1.

Table 4.1: FRAME NOTATIONS

Notation	Meaning
FRGHD	Datagram Header Fragment $FRAG1$
BRNACK	Buffer Reservation No Acknowledgement Packet
BRACK	Buffer Reservation Acknowledgement Packet
FRG	Datagram Fragment $FRAGN$
PRDACK	Packet Receipt Acknowledgement Packet
DPNACK	Duplicate Fragment No ACK Packet
RTO	Retransmission Timeout Event

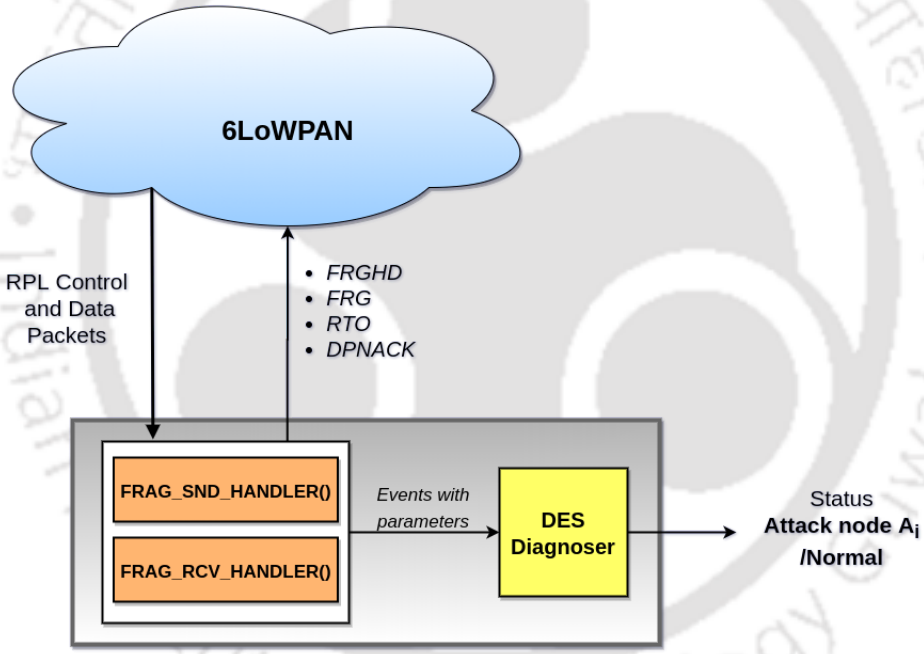


Figure 4.4: IDS architecture

Components in an IDS $\mathcal{I}_i$: The block diagram of proposed IDS with the basic components is shown in Figure 4.4 and are discussed here as follows:

- **FRAG_RCV_HANDLER():** It monitors communication data packets from IDS nodes and extracts information such as IP address, MAC address and datagram tags. It reports a source IDS node if the packets contain overlapping fragments. The working procedure of this handler is described in Section 5.2.4.

- **FRAG_SND_HANDLER():** Its prime responsibility is to extract vital information such as source IP address, MAC address and datagram tags from 6LoWPAN packets. It generates events such as $FRGHD$, $BRACK$, $BRNACK$, FRG, $DPNACK$, $PRDACK$ and RTO. The generated events are passed to the DES diagnoser component. The working procedure of this handler is described in Sections 5.2.4 and 5.2.5.

- **Local I-DES Diagnoser:** This component diganoses the attacker node and is implemented as a software module. Given the knowledge of the I-DES model specifications pertaining to normal and attack type conditions, the local I-diagnoser can be constructed. $FRAG\_SND\_HANDLER()$ passes information regarding network events to the diagnoser. Based on the event parameters that are shared, the diagnoser generates an alert on attack detection or identification of malicious nodes. The usage and construction of the diagnoser is described in Section 5.2.6.

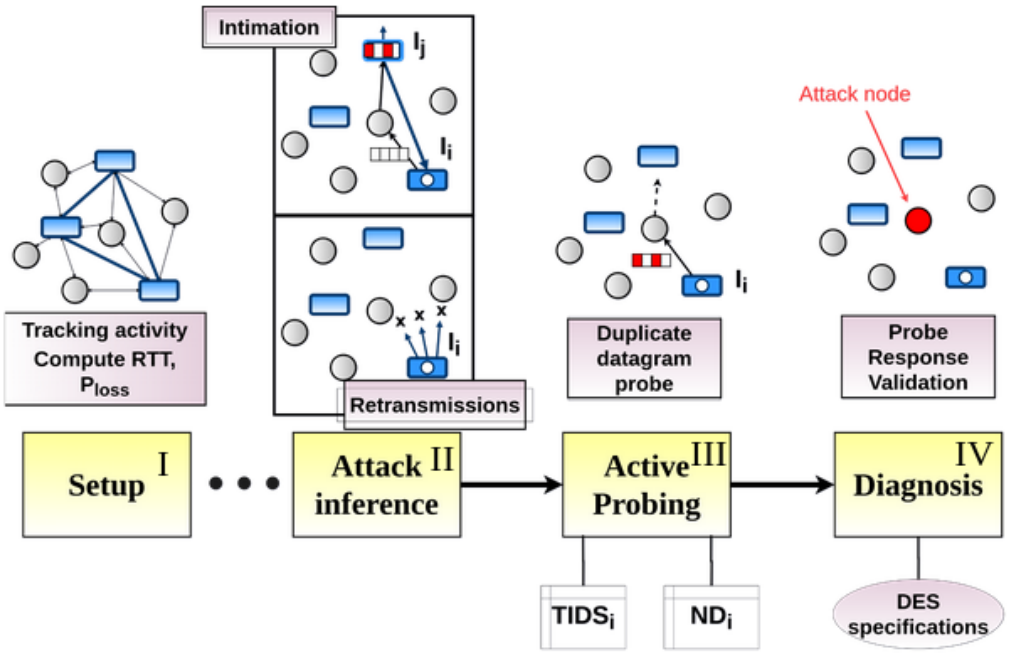


Figure 4.5: Flow of DES based IDS scheme

Attack detection and identification is sequentially carried out in phases. They are namely, **setup**, **inference**, **active probing** and **diagnosis**. The working methodology of our proposed scheme is demonstrated next. A schematic of the detection timeline is

shown using Figure 4.5. Prior to attack, network traffic is monitored and data is logged to setup the IDS as shown in the initial module. During this setup phase, the IDS perform all the normal functionalities of a 6LoWPAN node besides analysis. Tables used to store IDS and neighbour node related information are maintained and updated throughout. A fragment duplication attack is inferred in our scheme by an IDS source if the following two situations arise: (i) a packet needs to be retransmitted beyond a predetermined limit (ii) an intermediate or receiver IDS detects duplicate fragments during reassembly. In the latter scenario, the $FRAG\_RCV\_HANDLER()$ component of the IDS intimates about the duplication to the source IDS as obfuscated application data at random delay, to prevent an attacker or bot from profiling. The source IDS therefore conducts active probing if any of the above situations are recorded. The $FRAG\_SND\_HANDLER()$ on behalf of the source IDS sends fabricated IPv6 probe datagrams with random payload. Its diagnoser component (IDS engine) then validates the responses against I-DES modeled normal and attack behaviour specifications. An alert is generated to the system administrator if an attack behaviour is detected. This constitutes the diagnosis phase. Basically, a response to a probe datagram helps differentiate attack behaviour from the normal. Essentially, a malicious node behaviour is distinguishable if the crafted datagram is forwarded, as opposed to the fact that the datagram must be discarded in normal conditions. The phases in our detection procedure are now sequentially demonstrated.

4.3.2 Setup

This phase consists of administrator intervention for parameter setup. Traffic is monitored, relevant data is collected and parameters are measured for Network Traffic Analysis (NTA) purposes. At each IDS node, tables and variables corresponding to IDS as well as neighbour nodes are maintained and updated in this phase. An IDS node $\mathcal{I}_i$ uses table $\mathbf{TIDS_i}$, represented as an array of structures, for storing the other IDS information. An element of the array, $TIDS_i{}^j$ stores the IP address, MAC address for an IDS $\mathcal{I}_j$ and a round-trip time (RTT) value, Δ_i^j. $|TIDS_i|$ signifies the size of $TIDS_i$, i.e., the number of IDS nodes reachable from $\mathcal{I}_i$. RTT between a pair of communicating IDS nodes, let $\mathcal{I}_i$ and $\mathcal{I}_j$ can be found at the j^{th} and i^{th} table entries of $TIDS_i$ and $TIDS_j$, respectively. Essentially, Δ_i^j is computed as,

$$\Delta_i^j = T_{i,\mathcal{P}_{\mathcal{ACK}}}^j - T_{i,\mathcal{P}}^j \tag{4.1}$$

where, $T_{i,\mathcal{P}_{\mathcal{ACK}}}^j$ is the time instant when an ACK due for packet $\mathcal{P}$, $\mathcal{P}_{\mathcal{ACK}}$ is received at $\mathcal{I}_i$ from $\mathcal{I}_j$ and $T_{i,\mathcal{P}}^j$ represents the time instant when the packet $\mathcal{P}$ got transmitted from $\mathcal{I}_i$ to $\mathcal{I}_j$. Also, a variable Δ_i is used at $\mathcal{I}_i$ to hold the maximum round trip time (RTT) value, i.e., $\max_j\{\Delta_i^j\}$ and is continuously updated. Besides, $TIDS_i$, a table $\mathbf{ND_i}$ is maintained at $\mathcal{I}_i$ to store requisite parameters and information of its next-hop neighbour nodes. $|ND_i|$ signifies the size of ND_i, i.e., the number of next-hop neighbour nodes of $\mathcal{I}_i$. Each element of the array of the form ${ND_i}^j$ in IDS $\mathcal{I}_i$ holds the IP address, MAC address of a node j and a value $P_{i,loss}^j$ indicating the probability of packet losses using the link $\overline{\mathcal{I}_i j}$. $P_{i,loss}^j$ is calculated from the history of packets sent from $\mathcal{I}_i$ and ACKs received. We calculate $P_{i,loss}^j$ as,

$$P_{i,loss}^j = 1 - \frac{n_{i,r}^j}{n_{i,s}^j} \tag{4.2}$$

where, $n_{i,r}^j$ represents the number of due ACKs received at $\mathcal{I}_i$ via node j and $n_{i,s}^j$ represents the number of packets sent from $\mathcal{I}_i$ via node j. Correspondingly, a variable $P_{i,loss}$ is used at $\mathcal{I}_i$ to hold the packet loss probability of the link which is the minimum, i.e., $\min_j\{P_{i,loss}^j\}$. An IDS utilises the communication history in the subsequent phases. The attack inference phase is demonstrated in the following subsection.

ALGORITHM 8: $FRAG\_RCV\_HANDLER()$ of I_j

Local Variables: $\mathbf{TIDS_j}$
Input: Datagram $\mathcal{D}$
Output: Fragment duplication NACK

```
while ∃u, v|(F_u, F_v) ∈ D ∧ (u = v) do
    if D.sourceIP ∈ TIDS_j(IP) then
        Intimate NACK to D.sourceIP;
```

4.3.3 Attack Inference

As a prerequisite, IDS are assumed to be setup from monitoring network activities. An attack then takes place in the monitored 6LoWPAN. Attack detection and node identification relies on the inference of a fragment duplication attack activity by IDS nodes. Specifically, a source IDS node infers an attack while sending a datagram, let us suppose $\mathcal{D}$, as part

of its regular communication. Firstly, a suspicious duplication activity might be reported by an intermediate IDS node or a receiver IDS, or secondly, an attack might be inferred due to failed retransmissions. To understand the first scenario, let us consider that $\mathcal{D}$ is sent from an IDS $\mathcal{I}_i$. Now, as $\mathcal{D}$ gets reassembled and forwarded hop-by-hop, suppose an eavesdropping attack node passes a spoofed fragment of $\mathcal{D}$ towards a reassembling node. If this current reassembling node is an IDS node, let us suppose $\mathcal{I}_j$, it not only discards $\mathcal{D}$, but also intimates a negative acknowledgement to $\mathcal{I}_i$, since it is an IDS source. The $FRAG\_RCV\_HANDLER()$ component handles communication on behalf of the reassembling IDS $\mathcal{I}_j$. Piggybacked information is obfuscated and reported to $\mathcal{I}_i$. To prevent an attacker from profiling, the agents send the intimation packet with a random delay. Function of the $FRAG\_RCV\_HANDLER()$ component of $\mathcal{I}_j$ is explained using Algorithm 8. On receipt of the NACK from $\mathcal{I}_j$, $\mathcal{I}_i$ infers a fragmentation attack on $\mathcal{D}$.

The second attack inference situation occurs when a datagram, say $\mathcal{D}$, needs to be retransmitted beyond a certain limit. We look into how retransmissions are characterized in our scheme. Suppose $\mathcal{I}_i$ waits for an ACK after having sent datagram $\mathcal{D}$. Now, if an ACK is not received before a set ACK timeout expiry, then $\mathcal{D}$ is retransmitted. Consequently, a counter variable that keeps track of the retransmission count is incremented. If the number of retransmissions exceeds the retransmission limit, then $\mathcal{I}_i$ refrains from retransmitting and marks the activity as a suspicious attack. The ACK timeout value and retransmission limit are realized from round-trip delay and the probability of packet loss, respectively, in our scheme. Variable Δ_i, computed in the setup phase, that stores the maximum of RTT from $\mathcal{I}_i$ is used to quantify the ACK timeout period. Since a packet or ACK may be delayed in the network due to varied reasons, and to avoid inferring from such delay unreasonably, we consider the maximum timeout period. The retransmission limit used by $\mathcal{I}_i$ depends upon the next-hop node. A set, $\hat{\Theta}_i = \{\Theta_i^1, \Theta_i^2, \dots\}$, is maintained to this purpose. The number of elements of set $\hat{\Theta}_i$ is equal to the number of next-hop neighbours of $\mathcal{I}_i$ (size of table ND_i). A value Θ_i^j stores the retransmission limit that is used if the next-hop node is j. It can be computed as,

$$\Theta_i^j = \lceil \frac{1}{1 - P_{i,loss}^j} \rceil \tag{4.3}$$

We characterize the retransmission limit using the probability of successful packet

delivery $(1 - P^{j}_{i,loss})$, which is justified because an ACK timeout beyond the retransmission limit can be inferred to be occurring due to reasons beyond normal network behaviour.

In both of the situations discussed above, attack inference by $\mathcal{I}_i$ is handled due to the $FRAG\_SND\_HANDLER()$ component, shown in Algorithm 9. Outlined below is the $FRAG\_SND\_HANDLER()$ whose inputs are as follows:

- Duplicate Fragment Intimation NACK that are reported due to the $FRAG\_RCV\_HANDLER()$ on behalf of host IDS on receipt of duplicated fragments.
- Probe datagram from the buffer that are yet to be sent (this becomes possible as $FRAG\_SND\_HANDLER()$ is part of the modified 6LoWPAN).
- Datagram receipt ACK.
- TEST_FLAG indicates when to detect and identify the attack by sending probe packets to intended nodes.

The $FRAG\_SND\_HANDLER()$ component and our local I-diagnoser work in communion for attack detection and node identification in our scheme. Initially, on the values $\hat{\Theta}_i$ and Δ_i being computed, the diagnoser sets TEST_FLAG = 1 (Line 1). The two values are pre-computed during non-attack condition as discussed in Section 4.3.2. The outputs events of the handler, namely, $FRGHD$, $BRACK$, $BRNACK$, FRG, $DPNACK$, $PRDACK$ and RTO, are all passed to the DES diagnoser. The handler and diagnoser share the model variables c_i, Δ_i, $\hat{\Theta}_i$, N_i, $\dot{r}_i$, $macd$ and ipd between them. When the TEST_FLAG is set by the diagnoser, it means that the attack diagnosis phase can be started. The DES diagnoser gets executed and remains so till the monitored 6LoWPAN is operational. Diagnoser setting the TEST_FLAG = 1 is also the initial transition in the DES diagnoser graph discussed later. On receipt of a fragment duplication NACK, the first *if then* module is invoked and attack is inferred. The $FRAG\_SND\_HANDLER()$ generates event $DPNACK$ and passes it to the diagnoser. A variable N_i, used to keep track of the number of retransmissions, is initialized to 0 at the model start and is incremented when a buffer reservation ACK is received by $\mathcal{I}_i$ (event $BRACK$ is generated and N_i is shared with diagnoser). If an ACK timeout occurs, i.e., clock timer variable c_i counts beyond Δ_i, consecutively, for a Θ_i^{macd} time, the third *if then* module is invoked. This is

how attack is inferred from the number of retransmissions. Consequently, the event RTO is shared with diagnoser. $\mathcal{I}_i$ on inferring a possible fragmentation attack conducts active probing. $FRAG\_SND\_HANDLER()$ of $\mathcal{I}_i$ generates events during active probing that are key to diagnosis of attack node, as will be demonstrated in the subsequent sections.

4.3.4 Active probing

In 6LoWPAN based fragmentation attacks, attack specific signatures are naturally lacking. Moreover, devices are resource constrained and there are no in-place sender authenticity mechanism in 6LoWPAN. Hence, it is challenging to differentiate between a normal node behaviour and an attack behaviour, which generally appear same. We use an active probing technique, intelligently, to generate responses that make an attack behaviour distinguishable from normal. An IPv6 datagram with random payload, sent by an inferring node, are the active probes in our proposed scheme. This probe datagram, let $\mathcal{D}'$, is fabricated in a way such that a randomly chosen fragment in $\mathcal{D}'$ (using $RND1$) is duplicated. The duplicate fragment replaces a genuine fragment in $\mathcal{D}'$ (also randomly chosen using $RND2$). As demonstrated, $FRAG\_SND\_HANDLER()$ sends a probe fragment header on behalf of $\mathcal{I}_i$, after an attack is inferred. Event $FRGHD$ is generated here and passed to the diagnoser. Furthermore, an IDS node from $TIDS_i$ is chosen as the destination IP for probe datagram $\mathcal{D}'$, and assigned to model variable *ipd* and shared. A boolean variable $\dot{r}_i$ is set to 1 to flag the probe datagram, and is also shared with the diagnoser. On receipt of a buffer reservation ACK for $\mathcal{D}'$ the second *if then* module is invoked. If a fragment is sent, event FRG is generated and subsequently shared. Probe fragments in $\mathcal{D}'$ are sent such that duplicate fragments are received at the next-hop node. Meanwhile, diagnoser sets TEST_FLAG to 0 until a response due for $\mathcal{D}'$ is received. While waiting on response, table $TIDS_i$ is consulted, and RTT specific to the destination IDS node is considered for timeout calculation. Because of the receiver of $\mathcal{D}'$ being an IDS node, a probe datagram ACK or duplicate fragment NACK might be received, or a response timeout gets clocked over. Consequently, an event among $PRDACK$, $DPNACK$ or RTO is generated and passed to the diagnoser. It then validates the received events against DES-modeled normal and attack behaviour specifications, and from the model parameters that are shared. Depending on the response characteristics, a decision is made on the next-hop node to be normal or

attack, or further probing is continued. An alert is generated to the system administrator if an attack behaviour is diagnosed. This constitutes the diagnosis phase and is discussed in the following subsection.

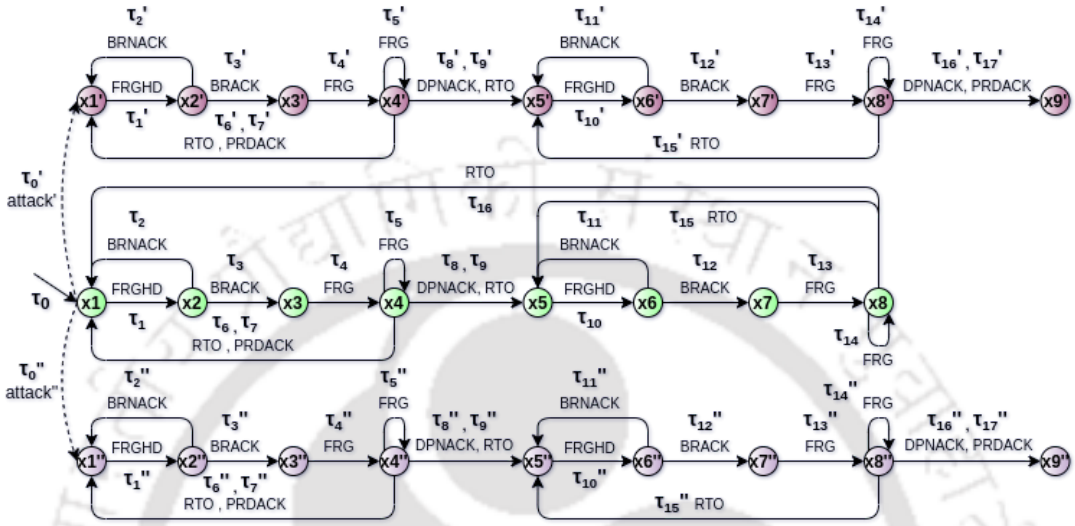


Figure 4.6: DES model H_i

4.3.5 I-DES Model and local I-Diagnoser

The I-DES model H is formally defined as a 6-tuple $H_i = \langle X, X_0, \Sigma, V, C, \Im \rangle$. Here, X is the set of states and is finite, $X_0 \subseteq X$ is the set of initial states, Σ is the finite set of events, V is the finite set of model variables, C is the finite set of clock variables and $\Im$ is the finite set of transitions. Elements of the set of model variables assume values from their respective domain sets. Suppose if $V = \{v_1, v_2, \ldots, v_n\}$ is the set of model variables (for some finite value of n) where each element v_i takes some values from its domain set Dom_i. The domain of each of the clock variables is the set of non-negative reals, R. A transition $\tau \in \Im$ is defined as a 7-tuple $\langle x, x^+, \sigma, \phi(V), \Phi(C), Reset(C), Assign(V) \rangle$, where x, x^+ are the source state and destination state of transition τ respectively. Due to the occurrence of the event $\sigma \in \Sigma$, the transition τ is enabled. $\phi(V)$ is defined as a boolean conjunction of equalities over some subset of the model variables, V, and which needs to hold true overall for a transition to be taken. $\Phi(C)$ is an invariant condition over some subset of the clock variables C. $Reset(C)$ is a subset of clock variables to be reset and $Assign(V)$ is a subset of model variables along with an assignment of values from their corresponding domains.

ALGORITHM 9: $FRAG\_SND\_HANDLER()$ of $\mathcal{I}_i$

Data: c_i, Δ_i, $\hat{\Theta}_i$, N_i, $\dot{r}_i$, $macd$, ipd
Input: Duplicate Fragment Intimation NACK, Datagram receipt ACK, Buffer ACK, $TEST\_FLAG$
Output: Events: $FRGHD$, $BRACK$, $BRNACK$, FRG, $DPNACK$, $PRDACK$, RTO

```
1  while Θ̂_i and Δ_i are not NULL do
2      if (TEST_FLAG == 1) then
3          Generate RND1 and RND2;
4      if Duplicate Fragment NACK then
5          Generate event DPNACK;
6          N_i ← 0;
7          if (c_i < Δ_i) ∧ (ṙ_i = 0) then
8              Generate event FRGHD;
9              ipd ← FRGHD.destIP;
10             Send datagram probe FRAG1;
11             Stop c_i();
12             ṙ_i ← 1;
13     if Buffer ACK then
14         Generate event BRACK;
15         N_i + +;
16         Start c_i();
17         if ṙ_i = 0 then
18             macd ← BRACK.sourceMAC;
19             foreach FRAGN of D do
20                 Generate event FRG;
21                 Send probe fragment FRAGN;
22         else if ṙ_i = 1 then
23             foreach FRAGN of D do
24                 if FRAGN.offset = RND2 × 8 then
25                     FRAGN.offset ← 8 × RND1;
26                 Generate event FRG;
27                 Send probe fragment FRAGN;
28             TEST_FLAG ← 0;
29     if ((c_i ≥ Δ_i) ∧ (ṙ_i = 0)) then
30         Generate event RTO;
31         Stop c_i();
32         Generate event FRGHD;
33         Send datagram probe FRAG1;
34         if (N_i = Θ_i^macd) then
35             ipd ← FRGHD.destIP;
36             N_i ← 0;
37             ṙ_i ← 1;
38     else if ((c_i ≥ Δ_i^ipd) ∧ (ṙ_i = 1) ∧ (N_i < |ND_i|)) then
39         Generate event RTO;
40         Stop c_i();
41         TEST_FLAG ← 1;
42         Generate event FRGHD;
43         ipd ← FRGHD.destIP;
44         Send datagram probe FRAG1;
45     if Datagram receipt ACK then
46         Generate event PRDACK;
47         Stop c_i();
48         if (c_i ≤ Δ_i) ∧ (ṙ_i = 0) then
49             N_i ← 0;
```

Some of the fields in the tuple representing a transition maybe be denoted by "-". For example, if "-" is used for $\phi(V)$ or $Assign(V)$, then it would mean that no condition needs to be met (i.e., the condition is implicitly TRUE) or NO assignment is required respectively.

The I-DES modeling of a networked system of IoT-6LoWPAN nodes is demonstrated here. The I-DES model H_i, drawn in Figure 4.6, has been used to represent the *Fragment Send Receive* sequences during normal and fragment duplication attack conditions. The principle of detection and identification by the local I-diagnoser is discussed later. We will then show that location of an attacker, if present, is accurately identified. The DES notations used are listed in Table 4.2 and the definitions of the various DES terminologies have been presented in Appendix A. The various components of H_i are discussed.

Table 4.2: LIST OF SYMBOLS

Symbol	Definition
H_i	DES model
Σ	Set of events of the DES model H_i
Σ_m	Set of measurable events of the DES model H_i
Σ_{um}	Set of unmeasurable events of the DES model H_i
V	Set of model variables of the DES model H_i
$\Im$	Set of transitions of the DES model H_i
τ	A transition $\tau \in \Im$
X	Set of states of the DES model H_i
X_0	Set of initial states of the DES model H_i
σ	Event on which a transition is enabled
$check(V)$	Condition(s) on a subset of model variables, V
$assign(V)$	Assignment(s) on a subset of model variables, V
$L(H)$	Set of all traces *generated* in H_i
A_k	k^{th} attacker
X_N	Set of normal states of the DES model H_i
X_F	Set of faulty states of the DES model H_i for fault type F
X_{A_k}	Set of attacker states of the DES model H_i for attacker A_k
σ_{A_k}	Event corresponding to attack launched by attacker A_k
O	Diagnoser of DES H_i
Z	Set of states of the diagnoser, O, also called O-states
Z_0	Set of initial nodes of the diagnoser, O
A	Set of transitions of the diagnoser, O, also called O-transitions

The state set X with initial set of states X_0 ($X_0 \subseteq X$) symbolise the control states of the $FRAG\_SND\_HANDLER()$ component of the IDS. The normal DES model states

and attacker type model states together constitute the state set $X = \{x1, x2, \ldots, x8, x1', x2', \ldots, x9', x1'', x2'', \ldots, x9''\}$. In our model, the set of model variables, $V = \{ips, ipd, macs, macd, dtag, N_i, id\}$. The model variables correspond to program and data variables that are internal to the IDS. Certain program variables are designated as the clock variables, C, which are absolute values of clock timer that can be SET and RESET using commands. In real-time applications, timing constraints are expressed by satisfying the conditions on the clock variables. We use a single clock variable in the set of clock variables, i.e., $C = \{c_i\}$. Event set Σ contains the packet communication events. In our model, the set of events, $\Sigma = \{FRGHD, BRNACK, BRACK, FRG, DPNACK, RTO, PRDACK, attack', attack''\}$. A transition is enabled if the conditions are satisfied and is said to be taken on the occurrence of the associated event. The transitions set Γ consists of transitions $\{\tau0, \tau1, \ldots, \tau16, \tau1', \tau2', \ldots, \tau17', \tau1'', \tau2'', \ldots, \tau17''\}$.

Considering that there is one attack node among n nodes in the network, the state set, X, can be partitioned into disjoint sets $X_N, X_{A_1}, X_{A_2}, \ldots, X_{A_n}$, where, X_N represents the set of states belonging to the normal behavior of the network, while states of the form X_{A_k}, $1 \leq k \leq n$, $k \in \mathcal{N}$, represent the behavior of the network if A_k is an attack node. For the sake of brevity, we model using 2 nodes, A_1 and A_2, among which one is an attack node, hence $X = X_N \cup X_{A_1} \cup X_{A_2}$. In Figure 4.6, the non-primed states are the states when the system behaves normally while the single and double primed states represent the system under attack by the nodes A_1 and A_2, respectively. The events of the system is disjoint union of measurable events and unmeasurable events Σ_m and Σ_{um}.

I-DES behavior under normal circumstances

The behavior of H_i under normal circumstances is shown in Figure 4.6. The system, when functioning normally, is represented using the states $\{x1, x2, \ldots, x8\}$ and the transitions $\{\tau0, \tau1, \ldots, \tau16\}$. The initial state of X_0 is $x1$. We next discuss the transitions in normal condition as follows:

- $\tau0$, the initial transition leads to the initial state $x1$ as shown in Figure 4.6. It is assumed while modeling that the constant timeout values, Δ_i and $\hat{\Theta}_i$ have been computed and then $\tau0$ takes place. There is no explicit event that triggers $\tau0$. Occurrence of $\tau0$ implies that the DES model is invoked when the timeout values

Table 4.3: TRANSITIONS $\Im$ IN H_i CORRESPONDING TO NETWORK PACKET FRAMES

Event (σ)	Transition	$\phi(V)$	Assign(V)	$\phi(C)$	Reset(C)
, **FRGHD**	$\langle x1,x2\rangle$,$\langle x1',x2'\rangle$,$\langle x1'',x2''\rangle$	-	$ips \leftarrow FRGHD_{IPS}$	-	-
	$\langle x5,x6\rangle$,$\langle x5',x6'\rangle$,$\langle x5'',x6''\rangle$	-	$ipd \leftarrow FRGHD_{IPD}$	-	-
		-	$macd \leftarrow FRGHD_{MACD}$	-	-
		-	$dtag \leftarrow FRGHD_{DATAGRAM\_TAG}$	-	-
BRNACK	$\langle x2,x1\rangle$,$\langle x2',x1'\rangle$,$\langle x2'',x1''\rangle$	$ips \equiv BRNACK_{IPD}$	-	-	-
		$dtag \equiv BRNACK_{DATATGRAM\_TAG}$	-	-	-
BRACK	$\langle x2,x3\rangle$,$\langle x2',x3'\rangle$,$\langle x2'',x3''\rangle$	$ips \equiv BRACK_{IPD}$	$N_i \leftarrow N_i + 1$	-	$c_i \leftarrow 0$
		$dtag \equiv BRACK_{DATAGRAM\_TAG}$	-	-	-
BRACK	$\langle x6,x7\rangle$,$\langle x6',x7'\rangle$,$\langle x6'',x7''\rangle$	$ips \equiv BRACK_{IPD}$	$N_i \leftarrow N_i + 1$	-	$c_i \leftarrow 0$
		$dtag \equiv BRACK_{DATAGRAM\_TAG}$	-	-	-
		-	$TEST\_FLAG \leftarrow 0$	-	-
FRG	$\langle x3,x4\rangle$,$\langle x3',x4'\rangle$,$\langle x3'',x4''\rangle$	$ips \equiv FRG_{IPS}$	-	-	-
		$ipd \equiv FRG_{IPD}$	-	-	-
		$dtag \equiv FRG_{DATAGRAM\_TAG}$	-	-	-
FRG	$\langle x7,x8\rangle$,$\langle x7',x8'\rangle$,$\langle x7'',x8''\rangle$	$ips \equiv FRG_{IPS}$	$FRG_{OFFSET} \leftarrow RND1$	-	-
		$ipd \equiv FRG_{IPD}$	-	-	-
		$dtag \equiv FRG_{DATAGRAM\_TAG}$	-	-	-
FRG	$\langle x8,x8\rangle$,$\langle x8',x8'\rangle$,$\langle x8'',x8''\rangle$	$ips \equiv FRG_{IPS}$	-	-	-
		$ipd \equiv FRG_{IPD}$	-	-	-
		$dtag \equiv FRG_{DATAGRAM\_TAG}$	-	-	-
		$RND2 \neq FRG_{OFFSET}$	-	-	-
DPNACK	$\langle x4,x5\rangle$,$\langle x4',x5'\rangle$,$\langle x4'',x5''\rangle$	$ips \equiv DPNACK_{IPD}$	-	$c_i < \Delta_i$	-
		$dtag \equiv DPNACK_{DATAGRAM\_TAG}$	-	-	-
DPNACK	$\langle x8',x9'\rangle$	$ips \equiv DPNACK_{IPD}$	-	$c_i < \Delta_i{}^{ipd}$	-
		$ipd \equiv DPNACK_{IPS}$	-	-	-
		$dtag \equiv DPNACK_{DATAGRAM\_TAG}$	-	-	-
		$macd \equiv mac'$	-	-	-
PRDACK	$\langle x4,x1\rangle$,$\langle x4',x1'\rangle$,$\langle x4'',x1''\rangle$	$ips \equiv PRDACK_{IPD}$	$N_i \leftarrow 0$	-	-
		$ipd \equiv PRDACK_{IPS}$	-	-	-
		$dtag \equiv PRDACK_{DATAGRAM\_TAG}$	-	$c_i < \Delta_i$	-
PRDACK	$\langle x8'',x9''\rangle$	$ips \equiv PRDACK_{IPD}$	$N_i \leftarrow 0$	-	-
		$ipd \equiv PRDACK_{IPS}$	-	-	-
		$dtag \equiv PRDACK_{DATAGRAM\_TAG}$	-	$c_i < \Delta_i{}^{ipd}$	-
		$macd \equiv mac''$	-	-	-
RTO	$\langle x4,x1\rangle$,$\langle x4',x1'\rangle$,$\langle x4'',x1''\rangle$	$N_i < \Theta_i{}^{macd}$	-	$c_i \geq \Delta_i$	-
RTO	$\langle x4,x5\rangle$,$\langle x4',x5'\rangle$,$\langle x4'',x5''\rangle$	$N_i = \Theta_i{}^{macd}$	-	$c_i \geq \Delta_i$	-
RTO	$\langle x8,x5\rangle$,$\langle x8',x5'\rangle$,$\langle x8'',x5''\rangle$	$N_i < \lvert ND_i\rvert$	$TEST\_FLAG \leftarrow 1$	$c_i \geq \Delta_i{}^{ipd}$	-
RTO	$\langle x8,x1\rangle$	$N_i = \lvert ND_i\rvert$	$TEST\_FLAG \leftarrow 1$	$c_i \geq \Delta_i{}^{ipd}$	-
attack'	$\langle x1,x1'\rangle$	-	$id \equiv mac'$	-	-

are not NULL. Table 5.2 shows $initial(\tau 0) = --$, i.e., there are no initial states and $final(\tau 0) = x1$. $\sigma = TRUE$ means that transition $\tau 0$ is always enabled and $x1$ is automatically reached at the start of the model. $check(V) = --$ implies that no condition over the model variables are checked and the condition is always satisfiable for the transition. Value 1 is assigned to variable TEST_FLAG as implied by $Assign(V) = \{TEST\_FLAG \leftarrow 1\}$, which in turn means that the detection of fragmentation attacker can be started.

- $\tau 1 : (x1 \rightarrow x2)$ When the model is started and the current state is at $x1$, the transition $\tau 1$ implies that an IPv6 packet fragment of FRAG1 header type is sent. Here, $initial(\tau 1) = x1$ and $final(\tau 1) = x2$. $\sigma = FRGHD$ implies that transition $\tau 1$ is enabled when $FRAG\_SND\_HANDLER()$ generates event $FRGHD$ (i.e., after a FRAG1 is sent). $check(V) = --$ meaning that no condition need to be satisfied and $Assign(V) = \{ips \leftarrow FRGHD_{IPS}, ipd \leftarrow FRGHD_{IPD}, macd \leftarrow FRGHD_{MACD}, dtag \leftarrow FRGHD_{DATAGRAM\_TAG}\}$. The parameters that uniquely

identify a fragment header of a datagram are source IP, destination IP, destination MAC and datagram tag. Consequently, all the parameters that correspond to the header are stored in the model variables, *ips*, *ipd*, *macd* and *dtag*.

- $\tau 2 : (x2 \rightarrow x1)$ At state $x2$, the transition $\tau 2$ implies that a negative buffer acknowledgement is received. $\sigma = BRNACK$ implies that $\tau 2$ is enabled when the $FRAG\_SND\_HANDLER()$ generates the event $BRNACK$ (i.e., after a buffer NACK is received). $check(V) = \{ips = BRNACK_{IPD}, dtag = BRNACK_{DTAG}\}$ and $Assign(V) = --$. Satisfaction of the conditions over the model variables, *ips* and *dtag*, ensure that the buffer NACK is a response to the header fragment sent in $\tau 1$. No assignments are done in this transition.

- $\tau 3 : (x2 \leftarrow x3)$ At state $x2$, the transition $\tau 3$ implies that a buffer acknowledgement has arrived from a next hop node for some sent FRAG1 type fragment. Here, $initial(\tau 3) = x2$ and $final(\tau 3) = x3$. $\sigma = BRACK$ corresponds to enabling transition $\tau 3$ after the $FRAG\_SND\_HANDLER()$ generates the event $BRACK$ implying that a buffer ACK has arrived and the condition on the model variables in $check(V)$ are satisfied. $check(V) = \{ips = BRACK_{IPD}, dtag = BRACK_{DATAGRAM\_TAG}\}$. Similar to the situation described for $\tau 2$, the conditions over the model variables, *ips* and *dtag*, ensure that the ACK packet is a response to $\tau 1$. $Assign(V) = \{N_i \leftarrow N_i + 1\}$. The counter variable, N, used to keep track of the buffer allocations, is incremented. $Reset(C) = \{c_i \leftarrow 0\}$ indicates that the retransmission timer, modeled using the clock variable c_i, is started.

- $\tau 4 : (x3 \rightarrow x4)$ At state $x3$, the transition $\tau 4$ corresponds to sending an IPv6 packet fragment of FRAGN header type. $\sigma = FRG$ implies that the transition $\tau 4$ is enabled when the $FRAG\_SND\_HANDLER()$ generates the event FRG. $check(V) = \{ips = FRG_{IPS}, ipd = FRG_{IPD}, dtag = FRG_{DATAGRAM\_TAG}\}$. The conditions over the model variables if satisfied ensure that the FRAGN belongs to the same datagram and bears the same source and destination address. No conditions on the clock variables need to be satisfied here.

- $\tau 8$ $(x4 \rightarrow x5)$ At state $x4$, the transition $\tau 8$ implies that a duplicate NACK intimation message is received. $\sigma = DPNACK$ implies that the transition is enabled when

the $FRAG\_SND\_HANDLER()$ generates the event $DPNACK$. $check(V) = \{ips = DPNACK_{IPD}, dtag = DPNACK_{DATAGRAM\_TAG}\}$. The condition check on the model variables ips, and $dtag$ are used to ensure that the intimation is received as a response to the datagram sent in $\tau 1$.

- $\tau 15 : (x8 \rightarrow x5)$ At state $x8$, the transition $\tau 15$ corresponds to a retransmission timeout. $\sigma = RTO$ implies that the transition $\tau 15$ is enabled when the $FRAG\_SND\_HANDLER()$ generates the event RTO. $check(V) = \{N_i < |ND_i|\}$, $Assign(V) = \{TEST\_FLAG \leftarrow 1\}$ and $\Phi(C) = \{c_i \geq \Delta_i{}^{ipd}\}$. The condition over the model variable N_i is checked to send datagram probes via other next-hop nodes. $Assign(V)$ makes $TEST\_FLAG = 1$ which means that the attack detection phase can restart, i.e., $FRAG\_SND\_HANDLER()$ can again receive acknowledgements on sent datagrams from receiver nodes and duplicate intimation from other IDS nodes. $\Phi(C)$ implies that $\tau 15$ is enabled when the clock variable overshoots the retransmission timeout.

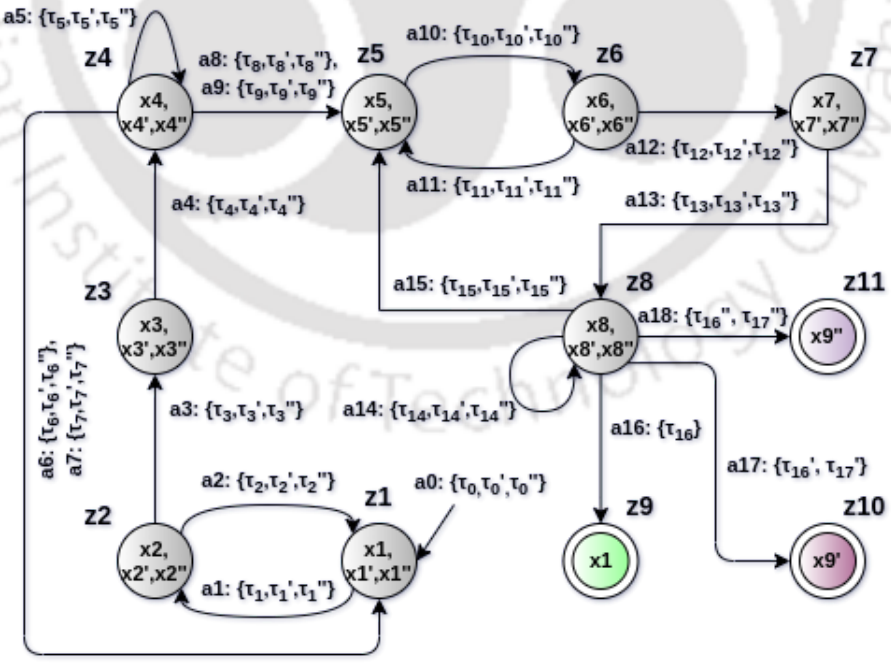


Figure 4.7: Diagnoser O for DES model H_i

I-DES behavior under attack circumstances

The DES model under rank or version attack condition launched by attacker A_1 is shown using the states in $X_{A_1} = \{x1', x2', \ldots, x9'\}$ and transitions, $\{\tau1', \tau2', \ldots, \tau17'\}$. Similarly for attacker type A_2, states and transitions are represented using double prime notation, $X_{A_2} = \{x1'', x2'', \ldots, x9''\}$ and transitions, $\{\tau1'', \tau2'', \ldots, \tau17''\}$ as shown in Figure 4.6. The DES model behavior under different attackers are mostly identical except two transitions that differentiate them which are discussed.

- At state $x1$, the system reaches an attacker type state $x1'$ or $x1''$ following an unmeasurable attack transition ((Definition 1, Appendix A)) $\tau0'$ or $\tau0''$, respectively.
- $\tau16' : (x8' \rightarrow x9')$ At state $x8'$, the transition $\tau16'$ corresponds to duplicate fragment intimation for the datagram probe sent via a known 1-hop node. $\sigma = DPNACK$ implies that the transition is enabled when the $FRAG\_SND\_HANDLER()$ generates the event $DPNACK$. $check(V) = \{ips = DPNACK_{IPD}, ipd = DPNACK_{IPS}, dtag = DPNACK_{dtag}, macd = mac'\}$. The conditions over the model variables, ips, ipd and $dtag$, ensure that the duplicate NACK intimation is a response to the datagram probe header fragment in $\tau10'$. The model variable $macd$ holds the MAC address of the next-hop node via which the probe was sent. $\tau16'$ ensures that $macd$ holds the IP address of attack node A_1. An intimation for probe sent via a node with MAC address stored in $macd$ is a fragment duplication attack. $\Phi(C) = \{c_i < {\Delta_i}^{ipd}\}$ means that c_i does not exceed the retransmission timeout period.

Local I-Diagnoser

The property of I-diagnosability pertaining to fault diagnosis in I-DES has been discussed in Section 5.2.6 of the previous chapter. Here we use a set of IDS each consisting of a local I-diagnoser. Each of the local I-diagnosers are basically observer automatons. Given a measurable trace executed on the model, the diagnosers, constructed from corresponding I-DES models, give an estimate of membership of the current system state in the model among normal or any attacker type state from H_i, locally. An alert is generated when it can be ascertained that the current state belongs to an attacker type. It is also notified as to which attacker type the corresponding belongs. Details of the diagnoser construction

procedure and definitions pertaining to diagnosability are highlighted in Appendix B. Now, an attack node is I-diagnosable in finite time, if the I-diagnosability condition is met (F_i-I-Diagnosability property is satisfied, see Definition 6, Appendix B). A lemma on the I-diagnosability property states that lack of attack A_i-indeterminate cycles (Definition 14, Appendix C) having an embedded indicator transition (Definition 13, Appendix C) guarantees I-diagnosability. Global I-diagnosis is guaranteed due to local I-diagnosis across any of the IDSs [169].

Figure 4.7 shows the constructed diagnoser for our DES model H_i, considered in Figure 4.7. The working mechanism of our diagnoser is summarised here by showing one or more executions of sequences of measured events (transitions) as follows:

1. The initial state of the model H_i, $x1$, and states $x1'$ and $x1''$ reachable via unmeasurable attack transitions, $\tau 0'$ and $\tau 0''$, form the initial state of the diagnoser, $z1$.

2. Let $\Im_{z1} = \{\tau 1, \tau 1', \tau 1''\}$, i.e., the outgoing transitions from model states $\{x1, x1', x1''\} \in z1$. All the transitions in $\Im_{z1}$ are equivalent ((Definition 2, Appendix A)) and hence cannot be further subdivided and hence justifies O-transition $a1$. The O-state corresponding to the transition $a1$ is $z2 = \{x2, x2', x2''\}$.

3. Let $\Im_{z2_1} = \{\tau 2, \tau 2', \tau 2''\}$, i.e., the outgoing transitions from model states $\{x2, x2', x2''\} \in z2$. All of outgoing transitions in $\Im_{z2}$ are measurement equivalent belonging to one measurement equivalence class of transitions, hence cannot be further partitioned. Therefore, it justifies O-transition $a2$. The O-state reached corresponding to the transition $a2$ is $z1 = \{x1, x1', x1''\}$.

4. Let $\Im_{z2_2} = \{\tau 3, \tau 3', \tau 3''\}$, i.e., the outgoing transitions from model states $\{x2, x2', x2''\} \in z2$. All the transitions in $\Im_{z2_2}$ are equivalent and hence cannot be partitioned further, justifying O-transition $a3$. The O-state corresponding to the transition $a3$ is $z3 = \{x3, x3', x3''\}$. In a similar manner, the diagnoser states $\{z4, z5, z6, z7, z8\}$ can be constructed using the corresponding O-transitions $\{a4, a6, a7, a9, a10\}$. The principle can be safely extended.

5. From the definition, we can compute the $Attacker_k$ certain O-states and the Normal certain O-states. In our example, the $Attacker_1$ certain O-state may be computed

as $z10 = \{x9'\}$ since it exclusively consists of states only belonging to attacker A_1. Similarly, $Attacker_2$ certain O-state may be computed as $z11 = \{x9''\}$ and the normal certain O-state can be computed as $z9 = \{x1\}$.

4.3.6 An example of fragment duplication attacker node identification using DES Diagnoser

Suppose ${\Theta_i}^{macd} = 1$ and the following events occur chronologically in the monitored RPL-6LoWPAN due to packets received or sent from the IDS node: $FRGHD$, $BRACK$, FRG, FRG, FRG, RTO, $FRGHD$, $BRACK$, FRG, FRG, FRG, $DPNACK$.

The diagnoser starts from the O-state $z1$ and on occurrence of the $FRGHD$ event, the diagnoser moves to O-state $z2$ via O-transition $a1$. The event occurs when a datagram fragment header is sent. Now, the transition $a1$ might have been taken by the diagnoser due to the occurrence of any of the H_i-transitions, $\tau 1$, $\tau 1'$ or $\tau 1''$. Since the transitions $\tau 1$, $\tau 1'$ and $\tau 1''$ are measurement equivalent, it cannot be certainly said at this point if an attack has occurred. A buffer reservation ACK is received next due to which the event $BRACK$ is passed to the diagnoser. The latter moves to O-state $z3$ via O-transition $a2$. Now, a fragment of FRAG1 header type is sent to the receiver and the event FRG occurs. Consequently, the diagnoser reaches O-state $z4$ via $a4$. Subsequent fragments of the same IPv6 datagram on being sent, the diagnoser stays in the same O-state $z4$ due to the self-loop transition $a5$. At this point, if no ACK is received before timeout, the event RTO is passed to the diagnoser. Since the retransmission limit, ${\Theta_i}^{macd}$ is set to 1 at the model start, and the model variable $N_i = 1$, hence the diagnoser moves to O-state $z5$ via the O-transition $a6$. The transition $a6$ is taken in the diagnoser due to the occurrence of any of the H_i-transitions, $\tau 6$, $\tau 6'$ or $\tau 6''$ which are measurement equivalent. The header fragment FRAG1 of a datagram probe is sent via an alternate next-hop node due to which the event $FRGHD$ occurs and the diagnoser moves to O-state $z6$. Analogously, the diagnoser O-state evolves to $z8$ via the O-transitions $a9$ and $a10$ when a buffer ACK is received and fragments are subsequently sent. Eventually, when the $DPNACK$ event occurs, suppose the diagnoser moves from O-state $z8$ to O-state $z10 = \{x9'\}$ via O-transition $a13$ due to the model transition $\tau 12'$. Since the O-state $z10$ reached by the diagnoser is an $Attacker_1$-certain O-state, it is ascertained that the system is under attack condition due to attacker node A_1.

Moreover, since there are no A_k-indeterminate cycles [46, 45], along all paths of the DES diagnoser, an unique malicious node A_k, when present, can be identified correctly. On each such occasion when the diagnoser reaches an $Attacker_k$-certain state due to an event trace, an alert is generated.

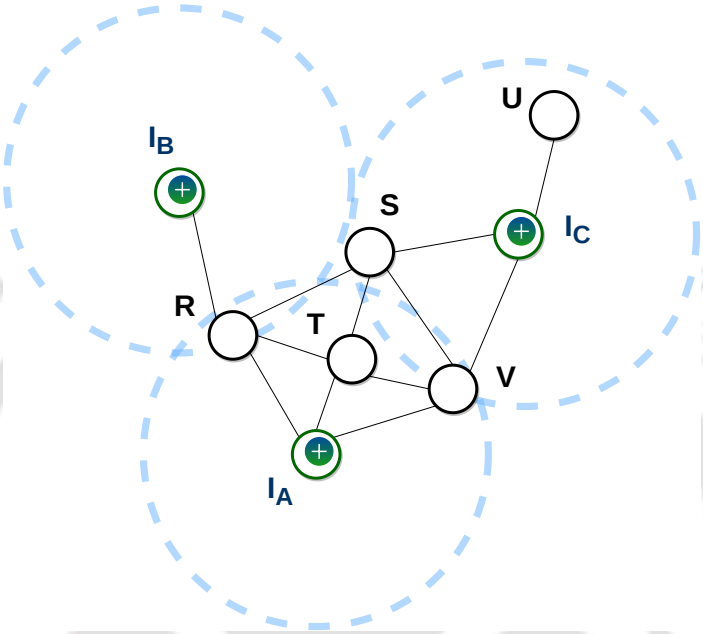


Figure 4.8: An arrangement of IDS and non-IDS 6LoWPAN nodes

4.3.7 Correctness and Completeness

DES modeling aids in formalizing a system to check correctness and completeness [44]. We demonstrate correctness and completeness of our proposed IDS here, by taking into consideration all possible cases of fragment duplication attack. We show that an attacker node is correctly identified. We use the RPL-6LoWPAN instance shown in Figure 4.3 for our proof, where $\mathcal{I}_A$, $\mathcal{I}_B$ and $\mathcal{I}_C$ are the IDS nodes. Among R, S, T, U and V, let us suppose R and S are the two suspected attack nodes and can be related to nodes A_1 and A_2 used in our DES model. Since there are no $Attacker_k$-indeterminate cycles in the diagnoser O shown in Figure 4.7, therefore the diagnosability condition is satisfied [45]. This means that location of an attacker A_k in the monitored network is always diagnosable.

We now prove the completeness by justifying why all attack cases can be detected from the traces in H_i. We show that R and V are always correctly identified as attack node

when the corresponding attacker certain state is reached in the diagnoser. An arrangement of nodes as shown in Figure 4.8 is considered, where an attack might have been launched by node R, S or V. T is not considered an attacker for demonstration of the following proof. For simplicity we assume $\Theta_i{}^{macd} = 1$ and datagram $\mathcal{D}$ has only 2 fragments, $F1$ and $F2$. Suppose $\mathcal{I}_A$ sends a datagram $\mathcal{D}$ to U. While $\mathcal{D}$ is reassembled and forwarded at the intermediate hops, namely, T, S, and $\mathcal{I}_C$, the packet is eavesdropped. As a result, overlapping fragments are reported to $\mathcal{I}_A$ by $\mathcal{I}_C$. All the attack cases where R and V are attack nodes are enumerated below. Identification of S can be carried out in the same procedure as demonstrated.

Case I: R, S, V are attack nodes In this case, we consider all three, R, S and V, are attack nodes. Suppose, a datagram $\mathcal{D}$ is to be sent from $\mathcal{I}_A$ to U. Accordingly, it is forwarded to the next-hop node T. In the diagnoser O, this can be understood as O-states evolving from $z1$ to $z4$ due to O-transitions $a1$, $a3$, $a4$, and $a5$. It may be noted that we do not consider explicit packet drops for our demonstration, unless dropped due to an attack activity. So T is forwarded to S in the next-hop. Since S and V are both attack nodes and S lies in the range of V, so overlapping fragments are present during reassembly at S due to V or R. Accordingly, since S is also an attack node, hence D with overlapping fragments are forwarded to $\mathcal{I}_C$. Now, $FRAG\_RCV\_HANDLER()$ at $\mathcal{I}_C$ on its behalf sends an intimation to the source node, $\mathcal{I}_A$. The $FRAG\_SND\_HANDLER()$ at $\mathcal{I}_A$ on receipt of the intimation packet generates event $DPNACK$ which is passed to the diagnoser. Consequently, it updates its O-state and moves to $z5$ due to O-transitions $a6$. Next, a datagram probe $\mathcal{D}'$ with overlapping fragment, let $F1$, is sent via R to $\mathcal{I}_B$. Also note that R lies in the range of S and outside the radio range of V. Now, the current state of the diagnoser reaches O-state $z8$. Since R is an attack node, it refrains from discarding the packet and forwards it to $\mathcal{I}_B$. In turn, $\mathcal{I}_B$ comes to know about the fragment duplication and intimates $\mathcal{I}_A$. Again, the $FRAG\_SND\_HANDLER()$ at $\mathcal{I}_A$ on receipt of this intimation packet generates event $DPNACK$. When this event is passed to the diagnoser, O moves from $z8$ to $z10$ which is $Attacker_1$-certain O-state (A_1 can be related to R, and A_2 to V). This is because the fabricated probe datagram was sent via next-hop node, R. Therefore, attack node R is correctly identified.

Case II: R, S are attack nodes and V is non-attack node Since R and S are the

attack nodes here, the attack characteristics are similar to the situation described above with the only difference that $\mathcal{D}$ is infiltrated due to R or S, a forwarding node, itself. Here again, the diagnoser reaches $Attacker_1$-certain O-state $z10$ and R is correctly identified as an attack node.

Case III: R, V are attack nodes and S is non-attack node Considering R and V are the only attack nodes, attack characteristics are similar except that D is discarded by S when duplicated by R or V. Hence, instead of a duplicate fragment intimation, event RTO is invoked by the $FRAG\_SND\_HANDLER()$ at $\mathcal{I}_A$ due to clock timer exceeding retransmission timeout Δ_i while waiting on an acknowledgement. The diagnoser consequently moves to state $z5$ due to the shared event, RTO, and since we assume ${\Theta_i}^{macd} = 1$. Again $\mathcal{D}'$ is forwarded to next-hop R. R being an attacker refrains from discarding the packet and forwards it to $\mathcal{I}_B$ in the manner similar to descriptions above. The diagnoser reaches $Attacker_1$-certain O-state $z10$ confirming that R is an attack node, correctly.

Case IV: R is an attack node while S, V are non-attack nodes The situation here is again analogous to the situation described above. When $\mathcal{D}$ and $\mathcal{D}'$ are both sent, the network event occurrences are all similar here except that R is the only eavesdropping attacker in both of the scenario. The diagnoser reaches $z10$ here as well and attack node R is correctly identified.

Case V: S, V are attack nodes and R is non-attack node Since S and V are the attack nodes here, in the first phase an intimation is received from $\mathcal{I}_C$ and the diagnoser reaches O-state $z5$. Subsequently, $\mathcal{D}'$ is sent by the $FRAG\_SND\_HANDLER()$ (on behalf of $\mathcal{I}_A$) to next-hop node R. The datagram is eavesdropped and S supposedly times the attack and sends a overlapping fragment, $F2$. Diagnoser O consequently moves to O-state $z8$. R notices a duplication during reassembly and discards the packet, since it is a normal node. On retransmission period clocked over, the event RTO is invoked and the diagnoser reaches $z5$. Now, the datagram $\mathcal{D}'$ is sent to next-hop node V with destination address of $\mathcal{I}_C$ in FRAG1. Since V is an attack node, it follows the same procedure as described for the case of R and S as attacker in Case I. Consequently, due to events generated, the diagnoser reaches O-state $z11$ which is an $Attacker_2$-certain O-state. V (is same as A_2) is therefore correctly identified.

Case VI: S is an attack node while R, V are non-attack nodes The situation

here is similar to the situation described above with the only difference that V is not an attack node this time. Therefore, at $z5$, when datagram $\mathcal{D}'$ is sent via next-hop node V, V discards it and consequently the diagnoser reaches O-state $z5$ on c exceeding Δ_i. Since R and V are not attack nodes, no decision is accordingly arrived at the diagnoser.

Case VII: V is an attack node while R, S are non-attack nodes The situation is similar to the one described in Case IV with V playing a similar role as R. Accordingly, the diagnoser reaches O-state $z11$ and V is correctly identified as an attack node.

So, all the possible cases of attack by R and V are analyzed. The diagnoser correctly reports the network condition by identifying the corresponding attacker type states, for each case.

4.4 Performance evaluation

We have executed three experiments in Contiki cooja [155] and one on a real testbed at FIT IoT-LAB [156]. Cooja and FIT IoT-LAB have been already introduced in Section 3.4. We have used two types of topology $T1$ and $T2$, as shown in Figure 4.9. In topology 1 ($T1$), IoT nodes are distributed very densely, whereas in topology 2 ($T2$) nodes are distributed sparsely. The hop count is more in $T2$ as compared to $T1$. The simulation/ experimental parameters of Contiki cooja and FIT IoT-LAB are presented in Table 4.4. To examine the experimental performance of our proposed solution, three types of scenarios are designed as part of the experimental setup: 1) Non-FDA scenario, 2) FDA scenario, and 3) FDA with the proposed solution. The comprehensive analysis of all the scenarios are given below.

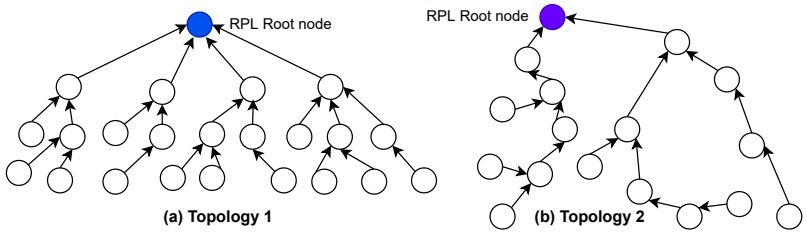


Figure 4.9: Topology considered for testbed and simulation experiments

Table 4.4: Simulation and real-time test-bed parameters

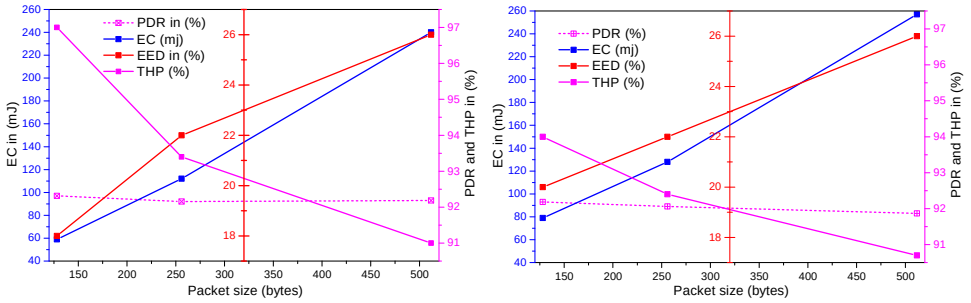
Parameter name	Simulation	Real time testbed
Operating system	Contiki 3.0, Contiki 4.5	Contiki-NG
Simulator/Testbed	Cooja Cooja	FIT IoT-LAB
Network size	32, 64, 128 nodes	
Radio Environment	UDGM	
Node Type	Tmote Sky	IoT-Lab A8
Routing Protocol	RPL	RPL Lite
RPL Objective Function	MRHOF - ETX, OF0	MRHOF - ETX
MAC/adaptation layer	Contiki MAC/6LoWPAN	
Transmitter output power	(dBm) 0 to -25	
Receiver sensitivity	(dBm) -94	
Radio frequency	2.4 GHz	
Attack Modeled	Fragment duplication attack (FDA)	
Experiment Duration	60 minutes	

4.4.1 Experiment 1: Non-FDA scenario

This experiment is conducted with varying number of IoT nodes (i.e., 32, 64, and 128 nodes) and it is noted as to how the IoT network parameter performances change. Similar to the non-rank attack scenario experiment outlined in Section 3.4.1, *Wireshark* and *power trace* tools are used for network traffic analysis and A8-type nodes with Grenoble areas are used in real testbed. The adopted parameters while running the testbed experiments are specified in Table 4.4. Figure 4.10 shows packet delivery ratio (PDR), average energy consumption (AEC), average end to end delay (AEED), and throughput (THP) with the respective different packet size (i.e., 128 byte, 256 byte, and 512 byte). Figure 4.10 (a) and (b) show experimental analysis with topology 1 (dense) topology 2 (sparse) in Contiki cooja and FIT IoT-LAB. This analysis shows PDR ($83.82\% - 98.30\%$), AEC ($68mJ - 298.3mJ$), AEED ($19\% - 26\%$) and THP is ($84.23\% - 97.63\%$) in a non-FDA scenario. This is due to the reason that in non-FDA scenario, packet delay, packet reordering, packet alteration, and dropping of legitimate packets are a minimum. Hence in a FDA scenario, this analysis shows that average THP, EED, EC and PDR is moderately good.

Experiment 2: FDA scenario

The FDA scenario is executed with IoT nodes ranging from 32 to 128 nodes, i.e., 32, 64, and 128. We incorporate attack nodes during our experiments. These nodes generate duplicate packet fragments and alter packets. We examine the PDR, EED, EC, and



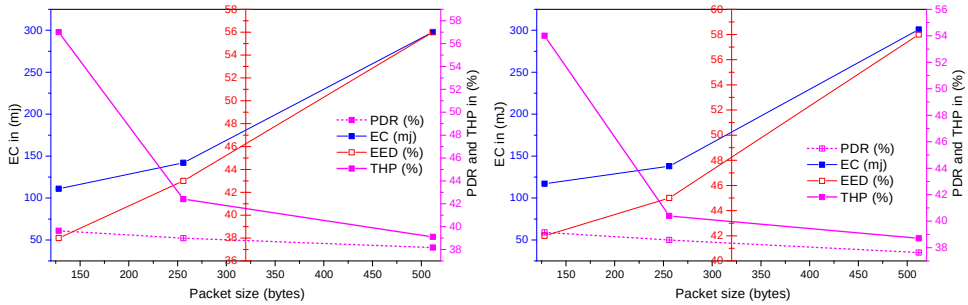
(a) Contiki Cooja (b) FIT-IoT-Lab

Figure 4.10: Analysis of average PDR, EC, EED, and THP over different packet size with 64 node (non-FDA scenario)

THP with different malicious nodes and by varying node density. Traffic generated from FDA scenario is analyzed using collect view modules for simulation analysis purposes. Similarly, we use `Sysstat` [157] and `iperf` tool [158] for real testbed analysis. Figure 4.11 (a) exhibits a considerable degradation in PDR, EC, EED, and THP, i.e., $(34.3\% - 52.2\%)$, $(114.3mJ - 308.9mJ)$, $(44.3\% - 55.9\%)$ and $(33.79\% - 51.87\%)$, respectively in Contiki cooja simulations. Figure 4.11 (b) also shows similar types of outcomes i.e., $(29.8\% - 48.5\%)$, $(124.3mJ - 321.9mJ)$, $(49.8\% - 59.3\%)$ and $(37.52\% - 54.27\%)$ in FIT IoT-LAB. Figures 4.11 (a) and (b) show the PDR and THP graph to be going down significantly. The average PDR and THP value are reduced in the FDA scenario, both in simulation and real testbed. All experiments show huge AEC with reduced PDR, and THP. The reason is that the FDA is launched using duplicate fragment, packet drop, and packet alterations. Table 4.5 presents the performance parameters (i.e., PDR, EC, EED, and THP) that significantly affect the IoT ecosystem performance. Based on the tabulated results presented in Table 4.5, it can be observed that increased IoT node count incurs a huge number of duplicate fragments. Hence it consumes extra network energy, due to usage, and node power. Table 4.5 also highlights the performance analysis with three different sizes of a packet and different node densities.

4.4.2 Experiment 3: FDA scenario with proposed solution

In this scenario, experiment 2 is executed with a proposed solution in simulation and a real testbed, several times. We consider 32, 64, and 128 IoT nodes, and experiments run for 3600 Sec with the proposed approach. The performance analysis of all experiments is based on the following metrics:



(a) Contiki Cooja (b) FIT-IoT-Lab

Figure 4.11: Analysis of average PDR, EC, EED, and THP over different packet size with 64 node (During FDA scenario)

- *Packet delivery ratio (PDR)*: It is the ratio between the number of packets received P_{recvi} at the receiver node and the total number of packets sent P_{send} from sender node. The PDR is estimated using the following Eq. (4.4).

$$PDR = \frac{P_{recvi}}{P_{send}} \times 100 \tag{4.4}$$

- *Average energy consumption (AEC)*: In general, it is calculated by taking the total amount of energy required for delivering packets during an experiment and dividing it by the number of packets delivered P_D. The energy consumption estimated using Eqn (4.5).

$$\begin{aligned} Energy(mJ) = [(CPU \times 1.8mA) + (LPM \times 0.0545mA) \\ + (Transmit \times 19.5mA) \\ + (Receive \times 21.8mA)] \times 3V/Rtimer \times 8 \end{aligned} \tag{4.5}$$

$$AEC\ (mJ) = \frac{Energy\,(mJ)}{P_D} \tag{4.6}$$

- *Average end to end delay (AEED)*: It defines the average time it takes to deliver packets under an experiment. The formula to calculate it is as follows.

$$AEED = \frac{Total\ EED}{Total\ Packets\ Delivered} \tag{4.7}$$

- *Throughput (THP)*: This metric measures the ratio of the network throughput in the presence of FDAs with respect to the observed network throughput in the absence of FDAs.

$$THP = \frac{throughput\ in\ FDAs\ scenario}{throughput\ in\ normal\ scenario} \times 100 \tag{4.8}$$

- *Accuracy (ACC)*: It denotes the percentage of correctly classified flows as true attack

Table 4.5: PDR, EC,EED and THP values for IoT ecosystem (During attack)

	PDR (%)			EC (mJ)			EED (%)			THP (%)		
Packet size 128 bytes												
Packet Size	32N	64N	128N	32N	64N	128N	32N	64N	128N	32N	64N	128N
T1	69.8	64.2	60.9	98.2	118	199	19.5	22.7	28.6	61.8	58.4	50.7
T2	73.5	68.9	64.1	110	132	221	22.4	25.9	32.7	64.6	62.2	55.5
Packet size 256 bytes												
T1	62.4	60.6	51.7	109	139	227	23.6	27.9	33.8	68.3	61.5	44.1
T2	67.3	63.2	58.4	125	176	289	27.3	38.3	44.7	76.9	69	48.7
Packet size 512 bytes												
T1	56.2	38.9	37.3	189	289	477	33.5	54.3	58.2	41.9	39.7	33.3
T2	59.2	41.7	37.9	209	302	498	37.4	59.5	61.3	48.5	44.2	37.9

or true legitimate flows with respect to total number of flows. Accuracy is given by:

$$ACC = \frac{p+r}{p+q+r+s} \times 100 \tag{4.9}$$

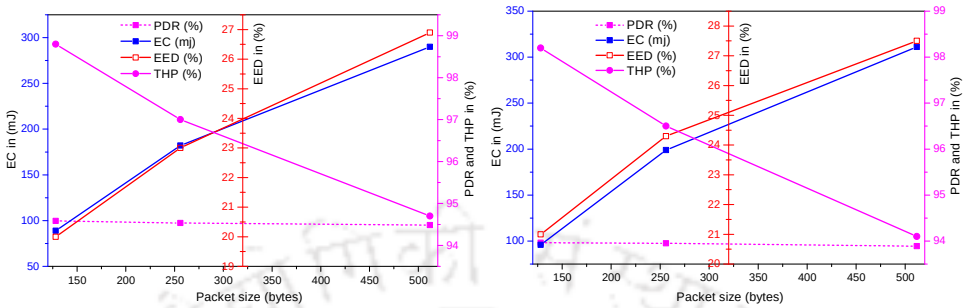
where, p = *Attacker nodes identified accurately* q = *Attacker nodes identified wrongly* r = *Genuine nodes identified accurately* s = *Genuine nodes identified wrongly.*

- *Memory Consumption (MEMC)*: It shows the percentage of memory utilization of the IoT devices to run proposed solution, throughout the experimentation.

- *Attack Detection Time*: This performance metric shows the time taken to identify the attack.

Figures 4.12 (a) and (b) show the performance analysis of our proposed solution during the simulation and real testbed, respectively. Figures 4.12 (a) and (b) show outstanding results like PDR ($94.6\% - 98.7\%$) , AEC ($112\ mj - 314\ mj$), AEED ($19.3\% - 26.4\%$), and THP ($96.5\% - 98.7\%$). In Figures 4.12 (a) and (b), the PDR and THP graph can be seen to significantly progressing upward. The average PDR and THP value were improved by ($42.6\% - 56.2\%$) and ($38.8\% - 48.4\%$), respectively. Figures 4.13 (a) and (b) also show ACC and FDA detection time graphs with topologies, $T1$ and $T2$. Table 4.6 presents the performance analysis during the recursive execution of our proposed solution in Contiki cooja and FIT-IoT LAB, by varying the count of IoT nodes and with different sizes of packets. Based on Table 4.6 outcome, we noticed that different size of network and packet sizes (i.e., 128, 256, and 512) give the distinct value of network performance metrics. Based on Tables 4.5 and 4.6, we observed that our solution takes the minimum amount of AEC

and AEED. The reason is that the proposed solution with probing identifies the malicious node and minimizes duplicate packet fragments, and modifies genuine packets.



(a) Contiki Cooja (b) FIT-IoT-Lab

Figure 4.12: Analysis of average PDR, EC, EED, and THP over different packet size with 64 node (After solution implementation)

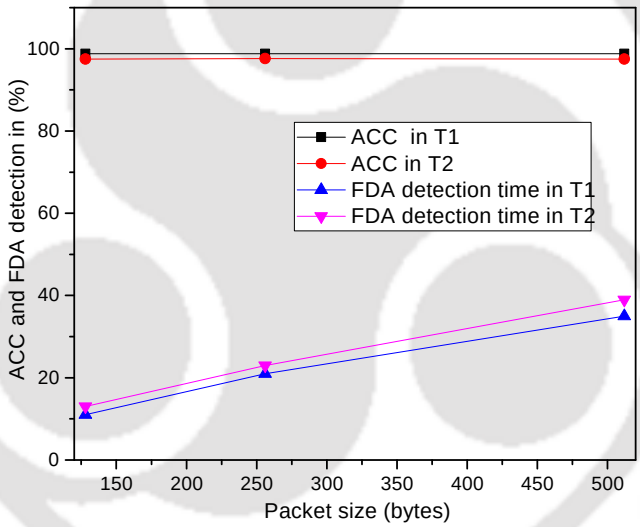


Figure 4.13: Analysis of ACC and FDA over different packet size with 64 node (Topology 1 and Topology 2 used)

4.4.3 Comparison with the existing works

This subsection presents the comparative analysis of the proposed FDA detection approach with state-of-the-art solutions. To achieve fair results, we conducted simulation and real testbed experiments. All experiments were repeated many times to create tight confidence intervals. In general, the results did not show much variance between simulation and real testbed. To measure the performance metrics, we have used *collect view modules*, *Sysstat*, and *iperf tool* for simulation and real testbed analysis. We consider seven different

Table 4.6: PDR, EC,EED and THP values for IoT ecosystem (After solution Implementation)

	PDR (%)			EC (mJ)			EED (%)			THP (%)		
Packet size 128 bytes												
Packet Size	32N	64N	128N	32N	64N	128N	32N	64N	128N	32N	64N	128N
T1	69.8	64.2	60.9	98.2	118	199	19.5	22.7	28.6	61.8	58.4	50.7
T2	73.5	68.9	64.1	110	132	221	22.4	25.9	32.7	64.6	62.2	55.5
Packet size 256 bytes												
T1	62.4	60.6	51.7	109	139	227	23.6	27.9	33.8	68.3	61.5	44.1
T2	67.3	63.2	58.4	125	176	289	27.3	38.3	44.7	76.9	69	48.7
Packet size 512 bytes												
T1	56.2	38.9	37.3	189	289	477	33.5	54.3	58.2	41.9	39.7	33.3
T2	59.2	41.7	37.9	209	302	498	37.4	59.5	61.3	48.5	44.2	37.9

Table 4.7: Comparison of the proposed scheme with the closely related works

References	SIM/TB	PDR	AEC	AEED	THP	ACC	MEMC (RAM/ROM)	SCAL	ADT (msec.)
Hummen [1]	Contiki cooja	98%	289.5 mJ	N/A	N/A	N/A	41750/9502 byte	No	(500-640) msec.
SecuPan[2]	Contiki cooja	97.6%	280 mJ	80 %	96.4%	N/A	N/A	Yes	(130-160) msec.
Nikarvan[3]	Testbed	N/A	251.4 mJ	N/A	N/A	N/A	46381/10211 byte	No	(3053-3330) msec.
[4] Proposed Approach	Contiki cooja Testbed	99.2%	139 mJ	92%	97.8%	99.8%	32568/7845 byte	Yes	(98-128) msec.

SIM: Simulation, TB: Testbed, SCAL: Scalability, ADT: Attack detection time, N/A:Not available

performance metrics, namely, PDR, AEC, AEED, THP, ACC, and ADT. State-of-the-art methods [112] [34], consume enormous AEC, and take AEED. Hence these solutions are very hard to implement in a constrained IoT ecosystem. Some closely related approaches [37] detect FDA, but the deficiency of these approaches are non-scalability and increased ADT. The performed comparative analysis is shown in Table 4.7. It can be observed that our security solution is energy-aware, highly accurate, and yet scalable.

4.5 Conclusion

A novel 6LoWPAN fragment duplication attacker identification scheme is presented. The scheme proposed is decentralized and uses an intelligent probing technique and I-DES based IDS. Traditional I-DES based IDS have been augmented, such that attacker type is also diagnosed. Active probe packets are used judiciously to differentiate an attack node from a normal node, which is normally lacking otherwise. Using this scheme, location of attack node can be identified accurately. Each I-diagnoser serves as a local IDS engine that generates an alert when an attack node is identified. Global I-diagnosis is achieved from local I-diagnosers. We perform node elimination using kill switch such that fresh avenues of

attack from same node is nullified. The correctness and completeness of our approach is also proved.

The performance analysis of our proposed scheme in simulation and real testbed considers both attack and non-attack behavior patterns, with a sufficiently large number of 6LoWPAN nodes. The average throughput, accuracy, and response times in our proposed approach are 97.8%, 99.8%, and $98 - 128$ *msec.*, respectively. The observed results show our approach is energy-efficient than existing works. It is scalable, achieves minimum false positives, and higher accuracy with lower detection time.

The network and adaptation layer attacks have been mitigated successfully using centralized I-DES and decentralized I-DES frameworks. In the proposed I-diagnosability based DES frameworks, indicator events are used to identify the attack node. Suitable active probe packets are sent while the generated responses are validated to test the ingenuinity of network traffic. However, there exists a class of spoofing attacks that exploit the application layer IoT protocol, CoAP, to make target either server or client nodes vulnerable, and cause Denial of Service (DoS). For CoAP request/response spoofing attacks, active probe packets are not sufficient for deterministic diagnosis, since spoofed CoAP probe request or response packets resemble the original ones. Moreover, attacker may be on-path which further complicates detection, since the original packet may be dropped. Hence, there exists paths in the I-DES model which lead to attack uncertain states, making I-diagnosis frameworks incapable to guarantee identification. To deal with such scenario, decentralized I^2-diagnosability based DES frameworks are adopted and extended to detect and identify CoAP request/response spoofing attacker. Our proposed scheme utilises I^2-DES based IDS, empowering events and indicator events. The probe packets injected are spoofed CoAP request/response packets. Attacker diagnosis is tested only through those paths that contain an empowering event and indicator event after an attack has been launched. The proposed identification scheme is successfully verified in simulation with variable number of nodes. The results show effectiveness of our technique with achieved accuracy of more than 99% and response times as low as 6 seconds.

CHAPTER

5

Mitigation of CoAP request/response spoofing attacks

The Internet of Things (IoT) technology has revolutionised the outlook of connected devices and IP-connected smart objects. A significant portion of such interconnected heterogeneous devices are being extensively deployed to perform mission-critical tasks in areas such as the health sector, energy management, industrial process control systems, etc. [4] For uninterrupted services over the internet, a reliable end-to-end communication is demanding for the family of constrained devices. The IoT protocol stack has been designed and adopted with an aim of achieving such a standard [10].

IETF has standardized CoAP as an Application layer web transfer protocol to provision for usage of Internet services in M2M applications constituting resource-constrained devices across lossy, low-power networks [114]. Specially designed for provisioning web interoperability, applications that make use of constrained sensing or actuating devices, having limited power, memory and processing capabilities, render them fragile to various external threats and DoS attacks. CoAP uses UDP as the transport protocol, which is unreliable and is devoid of handshaking mechanism between client and server. CoAP is susceptible to various attacks like Cross-Protocol attacks, Amplification attacks, Man-in-The-Middle (MiTM), etc. [115] Amongst these, IP address spoofed DoS attacks can be launched at ease and is the focus of our work here. Moreover, it can help mount stealthier attacks when used in combination, such as amplification attacks. Availability of devices and accessibility to services can be immensely compromised by a malicious endpoint that is exercising *read* and *write* access.

Research works reported in the literature have undertaken numerous countermeasures to effectively mitigate DoS attacks in general. Mostly, the adopted approaches either employ host-based, router-based or hybrid techniques [116]. Adaptive solutions employing frequency based approaches have been successfully used to detect mixed-rate IP spoofed DDoS attacks [117]. Off-path response spoofing attacks in TCP and DNS have been analyzed thoroughly in the literature [118, 119]. A countermeasure based on source-port randomization has been proposed for an off-path attack on TCP [120]. In, they launch an off-path attack by analyzing the remote server access support in CoAP and a request spoofing vulnerability. Request spoofing is also shown to be mitigated using machine learning based approaches [121].

The proposed schemes, except a few, do not address mitigation techniques for request spoofing attack in LoWPAN applications using CoAP. None of the reported works have attempted identification of malicious or misconfigured endpoints in the past. Owing to the similarity between diagnosis of faults and detection of network attacks in DES, we employ DES based IDS [122, 48]. However, I-diganosability frameworks are not sufficient to generate responses such that an attacker can be identified. We employ I^2-Diagnosability based DES framework that uses empowering events along with indicator transitions.

Enumerated below are our contributions in this work:

1. We design a novel request/response spoofing attack detection and attacker identification scheme in LoWPAN applications employing CoAP without DTLS support. Our scheme utilises decentralized DES based IDS, empowering events and indicator events.

2. We extend the power of traditional I^2-DES based IDS with attack type modeling for identification of malicious node.

3. We prove the correctness and completeness of our approach by enumerating all the attack cases.

4. The results demonstrate an energy efficient and scalable solution at considerably lesser response times compared to the related state-of-art solutions.

5.1 Background

5.1.1 CoAP

Similar to HyperText Transport Protocol (HTTP), CoAP realizes a subset of REpresentational State Transfer (REST) architecture that is meant to be suitable for constrained systems. It defines a request/response model for communicating endpoints while supporting POST, GET, PUT and DELETE methods. CoAP defines four message types, namely, confirmable messages (CON), Non-confirmable messages (NON), Acknowledgment messages (ACK) and Reset messages (RST).

5.1.2 CoAP Message Format

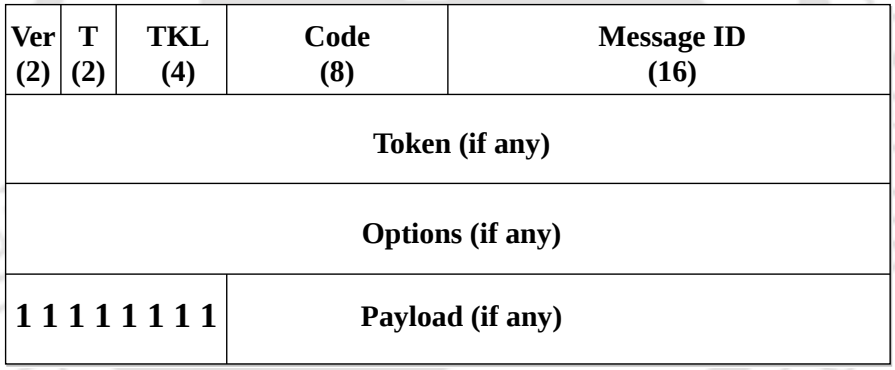


Figure 5.1: CoAP message format

The CoAP message format is shown in Figure 5.1. Messages in CoAP are binary encoded with a fixed size header of 4 bytes. The variable length Token value can range between 0 and 8 bytes. The header fields and their meanings are: **Version (Ver):** A 2-bit integer (unsigned) binary value that indicates the CoAP version in use. **Type (T):** It is used to specify type of messages (00) Confirmable (01) Non-confirmable (10) Acknowledgement (11) Reset. **Code:** An integer of 8-bits of the format "c.dd". 0.00 indicates an empty message. **Message ID:** An integer value of 16-bits used in duplicate detection and message type matching. **Token Length (TKL):** This 4-bit integer indicates length of the Token field (variable).

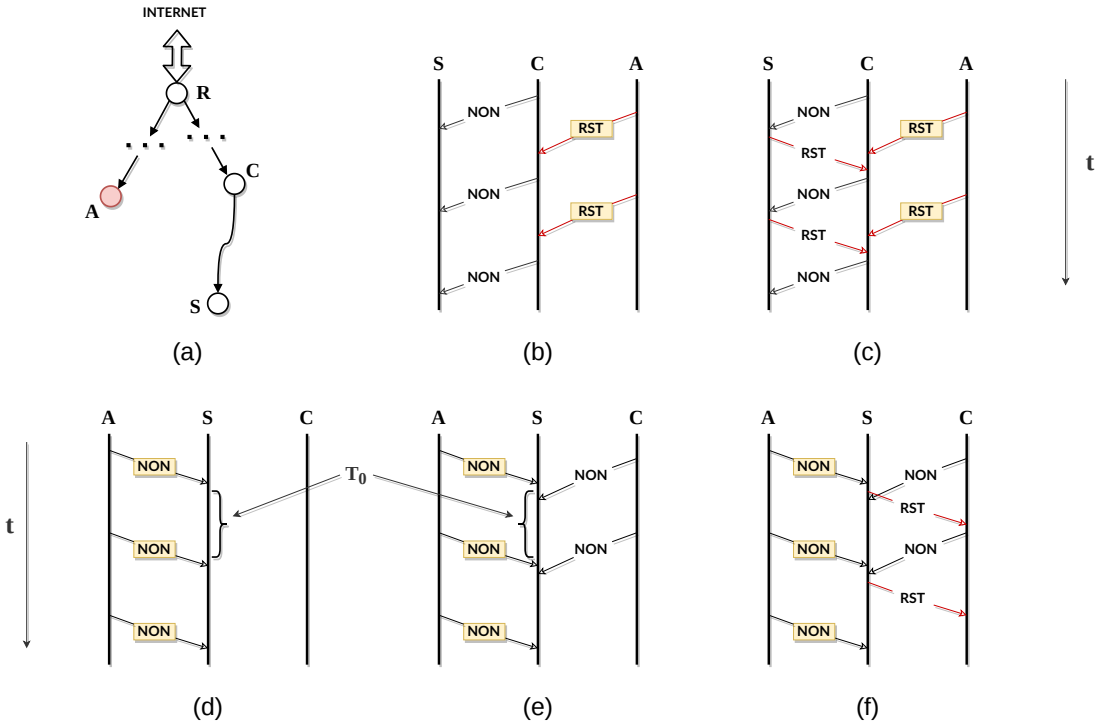


Figure 5.2: CoAP address spoofed attack timeline

5.1.3 CoAP - IP address spoofing attack

Though CoAP supports stop-and-wait mechanism for retransmission and NON/CON duplicate detection, there are occasions which can be evasive and require stricter measures to ensure security. The attack can be launched by an on-path attacker or an off-path attacker. An on-path attacker lies on the path connecting the client to the server node while an off-path attacker lies somewhere except the legitimate CoAP message delivery path. We discuss the off-path attack here. Irrespective of any cryptographic implementation,the attacker mimics the target node using address spoofing and sends identical forged packets. RFC 7252 discusses the various ways that response and multicast request spoofing may be performed. We here discuss the network event dynamics pertaining to non-confirmable messages. Figure 5.2(a) shows the network of resource constrained devices with an off-path internal attacker, A, a client node C, server node S and the root node (or, 6BR), R.

Response spoofing

The timeline of events arising from response spoofing are shown in Figure 5.2(b) and Figure 5.2(c). In the first case, C legitimately sends NON message each time to S. A times the attack and sends a spoofed RST message to C, thus, mimicing the server. C is deceived and each time retransmits the packet essentially resulting in DoS. In the second case, due to rightly received illegitimate packet, S sends back a RST message to C. Since A has already conveyed a RST message to C, duplicate detection capabilities of CoAP can help identify the forgery.

Request spoofing

The request spoofing attack timeline is shown through the Figures 5.2 (d)-(f). In the first instance, A keeps on sending NON messages using C's identity, illegitimately, and floods the server. The second case shows a fake NON message from A is delivered each time before C delivers one to S. The overhead is same as the previous scenario, but C never gets to know about the attack. In the third case, a RST is received by C due to the fake message sent from A behaving to be C.

5.2 Proposed Scheme

In this section our countermeasure scheme against CoAP request/response spoofing attack is presented. We introduce I^2-DES based IDS followed by an overview of the detection methodology using our proposed IDS. We then discuss the techniques used and algorithms devised to identify the attacker. The construction of DES normal and DES attack models and the diagnoser are discussed next. The correctness and completeness proof is presented subsequently.

5.2.1 I^2-DES based IDS

We introduce I-DES based IDS in Section 3.3.1. In I-DES based IDS an I-diagnoser is used as the IDS engine, constructed from the knowledge of normal and attack type I-DES models. Consequently, attacker location is identified only through the states that lie on the path after an indicator event has occurred. To summarize, by using I-DES based IDS,

and given all possible attack instances, it can be ascertained if an attack can always be exclusively identified, correctly and completely. However, there exists avenues when using I-DES based IDS does not suffice in attack detection or attacker location identification in spite of indicator events. Correspondingly, to overcome such limitations, I^2-DES based IDS is proposed in this work to identify an attack node location. Analogously, we augment traditional I (Induced)-I-DES based IDS with attack types in our work here. In I^2-DES based IDS, an I^2-diagnoser is used as our IDS engine. Empowering events are used along with indicator events for this DES based detection methodology. To summarize, by using our proposed I^2-DES based IDSs, and given all possible attack instances, it can be ascertained if an attack can always be exclusively identified, correctly and completely.

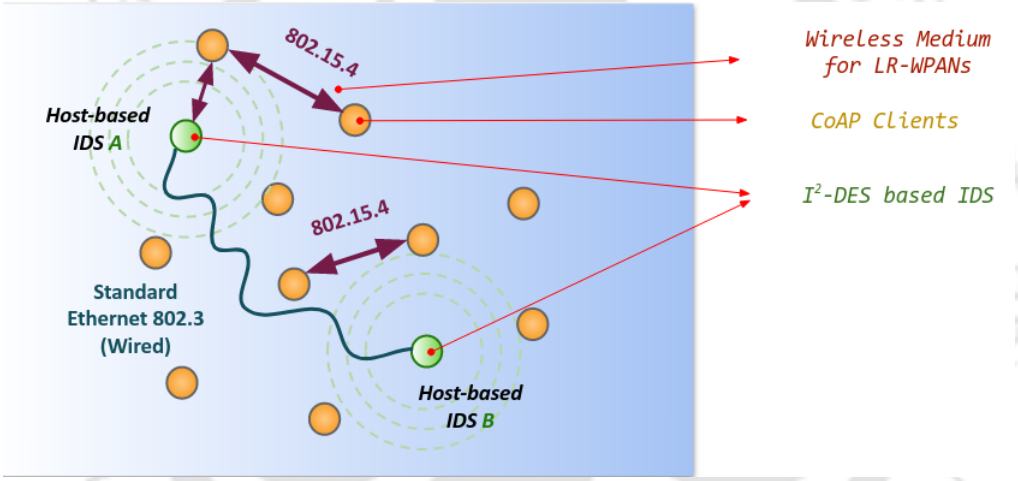


Figure 5.3: An example of IDS deployment

5.2.2 Design Overview

Main challenges in CoAP spoofing attack detection are (i) nodes and IoT network is resource constrained (ii) communication sequences across normal and attack conditions are same (iii) eavesdropping attacker may be off-path as well as on-path. Moreover an on-path attacker can delay or drop messages making attacker localisation cumbersome. To overcome these challenges, a decentralized approach is proposed in this work with a set of IDS, $\hat{I}$, that are distributed across the monitored network of IoT nodes. Each IDS, $I_i \in \hat{I}$, works analogously and independently. Besides the regular 802.15.4 communication, the set of IDS are also connected via a wired network for reserved communication (between IDS nodes only), e.g.,

intimation packets. Moreover, in our proposed countermeasure, the following is established as a general rule; a CoAP packet exchange between a pair of IDS nodes, via the wireless network, must always be preceded by a prior intimations via the wired network. The IDS make use of this wired intimation and an intelligent probing mechanism. The probe packets are basically crafted CoAP packets. Due to active probing, unique responses are generated that makes an attack node differentiable from a normal node. Table 1 lists the notations that will be used throughout. Figure 5.3 shows an example of IDS deployment in our scheme. For simplicity we show the wired and 802.15.4 connections with 2 IDS nodes.

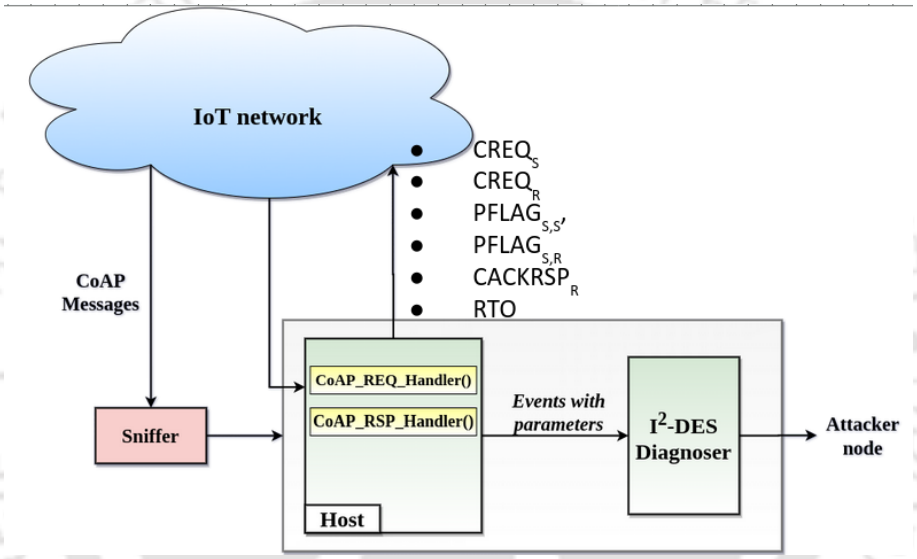


Figure 5.4: Architecture of the proposed IDS

Components of an IDS: The block diagram of proposed IDS with the basic components is shown in Figure 4.4 and are discussed here as follows:

- **Sniffer:** It captures control and data packets in the network while working in promiscuous mode. Relevant packets are sniffed and others are dropped. It then forwards the sniffed packets to the components, "$CoAP\_REQ\_HANDLER()$" and "$CoAP\_RSP\_HANDLER()$".

- $CoAP\_REQ\_HANDLER()$**:** The following handler module in IDS nodes are used to process client requests and plays the server role. It monitors communication data packets from IDS nodes and extracts information such as IP address, MAC address, etc. It intimates a source IDS node if the requests are proved. The working procedure

of this handler is described in Sections 5.2.4 and 5.2.5. A brief demonstration of their role can be seen in Figure 5.5

- $CoAP\_RSP\_HANDLER()$**:** The following handler module in IDS nodes are used to process server and plays the client role. Its prime responsibility is to extract vital information such as source IP address, MAC address from CoAP packets. It generates events such as $CREQ_S$, $CPREQ_S$, $FLAG_S$, $PFLAG_S$. The generated events are passed to the DES diagnoser component. The working procedure of this handler is described in Sections 5.2.4 and 5.2.5.

- **I^2-DES Diagnoser:** This component diganoses the attacker node and is implemented as a software module. Given the knowledge of the DES model specifications pertaining to normal and attack type conditions, the diagnoser can be constructed. $CoAP\_REQ\_HANDLER()$ and $CoAP\_RSP\_HANDLER()$ pass information regarding network events to the local I^2-diagnoser. Based on the event parameters that are shared, the diagnoser generates an alert on attack detection or identification of malicious nodes. The usage and construction of the diagnoser is described in Section 5.2.6.

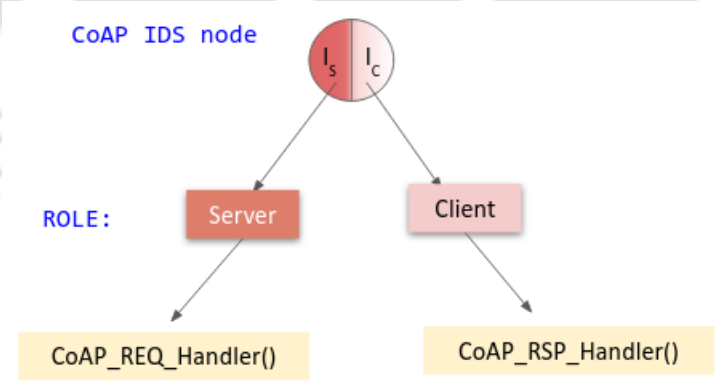


Figure 5.5: Role of IDS

Our identification procedure consists of four phases, namely, setup, intimation, active probing and diagnosis. We demonstrate the working methodology of our scheme here. A pictorial representation of the workflow can be seen in Figure 4. Initially, the set of IDS are setup using network traffic monitoring and logs relevant data. Otherwise the IDS node behaves as any other node in the network during this stage. Each IDS will locally store the

relevant detection parameters in array data structures, which are updated and maintained throughout. Each IDS node contains separate client and server module for role specific tasks. When a CoAP request or response spoofing attack is underway, an IDS server or client module, i.e., $CoAP\_REQ\_HANDLER()$ or $CoAP\_RSP\_HANDLER()$, respectively, infers it from the following situations: (i) more than one request or response is received from an IDS (ii) a request or response is received from an IDS without a wired intimation. On inference, the receiver IDS will send active probe packets. On behalf of the IDS, the handler module will send a probe intimation followed by a probe response or request packet with dummy application data to the source IDS. The generated response to the probe packets are validated by the I^2-DES diagnoser of the receiver IDS against the DES normal and attack specifications. The diagnoser generates an alert to the network admin if an attacker node is identified. This concludes the diagnosis phase. Essentially, the probe response characteristics separates an attack specific behaviour from the normal. The normal scenario corresponds to the receipt of only one probe response with matching application data or no response. The phases of our detection methodology are discussed below and is shown using Figure 5.6.

5.2.3 Setup

This phase requires administrator intervention for parametrized setup of IDS. For the purpose of Network Traffic Analysis (NTA), IDS monitors network traffic, collects the relevant data and stores them. Essential parameters are measured, the required information is stored in arrays and global constants are set during this stage. A variable t_{ifs} stores the inter-frame spacing with inputs from the lower layers. Each IDS maintains 2 arrays, $\hat{TRES}$ and $\hat{\Delta}$. A CoAP client request embeds information, namely host IP address, port number of the URI (Unified Resource Identifier), absolute path and URI-query, directed to intended resources. A resource URI is a string of the form "coap://<host>[:<port>]<path>[?<query>]". Some examples of resource URI are *coap://example.com:5682/ sensors/readings.xml*, *coap://FF05::FD:5682/ .well-known/core*, *coaps://127.0.0.1/.well-known/core?rt=core1.ps*, etc. Each IDS shares and maintains a record of resource URI and client nodes depending on the history of access requests. $\hat{TRES}$ is an array of linked list pointers used by an IDS to store resource URI information of its associated nodes. Each array element points to a memory address

where the list of strings, each corresponding to an unique resource URI for a particular associated node, can be accessed. An element of the array, $TRES^j$, therefore points to a list of resource uris for node j. $|T\hat{R}ES|$ signifies the size of $T\hat{R}ES$, i.e., the number of nodes in the monitored network that have an association with it. $\hat{\Delta}$ stores the transmission timeout values between pairs of IDS nodes. Given an IDS node I_i, Δ_j contains the average transmission time taken for a CoAP packet to reach IDS I_i from IDS I_j. Again, $|\hat{\Delta}|$, the size of $\hat{\Delta}$, is equal to one less than the number of IDS nodes.

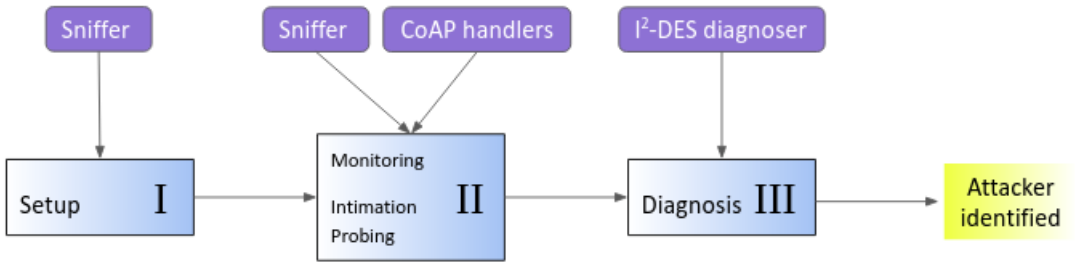


Figure 5.6: Flow of proposed scheme

5.2.4 Inference

We discuss how a CoAP request/response spoofing attack is inferred after the IDS has been setup. Correct inference of the attack is vital to the detection and identification of the malicious node. Our proposed IDSs consist of two handler components, $CoAP\_REQ\_HANDLER()$ and $CoAP\_RSP\_HANDLER()$. The former plays the role of a server module while $CoAP\_RSP\_HANDLER()$ plays the client role. The former handles the requests of clients on behalf of the host IDS, processes the information and sends the appropriate responses. They are responsible for request spoofing attack inference as well as detection in our scheme. The latter handles server responses and is responsible for response spoofed attack inference and detection. Furthermore, these modules share the generated events with the DES diagnoser. As already mentioned, these handler components issue an intimation via the wired network, i.e., $Channel2$ (referred as $Ch2$), before sending a CoAP packet via the wireless network, $Channel1$ (referred as $Ch1$), to any IDS node.

For a CoAP receiver IDS to infer an attack in our scheme, it is a prerequisite that the spoofed requests or responses are sent from an IDS source. For a receiver IDS, let us

suppose I_r, the following four cases occur.

- a request from a source IDS, let I_s, is received without intimation
- two successive identical requests are received from I_s within the distributed inter-frame spacing
- a response from I_s is received without intimation
- two successive identical responses are received from I_s within the distributed inter-frame spacing

Cases 1 and 2 are handled by $CoAP\_REQ\_HANDLER()$ on behalf of I_r. Cases 3 and 4 are handled by $CoAP\_RSP\_HANDLER()$ on behalf of I_r. Since a source IDS intimation is not received apriori, it can be safely inferred from Cases 1 and 3 that the request or responses received are spoofed. Also, since two consecutive identical requests or responses are received within t_{ifs}, it can be also concluded from Cases 2 and 4 that one of the request or response packet is spoofed. These are shown through the lines of the Algorithms 10 and 13.

5.2.5 Probing

In CoAP request/response spoofing attacks, due to lack of authentication mechanism and because of unreliability in transmission, it becomes challenging to verify the ingenuity of CoAP packets. In case only one spoofed request or response is received, there are no means to know sender authenticity. Moreover, devices are mostly resource constrained and verification of per packet originality is detrimental to network performance. Furthermore, even when more than two identical packets are received, consecutively, there are no mechanisms to judge as to which one is original. Packets therefore need to be retransmitted in such a scenario. Again, it does not hinder the attacker from launching fresh attacks. Therefore it becomes quintessential to identify the malicious node and also create distinguishable traffic. In our scheme, wired intimation helps infer an attack with surety. We use an intelligent active probing technique that helps issue differentiable responses which are passed to the diagnoser. Given DES attack and normal models, the I^2-DES diagnoser helps generate an alert and identify the attacker.

As soon as attack is inferred by an IDS, I_r, it sends active probe packets to the source IDS. First, I_r intimates I_s via $Ch2$. Particulars of a possible attack node is shared with I_s via this intimation. If the attack was inferred by $CoAP\_REQ\_HANDLER()$ of I_r, then it sends a normal CoAP response RST or ACK packet to I_s after some random delay. These are probe packets that the $CoAP\_RSP\_HANDLER()$ of I_s understands due to the probe intimation received beforehand. Accordingly, $CoAP\_RSP\_HANDLER()$ on behalf of I_s will send probe response intimation packet to I_r via $Ch2$, which it expects. Random application data is piggybacked with the intimation. I_r will set the requisite $flag$ and $pflag$ variables on receipt of this intimation. Meanwhile, diagnoser sets TEST_FLAG to 0 until a response is received. While waiting on a response, array Δ_s is consulted for timeout calculation.

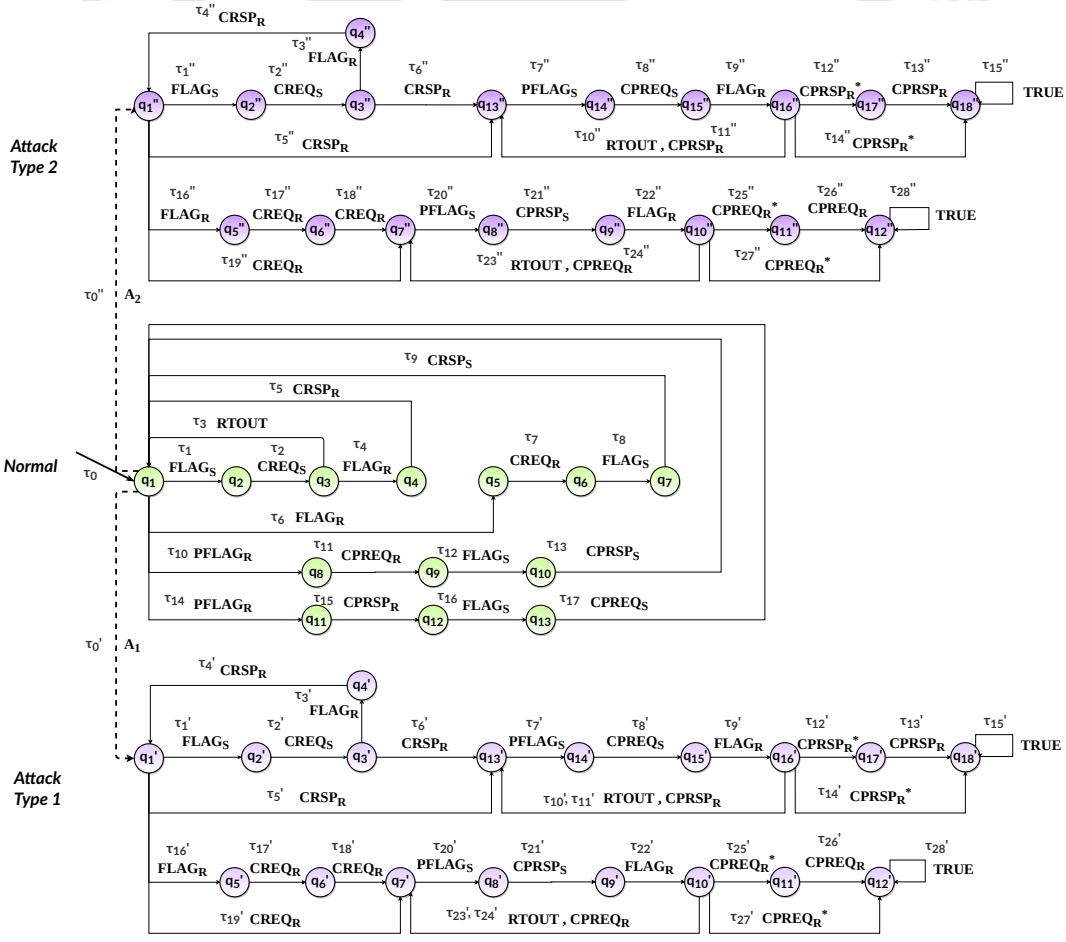


Figure 5.7: DES model H_i

ALGORITHM 10: $CoAP\_REQ\_HANDLER()$ of $\mathcal{I}_i$

Data: c_i, $\hat{\Delta}$, $\hat{\Theta}_i$, N_i, $\dot{r}_i$, $macd$, ipd
Input: Client flag, Client probe flag, CoAP request packet, CoAP probe request packet, $TEST\_FLAG$
Output: Events $CREQ_R$, $FLAG_R$, $FLAG_S$, $CPREQ_R$, $CPRSP_S$, $PFLAG_R$, $PFLAG_S$

while $\hat{\Theta}_i$ *and* Δ *are not NULL* **do**
  if $(TEST\_FLAG == 1)$ **then**
    Generate $RND1$ and $RND2$;
  if *Flag received* **then**
    if $probing^s$ = *false* **then**
      Generate event $FLAG_R$;
      $j^s \leftarrow FLAG_{RIPS}$;
      $flag^s_{i,j} \leftarrow 1$;
    if $probing^s$ = *true* **then**
      Generate event $FLAG_R$;
      if $ipd^s = FLAG_{RIPS}$ **then**
        $data^s = FLAG_{RPAYLOAD}$;
        $flag^s_{i,ipd} \leftarrow 1$;
      else if $ipd^s \neq FLAG_{RIPS}$ **then**
        $j^s \leftarrow FLAG_{RIPS}$;
        $flag^s_{i,j} \leftarrow 1$;
  if $pFlag$ *received* $\wedge$ $probed^s$ = *false* **then**
    Generate event $PFLAG_R$;
    `// Set client flag`
    $probed^s$ = true;
    $ips^s \leftarrow PFLAG_{RIPS}$;
    $csid^s \leftarrow PFLAG_{RSESSION\_ID}$;
  if *Probe Request received* $\wedge$ $probed^s$ = *true* $\wedge$ $probing^s$ = *false* **then**
    Generate event $CPREQ_R$;
    if $ips^s = CPREQ_{RIPS}$ $\wedge$ $csid^s = CPREQ_{RSESSION\_ID}$ **then**
      Generate event $FLAG_S$;
      SELECT $random(hex)$ and assign to $FLAG_{SPAYLOAD}$;
      Send flag variable status as SET to ips^s;
      Generate event $CPRSP_S$;
      Send CoAP ACK response to ips^s; $probed^s$ = false;
  if *Probe Request received* $\wedge$ $probed^s$ = *false* $\wedge$ $probing^s$ = *true* **then**
    Generate event $CPREQ_R$;
    $probing^s$ = false;
    $TEST\_FLAG \leftarrow 0$;
  if *Request received* **then**
    Generate event $CREQ_R$;
    $j^s \leftarrow CREQ_{RIPS}$;
    if $j^s \in \mathcal{I}$ *and* $flag^s_{i,j} = 1$ **then**
      $N^s_{i,j}++$;
      if $N^s_{i,j} = 1$ **then**
        Start clock timer $c^s_2()$;
      if $N^s_{i,j} > 1 \wedge c^s_2() < t_{ifs}$ **then**
        `// Function call to probing module`
    if $j^s \in \mathcal{I}$ *and* $flag^s_{i,j} = 0$ **then**
      `// Function call to probing module`
  if $((c^s_1() \geq \Delta^s_j)$ **then**
    Stop clock timer $c^s_1()$;
    `// Function call to probing module`

Table 5.1: TRANSITIONS $\Im$ IN H_i CORRESPONDING TO CoAP PACKETS

Event (σ)	Transition	$\phi(V)$	Assign(V)	$\phi(C)$	Reset(C)
$FLAG_R$	$\langle q_3,q_4\rangle,\langle q'_3,q'_4\rangle,\langle q''_3,q''_4\rangle$	$ips^c \equiv FLAG_{RIPS}$	$flag^c_{i,ips} \leftarrow 1$	-	-
		$macs^c \equiv FLAG_{RMACS}$	-	-	-
		$csid^c \equiv FLAG_{RSESSIONID}$	-	-	-
$FLAG_R$	$\langle q'_{15},q'_{16}\rangle,\langle q''_{15},q''_{16}\rangle$	$ips^c \equiv FLAG_{RIPS}$	$flag^c_{i,ips} \leftarrow 1$	-	-
		$macs^c \equiv FLAG_{RMACS}$	$data^c \leftarrow FLAG_{RPAYLOAD}$	-	-
		$csid^c \equiv FLAG_{RSESSIONID}$	$TEST\_FLAG \leftarrow 0$	-	-
		$probing^c \equiv true$	-	-	-
$CRSP_R$	$\langle q_4,q_1\rangle,\langle q'_4,q'_1\rangle,\langle q''_4,q''_1\rangle$	$ips^c \equiv CRSP_{RIPS}$	-	$c^c_1 < \Delta^c_{ips}$	$c^c_2 \leftarrow 0$
		$macs^c \equiv CRSP_{RMACS}$	-	-	-
		$csid^c \equiv CRSP_{RSESSION\_ID}$	-	-	-
		$flag^c_{i,ips} \equiv true$	-	-	-
$CRSP_R$	$\langle q'_1,q'_{13}\rangle,\langle q''_1,q''_{13}\rangle$	$ips^c \equiv CRSP_{RIPS}$	-	$c^c_1 < \Delta^c_{ips}$	-
		$macs^c \equiv CRSP_{RMACS}$	-	$c^c_2 < t_{ifs}$	-
		$csid^c \equiv CRSP_{RSESSION\_ID}$	-	-	-
		$flag^c_{i,ips} \equiv false$	-	-	-
$PFLAG_R$	$\langle q_1,q_{11}\rangle$	-	$ips^c \leftarrow PFLAG_{RIPS}$	-	-
		-	$macs^c \leftarrow PFLAG_{RMACS}$	-	-
		-	$csid^c \leftarrow PFLAG_{RSESSION\_ID}$	-	-
		-	$x^c \leftarrow PFLAG_{RAIP}$	-	-
		-	$probed^c \leftarrow true$	-	-
$CPRSP_R$	$\langle q'_{17},q'_{18}\rangle,\langle q''_{17},q''_{18}\rangle$	$ips^c \equiv CPRSP_{RIPS}$	-	$c^c_1 < \Delta^c_{ips}$	-
	$\langle q'_{16},q'_{13}\rangle,\langle q''_{16},q''_{13}\rangle$	$macs^c \equiv CPRSP_{RMACS}$	-	-	-
		$csid^c \equiv CPRSP_{RSESSION\_ID}$	-	-	-
		$probing^c \equiv true$	-	-	-
		$data^c \equiv CPRSP_{RPAYLOAD}$	-	-	-
$CPRSP^*_R$	$\langle q'_{16},q'_{17}\rangle,\langle q'_{16},q'_{18}\rangle$	$ips^c \equiv CPRSP^*_{RIPS}$	-	$c^c_1 < \Delta^c_{ips}$	-
		$macs^c \equiv CPRSP^*_{RMACS}$	-	-	-
		$csid^c \equiv CPRSP^*_{RSESSION\_ID}$	-	-	-
		$probing^c \equiv true$	-	-	-
		$data^c \neq CPRSP^*_{RPAYLOAD}$	-	-	-
		$ips^c \equiv ip\prime$	-	-	-
$FLAG_S$	$\langle q_1,q_2\rangle,\langle q'_1,q'_2\rangle,\langle q''_1,q''_2\rangle$	-	$ips^c \leftarrow FLAG_{SIPD}$	-	-
		-	$macs^c \leftarrow FLAG_{SMACD}$	-	-
		-	$csid^c \leftarrow FLAG_{SSESSION\_ID}$	-	-
		-	$flag^c_{i,ips} \leftarrow false$	-	-
$CREQ_S$	$\langle q_2,q_3\rangle,\langle q'_2,q'_3\rangle,\langle q''_2,q''_3\rangle$	$ips^c \equiv CREQ_{SIPD}$	-	-	-
		$macs^c \equiv CREQ_{SMACD}$	-	-	-
		$csid^c \equiv CREQ_{SSESSION\_ID}$	-	-	-
$PFLAG_S$	$\langle q'_{13},q'_{14}\rangle,\langle q''_{13},q''_{14}\rangle$	$ips^c \equiv PFLAG_{SIPD}$	$K_i \leftarrow K_i + 1$	-	-
		$macs^c \equiv PFLAG_{SMACD}$	-	-	-
		$csid^c \equiv PFLAG_{SSESSION\_ID}$	-	-	-
		$x^c \equiv PFLAG_{SAIP}$		-	-
		-	$probing^c \equiv true$	-	-
$CPREQ_S$	$\langle q_{13},q_1\rangle,\langle q'_{14},q'_{15}\rangle,\langle q''_{14},q''_{15}\rangle$	$ips^c \equiv CPREQ_{SIPD}$	-	-	$c^c_1 \leftarrow 0$
		$macs^c \equiv CPREQ_{SMACD}$	-	-	-
		$csid^c \equiv CPREQ_{SSESSION\_ID}$	-	-	-
		$data^c \equiv CPREQ_{SPAYLOAD}$	-	-	-

5.2.6 I^2-DES modeling

We model the LoWPAN of our interest as an I^2 (Induced I)-DES H_i, shown in Figure 5.7. The networked system model represents the *CoAP request response* event dynamics occurring during normal as well as attack scenario. We formally define H_i as a 6-tuple, H_i=$\langle Q, Q_0, \Sigma, V, C, \Im\rangle$. Q designates the set of finite states while a subset of Q designating the initial set of states is represented using Q_0 ($Q_0 \subseteq Q$). Σ is used to represent the event set, V is the finite set of model variables, C represents a set of clock variables and $\Im$ is the set of finite transitions in H_i. The notion of *final state* is dropped, since the LoWPAN is assumed to be always up with frame exchanges being continuously monitored. The definitions of the DES terminologies are presented in Appendix A. The various components

Table 5.2: TRANSITIONS $\Im$ IN H_i CORRESPONDING TO CoAP PACKETS

Event (σ)	Transition	$\phi(V)$	Assign(V)	$\phi(C)$	Reset(C)
$FLAG_R$	$\langle q_1, q_5\rangle$,$\langle q_1', q_5'\rangle$,$\langle q_1'', q_5''\rangle$	-	$ips^s \leftarrow FLAG_{RIPS}$	-	-
		-	$macs^s \leftarrow FLAG_{RMACS}$	-	-
		-	$flag^s_{i,ips} \leftarrow 1$	-	-
$FLAG_R$	$\langle q_9', q_{10}'\rangle$,$\langle q_9'', q_{10}''\rangle$	$ipd^s \equiv FLAG_{RIPS}$	$flag^s_{i,ips} \leftarrow 1$	-	-
		$macd^s \equiv FLAG_{RMACS}$	$data^s \leftarrow FLAG_{RPAYLOAD}$	-	-
		$csid^s \equiv FLAG_{RSESSIONID}$	$TEST\_FLAG \leftarrow 0$	-	-
		$probing^s \equiv true$	-	-	-
$CREQ_R$	$\langle q_5, q_6\rangle$,$\langle q_5', q_6'\rangle$,$\langle q_5'', q_6''\rangle$	$ips^s \equiv CREQ_{RIPS}$	-	$c_1^s < \Delta^s_{ips}$	$c_2^s \leftarrow 0$
		$macs^s \equiv CREQ_{RMACS}$	-	-	-
		$csid^s \equiv CREQ_{RSESSION\_ID}$	-	-	-
		$flag^s_{i,ips} \equiv false$	-	-	-
$CREQ_R$	$\langle q_6', q_7'\rangle$,$\langle q_6'', q_7''\rangle$	$ips^s \equiv CREQ_{RIPS}$	-	$c_1^s < \Delta^s_{ips}$	-
		$macs^s \equiv CREQ_{RMACS}$	-	$c_2^s < t_{ifs}$	-
		$csid^s \equiv CREQ_{RSESSION\_ID}$	-	-	-
		$flag^s_{i,ips} \equiv true$	-	-	-
$PFLAG_R$	$\langle q_1, q_8\rangle$	-	$ips^s \leftarrow PFLAG_{RIPS}$	-	-
		-	$macs^s \leftarrow PFLAG_{RMACS}$	-	-
		-	$csid^s \leftarrow PFLAG_{RSESSION\_ID}$	-	-
		-	$x^s \leftarrow PFLAG_{RAIP}$	-	-
		-	$probed^s \leftarrow true$	-	-
$CPREQ_R$	$\langle q_{11}', q_{12}'\rangle$,$\langle q_{11}'', q_{12}''\rangle$	$ips^s \equiv CPREQ_{RIPS}$	-	$c_1^s < \Delta^s_{ips}$	-
	$\langle q_{10}', q_7'\rangle$,$\langle q_{10}'', q_7''\rangle$	$macs^s \equiv CPREQ_{RMACS}$	-	-	-
		$csid^s \equiv CPREQ_{RSESSION\_ID}$	-	-	-
		$probing^s \equiv true$	-	-	-
		$data^s \equiv CPREQ_{RPAYLOAD}$	-	-	-
$CPREQ^*_R$	$\langle q_{10}'', q_{11}''\rangle$,$\langle q_{10}'', q_{12}''\rangle$	$ips^s \equiv CPREQ^*_{RIPS}$	-	$c_1^s < \Delta^s_{ips}$	-
		$macs^s \equiv CPREQ^*_{RMACS}$	-	-	-
		$csid^s \equiv CPREQ^*_{RSESSION\_ID}$	-	-	-
		$probing^s \equiv true$	-	-	-
		$data^s \equiv CPREQ^*_{RPAYLOAD}$	-	-	-
		$ips^s \equiv ip''$	-	-	-
$FLAG_S$	$\langle q_6, q_7\rangle$	$ips^s \equiv FLAG_{SIPS}$	-	-	-
		$macs^s \equiv FLAG_{SMACS}$	-	-	-
		$csid^s \equiv FLAG_{SSESSION\_ID}$	-	-	-
		-	$flag^s_{i,ips} \leftarrow false$	-	-
$CRSP_S$	$\langle q_7, q_1\rangle$	$ips^s \equiv CRSP_{SIPS}$	-	-	-
		$macs^s \equiv CRSP_{SMACS}$	-	-	-
		$csid^s \equiv CRSP_{SSESSION\_ID}$	-	-	-
$PFLAG_S$	$\langle q_7', q_8'\rangle$,$\langle q_7'', q_8''\rangle$	$ips^s \equiv PFLAG_{SIPD}$	$K_i \leftarrow K_i + 1$	-	-
		$macs^s \equiv PFLAG_{SMACD}$	-	-	-
		$csid^s \equiv PFLAG_{SSESSION\_ID}$	-	-	-
		$x^s \equiv PFLAG_{SAIP}$		-	-
		-	$probing^s \equiv true$	-	-
$CPRSP_S$	$\langle q_{10}, q_1\rangle$,$\langle q_8', q_9'\rangle$,$\langle q_8'', q_9''\rangle$	$x^s \equiv CPRSP_{SIPD}$	-	-	$c_1^s \leftarrow 0$
		$macs^s \equiv CPRSP_{SMACD}$	-	-	-
		$csid^s \equiv CPRSP_{SSESSION\_ID}$	-	-	-
		$data^s \equiv CPREQ_{SPAYLOAD}$	-	-	-
$RTOUT$	$\langle q_{10}', q_7'\rangle$,$\langle q_{10}'', q_7''\rangle$	$K_i < M$	-	-	-
	$\langle q_{16}', q_{13}'\rangle$,$\langle q_{16}'', q_{13}''\rangle$	-	-	-	-

of H_i are discussed here.

There exist domain sets of variables from which each model variable assumes values. If the set of model variables, V, is $\{v_1, v_2, \ldots, v_n\}$, then each v_i, where i is a positive integer and $i \in [1, n]$, takes some value from D_i, the domain set. The clock variables assume values from the set of non-negative real numbers, R ($R = R^+ \cup \{0\}$). Let τ be a transition in H_i, $\tau \in \Im$. We define it as a 7-tuple, $\tau = \langle x, x^+, \sigma, \phi(V), \Phi(C), Reset(C), Assign(V)\rangle$, where x is the source state and x^+ denotes the destination state. The occurrence of event σ, $\sigma \in \Im$, enables transition τ. $\phi(V)$ denotes a boolean conjunction of equality or non-equality conditions defined over a subset of the model variables. It needs to be TRUE for a transition

ALGORITHM 11: $CoAP\_REQ\_HANDLER()$ Probing Module of $\mathcal{I}_i$

Generate event $PFLAG_S$;
SELECT $random(x)$, and assign x to $PFLAG_{SAIP}$, $x \in IPLIST$;
Send probe flag to j^s;
$N^s_{i,j} = 0$; $Wait()$;
Generate event $CPRSP_S$;
$ipd^s \leftarrow CPRSP_{SIPD}$;
$csid^s \leftarrow CPRSP_{SIPD}$;
$probing^s$ = true;
Send CoAP probe response to j^s; Start clock timer $c^s_1()$;

ALGORITHM 12: $CoAP\_RSP\_HANDLER()$ Probing Module of $\mathcal{I}_i$

Generate event $PFLAG_S$;
SELECT $random(x)$, and assign x to $PFLAG_{SAIP}$, $x \in IPLIST$;
Send probe flag to j^c;
$N^c_{i,j} = 0$; $Wait()$;
Generate event $CPREQ_S$;
$ipd^c \leftarrow CPREQ_{SIPD}$;
$csid^c \leftarrow CPREQ_{SIPD}$;
$probing^c$ = true;
Send CoAP request to j^c; Start clock timer $c^c_1()$;
Stop clock timer $c^c_2()$;

to fire. An invariance condition defined over a subset of the clock variables, C is denoted using $\Phi(C)$ while $Reset(C)$ denotes reset of a subset of clock variables. A subset of model variables are assigned values, and a set of such assignments is represented using $Assign(V)$. If in some transition, τ, $\phi(V)$ or $Assign(V)$ is denoted using "-", it might mean that a condition need not be met (i.e., implicitly TRUE) or an assignment is not required.

Communication in the LoWPAN is modeled using the below mentioned variables:

$V = \{\hat{\Delta}, \hat{\Theta}_i, N_i, \dot{r}_i, macd, ipd, ip\prime, ip\prime\prime\}$ is the set of model variables.

$C = \{c^c_1, c^s_1, c^c_2, c^s_2\}$ is the set of clock variables.

The domain of the model variables $\{ips, ipd, ip\prime, ip\prime\prime\}$ is of the form $\{x.x.x.x\}$, where x denotes an integer value in the range $[0, 255]$. The state set, Q, including the initial set of states in Q_0, is disjointly partitioned into sets Q_N, Q_{A_1} and Q_{A_2} as shown in Figure 5.7. Q_N refers to the set of states visited during normal operation of the LoWPAN at the Application layer, while the states of the form Q_{A_i}, $1 \leq i \leq 2$ (designated here using primed states), belong to the network operation when under attack launched from actual node, A_i, independently. Tables 5.1 and 5.2 list elements from the event set, Σ, with their associated transitions. There exists events occurring in the system, which we model here, that may be unmeasurable. Hence event set can be expressed as a disjoint union of measurable events and unmeasurable events Σ_m and Σ_{um}, respectively. That is, $\Sigma = \Sigma_m \cup \Sigma_{um}$.

ALGORITHM 13: $CoAP\_RSP\_HANDLER()$ of $\mathcal{I}_i$

Data: c_i, Δ, $\hat{\Theta}_i$, N_i, $\dot{r}_i$, $macd$, ipd
Input: Server flag, Server probe flag, CoAP response packet, CoAP probe response packet, $TEST\_FLAG$
Output: Events $CRSP_R$, $FLAG_R$, $FLAG_S$, $CPRSP_R$, $CPREQ_S$, $PFLAG_R$, $PFLAG_S$

while $\hat{\Theta}_i$ *and* Δ *are not NULL* **do**
  if $(TEST\_FLAG == 1)$ **then**
    Generate $RND1$ and $RND2$;
  if *Flag received* **then**
    if $probing^c$ = *false* **then**
      Generate event $FLAG_R$;
      $j^c \leftarrow FLAG_{RIPS}$;
      $flag^c_{i,j} \leftarrow 1$;
    if $probing^c$ = *true* **then**
      Generate event $FLAG_R$;
      if $ipd^c = FLAG_{RIPS}$ **then**
        $data^c = FLAG_{RPAYLOAD}$;
        $flag^c_{i,ipd} \leftarrow 1$;
      else if $ipd^c \neq FLAG_{RIPS}$ **then**
        $j^c \leftarrow FLAG_{RIPS}$;
        $flag^c_{i,j} \leftarrow 1$;
  if $pFlag$ *received* $\wedge$ $probed^c$ = *false* **then**
    Generate event $PFLAG_R$;
    `// Set client flag`
    $probed^c$ = true;
    $ips^c \leftarrow PFLAG_{RIPS}$;
    $csid^c \leftarrow PFLAG_{RSESSION\_ID}$;
  if *Probe Response received* $\wedge$ $probed^c$ = *true* $\wedge$ $probing^c$ = *false* **then**
    Generate event $CPRSP_R$;
    if $ips^c = CPRSP_{RIPS} \wedge csid^c = CPRSP_{RSESSION\_ID}$ **then**
      Generate event $FLAG_S$;
      SELECT $random(hex)$ and assign to $FLAG_{SPAYLOAD}$;
      Send flag variable status as SET to ips^c;
      Generate event $CPREQ_S$;
      Send CoAP request to ips^c; $probed^c$ = false;
  if *Probe Response received* $\wedge$ $probed^c$ = *false* $\wedge$ $probing^c$ = *true* **then**
    Generate event $CPRSP_R$;
    $probing^c$ = false;
    $TEST\_FLAG \leftarrow 0$;
  if *Response received* **then**
    Generate event $CRSP_R$;
    $j^c \leftarrow CRSP_{RIPS}$;
    if $j^c \in \mathcal{I}$ *and* $flag^c_{i,j} = 1$ **then**
      $N^c_{i,j}++$;
      if $N^c_{i,j} = 1$ **then**
        Start clock timer $c^c_2()$;
      if $N^c_{i,j} > 1 \wedge c^c_2() < t_{ifs}$ **then**
        `// Function call to probing module`
    if $j^c \in \mathcal{I}$ *and* $flag^c_{i,j} = 0$ **then**
      `// Function call to probing module`
  if $((c^c_1() \geq \Delta^c_j)$ **then**
    Stop clock timer $c^c_1()$;
    `// Function call to probing module`

Table 5.3: Notations used

Notation	Explanation
$CREQ_R$, $CREQ_S$	CoAP request packet received, sent
$CRSP_R$, $CRSP_S$	CoAP response packet received, sent
$FLAG_R$, $FLAG_S$	Flag intimation packet received, sent
$PFLAG_R$, $PFLAG_S$	Probe flag intimation packet received, sent
$CPREQ_R$, $CPREQ_S$	CoAP probe request packet received, sent
$CPRSP_R$, $CPRSP_S$	CoAP probe response packet received, sent
$RTOUT$	Retransmission timeout

I^2-DES behaviour under normal circumstances

The behavior of H_i under normal circumstances is shown in Figure 5.7. The system, when functioning normally, is represented using the states $\{q_1, q_2, \ldots, q_{13}\}$ and the transitions $\{\tau_0, \tau_1, \ldots, \tau_{17}\}$. The initial state of Q_0 is q_1. The transitions are listed in Tables 5.2 and 5.1. We next discuss the transitions in normal condition as follows:

- τ_0, the initial transition leads to the initial state q_1 as shown in Figure 5.7. It is assumed while modeling that the constant timeout values, Δ and $\hat{\Theta}_i$ have been computed and then τ_0 takes place. There is no explicit event that triggers τ_0. Occurrence of τ_0 implies that the DES model is invoked when the timeout values are not NULL. Hence, $initial(\tau_0) = --$, i.e., there are no initial states and $final(\tau_0) = x1$. $\sigma = true$ means that transition τ_0 is always enabled and q_1 is automatically reached at the start of the model. $check(V) = --$ implies that no condition over the model variables are checked and the condition is always satisfiable for the transition. Value 1 is assigned to variable TEST_FLAG as implied by $Assign(V) = \{TEST\_FLAG \leftarrow 1\}$, which in turn means that the detection of fragmentation attacker can be started.

- $\tau_1 : (q_1 \rightarrow q_2)$ When the model is started and the current state is at q_1, the transition τ_1 implies that client intimates about its future communication through wired channel $(Ch2)$ by setting a flag variable . Here, $initial(\tau_1) = q_1$ and $final(\tau_1) = q_2$. $\sigma = FLAG_S$ implies that transition τ_1 is enabled when $CoAP\_RSP\_HANDLER()$ generates event $FLAG_S$ (i.e., after status is flagged). $check(V) = --$ meaning that no condition need to be satisfied and $Assign(V) = \{ips^c \leftarrow FLAG_{SIPD}, macs^c \leftarrow FLAG_{SMACD}, csid^c \leftarrow FLAG_{SSESSION\_ID}\}$. The parameters that uniquely identify a flag status packet are destination IP, destination MAC and a session ID. Consequently,

all the parameters that correspond to the header are stored in the model variables, ips^c, $macs^c$ and $csid^c$.

- τ_2 : $(q_2 \leftarrow q_3)$ At state q_2, the transition τ_2 implies that now a CoAP request packet is sent by the host client to the same server node to which the flag status was communicated earlier. It is sent via the 802.15.4 network ($Ch1$). Here, $initial(\tau_2) = q_2$ and $final(\tau_2) = q_3$. $\sigma = CREQ_S$ corresponds to enabling transition τ_2 after the $CoAP\_RSP\_HANDLER()$ generates the event $CREQ_S$ implying that a request is sent on the condition over the model variables in $check(V)$ being satisfied. $check(V) = \{ips^c = CREQ_{SIPD}, macs^c = CREQ_{SMACD}, csid^c = CREQ_{SSESSION\_ID}\}$. $Assign(V) = --$, meaning that no assignments are done in this transition.

- τ_4 : $(q_3 \rightarrow q_4)$ At state q_3, the transition τ_4 implies that a flag status intimation is received from the server node, via $Ch2$, to which the request packet was sent in τ_2. $\sigma = FLAG_R$ implies that τ_4 is enabled when the $CoAP\_RSP\_HANDLER()$ generates the event $FLAG_R$ (i.e., after flag SET status is received). $check(V) = \{ips^c = FLAG_{RIPS}, macs^c = FLAG_{RMACS}, csid^c = FLAG_{RSESSION\_ID}\}$ and $Assign(V) = \{flag^c_{i,ips} \leftarrow 1\}$. Satisfaction of the conditions over the model variables, ips^c, $macs^c$ and $csid^c$, ensure that the flag status is received from the same IDS node to which request was sent. The flag variable corresponding to the source IDS node is SET.

- τ_5 : $(q_4 \rightarrow q_1)$ At state q_4, the transition τ_5 corresponds to the receipt of a response due to the request packet sent. $\sigma = CRSP_R$ implies that the transition τ_5 is enabled when the $CoAP\_RSP\_HANDLER()$ generates the event $CRSP_R$. $check(V) = \{ips^c = CRSP_{RIPS}, macs^c = CRSP_{RMACS}, csid^c = CRSP_{RSESSION\_ID}, flag^c_{i,ips} = true\}$. The conditions over the model variables if satisfied ensure that the CoAP response is due to the request packet sent in transition τ_2. Furthermore, the clock variable c^c_1 ensures that response is received before the retransmission timeout window expires and variable c^c_2, modeling the timer that checks inter-frame spacing, is started.

Similarly, the other states and transitions pertaining to the $CoAP\_RSP\_HANDLER()$ with their implications are briefly discussed as follows:

- τ_{14}: $(q_1 \rightarrow q_{11})$ At state q_1, this transition corresponds to the receipt of probing intimation by the host node via $Ch2$. Accordingly the $CoAP\_RSP\_HANDLER()$ generates the event $PFLAG_R$ and the packet parameters received are assigned to the variables ips^c, $macs^c$, $csid^c$, $probed^c$ and x^c.

- τ_{15}: $(q_{11} \rightarrow q_{12})$ At state q_{11}, this transition corresponds to the receipt of a CoAP probe response packet by the host node via $Ch1$. Accordingly the $CoAP\_RSP\_HAN- -DLER()$ generates the event $CPRSP_R$ and the packet parameters received are checked against variables ips^c, $macs^c$ and $csid^c$.

- τ_{17}: $(q_{13} \rightarrow q_1)$ At state q_{13}, this transition is taken on sending a CoAP probe request packet via $Ch1$. Accordingly the $CoAP\_RSP\_HANDLER()$ generates the event $CPREQ_S$ and the packet parameters received are checked against variables ips^c, $macs^c$, $csid^c$ and $data^c$. $data^c$ is checked to ensure that the random payload communicated is same across $Ch1$ and $Ch2$.

Next, the states and transitions corresponding to the $CoAP\_REQ\_HANDLER()$ in the normal behaviour are discussed as follows:

- τ_7: $(q_5 \rightarrow q_6)$ At state q_5, this transition is taken on receipt of a CoAP request packet via $Ch1$ following a flag status receipt. Accordingly here, the event $CREQ_R$ is generated after the $CoAP\_REQ\_HANDLER()$ is invoked and the packet parameters received are checked against the variables ips^s, $macs^s$, $csid^s$. $flag^s_{i,ips}$ is ensured to be true.

- τ_{11}: $(q_8 \rightarrow q_9)$ At state q_8, this transition implies that CoAP probe request packet via $Ch1$ is received following a probe flag status update. Accordingly the event $CPREQ_R$ is generated by the $CoAP\_REQ\_HANDLER()$ and the packet parameters that are received are checked against variables ips^s, $macs^s$, $csid^s$ to ensure that they are received from the same client IDS node that communicated the probe flag status. $probed^s$ status is checked such that it holds true.

- τ_{13}: $(q_{10} \rightarrow q_1)$ At state q_{10}, this transition implies sending a CoAP probe response packet via $Ch1$. This is sent in response to the probe request packet that is received in transition τ_{11}. The $CoAP\_REQ\_HANDLER()$ generates the event $CPRSP_S$

and the packet parameters received are checked against variables x^s, $macs^s$, $csid^s$ and $data^s$.

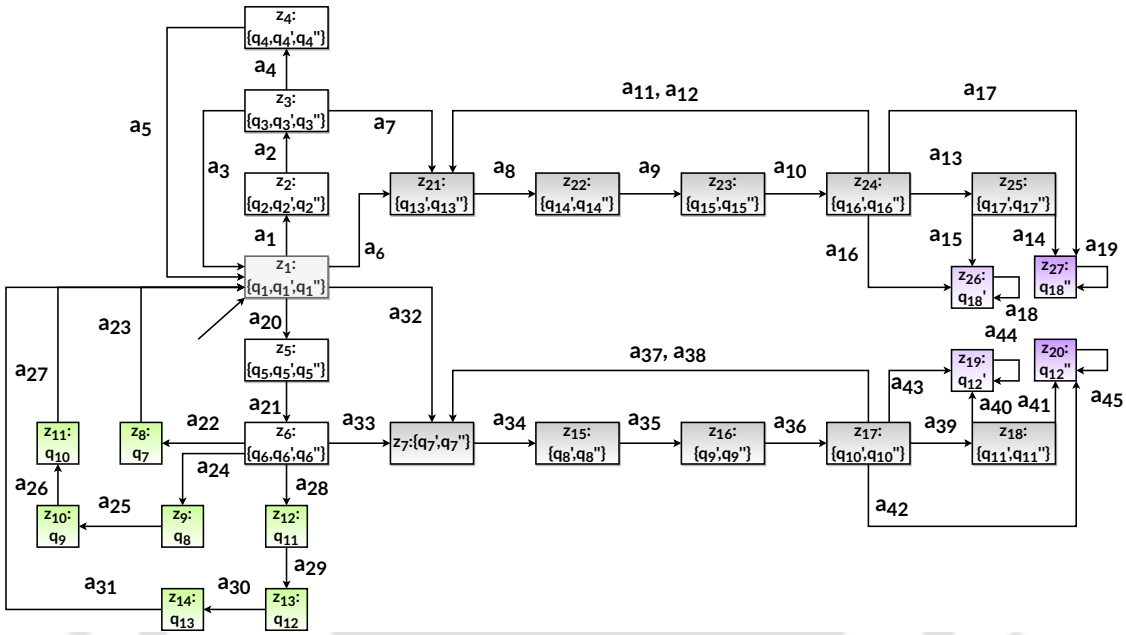


Figure 5.8: Diagnoser O for DES model H_i

I^2-DES behaviour under attack circumstances

The DES model under CoAP spoofing attack condition launched by attacker A_1 is shown using the states in $Q_{A_1} = \{q'_1, q'_2, \ldots, q'_{18}\}$ and transitions, $\{\tau'_1, \tau'_2, \ldots, \tau'_{28}\}$. Similarly for attacker type A_2, states and transitions are represented using double prime notation, $Q_{A_2} = \{q''_1, q''_2, \ldots, q''_{18}\}$ and transitions, $\{\tau''_1, \tau''_2, \ldots, \tau''_{28}\}$ as shown in Figure 5.7. The DES model behavior under different attackers are identical except four transitions that differentiate them which are discussed.

- At state q_1, the system reaches an attacker type state q'_1 or q''_1 following an unmeasurable attack transition τ'_0 or τ''_0, respectively.

- $\tau'_{12} : (q'_{16} \rightarrow q'_{17}, q'_{16} \rightarrow q'_{18})$ At state q'_{16}, the transition τ'_{12} corresponds to a spoofed CoAP response packet. $\sigma = CPRSP^*_R$ implies that the transition is enabled when the $CoAP\_RSP\_HANDLER()$ generates the event $CPRSP^*_R$. $check(V) = \{ips^c = CPRSP^*_{RIPS}, macs^c = CPRSP^*_{RMACS}, csid^c = CPRSP^*_{RSESSION\_ID}, ips^c = ip\prime\}$.

The conditions over the model variables, ips^c, $macs^c$ and $dtag$, ensure that the duplicate NACK intimation is a response to the datagram probe header fragment in $\tau 10'$. The model variable $macd$ holds the MAC address of the next-hop node via which the probe was sent. $\tau 16'$ ensures that $macd$ holds the IP address of attack node A_1. An intimation for probe sent via a node with MAC address stored in $macd$ is a fragment duplication attack. $\Phi(C) = \{c_1^c < \Delta_{ips}^c\}$ means that c_1^c does not exceed the retransmission timeout period.

- τ_{25}'' : $(q_{10}'' \rightarrow q_{11}'', q_{10}'' \rightarrow q_{12}'')$ At state q_{10}'', the transition τ_{25}'' corresponds to duplicate fragment intimation for the datagram probe sent via a known 1-hop node. $\sigma = CPREQ_R^*$ implies that the transition is enabled when the $CoAP\_REQ\_HANDLER()$ generates the event $CPREQ_R^*$. $check(V) = \{ips = DPNACK_{IPD}, ipd = DPNACK_{IPS}, dtag = DPNACK_{dtag}, macd = mac'\}$. The conditions over the model variables, ips, ipd and $dtag$, ensure that the duplicate NACK intimation is a response to the datagram probe header fragment in $\tau 10'$. The model variable $macd$ holds the MAC address of the next-hop node via which the probe was sent. $\tau 16'$ ensures that $macd$ holds the IP address of attack node A_1. An intimation for probe sent via a node with MAC address stored in $macd$ is a fragment duplication attack. $\Phi(C) = \{c_1^s < \Delta_{ips}^s\}$ means that c_1^s does not exceed the retransmission timeout period.

I^2-Diagnoser

As discussed earlier, for a system to be DES diagnosable, no fault indeterminate cycles should be present. The stringency of this requirement makes a lot of systems non-diagnosable. Hence, to render a large class of such non-diagnosable systems to be diagnosable, a weaker and relaxed notion, I-diagnosability was proposed (see Section 5.2.6). I-diagnosability associates failures with indicator events. An I-DES modeled system is I-diagnosable if there exists an indicator event following a failure and the occurrence of the failure can be detected only by testing system through paths after the indicator event occurs. There still exists system failures that cannot be diagnosed by the I-diagnosability framework as well in spite of observable indicator events after the failure. To overcome this limitation, a notion of I^2-diagnosability is proposed in this work. Under I^2-diagnosability framework, given indicator events and the DES model, an empowering event ensures that an indicator event

actually sensitizes the failure. Here we use a set of IDS each consisting of a local I^2-diagnoser. Each of the local I^2-diagnosers are basically observer automatons. They track the local system behavior and gives a decision on the diagnosis of monitored events. Details of the diagnoser construction procedure and definitions pertaining to diagnosability are highlighted in Appendix B. Now, an attack type I^2-diagnosable in finite time, if the I^2-diagnosability condition is met (F_i-I^2-Diagnosability property is satisfied, see Definition 7, Appendix B). Since fault occurrence and attack events exhibit identical deviations from the normal behaviour and in both scenarios they are unmeasurable, taking place without the knowledge of the system administrator, hence fault diagnosis has been successfully applicable in attack detection and attack type identification too. A lemma on the I^2-diagnosability property states that lack of attack A_i-indeterminate cycles (Definition 14, Appendix C) having an embedded indicator transition (Definition 13, Appendix C) guarantees I^2-diagnosability. It means that the diagnoser is able to give a decision in finite time on the occurrence of the attack event type. Satisfaction of the I^2-diagnosability property locally, considering the limitations in measurement, ensures efficient attack detection as well as attack node location diagnosis globally [169].

Figure 5.8 shows the constructed diagnoser for our DES model H_i, considered in Figure 5.7. The working mechanism of our diagnoser is summarised here by showing one or more executions of sequences of measured events (transitions) as follows:

1. The initial state of the model H_i, q_1, and states q_1' and q_1'' that are reachable via unmeasurable attack transitions, τ_0' and τ_0'', form the initial state, z_1 of the diagnoser O.

2. Let $\Im_{z1_1} = \{\tau_1, \tau_1', \tau_1''\}$, i.e., the outgoing transitions from model states $\{q_1, q_1', q_1''\} \in z1$. All of outgoing transitions in $\Im_{z1_1}$ are measurement equivalent (Definition 2, Appendix A) belonging to one measurement equivalence class of transitions, hence cannot be further partitioned. Therefore, it justifies O-transition a_1. The O-state reached corresponding to the transition a_1 is $z_2 = \{q_2, q_2', q_2''\}$.

3. Let $\Im_{z1_2} = \{\tau_6, \tau_{16}', \tau_{16}''\}$, i.e., the outgoing transitions from model states $\{q_1, q_1', q_1''\} \in z1$. All the transitions in $\Im_{z1_2}$ are equivalent and hence cannot be partitioned further, justifying O-transition a_{20}. The O-state corresponding to the transition a_{20}

is $z_5 = \{q_5, q_5', q_5''\}$.

4. Let $\Im_{z1_3} = \{\tau_5', \tau_5''\}$, i.e., the outgoing transitions from model states $\{q_1', q_1''\} \in z1$. All the transitions in $\Im_{z1_3}$ are also measurement equivalent and hence cannot be partitioned further, justifying O-transition a_6. The O-state corresponding to the transition a_6 is $z_{21} = \{q_{13}', q_{13}''\}$. Since z_{21} consists of states only from the attack types 1 and 2, hence it is an *Attack* certain O-state (Definition 9, Appendix C).

5. Let $\Im_{z1_4} = \{\tau_{19}', \tau_{19}''\}$, i.e., the outgoing transitions from model states $\{q_1', q_1''\} \in z1$. All the transitions in $\Im_{z1_4}$ are also measurement equivalent and hence cannot be partitioned further, justifying O-transition a_{32}. The O-state corresponding to the transition a_{32} is $z_7 = \{q_7', q_7''\}$. Since z_7 consists of states only from the attack types 1 and 2, hence it is an *Attack* certain state.

   In a similar manner, the diagnoser states $\{z4, z5, z6, z7, z8\}$ can be constructed using the corresponding O-transitions $\{a4, a6, a7, a9, a10\}$. The principle can be safely extended.

6. From the definition, we can compute the A_k-certain O-states and the Normal certain O-states. In our example, the A_1 certain O-state may be computed as $z10 = \{x9'\}$ since it exclusively consists of states only belonging to attacker A_1. Similarly, A_2 certain O-state may be computed as $z11 = \{x9''\}$ and the normal certain O-state can be computed as $z9 = \{x1\}$.

5.2.7 An example of CoAP spoofing attacker node identification using DES Diagnoser

Suppose $\hat{\Delta}$, t_{ifs} and $\hat{TRES}$, IPLIST are computed and the following pair of event sequences occur chronologically in the monitored network due to CoAP packets received or sent from an IDS node: (i) $FLAG_S$, $CREQ_S$, $CRSP_R$, $PFLAG_S$, $CPREQ_S$, $FLAG_R$, $CPRSP_R^*$ (ii) $CREQ_R$, $PFLAG_S$, $CPRSP_S$, $FLAG_R$, $CPREQ_R^*$, $CPREQ_R$.

In case (i), the diagnoser starts from the O-state z_1 and on occurrence of the $FLAG_S$ event, the diagnoser moves to O-state z_2 via O-transition a_1. The event occurs when a flag status update is sent by IDS node $\mathcal{I}_i$ to another IDS node, suppose $\mathcal{I}_j$. Now, the transition a_1 might have been taken by the diagnoser due to the occurrence of any of the

H_i-transitions, τ_1, τ_1' or τ_1''. Since the transitions τ_1, τ_1' and τ_1'' are measurement equivalent, it cannot be certainly said at this point if an attack has occurred. Following this, a CoAP request packet is sent by $\mathcal{I}_i$ to $\mathcal{I}_j$ due to which the event $CREQ_S$ is passed to the diagnoser. The diagnoser of $\mathcal{I}_i$ moves to O-state z_3 via O-transition a_2. Now when a $CRSP_R$ packet is received with $flag^c_{i,j} = false$, the event $CRSP_R$ is recorded and the diagnoser reaches O-state z_{21} via O-transition a_6, which is an *Attack*-certain state. Therefore, at state z_{21}, a probe flag status packet, $PFLAG_S$, with attacker node information is passed via $Ch2$ to $\mathcal{I}_j$ by the $CoAP\_RSP\_HANDLER()$ component of behalf of $\mathcal{I}_i$. The diagnoser moves to state z_{22} via O-transition a_8 on the event $PFLAG_S$ being passed to it. Consequently, a CoAP probe request packet is sent to $\mathcal{I}_j$ via $Ch1$. The event generated is $CPREQ_S$ and the diagnoser moves to state z_{23} via a_9. On receipt of a flag status update from $\mathcal{I}_j$ with dummy data, the event $FLAG_R$ is passed to the diagnoser which now moves to z_{24}. Here, a CoAP probe response packet is received by $\mathcal{I}_i$ such that the payload does not match the data intimated in O-transition a_{10}. Therefore, $CPRSP^*_R$ event is generated. No further packet is received by $\mathcal{I}_i$ from $\mathcal{I}_j$ until timeout Δ_j occurs and the diagnoser moves to state z_{26} or z_{27}, both of which are A_k-certain states. Hence, it is ascertained that the system is under attack condition due to attacker node A_1 or A_2. Moreover, since there are no A_k-indeterminate cycles [46, 45], along all paths of the I^2-DES diagnoser, an unique malicious node A_k, when present, can be identified correctly. On each such occasion when the diagnoser reaches an A_k-certain state due to an event trace, an alert is generated.

Whereas in case (ii), the diagnoser starts from the O-state z_1 and on receipt of a CoAP request packet with $flag^c_{i,j} = false$, the event $CREQ_R$ is generated by the $CoAP\_REQ\_HANDLER()$ and passed to the diagnoser. The diagnoser reaches O-state z_7 via O-transition a_{32} which is an *Attack*-certain state consisting of states from attack models q_7' and q_7''. As a consequence, a probe flag status packet, $PFLAG_S$, with attacker node information is passed via $Ch2$ to $\mathcal{I}_j$ by the $CoAP\_REQ\_HANDLER()$ component of behalf of $\mathcal{I}_i$. Therefore, the diagnoser reaches O-state z_{15} via O-transition a_{34} from O-state z_7 on the event $PFLAG_S$ being passed to it. Vis-a-vis, a CoAP probe response packet is sent to $\mathcal{I}_j$ via $Ch1$. The event generated is $CPRSP_S$ and the diagnoser moves to state z_{16} via a_{35}. Next, when a flag status update is received at $\mathcal{I}_i$, the event $PFLAG_R$ is generated and passed to the diagnoser while the payload is locally stored. The diagnoser

moves correspondingly to state z_{17} via transition a_{36}. Eventually, when two CoAP response packets with varying payloads are received within the retransmission timeout period, the events $CPREQ^*_R$ and $CPREQ_R$ are passed to the diagnoser. The diagnoser moves finally to O-states z_{19} or z_{20} depending on the transitions a_{40} or a_{41} that may be taken due to value of attacker node IP address is $ip\prime$ or $ip\prime\prime$. Both z_{19} and z_{20} are A_k-certain states. Hence, it can be ascertained that the system is under attack condition due to attacker node either A_1 or A_2.

5.2.8 Correctness

DES modeling helps in system level formalisms for checking correctness and completeness. Our proposed IDS correctness is demonstrated here by considering all possible CoAP spoofing attack cases. The aim is to show that a malicious node is correctly identified each time. For our proof, we assume a network fragment consisting of 2 IDS nodes, I_r and I_s, and 4 non-IDS nodes, A, B, X and Y. Let us suppose that A is a request spoofing attacker node and X, a response spoofing attacker node. They are relatable to A_1 and A_2 nodes in our DES modeling. Since our diagnoser O lacks A_k-indeterminate cycles, diagnosability condition is hence satisfied, meaning that A_k is always identifiable (unique A_k is diagnosable). To prove completeness, we justify that all possible attack scenarios are detected by simulating traces in H_i. We show A and X are always correctly identified on reaching the corresponding A_k certain state in the diagnoser. We consider the following situations to occur; a CoAP request packet, $\mathcal{Q}$, is received at I_r on one occasion (Case 1) and on another, a CoAP response packet, $\mathcal{P}$, is received (Case 2). We also assume that X lies on-path node that forwards $\mathcal{P}$.

Case 1.1 (I_r receives $\mathcal{Q}$ on $Ch1$ without any prior intimation): This can be fairly understood as a CoAP request spoofing attack inference. Consequently, the current O-state of the diagnoser evolves from z_1 to z_7 via O-transition a_{32}. At this point it cannot be concluded that which one among A, B, X and Y had sent the spoofed packet, $\mathcal{Q}$. Now, a probe intimation is accordingly sent by the $CoAP\_REQ\_HANDLER()$ on behalf of I_r with AIP of A. The event $PFLAG_S$ is generated and passed to the diagnoser which reaches O-state z_{15} by this time. A probe response packet, $CPRSP_S$, is sent by I_r to I_s and diagnoser reaches z_{16} via a_{35}. Next, a flag intimation is received from I_s. The event

$FLAG_R$ is generated and passed to O. Relevant application data is stored in the variable $data^s$. The diagnoser current O-state is z_{17}. Now, I_s will send a response with the intimated application data with spoofed IP of the attack node A. The CoAP packet requests for a resource uri that A contains. A eavesdrops the packet and sends a spoofed request packet to deny node I_s. Hence, the events $CPREQ^*_R$ and $CPREQ_R$ are generated and on being passed to the diagnoser, O-state $_{19}$ is reached via O-transition a_{43}. This is an attacker A_1-certain state. Hence, attack node A is correctly identified.

Case 1.2 (I_r receives a duplicate $\mathcal{Q}$ on $Ch1$ with intimation): This is also a CoAP request spoofing attack inference. In this case the current O-state of the diagnoser evolves from z_1, z_5, z_6 to ultimately z_7 via O-transitions a_{20}, a_{21} and a_{30}. Again it cannot be concluded that which one among A, B, X and Y had sent a spoofed packet, $\mathcal{Q}$. This time, a probe intimation is accordingly sent by the $CoAP\_REQ\_HANDLER()$ on behalf of I_r with AIP of B. Probing is carried out similar to the scenario discussed in Case 1.1. The only difference arises is when a probe request is expected and the diagnoser is at state z_{17}, only the probe request from I_s with matching application data is received on this occasion. The event $CPREQ_R$ is generated and passed to the diagnoser which moves back to O-state z_7. Now, another probe intimation packet is sent by I_r with AIP of A. The events that follow are similar to the sequence of Case 1.1. Hence attack node A is correctly identified.

Case 2.1 (I_r receives $\mathcal{P}$ on $Ch1$ without any prior intimation): This is a CoAP response spoofing attack inference. Consequently, the current O-state of the diagnoser will now evolve from z_1 to z_{21} via O-transition a_6. At this point it cannot be concluded that which one among A, B, X and Y had sent the spoofed packet, $\mathcal{P}$. A probe intimation is accordingly sent by the $CoAP\_RSP\_HANDLER()$ on behalf of I_r with AIP of X. The event $PFLAG_S$ is generated and passed to the diagnoser which reaches O-state z_{22} by this time. A probe request packet, $CPREQ_S$, is sent by I_r to I_s and diagnoser reaches z_{23} via a_9. Next, a flag intimation is received from I_s. The event $FLAG_R$ is generated and passed to O. Relevant application data is stored in the variable $data^c$. The diagnoser current O-state is z_{24}. Now, I_s will send a response with the intimated application data with spoofed IP of the attack node X. The CoAP packet response is regarding a resource uri that X contains. X eavesdrops the packet and sends a spoofed response packet to deny node I_s. Hence, the events $CPRSP^*_R$ and $CPRSP_R$ are generated and on being passed to

the diagnoser, O-state $_{27}$ is reached via O-transition a_{14}. This is an attacker A_2-certain state. Hence, attack node X is correctly identified.

Case 2.2 (I_r receives a duplicate $\mathcal{P}$ on $Ch1$ with intimation): This is again a CoAP response spoofing attack inference. In this case the current O-state of the diagnoser evolves from z_1, z_2, z_3 to ultimately z_{21} via O-transitions a_1, a_2 and a_7. Again it cannot be concluded that which one among A, B, X and Y had sent a spoofed packet, $\mathcal{P}$. This time, a probe intimation is accordingly sent by the $CoAP\_RSP\_HANDLER()$ on behalf of I_r with AIP of Y. Probing is carried out similar to the scenario discussed in Case 2.1. The only difference arises is when a probe response is expected and the diagnoser is at state z_{24}, only the probe response from I_s with matching application data is received on this occasion. The event $CPRSP_R$ is generated and passed to the diagnoser which moves back to O-state z_{21}. Now, another probe intimation packet is sent by I_r with AIP of X. The events that follow are similar to the sequence of Case 2.1. Hence attack node X is correctly identified.

So, all possible attack cases by A and X are analyzed. Each time it can be seen that diagnoser reports the network condition and attack node correctly. Each time a correct attacker is concluded when an A_k-certain state is reached.

Table 5.4: Throughput, Accuracy, and Response Time During DoS and after implementation of Intended Approach

N/W Parameter	**During DoS Attack Execution**					**After Implementation of Intended Approach**				
No. of IoT Node	**8N**	**16N**	**32N**	**64N**	**128N**	**8N**	**16N**	**32N**	**64N**	**128N**
Throughput	87.6%	84.3%	79%	74%	69.3%	98.3%	95.6%	92.5%	83.9%	80.6%
Accuracy (%)	NA	NA	NA	NA	NA	99.7%	99.4%	99,2%	98.7%	93%
Response Time (Sec.)	NA	NA	NA	NA	NA	4.3 Sec	4.7Sec.	5.7 Sec.	6.4 Sec.	9.3 Sec.

5.3 Experiments, results, and discussion

In this section, we analyze network performance with the help of three types of experiments as follows:

Table 5.5: Comparison between closely related works and proposed approach

Refrences	**Attack Detection or Identification**	**Energy Usage (mJ)**	**RAM/ROM (in Byte)**	**Scalability**	**Response Time (s)**	**Accuracy (%)**
Bhale et al. (2018) [117]	Detection	94753	9883/56713	N	34.2-68.9	99.7%
Roselin et al.(2019) [121]	Both	NA	12785/61292	Y	NA	93.5%
JerryJohn et al (2020) [170]	Detection	79920	NA	N	28.9- 38.5	99.4%
Prahlad et al. (2021) [171]	Detection	NA	13478/58952	Y	94.7-198	97.8%
Proposed Approach	**Both**	**84282**	**5250/34955**	**Y**	**6.1-10.7**	**99.2%**

5.3.1 Network Performance under non-attack scenarios

In the non-attack scenario, we analyze the network performance with 8, 16, 32, 64, 128 nodes, as shown in Fig 5.9. All these nodes are legitimate and send requests to the *skywebsence* web server for getting their aids (e.g., temperature, intensity, and light values). We use WireShark to monitor the packet arrival time and flow count. The power consumption, and RAM/ROM in non-attack scenarios are 7834mJ, 4873/32785 (in bytes), respectively. These parameter values are consistently disseminated over an experimental period.

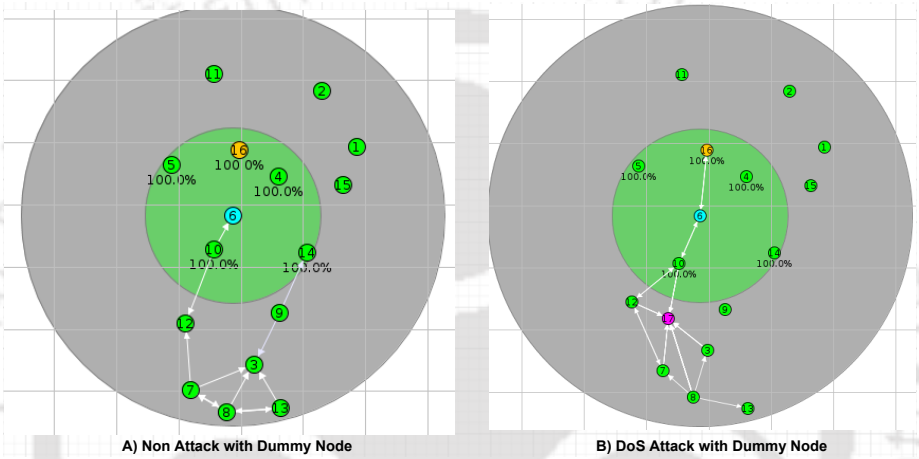


Figure 5.9: Snapshot of Non attack and DDoS attack scenario in IoT Ecosystem

5.3.2 Network Performance under DoS attack scenarios

Several tools like *libcoap*, *hping*, XOIC, HULK, HOIC, and Contiki Cooja are utilized to insert DoS attacks. Traffic generated based on these tools limit the web-server's liveliness by forming recurring CoAP connections. Network parameters like power consumption, and RAM/ROM values in DoS attack scenarios are shown in Table 5.5 . The exhibited throughput of the system drops as compared to the non-attack scenarios, due to recurring CoAP connections, are listed in Table 5.4.

5.3.3 Network Performance with the proposed approach:

Table 5.4 shows throughput, accuracy, and response time with the multiple numbers of IoT nodes. Based on the experimental results, the intended approach is energy-efficient and scalable. It also gives comparable energy usage, RAM/ROM utilization, response times,

and accuracy. Table 5.5 shows an average energy consumption, RAM/ROM, response time, and accuracy are *84282 mJ*, *5250B/34955B*, *6.1-10.7 (s)*, and *99.2%*, respectively.

5.4 Conclusion

In this chapter, a novel I^2-DES based IDS scheme is presented that detects as well as identifies an IP address spoofing attacker in CoAP based LoWPAN scenario. We show using the notion of diagnosability that an attacker can be correctly identified from analysing the characteristic network dynamics by using empowering events and a one of a kind probing schema design. The security solution we propose is an energy-efficient approach that achieves 99.2% accuracy with response time as low as 6.1 seconds, thus, mitigating both request and response spoofing.

The experimental results are illustrative of both attacked and non-attacked IoT ecosystem behavioural patterns. In future, we would like to extend our approach to MQTT security and further venture into distributed denial-of-service attack mitigation schemes using lightweight solutions.

CHAPTER

6

Conclusions and Future Work

The world has already witnessed a paradigm shift in modern day communication and connectivity with the phenomenal and speedy growth of IoT and the heavy reliance of modern day lifestyle on it. IoT services, made available to the tiniest of devices and sensors, are being vastly applied and used in mission-critical systems, healthcare, manufacturing systems, smart homes, smart cities, electricity meters, etc. A majority of the IoT networks and devices are lossy and constrained in nature, making them an easy target for malicious entities. Moreover, authentication capabilities are lacking in IoT systems with resource limitations also making strong cryptography and encryption inapplicable. An attacker that gets hold of the device, or is functioning as a regular node in an IoT network, can easily target insecure side-channels or launch various internal attacks exploiting the inherent IoT protocol vulnerabilities on battery-powered devices that can be easily compromised.

IoT devices popularly consist of scan chains for testing purposes. As a result enhanced observability and controllability of the internal register contents of the device is induced. There exists a class of intrusive (non-invasive) device-level attacks on cryptographic IoT devices that makes use of this testability-induced vulnerability to leak out confidential contents. In a scan-based side-channel attack, the secret key values can be easily differentiated from the non-secret ones, inflicting loss. These attacks create differentiable characteristics that make it challenging to preserve a secret. Also, an IDS is deployed nowadays in most IoT networks as the fundamental network security component. Task of an IDS is to monitor the host network for infiltration or malicious activities and raise an alarm to the

network administrator when found. Thus, alarm generation provides an opportunity for the administrator to take corrective measures, such that effects of the threat are quarantined or mitigated so as not to prevent further damage. IDS can be broadly categorized into signature based and anomaly based depending on the working procedure. Signature-based IDS uses fixed signatures (or patterns) while anomaly-based IDS is mathematical/statistical based. However, we identify there exist certain types of attacks on IoT protocols for which such IDS fail to generate alarms as effectively. Specifically, a large number of false positives ensue. Examples of such attacks at the network level are version attacks, rank attacks, fragmentation attacks and request/response spoofing attacks. Due to such attacks, the sequence of network events and semantics remain the same, hence attack patterns cannot be written. Furthermore, these attacks do not change the system behaviour considerably enough for anomaly-based IDS to be accurate.

Existing methods to protect scan-based side channel attacks are mostly LFSR or obfuscation based. Moreover, we devise a certain class of attacks for which there are no known effective countermeasures. Also present methods either compromise to testability or are resource consuming. Also network and application layer attacks are defended in the research community using user level, cryptographic level, protocol based or machine learning based. Each of these approaches suffer from drawbacks like costly deployment and setup, protocol modifications, lacking formalism, lacking scalability, or extensive training time. Each of the attacks portray unique characteristics that require hardware and software countermeasures to be minute and tuned in.

With the exponential growth in complex dynamic systems, Discrete Event System (DES) framework is being widely used to model these systems in avenues such as embedded systems, networked systems, communication systems, manufacturing plants, chemical processes, traffic systems, cyber physical systems, etc. The theory of FDD of DES finds wide applicability in determining a system to be functioning in faulty or normal condition.

6.1 Summary of Thesis Contributions

In this thesis, we have developed lightweight countermeasure strategies/mechanisms to secure the IoT ecosystem from different types of device-level, network-level and application-level attacks. For modeling and verification purposes, we use DES security notion of Opacity

to prevent device-level attacks. Analogously, the FDD theory of DES is used to detect and identify an attacker node that has launched various attacks on the IoT network and application layer protocols. The corresponding contributions of each chapter of the chapters are presented below:

6.1.1 Contributions of Chapter 2:

Scan chain is the de facto standard for testing manufacturing defects of cryptographic ICs in the semiconductor industry. As a result of scan chains, sequential elements are more observable and controllable. Hence, an attacker who gets access to IoT devices having a cryptographic implementation can launch various differential scan based attacks to reveal the embedded secret user key of such devices. To launch such attacks, the attacker just needs to apply plaintext inputs and observe the ciphertext output. The key can then be retrieved from offline computation. In this contribution, we propose a novel differential scan attack, namely co-relation scan attack (COSA) and a hardware controller based countermeasure that thwarts differential scan-based attacks in general. Our proposed COSA attack belongs to the class of differential scan attacks and is more comprehensive and effective than the existing differential scan attack that is launched by targeting unique hamming weight pairs only. The motivation behind this attack is that the existing attacks can be easily defended by carefully crafting additional bits in order to convert a unique hamming weight to non unique one. Our proposed attack can work using any possible hamming weight model necessitating protection of the circuit under test (CUT) from a relatively larger attack surface.

Next, we devise a countermeasure that preserves the secrecy of an embedded key in a cryptographic integrated circuit of an IoT device running an Advanced Encryption Standard (AES) implementation. A novel design involving a hardware unit is illustrated that circumvents all differential scan attacks by essentially performing bit flips deterministically, using a pre-computed mask value. This helps secure the chip while retaining full testability. The controller logic directly depends on a mask determination algorithm that can defend against any scan attack with O(1) theoretical complexity. Security analysis of our proposed defense procedure is performed in the framework of Discrete Event Systems (DES). The sequential scan circuit of an AES cryptosystem during normal operation and flipped operation

is modeled as a DES using Finite State Automata. A security notion, *Opacity*, is used to quantify and formally verify the security aspects of our controlled system, which shows that the entropy of the secret key is preserved. A case study is performed that shows to mitigate state-of-the-art differential scan attacks successfully at a nominal extra overhead of 1.78%.

6.1.2 Contributions of Chapter 3:

Rank and version value fields are used in RPL control packets for its efficient operation. RPL participating nodes are ordered as an acyclic tree, DODAG, and these values help in their creation and maintenance. A malicious node internal to the RPL DODAG may falsely modify these values to create unoptimized paths and loops essentially degrading performance of the IoT network. In this contribution we propose a novel software countermeasure that not only detects RPL rank and version attacks but also identifies the attacker node. As compared to the differential scan attacks that jeopardize a secret user key, rank and version attacks are non-invasive. Adverse effects of such low overhead IoT network attacks mostly go unnoticed since they do not change the network operations much. Hence, the normal network behaviour cannot be differentiated from attack type behaviour directly. Therefore, RPL rank and version attacks cannot be detected just by passively observing the RPL control and data packets that are exchanged in the IoT network. For these attacks, the DES states belonging to the attack behaviour cannot be deterministically diagnosed because of the inherent uncertainties of the genuinity of DIO packets disseminated by the attacker. So there exists paths in the DES that lead to indeterminate cycles. A mechanism is therefore required to generate distinguishable behaviour during an attack than when operating normally, such that an attack certain state can be reached in finite time.

Our countermeasure utilises active probing mechanism and I (Indicator)-DES based IDS. Probe packets sent by the IDS helps create differentiable attack behaviour from the normal. They are normal RPL data packets that signify the indicator events that are necessary for successful identification in our scheme. The identification principle is based on the RPL control and data packets exchanged between IDS root, agent nodes and all of the other nodes participating in the DODAG instance.The packet sequences are analyzed to generate the intelligent probing mechanism. A rank attacker node is ascertained from the history of probe response times measured by the IDS. Further, I-DES based IDS framework

is adopted and extended to model the normal and attacker type specifications, such that an attacker (attack type) can be identified. An I-diagnoser, constructed from the I-DES models, generates an alert when a malicious node is identified (corresponding attacker$_k$ certain state is reached). We also prove the correctness and completeness of our scheme. The DES framework is implemented only at the root node, therefore using our IDS does not require any heavy deployment, protocol modifications, or training. To assess the performance of our solution, we implement our proposed method in simulation as well as real testbed experiments with varying numbers of IoT nodes. The results show that our solution is scalable, has least false positives, energy-efficient and more accurate compared to most state-of-the-art schemes.

6.1.3 Contributions of Chapter 4:

IoT-LLNs mostly employ 6LoWPAN, an IETF-standardized adaptation layer, for IPv6 based communication. Secure mechanisms protect 6LoWPAN from external attackers, yet, lack of authentication capabilities and the scarcity of resources render it susceptible to various designed internal attacks. Especially, the 6LoWPAN fragmentation mechanism is easily exploited by replaying spoofed fragments, timely slipped in by an eavesdropping attacker. In a constrained environment, neither the original fragment nor the sender node authenticity is differentiable here making solution techniques challenging. In this contribution, we devise a 6LoPWAN fragment duplication attack detection and attacker localisation scheme that utilises an intelligent active probing mechanism for 6LoWPAN attacker localisation and I-DES based IDS. As compared to rank and version attacks that have far reaching effects across the RPL network, fragmentation attacks exploit nodes that are at 1-hop distance only. Moreover, as opposed to control packets, fragmentation attacks exploit the reassembly mechanism during regular end-to-end transmission and an attacker may be located on the forwarding path or can be off-path as well. So, a centralized IDS scheme is not suitable here and closer monitoring is necessary. Consequently, our proposed scheme is decentralized, utilizing a set of I-DES based IDS. Also, IDS sends fabricated IPv6 datagrams with random payload as part of active probing, which signify the indicator events. Basically, a response to the probe datagram helps differentiate attack behaviour from the normal. A malicious node behaviour is distinguishable if the crafted datagram is forwarded, as opposed to the fact

that the datagram must be discarded in normal condition. Further, a decentralized I-DES based IDS framework is proposed to model the normal and attacker type specifications, such that an attacker (attack type) can be identified. Local I-diagnosers are constructed from the local I-DES models. They generate an alert when a malicious node is identified (corresponding attacker-k certain state is reached in the local diagnoser). Global I-diagnosis is ascertained when an attacker$_k$ certain state is reached by any local I-diagnoser. Moreover, we eliminate the localised node using the kill switch mechanism to secure the 6LoWPAN. Correctness and completeness of our solution is proved and we implement it in simulation as well as real testbed with a large number of 6LoWPAN nodes. The results are observed to be superior to existing works. Our scheme achieves minimum false positives and achieves more than 99.8% accuracy in identifying the malicious nodes. The scheme is energy efficient and takes lower detection time for attacker identification.

6.1.4 Contributions of Chapter 5:

CoAP is a web transfer protocol similar to HTTP that is specially designed to facilitate IoT-LLN. It uses User Datagram Protocol (UDP) as the transport layer protocol which is unreliable and lacks a handshaking mechanism. A malicious endpoint with read and write access may just spoof requests or responses and launch low overhead IP address spoofing attacks. Moreover, it even helps mount other complex Distributed Denial-of-Service (DDoS) attacks like amplification attacks. In the previous two contributions discussed, active probe packets are sent to differentiate the attack type behaviour from the normal behaviour. Accordingly, I-diagnosability framework has been successfully adopted and extended to guarantee identification on each of these occasions. But, to identify an IP spoofing attacker, using just a probe response is insufficient. In this contribution, we adopt and extend the I^2 (Induced I)-diagnosability framework where we define empowering and indicator events. A CoAP request/response IP spoofing attack detection and identification scheme is devised that utilizes active probing and I2-DES based IDS in this contribution. Our scheme uses a decentralized architecture with each IDS functioning independently. IDS use empowering events and an intelligent probing technique that together help distinguish normal and attack behaviour. Wired network intimation is used as the empowering events. Also, probe packets are spoofed CoAP request/response packets with random payload and contain resource Uri

information that is relevant only to a malfunctioning node. Our proposed mechanism ensures that an attacker is identified as soon as it responds to probe packet after a wired intimation has been communicated between IDS nodes. Attack node identification is based on the principle that a true attacker node will respond or behave such as to protect its resource content. Our IDS is capable of detecting all instances of request/response spoofing. Further, I^2-DES is used to model the normal and attack specifications. An I^2-DES diagnoser, constructed from the I^2-DES models, generates an alert when a malicious node is identified. We consequently prove the correctness and completeness of our scheme. Proposed method is implemented in Contiki Cooja, with a sufficiently large number of IoT nodes. On comparing our scheme to state-of-the-art approaches, our performance is found to be energy-efficient, having minimal false positives and achieving more than 99.2% accuracy with response time of approx 6 sec.

6.2 Scope of Future Work

In this dissertation, the countermeasures presented successfully detect certain classes of attacks prevalent in the device-level, network-level and application-level of the IoT ecosystem. Various DES notions and paradigms like Opacity, I-Diagnosability, decentralized I-diagnosability and I^2-Diagnosability have been utilised depending on the corresponding attack characteristics. The mechanisms demonstrated are effective in securing IoT systems against attacks and can well be applied and extended to secure cyber physical systems, softwares, etc. Listed below are some possible future research directions:

- In Chapter 2, we proposed an effective countermeasure against scan-based attacks on crypto-chips running an AES implementation. As part of the security analysis, we used Opacity while also considering only a particular instance of Hamming weight distribution (for example, when a plaintext difference $0x01$ is applied), which is a limitation of this work. It would be very interesting to analyze the security aspects of our countermeasure irrespective of any given instance of a Hamming distribution in the future. An implementation of multi-bit flip controller ensuring opacity might then suffice to thwart all state of the art differential scan attacks. We would also like to further investigate our defense countermeasure in the presence of advanced DfT

structures with test compression as well as on other block or stream ciphers in use.

- Chapter 3 deals with identification of attacker nodes that launch network layer attacks by illegitimately increasing RPL rank field and falsifying version field. Our scheme uses a 6BR root node IDS, a set of leaf agents and incorporates an intelligent probing technique using ICMPv6 packets to achieve the desired objective. However, it would be worth exploring techniques to strengthen our security countermeasure in cases when malfunctioning leaf agents are compromised. Our current solution can be further improved with the generation of optimal probe sequences. It would not only guarantee more improved response times but would also reduce complexity. Another field of considerable research is to improve on the placement of the agent leaf nodes which will ensure that the overhead is further reduced. Furthermore, our current solution as demonstrated in this thesis is good at circumventing the worst parent attack and decreased rank attack as well with some minor modifications. Among our future works, we plan to devise an improved solution that can identify the malicious node in the presence of other RPL attacks as well, such as the DODAG Information Solicitation attack, Black Hole attack and Distributed Denial-of-Service attack.

- Chapter 3, 4 and 5 presents DES-based IDS techniques that guarantee attacker identification in the IoT ecosystem. Our approach can be further strengthened by improving on parameter inference and selection of active nodes. IoT devices are characterized by their heterogeneity and dynamic connections, requiring adaptive security measures to effectively counter evolving cyber threats. The integration of machine learning techniques alongside the Discrete Event System (DES)-based Intrusion Detection Systems (IDS) framework is one avenue that is worth exploring towards strengthening IoT security. While leveraging DES models allows for precise simulation and control of IoT network behavior, ensuring efficient resource utilization and robust security, using machine learning can help to dynamically infer optimal parameters and identify key nodes for active monitoring by analyzing historical and real-time data. This will enhance the IDS's ability to detect and respond to anomalies swiftly.

- In Chapter 5, we present an I^2-DES based IDS framework that makes use of empow-

ering events along with indicator probe packets to identify a CoAP request response spoofing attacker, thereby providing IoT application layer security against such types of attacks. However, there remains a wide variety of CoAP attacks such as, the selective blocking attack, request delay attack, response delay and mismatch attacks that make use of UDP which is unreliable and is effective yet when DTLS is present. Such attacks cannot be prevented with our countermeasure as it is. In the future, we plan to come up with a Timed I^2-DES framework to resist such attacks.

- Industrial IoT (IIoT) is a major enablement in Cyber Physical Systems (CPS). Securing IIoT is crucial to the smooth functioning of various applications across domains like healthcare, manufacturing, aircraft maintenance, and other mission critical systems. Devices in IIoT do not suffer much from resource constrainment, yet security is critical in such systems with necessity for more quicker and accurate approaches. Message Queue Telemetry Transport (MQTT) is one of the most important application layer data protocols of IoT networks, especially for Industrial IoT environments. It is based on the publish-subscribe model with three participants, namely client (publisher), broker, client (subscriber). MQTT 5.0 protocol security is challenging since it implements authentication methods like SCRAM or Kerberos. It is worth exploring the normal and attack event sequences when MQTT is in place, to identify a malfunctioning node uniquely. DES frameworks can be extended to ensure secure operation of CPS under the IIoT Ecosystem when a MQTT attack is in place. In future, we would like to extend our approaches to MQTT security and further venture into distributed denial-of-service attack mitigation schemes using lightweight solutions.

- In Chapters 3, 4 and 5, we discuss IoT layer-specific attack countermeasures that incorporate techniques suitable to identify an attack node, given a particular attack type and a protocol. However, a comprehensive IDS-based framework is necessary to mitigate cross layer IoT attacks, that is any IoT attacker node can be identified correctly irrespective of the layer and protocol. However, designing such an unified DES-based IDS framework will be fairly challenging since it would result in a vast number of states and transitions. The future work in this direction can be fruitful if modular DES-based IDS frameworks are considered to handle the state-space explosion problem.

Bibliography

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future generation computer systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[2] O. Vermesan and P. Friess, *Internet of things: converging technologies for smart environments and integrated ecosystems*. River publishers, 2013.

[3] S. Madakam, V. Lake, V. Lake, V. Lake *et al.*, "Internet of things (iot): A literature review," *Journal of Computer and Communications*, vol. 3, no. 05, p. 164, 2015.

[4] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE communications surveys & tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[5] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[6] C. Doukas and I. Maglogiannis, "Bringing iot and cloud computing towards pervasive healthcare," in *2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*. IEEE, 2012, pp. 922–926.

[7] M. Yun and B. Yuxin, "Research on the architecture and key technology of internet of things (iot) applied on smart grid," in *2010 international conference on advances in energy engineering*. IEEE, 2010, pp. 69–72.

[8] X. Jia, Q. Feng, T. Fan, and Q. Lei, "Rfid technology and its applications in internet of things (iot)," in *2012 2nd international conference on consumer electronics, communications and networks (CECNet)*. IEEE, 2012, pp. 1282–1285.

[9] T. Winter, P. Thubert, A. Brandt, J. W. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J.-P. Vasseur, R. K. Alexander *et al.*, "RPL: IPv6 Routing Protocol for Low power and Lossy Networks," *rfc*, vol. 6550, pp. 1–157, 2012.

[10] M. R. Palattella, N. Accettura, X. Vilajosana, T. Watteyne, L. A. Grieco, G. Boggia, and M. Dohler, "Standardized protocol stack for the internet of (important) things," *IEEE communications surveys & tutorials*, vol. 15, no. 3, pp. 1389–1406, 2012.

[11] L. Wallgren, S. Raza, and T. Voigt, "Routing Attacks and Countermeasures in the RPL-Based Internet of Things," *International Journal of Distributed Sensor Networks*, vol. 9, no. 8, pp. 1–11, 2013.

[12] Z. Shelby, K. Hartke, and C. Bormann, "The constrained application protocol (coap)," 2014.

[13] P. J. Denning and T. G. Lewis, "Exponential laws of computing growth," *Communications of the ACM*, vol. 60, no. 1, pp. 54–65, 2016.

[14] A. Lohachab and B. Karambir, "Critical analysis of ddos—an emerging security threat over iot networks," *Journal of Communications and Information Networks*, vol. 3, pp. 57–78, 2018.

[15] B. Russell and D. Van Duren, *Practical Internet of Things Security: Design a security framework for an Internet connected ecosystem.* Packt Publishing Ltd, 2018.

[16] V. Sachidananda, S. Siboni, A. Shabtai, J. Toh, S. Bhairav, and Y. Elovici, "Let the cat out of the bag: A holistic approach towards security analysis of the internet of things," in *Proceedings of the 3rd ACM International Workshop on IoT Privacy, Trust, and Security*, 2017, pp. 3–10.

[17] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani, "Demystifying iot security: An exhaustive survey on iot vulnerabilities and a first empirical look on internet-scale iot exploitations," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2702–2733, 2019.

[18] M. B. Barcena and C. Wueest, "Insecurity in the internet of things," *Security response, symantec*, vol. 20, 2015.

[19] M. Zulkifli and Z. W. Mohd, "Attack on cryptography," *Comput. Secur*, vol. 12, no. 5, pp. 33–45, 2008.

[20] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-baiot—network-based detection of iot botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018.

[21] Z.-K. Zhang, M. C. Y. Cho, C.-W. Wang, C.-W. Hsu, C.-K. Chen, and S. Shieh, "Iot security: ongoing challenges and research opportunities," in *2014 IEEE 7th international conference on service-oriented computing and applications*. IEEE, 2014, pp. 230–234.

[22] E. C. Ngai, J. Liu, and M. R. Lyu, "On the intruder detection for sinkhole attack in wireless sensor networks," in *2006 IEEE international conference on communications*, vol. 8. IEEE, 2006, pp. 3383–3389.

[23] Q. Jing, A. V. Vasilakos, J. Wan, J. Lu, and D. Qiu, "Security of the internet of things: perspectives and challenges," *Wireless Networks*, vol. 20, pp. 2481–2501, 2014.

[24] I. Andrea, C. Chrysostomou, and G. Hadjichristofi, "Internet of things: Security vulnerabilities and challenges," in *2015 IEEE symposium on computers and communication (ISCC)*. IEEE, 2015, pp. 180–187.

[25] P. N. Mahalle, N. R. Prasad, and R. Prasad, "Threshold cryptography-based group authentication (tcga) scheme for the internet of things (iot)," in *2014 4th International Conference on Wireless Communications, Vehicular Technology, Information Theory and Aerospace & Electronic Systems (VITAE)*. IEEE, 2014, pp. 1–5.

[26] E. Cozzi, M. Graziano, Y. Fratantonio, and D. Balzarotti, "Understanding linux malware," in *2018 IEEE symposium on security and privacy (SP)*. IEEE, 2018, pp. 161–175.

[27] A. Mangino, M. S. Pour, and E. Bou-Harb, "Internet-scale insecurity of consumer internet of things: An empirical measurements perspective," *ACM Transactions on Management Information Systems (TMIS)*, vol. 11, no. 4, pp. 1–24, 2020.

[28] E. Anthi, A. Javed, O. Rana, and G. Theodorakopoulos, "Secure data sharing and analysis in cloud-based energy management systems," in *Cloud Infrastructures, Services, and IoT Systems for Smart Cities: Second EAI International Conference, IISSC*

2017 and CN4IoT 2017, Brindisi, Italy, April 20–21, 2017, Proceedings 2. Springer, 2018, pp. 228–242.

[29] A. K. Simpson, F. Roesner, and T. Kohno, "Securing vulnerable home iot devices with an in-hub security manager," in *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 2017, pp. 551–556.

[30] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013.

[31] M. F. Elrawy, A. I. Awad, and H. F. Hamed, "Intrusion detection systems for iot-based smart environments: a survey," *Journal of Cloud Computing*, vol. 7, no. 1, pp. 1–20, 2018.

[32] F. Y. Yavuz, Ü. Devrim, and G. Ensar, "Deep Learning for Detection of Routing Attacks in the Internet of Things," *International Journal of Computational Intelligence Systems*, vol. 12, no. 1, pp. 39–58, 2018.

[33] A. A. Diro and N. Chilamkurti, "Distributed Attack Detection Scheme using Deep Learning Approach for Internet of Things," *Future Generation Computer Systems*, vol. 82, pp. 761–768, 2018.

[34] M. Hossain, Y. Karim, and R. Hasan, "Secupan: A security scheme to mitigate fragmentation-based network attacks in 6lowpan," in *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*. ACM, 2018, pp. 307–318.

[35] A. Dvir, L. Buttyan *et al.*, "VeRA - Version Number and Rank Authentication in RPL," in *2011 IEEE Eighth International Conference on Mobile Ad-Hoc and Sensor Systems*. IEEE, 2011, pp. 709–714.

[36] H. Perrey, M. Landsmann, O. Ugus, M. Wählisch, and T. C. Schmidt, "TRAIL: Topology Authentication in RPL," in *Proceedings of the 2016 International Conference on Embedded Wireless Systems and Networks*, ser. EWSN '16. Junction Publishing, 2016, p. 59–64.

[37] M. Nikravan, A. Movaghar, and M. Hosseinzadeh, "A lightweight signcryption scheme for defense against fragment duplication attack in the 6LoWPAN networks," *Peer-to-Peer Networking and Applications*, vol. 12, no. 1, pp. 209–226, 2019.

[38] Z. A. Almusaylim, A. Alhumam, and N. Jhanjhi, "Proposing a Secure RPL based Internet of Things Routing Protocol: A Review," *Ad Hoc Networks*, vol. 101, pp. 1–17, 2020.

[39] A. Bang and U. P. Rao, "Embof-rpl: Improved rpl for early detection and isolation of rank attack in rpl-based internet of things," *Peer-to-Peer Networking and Applications*, pp. 1–24, 2022.

[40] A. Seyfollahi, M. Moodi, and A. Ghaffari, "MFO-RPL: A secure RPL-based routing protocol utilizing moth-flame optimizer for the IoT applications," *Computer Standards & Interfaces*, vol. 82, pp. 1–19, 2022.

[41] S. J. Johnston, M. Scott, and S. J. Cox, "Recommendations for securing Internet of Things devices using commodity hardware," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*. IEEE, 2016, pp. 307–310.

[42] F. Lin, "Opacity of discrete event systems and its applications," *Automatica*, vol. 47, no. 3, pp. 496–503, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109811000173

[43] A. Saboori and C. N. Hadjicostis, "Notions of security and opacity in discrete event systems," in *2007 46th IEEE Conference on Decision and Control*, 2007, pp. 5056–5061.

[44] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Diagnosability of Discrete-Event Systems," *IEEE Transactions on automatic control*, vol. 40, no. 9, pp. 1555–1575, 1995.

[45] S. H. Zad, R. H. Kwong, and W. M. Wonham, "Fault Diagnosis in Discrete-Event Systems: Framework and Model Reduction," *IEEE Transactions on Automatic Control*, vol. 48, no. 7, pp. 1199–1212, 2003.

[46] C. G. Cassandras, *Discrete Event Systems: Modeling and Performance Analysis*. CRC, 1993.

[47] N. Hubballi, S. Biswas, S. Roopa, R. Ratti, and S. Nandi, "LAN attack detection using Discrete Event Systems," *ISA transactions*, vol. 50, no. 1, pp. 119–130, 2011.

[48] M. Agarwal, S. Biswas, and S. Nandi, "Discrete event system framework for fault diagnosis with measurement inconsistency: case study of rogue dhcp attack," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 3, pp. 789–806, 2017.

[49] B. Yang, K. Wu, and R. Karri, "Secure scan: A design-for-test architecture for crypto chips," vol. 25, 07 2005, pp. 135– 140.

[50] ——, "Scan based side channel attack on dedicated hardware implementations of data encryption standard," in *2004 International Conferce on Test*, 2004, pp. 339–344.

[51] J. Da Rolt, A. Das, G. Di Natale, M.-L. Flottes, B. Rouzeyre, and I. Verbauwhede, "A scan-based attack on elliptic curve cryptosystems in presence of industrial design-for-testability structures," in *2012 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 2012, pp. 43–48.

[52] R. Nara, K. Satoh, M. Yanagisawa, T. Ohtsuki, and N. Togawa, "Scan-based side-channel attack against rsa cryptosystems using scan signatures," *IEICE Transactions*, vol. 93-A, pp. 2481–2489, 12 2010.

[53] A. A. Kamal and A. M. Youssef, "A scan-based side channel attack on the ntruencrypt cryptosystem," in *2012 Seventh International Conference on Availability, Reliability and Security*, 2012, pp. 402–409.

[54] D. Hely, F. Bancel, M. Flottes, and B. Rouzeyre, "Test control for secure scan designs," in *European Test Symposium (ETS'05)*, 2005, pp. 190–195.

[55] S. S. Ali, S. M. Saeed, O. Sinanoglu, and R. Karri, "Scan attack in presence of mode-reset countermeasure," in *2013 IEEE 19th International On-Line Testing Symposium (IOLTS)*, 2013, pp. 230–231.

[56] S. S. Ali, O. Sinanoglu, and R. Karri, "Test-mode-only scan attack using the boundary scan chain," in *2014 19th IEEE European Test Symposium (ETS)*, 2014, pp. 1–6.

[57] S. S. Ali, S. M. Saeed, O. Sinanoglu, and R. Karri, "Novel test-mode-only scan attack and countermeasure for compression-based scan architectures," *IEEE Transactions on*

Computer-Aided Design of Integrated Circuits and Systems, vol. 34, no. 5, pp. 808–821, 2015.

[58] J. Lee, M. Tehranipoor, C. Patel, and J. Plusquellic, "Securing designs against scan-based side-channel attacks," *IEEE Transactions on Dependable and Secure Computing*, vol. 4, no. 4, pp. 325–336, 2007.

[59] M. T. Rahman, D. Forte, and M. Tehranipoor, *Protection of Assets from Scan Chain Vulnerabilities Through Obfuscation*, 01 2017, pp. 135–158.

[60] Y. Atobe, Y. Shi, M. Yanagisawa, and N. Togawa, "Secure scan design with dynamically configurable connection," in *2013 IEEE 19th Pacific Rim International Symposium on Dependable Computing*, 2013, pp. 256–262.

[61] J. Dworak, A. Crouch, J. Potter, A. Zygmontowicz, and M. Thornton, "Don't forget to lock your sib: hiding instruments using p1687," in *2013 IEEE International Test Conference (ITC)*, 2013, pp. 1–10.

[62] J. Dworak, Z. Conroy, A. Crouch, and J. Potter, "Board security enhancement using new locking sib-based architectures," in *2014 International Test Conference*, 2014, pp. 1–10.

[63] Y. Sao, A. Riaz, S. Ahlawat, and S. S. Ali, "Evaluating security of new locking sib-based architectures," in *2022 IEEE European Test Symposium (ETS)*, 2022, pp. 1–6.

[64] A. Das, B. Ege, S. Ghosh, L. Batina, and I. Verbauwhede, "Security analysis of industrial test compression schemes," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 12, pp. 1966–1977, 2013.

[65] T. Yu, A. Cui, M. Li, and A. Ivanov, "A new decompressor with ordered parallel scan design for reduction of test data and test time," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2015, pp. 641–644.

[66] C. Liu and Y. Huang, "Effects of embedded decompression and compaction architectures on side-channel attack resistance," in *25th IEEE VLSI Test Symposium (VTS'07)*, 2007, pp. 461–468.

[67] J. Da Rolt, G. Di Natale, M.-L. Flottes, and B. Rouzeyre, "New security threats against chips containing scan chain structures," in *2011 IEEE International Symposium on Hardware-Oriented Security and Trust*, 2011, pp. 110–110.

[68] Y. Sao, K. Soundra Pandian, and S. Subidh Ali, "Revisiting the security of static masking and compaction: Discovering new vulnerability and improved scan attack on aes," in *2020 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, 2020, pp. 1–6.

[69] A. Cui, Y. Luo, and C.-H. Chang, "Static and dynamic obfuscations of scan data against scan-based side-channel attacks," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 2, pp. 363–376, 2017.

[70] Y. Sao and S. S. Ali, "Security analysis of state-of-the-art scan obfuscation technique," in *2021 IEEE 39th International Conference on Computer Design (ICCD)*, 2021, pp. 599–602.

[71] K. Z. Azar, H. M. Kamali, H. Homayoun, and A. Sasan, "From cryptography to logic locking: A survey on the architecture evolution of secure scan chains," *IEEE Access*, vol. 9, pp. 73 133–73 151, 2021.

[72] D. Zhang, M. He, X. Wang, and M. Tehranipoor, "Dynamically obfuscated scan for protecting ips against scan-based attacks throughout supply chain," in *2017 IEEE 35th VLSI Test Symposium (VTS)*, 2017, pp. 1–6.

[73] X. Wang, D. Zhang, M. He, D. Su, and M. Tehranipoor, "Secure scan and test using obfuscation throughout supply chain," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 9, pp. 1867–1880, 2018.

[74] L. Alrahis, M. Yasin, N. Limaye, H. Saleh, B. Mohammad, M. Al-Qutayri, and O. Sinanoglu, "Scansat: Unlocking static and dynamic scan obfuscation," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 4, pp. 1867–1882, 2021.

[75] R. Karmakar, S. Chattopadhyay, and R. Kapur, "A scan obfuscation guided design-for-security approach for sequential circuits," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 3, pp. 546–550, 2020.

[76] N. Limaye and O. Sinanoglu, "Dynunlock: Unlocking scan chains obfuscated using dynamic keys," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, 2020, pp. 270–273.

[77] M. S. Rahman, A. Nahiyan, F. Rahman, S. Fazzari, K. Plaks, F. Farahmandi, D. Forte, and M. Tehranipoor, "Security assessment of dynamically obfuscated scan chain against oracle-guided attacks," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 26, no. 4, pp. 1–27, mar 2021. [Online]. Available: https://doi.org/10.1145/3444960

[78] Z. Zhang, S. Reddy, I. Pomeranz, X. Lin, and J. Rajski, "Scan tests with multiple fault activation cycles for delay faults," in *24th IEEE VLSI Test Symposium*, 2006, pp. 6 pp.–348.

[79] A. Cui, M. Li, G. Qu, and H. Li, "A guaranteed secure scan design based on test data obfuscation by cryptographic hash," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 12, pp. 4524–4536, 2020.

[80] M. Da Silva, M.-L. Flottes, G. Di Natale, and B. Rouzeyre, "Preventing scan attacks on secure circuits through scan chain encryption," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 3, pp. 538–550, 2019.

[81] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, 2nd ed. New York: Springer, 2008.

[82] P. Ramadge and W. Wonham, "The control of discrete event systems," *Proceedings of the IEEE*, vol. 77, no. 1, pp. 81–98, 1989.

[83] D. Ray, S. Singh, S. S. Ali, and S. Biswas, "Co-relation scan attack analysis (cosaa) on aes: A comprehensive approach," in *2019 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 2019, pp. 1–6.

[84] Y. Sao, S. S. Ali, D. Ray, S. Singh, and S. Biswas, "Co-relation scan attack analysis (cosaa) on aes: A comprehensive approach," *Microelectronics Reliability*, vol. 123, p. 114216, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0026271421001827

[85] P. Pongle and G. Chavan, "A survey: Attacks on RPL and 6LoWPAN in IoT," in *2015 International conference on pervasive computing (ICPC)*. IEEE, 2015, pp. 1–6.

[86] A. Mayzaud, R. Badonnel, and I. Chrisment, "A Taxonomy of Attacks in RPL-based Internet of Things," *International Journal of Network Security*, vol. 18, no. 3, pp. 459–473, 2016.

[87] A. Mayzaud, A. Sehgal, R. Badonnel, I. Chrisment, and J. Schönwälder, "A Study of RPL DODAG Version Attacks," in *IFIP international conference on autonomous infrastructure, management and security*. Springer, 2014, pp. 92–104.

[88] C. Pu and L. Carpenter, "Digital Signature Based Countermeasure Against Puppet Attack in the Internet of Things," in *2019 IEEE 18th International Symposium on Network Computing and Applications (NCA)*. IEEE, 2019, pp. 1–4.

[89] C. Pu, J. Brown, and L. Carpenter, "A Theil Index-Based Countermeasure Against Advanced Vampire Attack in Internet of Things," in *2020 IEEE 21st International Conference on High Performance Switching and Routing (HPSR)*. IEEE, 2020, pp. 1–6.

[90] C. Pu and B. Groves, "Energy Depletion Attack in Low Power and Lossy Networks: Analysis and Defenses," in *2019 2nd International Conference on Data Intelligence and Security (ICDIS)*. IEEE, 2019, pp. 14–21.

[91] E. Y. Vasserman and N. Hopper, "Vampire Attacks: Draining Life from Wireless Ad Hoc Sensor Networks," *IEEE transactions on mobile computing*, vol. 12, no. 2, pp. 318–332, 2011.

[92] C. Pu and K.-K. R. Choo, "Lightweight Sybil Attack Detection in IoT based on Bloom Filter and Physical Unclonable Function," *computers & security*, vol. 113, p. 102541, 2022.

[93] C. Pu, "Spam DIS Attack Against Routing Protocol in the Internet of Things," in *2019 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2019, pp. 73–77.

[94] C. Pu and X. Zhou, "Suppression Attack Against Multicast Protocol in Low Power and Lossy Networks: Analysis and Defenses," *Sensors*, vol. 18, no. 10, p. 3236, 2018.

[95] A. Raoof, A. Matrawy, and C.-H. Lung, "Routing Attacks and Mitigation Methods for RPL-Based Internet of Things," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1582–1606, 2018.

[96] G. Glissa, A. Rachedi, and A. Meddeb, "A Secure Routing Protocol Based on RPL for Internet of Things," in *2016 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2016, pp. 1–7.

[97] M. Nikravan, A. Movaghar, and M. Hosseinzadeh, "A Lightweight Defense Approach to Mitigate Version Number and Rank Attacks in Low-Power and Lossy Networks," *Wireless Personal Communications*, vol. 99, no. 2, pp. 1035–1059, 2018.

[98] M. Zaminkar, F. Sarkohaki, and R. Fotohi, "DSH-RPL: A Method based on Encryption and Node Rating for Securing the RPL protocol Communications in the IoT Ecosystem," *International Journal of Communication Systems*, vol. 34, no. 3, pp. 1–24, 2021.

[99] D. Airehrour, J. A. Gutierrez, and S. K. Ray, "SecTrust-RPL: A secure trust-aware RPL routing protocol for Internet of Things," *Future Generation Computer Systems*, vol. 93, pp. 860–876, 2019.

[100] K. Iuchi, T. Matsunaga, K. Toyoda, and I. Sasase, "Secure Parent Node Selection Scheme in Route Construction to Exclude Attacking Nodes From RPL Network," in *2015 21st Asia-Pacific Conference on Communications (APCC)*. IEEE, 2015, pp. 299–303.

[101] N. Djedjig, D. Tandjaoui, F. Medjek, and I. Romdhani, "Trust-aware and cooperative routing protocol for IoT security," *Journal of Information Security and Applications*, vol. 52, pp. 1–25, 2020.

[102] M. Osman, J. He, F. M. M. Mokbal, N. Zhu, and S. Qureshi, "ML-LGBM: A Machine Learning Model Based on Light Gradient Boosting Machine for the Detection of Version Number Attacks in RPL-Based Networks," *IEEE Access*, vol. 9, pp. 83 654–83 665, 2021.

[103] S. Cakir, S. Toklu, and N. Yalcin, "RPL Attack Detection and Prevention in the Internet of Things Networks Using a GRU Based Deep Learning," *IEEE Access*, vol. 8, pp. 183 678–183 689, 2020.

[104] M. Osman, J. He, F. M. M. Mokbal, and N. Zhu, "Artificial Neural Network Model for Decreased Rank Attack Detection in RPL Based on IoT Networks," *International Journal of Network Security*, vol. 23, no. 3, pp. 496–503, 2021.

[105] F. Barbhuiya, M. Agarwal, S. Purwar, S. Biswas, and S. Nandi, "Application of Stochastic Discrete Event System Framework for Detection of Induced Low Rate TCP Attack," *Isa Transactions*, vol. 58, pp. 474–492, 2015.

[106] V. Ramachandran and S. Nandi, "Detecting ARP Spoofing: An Active Technique," in *Information Systems Security: First International Conference, ICISS 2005, Kolkata, India, December 19-21, 2005. Proceedings 1.* Springer, 2005, pp. 239–250.

[107] F. A. Barbhuiya, S. Biswas, and S. Nandi, "An active host-based intrusion detection system for ARP-related attacks and its verification," *arXiv preprint arXiv:1306.1332*, 2013.

[108] M. Humayun, N. Jhanjhi, and M. Alamri, "Smart Secure and Energy Efficient Scheme for E-Health Applications using IoT: A Review," *International Journal of Computer Science and Network Security*, vol. 20, no. 4, pp. 55–74, 2020.

[109] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1125–1142, 2017.

[110] G. Montenegro, N. Kushalnagar, J. Hui, D. Culler *et al.*, "Transmission of ipv6 packets over ieee 802.15. 4 networks," *Internet proposed standard RFC*, vol. 4944, p. 130, 2007.

[111] E. Kim, D. Kaspar, and J. Vasseur, "Design and application spaces for ipv6 over low-power wireless personal area networks (6lowpans)," *RFC6568*, 2012.

[112] R. Hummen, J. Hiller, H. Wirtz, M. Henze, H. Shafagh, and K. Wehrle, "6lowpan fragmentation attacks and mitigation mechanisms," in *Proceedings of the sixth ACM*

conference on Security and privacy in wireless and mobile networks. ACM, 2013, pp. 55–66.

[113] H. Kim, "Protection against packet fragmentation attacks at 6lowpan adaptation layer," in *2008 International Conference on Convergence and Hybrid Information Technology*. IEEE, 2008, pp. 796–801.

[114] C. Bormann, A. P. Castellani, and Z. Shelby, "Coap: An application protocol for billions of tiny internet nodes," *IEEE Internet Computing*, vol. 16, no. 2, pp. 62–67, 2012.

[115] F. Maggi, R. Vosseler, and D. Quarta, "The fragility of industrial iot's data backbone," *Trend Micro Inc*, 2018.

[116] H. Wang, C. Jin, and K. G. Shin, "Defense against spoofed ip traffic using hop-count filtering," *IEEE/ACM Transactions on networking*, vol. 15, no. 1, pp. 40–53, 2007.

[117] P. Bhale, S. Biswas, and S. Nandi, "An adaptive and lightweight solution to detect mixed rate ip spoofed ddos attack in iot ecosystem," in *2018 15th IEEE India Council International Conference (INDICON)*. IEEE, 2018, pp. 1–6.

[118] Y. Gilad and A. Herzberg, "Off-path attacking the web." in *WOOT*, 2012, pp. 41–52.

[119] Y. Gilad, A. Herzberg, and H. Shulman, "Off-path hacking: The illusion of challenge-response authentication," *IEEE Security & Privacy*, vol. 12, no. 5, pp. 68–77, 2013.

[120] E. Fernandes, J. Jung, and A. Prakash, "Security analysis of emerging smart home applications," in *2016 IEEE symposium on security and privacy (SP)*. IEEE, 2016, pp. 636–654.

[121] A. G. Roselin, P. Nanda, S. Nepal, X. He, and J. Wright, "Exploiting the remote server access support of coap protocol," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 9338–9349, 2019.

[122] C. G. Cassandras, S. Lafortune *et al.*, *Introduction to discrete event systems*. Springer, 2008, vol. 2.

[123] R. Nara, N. Togawa, M. Yanagisawa, and T. Ohtsuki, "A Scan-Based Attack Based on Discriminators for AES Cryptosystems," *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, vol. 92, no. 12, pp. 3229–3237, Jan. 2009.

[124] H. Kodera, M. Yanagisawa, and N. Togawa, "Scan-based attack against des cryptosystems using scan signatures," in *2012 IEEE Asia Pacific Conference on Circuits and Systems*, 2012, pp. 599–602.

[125] A. Saboori and C. N. Hadjicostis, "Verification of initial-state opacity in security applications of des," in *2008 9th International Workshop on Discrete Event Systems*, 2008, pp. 328–333.

[126] B. Zhang, S. Shu, and F. Lin, "Maximum information release while ensuring opacity in discrete event systems," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 3285–3290.

[127] J. Dubreil, P. Darondeau, and H. Marchand, "Supervisory control for opacity," *IEEE Transactions on Automatic Control*, vol. 55, no. 5, pp. 1089–1100, 2010.

[128] "AES IP Core," Dec 2020, https://opencores.org/projects/aes_core.

[129] L. Alrahis, M. Yasin, H. Saleh, B. Mohammad, M. Al-Qutayri, and O. Sinanoglu, "Scansat: Unlocking obfuscated scan chains," in *ASPDAC*. ACM, 2019, pp. 352–357.

[130] M. Surendar and A. Umamakeswari, "InDReS: An Intrusion Detection and response system for Internet of Things with 6LoWPAN," in *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*. IEEE, 2016, pp. 1903–1908.

[131] A. Le, J. Loo, K. K. Chai, and M. Aiash, "A Specification-Based IDS for Detecting Attacks on RPL-Based Network Topology," *Information*, vol. 7, no. 2, pp. 1–19, 2016.

[132] S. Raza, L. Wallgren, and T. Voigt, "Svelte: Real-time intrusion detection in the internet of things," *Ad hoc networks*, vol. 11, no. 8, pp. 2661–2674, 2013.

[133] T. ul Hassan, M. Asim, T. Baker, J. Hassan, and N. Tariq, "CTrust-RPL: A control layer-based trust mechanism for supporting secure routing in routing protocol for low

power and lossy networks-based Internet of Things applications," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 3, p. e4224, 2021.

[134] S. Y. Hashemi and F. Shams Aliee, "Dynamic and comprehensive trust model for IoT and its integration into RPL," *The Journal of Supercomputing*, vol. 75, no. 7, pp. 3555–3584, 2019.

[135] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in Internet of Things," *Journal of Network and Computer Applications*, vol. 84, pp. 25–37, 2017.

[136] N. Mishra and S. Pandya, "Internet of Things Applications, Security Challenges, Attacks, Intrusion Detection, and Future Visions: A Systematic Review," *IEEE Access*, vol. 9, pp. 59 353–59 377, 2021.

[137] S. U. Jan, S. Ahmed, V. Shakhov, and I. Koo, "Toward a Lightweight Intrusion Detection System for the Internet of Things," *IEEE Access*, vol. 7, pp. 42 450–42 471, 2019.

[138] A. Althubaity, T. Gong, K.-K. Raymond, M. Nixon, R. Ammar, and S. Han, "Specification-based Distributed Detection of Rank-related Attacks in RPL-based Resource-Constrained Real-Time Wireless Networks," in *2020 IEEE Conference on Industrial Cyberphysical Systems (ICPS)*, vol. 1. IEEE, 2020, pp. 168–175.

[139] U. Shafique, A. Khan, A. Rehman, F. Bashir, and M. Alam, "Detection of rank attack in routing protocol for Low Power and Lossy Networks," *Annals of Telecommunications*, vol. 73, no. 7, pp. 429–438, 2018.

[140] A. D. Seth, S. Biswas, and A. K. Dhar, "LDES: Detector Design for Version Number Attack Detection using Linear Temporal Logic based on Discrete Event System," *International Journal of Information Security*, pp. 1–25, 2023.

[141] H. Sedjelmaci, S. M. Senouci, and M. Al-Bahri, "A Lightweight Anomaly Detection Technique for Low-Resource IoT Devices: A Game-Theoretic Methodology," in *2016 IEEE international conference on communications (ICC)*. IEEE, 2016, pp. 1–6.

[142] C. Cervantes, D. Poplade, M. Nogueira, and A. Santos, "Detection of Sinkhole Attacks for Supporting Secure Routing on 6LoWPAN for Internet of Things," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 2015, pp. 606–611.

[143] Z. A. Khan and P. Herrmann, "A Trust Based Distributed Intrusion Detection Mechanism for Internet of Things," in *2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)*. IEEE, 2017, pp. 1169–1176.

[144] A. Mayzaud, R. Badonnel, and I. Chrisment, "A Distributed Monitoring Strategy for Detecting Version Number Attacks in RPL-Based Networks," *IEEE Transactions on Network and Service Management*, vol. 14, no. 2, pp. 472–486, 2017.

[145] A. Agiollo, M. Conti, P. Kaliyar, T.-N. Lin, and L. Pajola, "DETONAR: Detection of Routing Attacks in RPL-Based IoT," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1178–1190, 2021.

[146] D. Ray, P. Bhale, S. Biswas, S. Nandi, and P. Mitra, "DAISS: Design of an Attacker Identification Scheme in CoAP Request/Response Spoofing," in *TENCON 2021-2021 IEEE Region 10 Conference (TENCON)*. IEEE, 2021, pp. 941–946.

[147] J. Yi, T. Clausen, and Y. Igarashi, "Evaluation of Routing Protocol for Low Power and Lossy Networks: LOADng and RPL," in *2013 IEEE Conference on wireless sensor (ICWISE)*. IEEE, 2013, pp. 19–24.

[148] J. Vasseur, M. Kim, K. Pister, N. Dejean, and D. Barthel, "Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks," in *RFC 6551*. IETF, 2012, pp. 1–30.

[149] P. Levis, T. Clausen, J. Hui, O. Gnawali, and J. Ko, "The Trickle Algorithm," *Internet Engineering Task Force, RFC6206*, 2011.

[150] D. Ray, P. Bhale, S. Biswas, P. Mitra, and S. Nandi, "A novel energy-efficient scheme for rpl attacker identification in iot networks using discrete event modeling," *IEEE Access*, vol. 11, pp. 77 267–77 291, 2023.

[151] D. Ray, P. Bhale, S. Biswas, S. Nandi, and P. Mitra, "ArsPAN: Attacker Revelation Scheme using Discrete Event System in 6LoWPAN based Buffer Reservation Attack," in *2020 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*. IEEE, 2020, pp. 1–6.

[152] K.-T. Cheng and A. S. Krishnakumar, "Automatic Functional Test Generation Using The Extended Finite State Machine Model," in *30th ACM/IEEE Design Automation Conference*. IEEE, 1993, pp. 86–91.

[153] R. Sekar, M. Bendre, D. Dhurjati, and P. Bollineni, "A Fast Automaton-Based Method for Detecting Anomalous Program Behaviors," in *Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001*. IEEE, 2000, pp. 144–155.

[154] R. Sekar, A. Gupta, J. Frullo, T. Shanbhag, A. Tiwari, H. Yang, and S. Zhou, "Specification-based Anomaly Detection: A New Approach for Detecting Network Intrusions," in *Proceedings of the 9th ACM conference on Computer and communications security*, 2002, pp. 265–274.

[155] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - A Lightweight and Flexible Operating System for Tiny Networked Sensors," in *International conference on local computer networks*, 2004, pp. 455–462.

[156] C. Adjih, E. Baccelli, E. Fleury, G. Harter, N. Mitton, T. Noel, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele *et al.*, "FIT IoT-LAB: A large scale open experimental IoT testbed," in *World Forum on Internet of Things (WF-IoT)*, 2015, pp. 459–464.

[157] S. C. R. or Save System Activity Information, Feb. 2019. [Online]. Available: https://dx.doi.org/10.21227/fesz-dm97

[158] A. Tirumala, "Iperf: The TCP/UDP bandwidth measurement tool," *http://dast.nlanr.net/Projects/Iperf/*, 1999.

[159] S. Nayak, N. Ahmed, and S. Misra, "Deep Learning-Based Reliable Routing Attack Detection Mechanism for Industrial Internet of Things," *Ad Hoc Networks*, vol. 123, pp. 1–11, 2021.

[160] I. S. Alsukayti and A. Singh, "A Lightweight Scheme for Mitigating RPL Version Number Attacks in IoT Networks," *IEEE Access*, 2022.

[161] A. Le, J. Loo, Y. Luo, and A. Lasebae, "Specification-based IDS for securing RPL from topology attacks," in *2011 IFIP Wireless Days (WD)*. IEEE, 2011, pp. 1–3.

[162] S. Sharma and V. K. Verma, "Security explorations for routing attacks in low power networks on internet of things," *The Journal of Supercomputing*, pp. 1–35, 2020.

[163] Z. A Almusaylim, N. Jhanjhi, and A. Alhumam, "Detection and Mitigation of RPL Rank and Version Number Attacks in the Internet of Things: SRPL-RP," *Sensors*, vol. 20, no. 21, pp. 1–25, 2020.

[164] R. Sahay, G. Geethakumari, and B. Mitra, "A novel blockchain based framework to secure IoT-LLNs against routing attacks," *Computing*, vol. 102, no. 11, pp. 2445–2470, 2020.

[165] H. Neminath, S. Biswas, S. Roopa, R. Ratti, S. Nandi, F. Barbhuiya, A. Sur, and V. Ramachandran, "A DES approach to Intrusion Detection System for ARP Spoofing Attacks," in *18th Mediterranean Conference on Control and Automation, MED'10*. IEEE, 2010, pp. 695–700.

[166] F. A. Barbhuiya, S. Biswas, N. Hubballi, and S. Nandi, "A Host Based DES Approach for Detecting ARP Spoofing," in *2011 IEEE Symposium on Computational Intelligence in Cyber Security (CICS)*. IEEE, 2011, pp. 114–121.

[167] A. Le, J. Loo, A. Lasebae, A. Vinel, Y. Chen, and M. Chai, "The Impact of Rank Attack on Network Topology of Routing Protocol for Low-Power and Lossy Networks," *IEEE Sensors Journal*, vol. 13, no. 10, pp. 3685–3692, 2013.

[168] Z. Shelby and C. Bormann, *6LoWPAN: The wireless embedded Internet*. John Wiley & Sons, 2011, vol. 43.

[169] H.-x. Hu, A.-l. Gehin, and M. Bayart, "A state-based approach for decentralized fault diagnosis in discrete-event systems," *IFAC Proceedings Volumes*, vol. 40, no. 1, pp. 142–147, 2007.

[170] J. J. Kponyo, J. O. Agyemang, G. S. Klogo, and J. O. Boateng, "Lightweight and host-based denial of service (dos) detection and defense mechanism for resource-constrained iot devices," *Internet of Things*, vol. 12, p. 100319, 2020.

[171] P. Kumar, H. Bagga, B. S. Netam, and V. Uduthalapally, "Sad-iot: Security analysis of ddos attacks in iot networks," *Wireless Personal Communications*, pp. 1–22, 2021.

Appendix

A Discrete Event System Definitions

Due to certain measurement limitations, some events cannot be measured. Such events are called unmeasurable events. The event set can be expressed as a disjoint union of measurable and unmeasurable events. In notation, $\Sigma = \Sigma_m \cup \Sigma_{um}$.

Definition 1. Measurable and Unmeasurable transitions A transition, τ, that is enabled under the influence of an event σ is said to be measurable if the corresponding event, σ, is measurable. Similarly, a transition associated with an unmeasurable event is said to be an unmeasurable transition. $\Im_m$ and $\Im_{um}$ denote the set of measurable and unmeasurable transitions.

Definition 2. Measurement equivalent transitions (states) A pair of transitions $\tau_1 = \langle x_1,\ x_1^+,\ \sigma_1,\ \phi_1(V),\ \Phi_1(C),\ Reset_1(C),\ Assign_1(V)\rangle$ and $\tau_2 = \langle x_2,\ x_2^+,\ \sigma_2,\ \phi_2(V),\ \Phi_2(C),\ Reset_2(C),\ Assign_2(V)\rangle$ are said to be measurement equivalent *iff* $\sigma_1 = \sigma_2$, $\phi_1(V) = \phi_2(V)$, $\Phi_1(C) = \Phi_2(C)$, $Reset_1(C) = Reset_2(C)$ and $Assign_1(V) = Assign_2(V)$. If a pair of transitions are equivalent, then their source states and destination states are equivalent states pair-wise. In simple terms, if the system current state is an initial state of a transition that has at least one more equivalent state, then the final states reached, from each of these states due to an equivalent transition, are also equivalent.

Definition 3. Projection and Inverse Projection Operator A projection operator $P : \Im^* \rightarrow \Im_m^*$ is defined as: $P(\epsilon) = \epsilon$ (null string); $P(\tau) = \tau$ if $\tau \in \Im_m$; $P(\tau) = \epsilon$ if $\tau \in \Im_{um}$; $P(s\tau) = P(s)P(\tau)$, where $s \in L_f(H), \tau \in \Im$. The function P erases the unmeasurable transitions from the argument finite trace. $P(s)$ is termed as the measurable finite trace corresponding to the finite trace s.

Definition 4. Normal H-state (H-transition) and Faulty (Attack) H-state (H-transition) States that are traversed by the system when operating without any fault (attack) are known as Normal H-states. X_N denotes the set of all normal states. A H-transition $\langle x, x^+\rangle$ is called a normal H-transition if $x, x^+ \in X_N$. States that are traversed by the system when operating under faulty (attack) circumstances are known as faulty H-states (attack H-states). X_{F_i} denotes the set of all faulty states. A H-transition $\langle x, x^+\rangle$ is called a faulty H-transition if $x, x^+ \in X_{F_i}$. Analogously, X_{A_i} denotes the set of all attack states. A H-transition $\langle x, x^+\rangle$ is called an attack H-transition if $x, x^+ \in X_{A_i}$.

B DES Diagnosability

Definition 5. F_i-Diagnosability Let $\Psi(X_{F_i}) = \{s | s \in L_f(H)$ and $final(s) \in X_{F_i}$ and s ends in a measurable transition$\}$. A DES model H is said to be diagnosable for fault F_i *iff* the following holds:

$$(\exists n_j \in \mathcal{N})[\forall s \in \Psi(X_{F_i})](\forall t \in L_f(G)/s)[|t| \geq n_j) \Rightarrow D] \quad (1)$$

where, D is $\forall x \in \{P^{-1}[P(st)]\}$, $final(x) \in X_{F_i}$.

Definition 6. F_i-I-**Diagnosability** Let $\Psi(X_{F_i}) = \{s|s \in L_f(H)$ and $final(s) \in X_{F_i}$ and s ends in a measurable transition$\}$. A I-DES model H is said to be I-diagnosable for fault F_i *iff* the following holds:

$$(\exists n_j \in \mathcal{N})[\forall s \in \Psi(X_{F_i})](\forall it \in L_f(G)/s)[|t| \geq n_j) \Rightarrow D] \quad (2)$$

where, D is $\forall x \in \{P^{-1}[P(sit)]\}$, $final(x) \in X_{F_i}$.

Suppose s is a finite trace in H containing a F_i state and i is an indicator transition and t is continuation of trace si that is sufficiently long. To satisfy the I-Diagnosability condition D then requires every trace that is measurement equivalent to sit to end in a F_i state. This implies that in absence of F_i-I-indeterminate cycles (A_i-I-indeterminate cycles), along every continuation it of s, occurrence of a fault F_i (attacker A_i) can be detected within at most n system transitions following si.

Definition 7. F_i-I^2-**Diagnosability** Let $\Psi(X_{F_i}) = \{s|s \in L_f(H)$ and $final(s) \in X_{F_i}$ and s ends in a measurable transition$\}$. A I^2-DES model H is said to be I^2-diagnosable for fault F_i *iff* the following holds:

$$(\exists n_j \in \mathcal{N})[\forall s \in \Psi(X_{F_i})](\forall ejit \in L_f(G)/s)[|t| \geq n_j) \Rightarrow D] \quad (3)$$

where, D is $\forall x \in \{P^{-1}[P(sejitt)]\}$, $final(x) \in X_{F_i}$.

Implications of s,i,t are the same as in Definition 6. e denotes a transition consisting of an empowering event and j is an optional trace that occurs after e and till indicator transition i. To satisfy the I^2-diagnosability condition requires e to ensure that i sensitizes the fault F_i (attacker A_i) occurrence within at most n system transitions following the trace $seji$.

Construction of the diagnoser:

DES diagnosers give estimates of system states that are modeled using a DES. Consequently, DES I-diagnosers and DES I^2-diagnosers are modeled from I-DES consisting of indicator transitions and I^2-DES consisting of empowering events apart from indicator transitions. Each of these diagnosers are represented as a directed graph, $O = \langle Z, A, Z_0 \rangle$. Here, Z is the set consisting of the states of the diagnoser O, called O-*states*, Z_0 is the set of initial O-states of the diagnoser and A is the set consisting of the transitions (edges) of the diagnoser, called O-*transitions*, where $A \subseteq Z \times Z$. Each O-state z is an estimate of the actual system state and consists of one or more states of DES H, $z \in 2^X$, the power set of X, signifying membership uncertainty. On a similar note, each of the O-transition a consists of one or more measurement equivalent transition of modeled DES H and represents an uncertainty in the actual measurable transition that takes place. They are of the form $\langle z_i, z_f \rangle$. We denote the unmeasurable successor set of a state set X as $\mathcal{U}(X)$ and is defined as $\mathcal{U}(X) = \bigcup_{x \in X}\{x^+|\tau = \langle x, x^+ \rangle \in \Im_u\}$. The unmeasurable reach of a state set X, $\mathcal{U}^*(X)$, is the reflexive-transitive closure of $\mathcal{U}(X)$.

The sets we consider are finite sets. To construct the diagnoser, transitions and states

are appended to the diagnoser based on the measurable system traces from the initial set of states. The set of states contained in an initial O-state are the initial states of DES H and the states that are reachable from each of those initial states using sequences of unmeasurable transitions, i.e., $\{X_0 \cup \mathcal{U}(X_0)\}$. The initial O-state thus comprises of states that belong to normal state set or any attacker type state from H, because an attack transition is inherently unmeasurable. Consequently, any O-state may comprise of equivalent states from normal as well as attacker type states. Hence, all states included in O are measurable equivalent. The O-transitions on the other hand are a set of equivalent H-transitions which are directed from a set of equivalent source states to a set of equivalent destination states. Any O-transition can therefore take either of these following forms:

- $\langle (x_a, x_a^+), (x_b, x_b^+) \rangle$ if $\langle x_a, x_b, \sigma_a, \phi_a(V), \Phi_a(C), Assign_a(V), Reset_a(C) \rangle \equiv \langle x_a^+, x_b^+, \sigma_{a^+}, \phi_{a^+}(V), \Phi_{a^+}(C), Assign_{a^+}(V), Reset_{a^+}(C) \rangle$
- $\langle (x_a, x_a^+), (x_b) \rangle$, $\langle (x_a, x_a^+), (x_b^+) \rangle$ if $\langle x_a, x_b, \sigma_a, \phi_a(V), \Phi_a(C), Assign_a(V), Reset_a(C) \rangle \not\equiv \langle x_a^+, x_b^+, \sigma_{a^+}, \phi_{a^+}(V), \Phi_{a^+}(C), Assign_{a^+}(V), Reset_{a^+}(C) \rangle$ and $x_a \equiv x_a^+$
- $\langle (x_a), (x_b) \rangle \langle (x_a^+), (x_b^+) \rangle$, otherwise.

In Algorithm 14, the step-wise procedure for a DES diagnoser construction is shown. Their construction procedure remains the same irrespective of indicator transitions or empowering transitions in the corresponding DES model.

ALGORITHM 14: Diagnoser construction O for DES model H

Input: DES model H
Output: DES Diagnoser
/* PARTITION $X_0 \rightarrow$ Measurement equivalent classes, X_{01}, X_{02}, ... , X_{0m} */
1 **for** *all* i, $1 \leq i \leq m$ **do**
2 $z_{0i} \leftarrow \mathcal{U}^*(X_{0i})$
3 $Z_0 \leftarrow z_{01} \cup \cdots \cup z_{0m}$
4 $Z \leftarrow Z_0$
5 $A \leftarrow \phi$
6 **for** *all* $z \in Z$ **do**
  /* Find the set of measurable H-transitions ($\Im_{mz}$) outgoing from z */
7 $\Im_{mz} \leftarrow \{\tau | \tau \in \Im_m \wedge initial(\tau) \in z\}$
  /* Find the set of all measurement equivalent classes A_z, of $\Im_{mz}$ */
8 **for** *all* $a \in A_z$ **do**
9 $z_a^+ = \{final(\tau) | \tau \in a\}$
10 $z^+ = \mathcal{U}^*(z_a^+)$
11 $Z \leftarrow Z \cup \{z^+\}$
12 $A = A \cup \{a\}$

C Diagnosability Definitions

F_i-certain O-state (A_i-certain O-state) and F_i-uncertain O-state (A_i-uncertain O-state) are two types of diagnoser states that relate to occurrence of a fault type F_i (attacker type A_i). Following are some of the definitions that pertain to the diagnoser:

Definition 8. Normal certain O-state A O-state that consists of states in H, all of which only belong to X_N.

Definition 9. Fault certain O-state (Attack certain O-state) Given n fault (attacker) types, a O-state that consists of states in H, all of which only belong to $\bigcup_{i\in n} F_i$ ($\bigcup_{i\in n} A_i$), purely.

Definition 10. F_i-certain O-state (A_i-certain O-state) A O-state that consists of states in H, all of which only belong to X_{F_i} (X_{A_i}).

Definition 11. F_i-uncertain O-state (A_i-uncertain O-state) A O-state that consists of states that may belong to F_i-H-states as well as states of DES H other than the fault type F_i (attacker type A_i).

Definition 12. F_i-uncertain cycle (A_i-uncertain cycle) It is defined as a cycle of O-states in which there are no F_i-certain O-state (A_i-certain O-state).

Definition 13. Indicator transition (I-transition) A transition containing a measurable event, called an indicator event, that sensitizes a fault (attack).

Definition 14. F_i-indeterminate cycle (A_i-indeterminate cycle) It is defined as a cycle of F_i-uncertain O-states (A_i-uncertain O-states) such that the transitions constituting this cycle also form a cycle in H using only non-F_i-states (non-A_i-states). A F_i-indeterminate cycle (A_i-indeterminate cycle) is therefore a special case of F_i-uncertain cycle (A_i-uncertain cycle).

Lemma: Existence of a F_i-indeterminate cycle (A_i-indeterminate cycle) renders non-diagnosability

Proof: Existence of a F_i-indeterminate cycle (A_i-indeterminate cycle) in diagnoser O implies the presence of at least two measurement equivalent traces in H, one consisting of only F_i-states (A_i-states) and another comprising of non-F_i-states (non-A_i-states). Therefore, system traces executed while in a F_i-indeterminate cycle (A_i-indeterminate cycle) imply that variables measured are identical in both normal and fault (attack) conditions. Hence, the estimates of the diagonser while entering and moving along such a cycle means non-diagnosability of fault F_i (attack A_i), since throughout the cycle transitions it remains uncertain if F_i (A_i) or non-F_i (non-A_i) occurs each time and as it is assumed that the faults (attacks) are permanent, the cycle may thus never be exited.

List of Publications

Publications from Thesis Work:

Refereed Journals

1. **Dipojjwal Ray***, Pradeepkumar Bhale, Santosh Biswas, Pinaki Mitra and Sukumar Nandi, "A Novel Energy-Efficient Scheme for RPL Attacker Identification in IoT Networks Using Discrete Event Modeling", *IEEE Access*, vol. 11, pp. 77267-77291, 2023.
   DOI: 10.1109/ACCESS.2023.3296558

2. **Dipojjwal Ray***, Yogendra Sao, Santosh Biswas, and Sk Subidh Ali, "On Securing Cryptographic ICs against Scan-based Attacks: A Hamming Weight Distribution Perspective", *ACM Journal of Emerging Technologies and Computing Systems (JETC)*, vol 19, Issue 2, Article 10 (April2023).
   DOI: https://doi.org/10.1145/35772152

Refereed Conferences

3. **Dipojjwal Ray***, Pradeepkumar Bhale, Santosh Biswas, Sukumar Nandi and Pinaki Mitra, "DAISS: Design of an Attacker Identification Scheme in CoAP Request/Response Spoofing", *IEEE Region 10 Conference (TENCON 2021)*, Auckland, NewZealand, 2021, pp.941-946. DOI: 10.1109/TENCON54134.2021.9707405

4. **Dipojjwal Ray***, Pradeepkumar Bhale, Santosh Biswas, Sukumar Nandi and Pinaki Mitra, "ArsPAN: Attacker Revelation Scheme using Discrete Event System in 6LoWPAN based Buffer Reservation Attack", *2020 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, New Delhi, India, 2020, pp.1-6.
   DOI: 10.1109/ANTS50601.2020.9342842

5. **Dipojjwal Ray***, Siddharth Singh, Sk Subidh Ali and Santosh Biswas, "Co-relation Scan Attack Analysis(COSAA) on AES: A Comprehensive Approach", *2019 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS)*, Noordwijk, Netherlands, 2019, pp.1-6.
   DOI: 10.1109/DFT.2019.8875272

Journal Submission Under Review

6. **Dipojjwal Ray***, Pradeepkumar Bhale, Santosh Biswas, Pinaki Mitra, Sukumar Nandi, "Fragment duplication attacker identification in 6LoWPAN using an energy-efficient probing scheme", *Wiley Transactions on Emerging Telecommunication Technolgies (TOETT)*.

7. **Dipojjwal Ray***, Santosh Biswas, Pinaki Mitra, "Design of an Attacker Identification Scheme in CoAP Request/Response Spoofing using CON messages", *IEEE IoT journal*.

Publications other than Thesis Work:

1. Yogendra Sao, Sk Subidh Ali, **Dipojjwal Ray***, Siddharth Singh, Santosh Biswas, "Co-relation scan attack analysis(COSAA) on AES: A comprehensive approach", *Microelectronics Reliability, Elsevier*, Volume 123, 2021, 114216.

2. Pradeepkumar Bhale, **Dipojjwal Ray**, Santosh Biswas and Sukumar Nandi, "WOMN: WOrMhole Attack DetectioN and Mitigation Using Lightweight Distributed IDS in IoT Network", *2023 IEEE Guwahati Subsection Conference (GCON)*, Guwahati, India, 2023, pp. 01-06.
DOI: 10.1109/GCON58516.2023.10183505

Doctoral Committee

Chairperson:
Prof. Jatindra Kumar Deka
Professor
Department of Computer Science and Engineering
Indian Institute of Technology Guwahati

Research Advisors:
Dr. Pinaki Mitra
Associate Professor
Department of Computer Science and Engineering
Indian Institute of Technology Guwahati

Prof. Santosh Biswas
Professor
Department of Computer Science and Engineering
Indian Institute of Technology Bhilai

Members:
Prof. Hemangee Kapoor
Professor
Department of Computer Science and Engineering
Indian Institute of Technology Guwahati

Dr. Aryabartta Sahu
Associate Professor
Department of Computer Science and Engineering
Indian Institute of Technology Guwahati

Vitae

Dipojjwal Ray joined the Dual (M.Tech + Ph.D.) programme in the Department of Computer Science and Engineering at Indian Institute of Technology (IIT) Guwahati, India in July 2015. Prior to joining IIT Guwahati, he was with Tata Consultancy Services as a Windows Phone Application Developer and later as a research fellow in Big Data at the Department of Computer Science and Engineering, Indian Institute of Technology (IIT) Bombay, Mumbai, Maharashtra, India. He earlier received a B.Tech. degree in Electronics and Communication Engineering from Institute of Engineering and Management (IEM), Kolkata, West Bengal, India in 2010. His domains of expertise are Formal Verification, Discrete Event Systems, Network Security, Hardware Security. His current research interests include but are not limited to Security Verification, Formal Verification of Learning-enabled Cyber Physical Systems, Game theoretic controller design, Combinatorial Optimization, Artificial Intelligence and Reactive Synthesis. He enjoys playing football, lawn tennis, cricket, badminton, table tennis, and traveling to new places.

Contact Information

E-mail: dipojjwal@iitg.ac.in
ray.dipojjwal@gmail.com

Address: S/o: Dr. Subrata Ray, 40 Pirtala, Baburbag,
Burdwan, West Bengal - 713104, India