Fall 2020

Internet-of-Things (IoT) Security Threats: Attacks on Communication Interface

Mohammad Mezanur Monjur
University of New Hampshire, Durham

Internet-of-Things (IoT) Security Threats: Attacks on Communication Interface

by

Mohammad Mezanur Rahman Monjur

DISSERTATION

Submitted to the University of New Hampshire
in Partial Fulfillment of the Requirements for the Degree of

Master of Science
in
Electrical and Computer Engineering

September, 2020

This thesis will be examined by and approved in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering by:

Thesis Director, Qiaoyan Yu, Ph.D.
Associate Professor
Department of Electrical & Computer Engineering

Md Shaad Mahmud, Ph.D.
Assistant Professor
Department of Electrical & Computer Engineering

Dongpeng Xu, Ph.D.
Assistant Professor
Department of Computer Science

On July, 2020

Original approval signatures are on file with the University of New Hampshire Graduate School.

ACKNOWLEDGEMENTS

First and foremost, I want to say thank you to my advisor Dr. Qiaoyan Yu. I received an enormous amount of help from her in both my course work and the project. Besides the academic aspect, she also gave me a lot of advice and encouragement when I made important decisions about my career. Most importantly, she helped me build up my interest and confidence in my research work.

I would like to thank Dr. Md Shaad Mahmud and Dr. Dongpeng Xu for their willingness to provide help on my thesis and other graduation work as my thesis committee members.

I would also like to thank my fellow graduate students in the UNH Reliable VLSI Systems Lab: Zhiming Zhang, Pruthvy Yellu, and Sandeep Sunkavilli. I got a lot of help and advice from them on my research, and I had a great time working with them.

Finally, I wish to thank my family for their support and care. I could not have come to this far in my life without their constant support, encouragement, and sacrifices of my family. Their support and care helped me overcome setbacks and stay focused on my graduate study.

Contents

List of Tables

List of Figures

Abstract

Internet-of-Things (IoT) Security Threats: Attacks on Communication Interface

by

Mohammad Mezanur Rahman Monjur

University of New Hampshire

Internet of Things (IoT) devices collect and process information from remote places and have significantly increased the productivity of distributed systems or individuals. Due to the limited budget on power consumption, IoT devices typically do not include security features such as advanced data encryption and device authentication. In general, the hardware components deployed in IoT devices are not from high end markets. As a result, the integrity and security assurance of most IoT devices are questionable. For example, adversary can implement a Hardware Trojan (HT) in the fabrication process for the IoT hardware devices to cause information leak or malfunctions. In this work, we investigate the security threats on IoT with a special emphasis on the attacks that aim for compromising the communication interface between IoT devices and their main processing host. First, we analyze the security threats on low-energy smart light bulbs, and then we exploit the limitation of Bluetooth protocols to monitor the unencrypted data packet from the air-gapped network. Second, we examine the security vulnerabilities of single-wire serial communication protocol used in data exchange between a sensor and a microcontroller. Third, we implement a Man-in-the-Middle (MITM) attack on a master-slave communication protocol adopted in Inter-integrated Circuit (I^2C) interface. Our MITM attack is executed by an analog hardware Trojan, which crosses the boundary between digital and analog worlds. Furthermore, an obfuscated Trojan detection method

(ADobf) is proposed to monitor the abnormal behaviors induced by analog Trojans on the I^2C interface.

Chapter 1

Introduction

1.1 Internet of Things

Various technological advancements have contributed to our way of life. With the rapid increase in technology, people are moving towards automated technology to make life easier. Among all those technologies, the Internet of Things (IoT) has potentially impacted us the most. IoT technologies harness a significant amount of data through the sensor network. The IoT industry is expected to manufacture 75.44 billion devices and generate 79.4 zettabytes of data by the end of 2025 [1] [2]. In the twenty-first century, artificial intelligence and big data technology are driving the mass influx of data, and IoT technology supports those sectors at the architecture level. IoT devices are growing enormously and will continue to grow in the coming years due to new sensors, more computing power, and reliable connectivity shown in Fig. 1.1. However, using more IoT devices have led to unavoidable sharing of personal information, thus leading to a potential breach of security and privacy.

IoT is an intelligent network connecting to several other IoT devices and data centers. The IoT device is continuously harnessing and analyzing data with the help of sensors

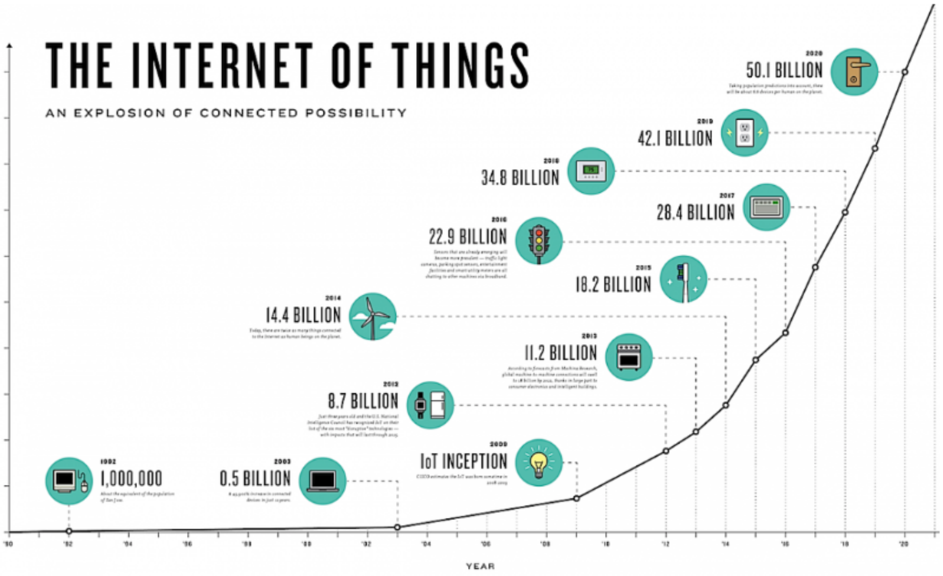


FIGURE 1.1: Number of active Internet of Things (IoT) devices [3]

from the surrounding environment. The majority of IoT devices are autonomous and function with minimal human intervention. Some IoT devices have unique identifications for authentication. Low energy consumption is the main feature of IoT devices. As a result, IoT devices can be placed in a remote area for data collection and transmission. IoT application is expanding into artificial intelligence, cloud, big data, smart systems such as smart homes, and smart offices. An IoT system typically consists of three major stages: collecting data, transferring data, and analyzing data. The first stage is responsible for data collection and transmission, where sensor antennas and microcontrollers are involved. This stage is also known as a physical layer. The second stage is for data transfer, where IoT hub and gateway or network are used. The final stage is for data analysis, includes user interfaces and the back end system, such as the cloud. Any breach in those stages will cause critical information to leak from IoT devices. Recently various vulnerabilities started to emerge in IoT devices due to the lack of advanced encryption and authentication system. Unfortunately, there are no commercial security solutions available on the market to protect against security attacks.

1.2 IoT Applications Vulnerability

The idea of IoT has been roaming around for a long time. IoT technologies' significant growth was possible due to the main feature, such as lower energy consumption than a conventional embedded system. Connectivity through the internet has made it convenient and efficient to transfer data from the user end to the host. IoT has seen massive growth in business and productivity for the last decade. With increasing connectivity, the consumer digital sector has created a large revenue flow and growth. Smart manufacturing, smart supply chain, intelligent power grid, and smart cities have seen promising growth due to automation and control lead by IoT devices at the industry level.

IoT application in a smart home brings benefits such as convenience, efficiency, and safety. A smart home system is equipped with embedded sensors, actuators, and other computing devices. IoT devices usually follow the Cisco IoT reference model and layers, as shown in Fig. 1.2 [4]. The physical layer contains different kinds of sensor applications, such as USB, wireless, and embedded sensors. The sensor's primary function is to collect data such as temperature, pressure, acceleration, optical measurement, and gas from the surrounding environment. In general, the sensor remains unprecedented by the developer. As a result, any unprotected and unsecured layer can act as an entry point for the adversary and launch an attack.

In industry, IoT applications are implemented in various autonomous sector. The robots in the Amazon warehouse are responsible for handling and transporting goods.

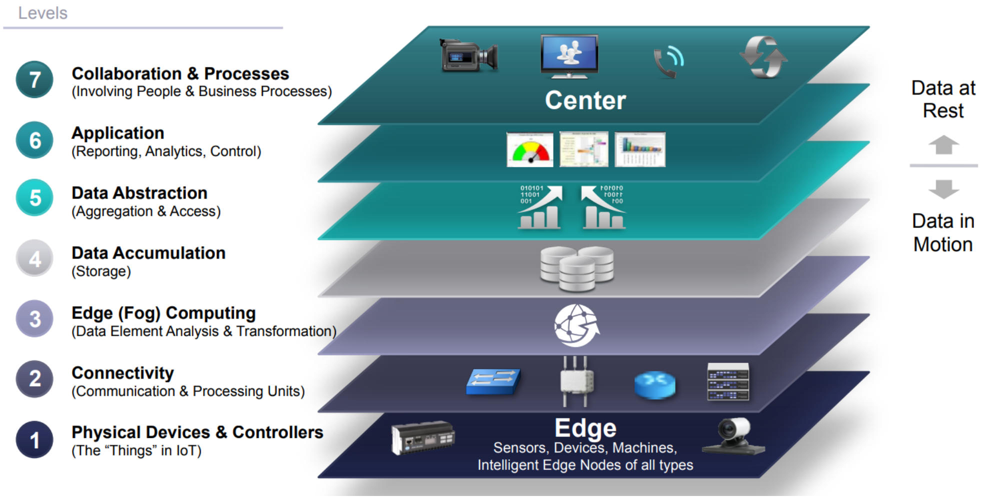

Levels
7 Collaboration & Processes
(Involving People & Business Processes)
6 Application
(Reporting, Analytics, Control)
5 Data Abstraction
(Aggregation & Access)
4 Data Accumulation
(Storage)
3 Edge (Fog) Computing
(Data Element Analysis & Transformation)
2 Connectivity
(Communication & Processing Units)
1 Physical Devices & Controllers
(The "Things" in IoT)
Center
Edge
Sensors, Devices, Machines,
Intelligent Edge Nodes of all types
Data at Rest
Data in Motion

FIGURE 1.2: Internet of Things Cisco reference model [4]

The robots have sensors to move around inside the warehouse without any human intervention. Those robots communicate through internal wireless such as Zigbee, Bluetooth protocols among themselves, and the central cloud system. However, security and privacy problems have emerged with the wide application of IoT devices. Zigbee's low power wireless protocol is based on the IEEE 802.15.4. The stack profile is based on the network layer and application layer. Zigbee security keys are symmetric keys, and this key sharing depends on the security mode of Zigbee [5]. Author Vidgren has shown that the Zigbee key can be easily compromised as during unencrypted key sharing [6]. As a result, sensitive information will be compromised, and the adversary will gain control of the IoT device due to network vulnerability.

According to author Morgner, connecting the Zigbee Light Link base to the host network can lead to any leak of sensitive information [7]. The author demonstrated that unsecured key management is caused by sharing pre-defined keys. According to the author Rodrigo Roman, four Key Management System classes, a key pool framework, a mathematical framework, a negotiation framework, and a public key framework are not applicable due to the unique characteristics demanded [8].

Due to the energy consumption limitation of IoT devices, the key management system can be compromised. According to author Simplicio, lightweight key authentication management schemes suffer insufficient key in large scale implementation [9]. Network protocols such as IPv6 and IPv4 are subject to remote access. Author Czyz demonstrated IoT devices could be remotely accessed over a command-line interface such as Telnet service [10]. IoT networks continuously handle extensive data follow. Data loss or denial of service can lead to high volume traffic in the network and lose control of the operation. Malware can stay dormant in IoT devices, and once triggered, it turns IoT device in botnet and lunch DDoS attack according to the author Angrishi [11].

In 2013, the Austrian and German power grid started to malfunction by self-inflicted DDoS attack and flooded the central command center with traffic [12]. The DDoS attack can temporarily blackout the communication network and protocol of the IoT devices. The smart grid automated system heavily relies on the IoT monitoring network. The sensor network gathers real-time information data to monitor the status of the equipment and control plants. As a consequence, the plant operation will lose the monitoring capability of the transmission and the distribution network. Any malware or spyware attack can have devastating consequences for a nuclear power plant. Stuxnet is an engineered malware, and the target plant can be infected through USB. Once the control system is infected, it will spy on the network's operation for gathering information. After collecting enough data, the Stuxnet started to confiscate the plant's control module and launch the attack to fail plant operation.

1.3 Key Contributions

The majority of research projects focus on exploring new ideas through theoretical analysis, formulating the problems, and validating the simulation idea. It is time-consuming and costly to conduct practical attacks on a system prototype. As attacks on real prototypes are extremely important for IoT idea evaluation, this thesis focuses on the investigation on IoT security threats with special emphasis on communication interface. Our key contributions are as follows:

1. We will investigate potential attacks that originated in unsecured software for IoT smart bulb devices. This attack involves the scanning of IoT devices, unique identification, and control smart bulb light color. We decoded the sound from an external light sensor and identified the corresponding numbers and the pattern of words.

2. We implemented an attack that tampers single-wire protocols. A man-in-the-middle (MITM) attack has been successfully implemented to harm the data flow from a temperature sensor to a microcontroller. The data path connected to the MSP430 microcontroller will trigger the attack. In the dormant stage, the data from the sensor pass to the microcontroller.

3. We propose an analog Trojan that will sabotage the integrity of I^{2}C protocols. This attack demonstrates the potential security threats on master-slave communication interfaces. Furthermore, we propose a detection mechanism for the analog Trojan with an obfuscated Trojan detection method (ADobf).

1.4 Thesis Outline

The thesis is organized as follows.

In Chapter 2, an overview of IoT applications and different attack models are introduced. Then we summarize existing research on the security problems of IoT systems.

In Chapter 3, we identify the security vulnerabilities of unsecured Bluetooth protocols for a smart light bulb. More specifically,

- At the preliminary stage, we start to investigate the wireless protocol for any vulnerabilities. Due to the low budget on power consumption, the Bluetooth protocols do not have any security measures. In the Bluetooth protocol layer, we exploit the Attribute Protocol (ATT), which allows certain data visible to the other devices, to control the device and implement covert data-channel. By examining the General Attribute Profile layer, a characteristic address of the device can be learned.

- We propose a reverse engineering attack on smart bulbs. First, we extracted the MAC address of the smart bulb's unique address after the host establishes the connection with the smart bulb. In this case, we assumed that both adversary and host are in the same network. Using Nrf-sniffer, we can sniff the data between Bluetooth connected devices. By analyzing the sniffed data, we identified the value of the RBG color command for the smart bulb .

- The captured data packets were analyzed in wireshark, and changing of the RGB values were selected as the payload; the pattern in nrf connect the bulb color can be changed. The observed pattern is used to write an automated script in Linux system to change the bulb color.

In Chapter 4, we analyzed the security vulnerabilities of a single wire protocol and demonstrated a MITM attack.

- In this stage, we focus our resources on serial communication protocols of the sensor network. First, we investigate the hand sharking procedure of a single wire protocol for any vulnerability. We implemented a man-in-the-middle attack with an external MSP430 microcontroller that intercepts single wire data. The MSP430 will receive temperature and pressure data from the sensor then perform data manipulation with a bitwise operation. The bitwise XOR will invert the specific bit of the original. The attack will be undetected by the host due to the mimicking capability of the MITM circuit.

In Chapter 5, we have implemented an attack on the boundary of analog and digital worlds. This attack is different from conventional digital-domain attacks. We select I^{2}C interface as a case study to analyze the attack crossing analog and digital domains.

- We use the I^{2}C master-slave interface to demonstrate practical attacks. In I^{2}C, data transmission profoundly can be affected by the clock mute and clock split. Connecting a resistor or capacitor will be sufficient to execute clock attacks by implementing the attack in the master-slave interface between a Xilinx FPGA chip and an off-chip temperature sensor. The attack will cause modification of the temperature data frame.
- We propose an obfuscated Trojan detection method, *ADobf*, to thwart clock attacks in I^{2}C communication. The detection method monitors the clock's abnormal behaviors, and an alert will notify the clock generator and prevent the data line from accepting malicious data frames.

Chapter 6 summarizes the main contributions of the thesis and future work related to this topic proposed.

Chapter 2

Background

2.1 Security Concerns on IoT Applications

Typically, an investigation in cybersecurity is based on a precise attack model. Security threat analysis and attack detection/mitigation methods are limited in one layer of the network. However, various attacks in real applications may be a combination of malicious efforts from multiple network layers. A defense technique designed for a particular layer of IoT and sensor networks may fail to thwart attacks from other layers. Thus, a full-stack network is needed to perform a thorough assessment. Moreover, simulation-based evaluation cannot measure accurate latency, power consumption, and side-channel leakage of the defense methods. As a result, it is imperative to have a more extensive investigation to attack resilient IoT and sensor networks. A simple assembly with off-shelf devices and instrumentation is insufficient to serve for prototyping sophisticated IoT and sensor networks, either. This research will analyze and develop comprehensive and configurable protection to address this urgent need for IoT and sensor applications' defense mechanisms.

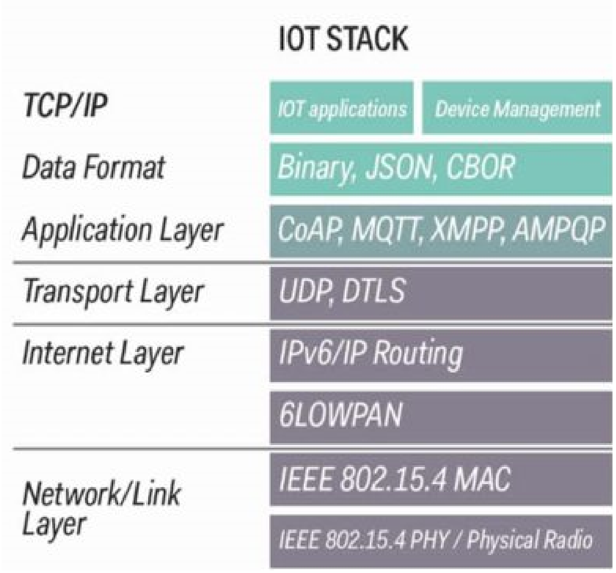


FIGURE 2.1: IoT protocol stack layer [13]

2.2 IoT Protocol

This section mostly focuses on various protocols for the Internet of Things for clients and users. IoT protocol stack has four significant layers shown in Fig. 2.1. More detail of those layers will be discussed in the subsection.

2.2.1 Network Layer

The network layer transmits digital data from the device to the IoT system's back end. The network layer usually consists of wireless devices that collect data and transmit through several gateways to the user end or cloud. Data transmission typically defines the low rate wireless personal networks (LR-WPANs). The physical layer consists of Low Power Wide Area Network (LPWAN), Bluetooth Low Energy (BLE), Zigbee, RFID, and IEEE 802.11 wireless LAN are major communication protocols.

2.2.2 Internet Layer

The Internet layer is responsible for packet forwarding as a gateway through the network nodes. The network layer's primary function is to guide the network packet from the transport layer to the data link layer. Standard protocols are Internet Protocol (IPv6), Internetwork Packet Exchange (IPX), 6LoWPAN and Internet Protocol Security (IPsec), and User Datagram Protocol. Among them, the most popular protocol is IPv6 and includes features such as address configuration and network routing. IPv6 provides an end to end data transmission and switching network nodes.

2.2.3 Transport Layer

The transport layer provides a connection between the host and the client. This layer manages all of the mass transit of data packages from source host to the final host. The main feature of the Transport Layer is to control the data flow and robust enough for error recovery. The most common example is the Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). TCP is responsible for establishing a secure connection between client and server. TCP also can provide multiple endpoints to single hosts and controls the flow to avoid oversubscription. Oversubscription can lead to the congestive collapse of the network. TCP makes sure data is transmitted to the actual address node and the reliability of the network. UDP is not compatible with the time-sensitive application as it may lead the packet to lose and cause packet drops. UPD has no handshaking protocols and can be subject to the unreliability of the network.

2.2.4 Application Layer

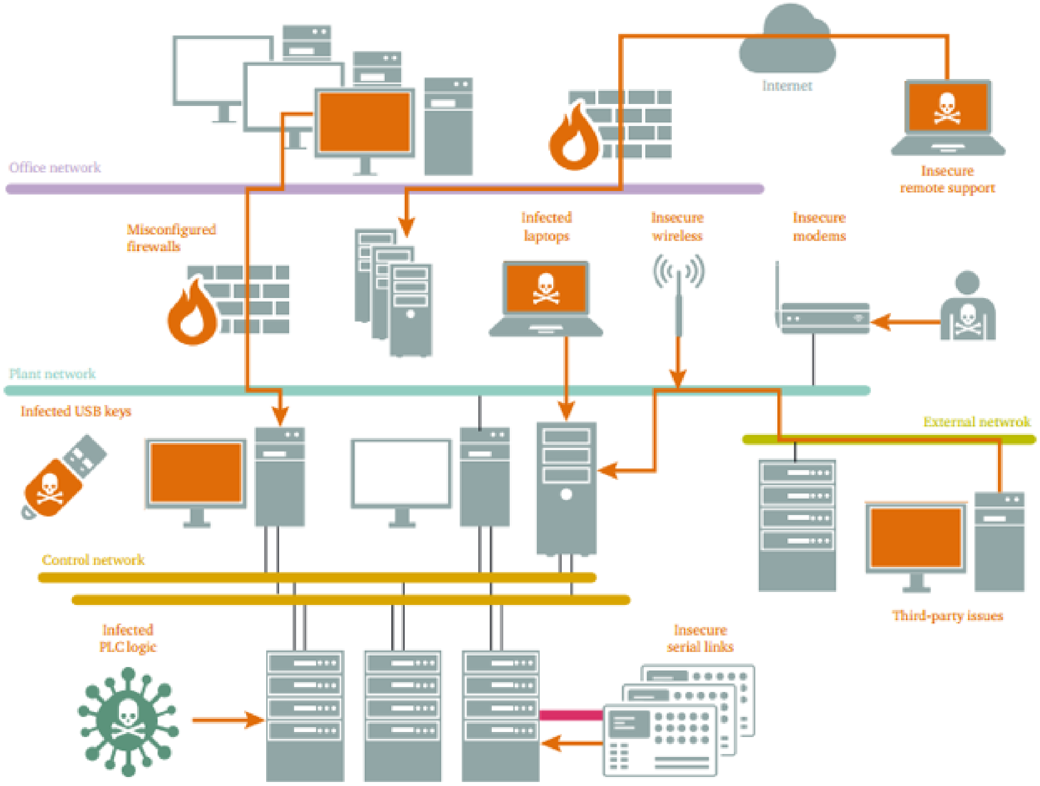
The application layer is the last layer of the IoT stack protocols. The application layer is responsible for user interface data handling. The most common example is the Constrained Application Protocol protocol (CoAP), a lightweight HTTP version. Other messaging protocols are AMQP, and XMPP also frequently used within IoT applications. Internet of Things is creating massive data transmission between devices and networks. All of the devices and sensors and systems and actuators are responsible for generating extensive volume packet data. So, a communications protocol is extremely vital. One of those protocols is message queue telemetry transport is a transmission control protocol based subscription and publish messaging protocol designed for lightweight devices to communications among them. This protocol not only handled interaction between humans interacting with IoT devices and sensors but also machines to machine communication.

2.3 IoT Vulnerability

Security is a serious concern in the era of big data, the Internet of Things (IoT), and artificial intelligence. The growing number of smart devices poses a pressing need to understand new security threats on IoT and sensor networks and develop effective countermeasures against those attacks accordingly. Various types of IoT vulnerabilities are shown in Fig. 2.2.

2.3.1 Software Vulnerability

Software vulnerability can have a catastrophic effect on the IoT. Adversaries can take control of the IoT device through a software vulnerability. Existing research works show



Source: Eric Byres, Byres Security.

FIGURE 2.2: IoT Device Vulnerability in modern nuclear power plant [14].

that a lack of security protection mechanisms directly causes IoT devices' security problems. Chapman [15] and Rodrigues [16] attack IoT devices with insecure configurations or weak authentication. Max [17] evaluated the security of a smart lock and reveals that the authentication and default configuration are insecure and weak. Fernandes et al. [18–20] demonstrated how the implicit trust harms Smart Home IoT devices' security by the third-party applications. Costin investigated is based on firmware updates and reported an array of flaws [21].

The cyber-attack usually carried out by a malicious program to leak information from a targeted attack to disrupt the normal operation flow. The smart grid automated system heavily relies on the IoT monitoring network. The sensor network gathers real-time information data to monitor the status of the equipment and control plants. Some of the power plants still run older version operating systems such as Windows XP or Windows 7. They are subject to worm attacks through internet surfing as they do not have host Firewall [22]. Once the Office network is compromised, the connected server will be compromised and affect other operating systems in the plant network and control network. Any kind of fault in the transmission line and distribution will be invisible to the control center and resulting in a blackout. In 2000 a Russian gas company was under attack by the Trojan virus and lost control of the gas pipeline [12]. The Trojan program can gather operation information, and an adversary can use them to launch an attack.

The adversary can gain access and take control of the IoT due to software vulnerability. According to author L. Markowsky [23], insufficient user authentication remains unsecured in IoT software. IoT device is a low energy device, and software optimization reduces power consumption as IoT places in remote areas. Firmware vulnerability is one of the significant security issues of IoT devices as it can defect regular operation of the device. Author Costin [24] able to retrieve the password as plaintext and 109 private RSA key from the 428 embedded firmware image and 56 SSL certificates out of 428 firmware.

2.3.2 Network Vulnerability

The security protection for those networks has not kept up with the booming speed of connecting low-end edge nodes. Cybersecurity is a serious concern in the era of big data, Internet of things (IoT), machine learning, and artificial intelligence. Any loophole in the security will create an opportunity for the adversary to execute attack vectors and leak sensitive information.

Recent reports show that hackers were able to compromise Amazon IoT devices such as Ring doorbell and Home security camera [25] [26]. The adversary was able to intercept the user names and passwords through an unencrypted HyperText Transfer Protocol (HTTP) of the local user network, thus gaining access to the IoT devices. Industry experts believe it is easy to breach the security of the IoT security camera and voice assistants such as Alexa and Google Home [27].

Many IoT devices rely on UPnP protocols to provide features like easy configuration and control. However, UPnP uses HTTP protocol, which does not offer any confidentiality and integrity protection. Garcia [28] presents various attack methods to break UPnP protocol due to its deficiency of authentication, validation, and logging. Therefore, existing works show that IoT devices in the Smart Home environment are vulnerable if not protected by any defensive security technique.

The security protection for IoT networks has not kept up with the booming speed of connecting low-end edge nodes. Cybersecurity is a serious concern in the era of big data, the Internet of things (IoT), machine learning, and artificial intelligence. Any loophole in the security will create an opportunity for the adversary to execute attack vectors and leak sensitive information. Recent reports show that hackers were able to compromise Amazon IoT devices such as Ring doorbell and Home security camera [25] [26]. The adversary was able to intercept the usernames and passwords through an unencrypted HyperText Transfer Protocol (HTTP) of the local user network, thus gaining access to the IoT devices. Industry experts believe it is easy to breach the security of the IoT security camera and voice assistants such as Alexa and Google Home [27].

According to author Morgner [7], connecting the Zigbee Light Link base to the host network can lead to leaks of sensitive information. The author demonstrated the unsecured key management cause by sharing pre-defined keys. According to the author Rodrigo Roman [8], four Key Management System classes, a key pool framework, a mathematical framework, a negotiation framework, and a public key framework are not applicable due to the unique characteristics demanded.

2.3.3 Hardware Vulnerability

For real-time data acquisition, sensor remotely measures and connect with the IoT device. These sensors can process the real-time data at the node and reduce the load of the central network. This type of sensor can also be implanted in the power generation side of the power plant. The sensor monitors the load demand and controls the turbine of the generator [29]. Most of the sensors are low energy IoT devices, and data are transmitted to the monitoring system encrypted. An adversary can implement a man in the middle attack to leak sensitive information or send a manipulated signal to the control center.

Attacks on the Advanced smart Metering Infrastructure (AMI) create a new entry point for attackers to breach information from the power network. The main consequences of AMI attacks include theft of data, theft of power, localized or widespread denial of power supply, and disruption of the power grid. Approximately 43% of U.S. households currently have smart meters installed [30]. Smart meters and data collectors for power grid communicate typically with RF techniques, the spectrum of which is 900MHz ISM band (industrial, scientific medical band). In the ISM band, many unlicensed devices compete for spectrum usage. As a result, a spoofing attack on the ISM band is easy to succeed. A mesh network for data communication from smart meters, data collectors in substations, and data management in utility analysis centers. If mesh topology is adopted, smart meters pass the measurement from one node to another; still, the data reaches the data collector in the substation. Due to the data propagation, a masquerade attack could intrude on the network and inject false monitoring data, disrupting the normal power distribution. An adversary could tamper the smart meter without being captured in the substation or meter data management system through the Replay attack, which re-send outdated measurement.

2.3.4 Chip Level Vulnerability

A Hardware Trojan (HT) is a malicious insertion to the existing circuit, causing alter of functionality upon activation. Adversaries insert HT payload in an IoT device to make it leak information or deliberately manipulate it or bypass the system's security. Integrate chip, such as application-specific system-on-chip and field-programmable gate array devices are vulnerable to digital and analog HTs. Due to the low power feature in IoT, the cryptography units are more unsecured than high-performance integrated circuits. A software bug can be easily fixed by a firmware update and is cost-effective. However, hardware vulnerability such as HTs cannot be removed by a simple as firmware and are more costly. Due to challenging the nature of restoring, HT may cause permanent damage and service degradation [31].

HT can be easily implemented by the untrusted third-party IP vendors, untrusted design house, and third-party Electronic Design Automation tool to fabricate microchip of IoT devices. The author Spreitzer demonstrated that the adversary observes the IoT device's behavior without disturbing It by using passive side-channel electromagnetic analysis [32]. Side-channel attacks are noninvasive types that can cause the physical implementation of the data extract technique. Side-channel attacks goal is to secret key of embedded IoT devices. A side-channel attack can extract encrypted key information of the crypto block module. Critical inform such passwords, and key can be guessed by monitoring and measuring power consumption and electromagnetic emissions of cryptography operations. The adversary can analyze the side-channel information such as voltage, current, thermal radiations, and timing information to decrypt the device's critical information. Author Pammu proposed a Correlation Electromagnetic Analysis attack to intercept the AES-128 encryption module and successfully retrieve the secret key [33]. Author Genkin illustrated that extraction of full 4096-bit RSA decryption keys attacks could be carried out using a sensitive microphone placed 4m away from IoT device [34].

2.4 Conclusion

IoT security is a serious apprehension in the era of big data and artificial intelligence. The growing number of smart devices poses a pressing need to understand IoT and sensor networks' security threats and develop effective countermeasures against those attacks accordingly. Existing defense mechanisms are typically assessed in an artificial attack environment defined by a specific attack model, thus maybe lacking resilience against other attacks. Aside from the cyber-attack, a physical attack can also have a catastrophic effect on the IoT. The physical attack usually carried out by an HT to leak information from a targeted attack to disrupt the normal operation flow.

Chapter 3

Bluetooth Low Energy Bulb Attack

3.1 Introduction

Internet of Things (IoT) is a system organize of embedded software, sensors, actuators, and other computing devices. Each of an IoT component is assigned with a unique identifier. Through the internet, IoT devices exchange data without human interaction. IoT is used in various sectors like automated homes, health and fitness, automobiles, and logistics. IoT-based smart parking lot monitors the free space for parking in the lot; smart farming system monitors soil moisture and temperature and waters the crop accordingly. Although IoTs have changed much of the world we live in, the use of IoTs brings in security threats. The things surround us, each of them transmitting valuable data. Attackers could breach IoT devices' connection to gain control of the smart devices and thus steal personal information like passwords. Research efforts have been made to investigate attack models and methods specific to IoT devices and networks. In this work, we focus on the cover data exfiltration in IoTs. A practical attack method is proposed, and a possible execution plan for attack implementation is provided, as well.

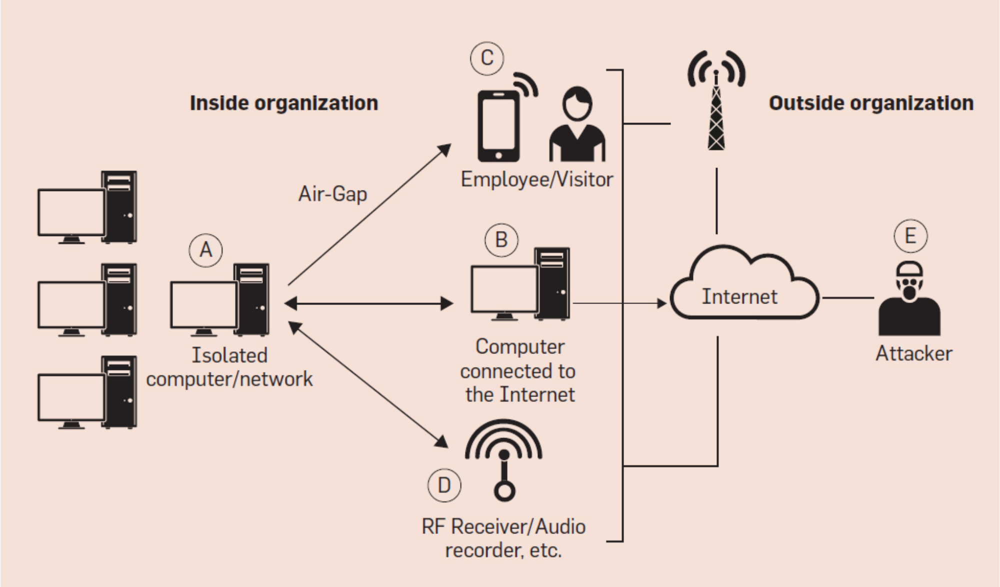

Inside organization
Outside organization
C
Employee/Visitor
Air-Gap
A
Isolated computer/network
B
Computer connected to the Internet
Internet
E
Attacker
D
RF Receiver/Audio recorder, etc.

FIGURE 3.1: Air-Gapped network [35]

3.2 Preliminaries

As required in the competition, the channel for covert data exfiltration is between an air-gapped network and an attacker accessible network. In this section, we introduce the security threats in air-gapped network and potential vulnerabilities of smart light bulbs.

3.2.1 Air-Gapped Networks

Air-gapped networks are those networks that are being separated from the unsecured network to provide cybersecurity. These air-gapped networks are physically isolated from the public networks so that the information in the air-gapped networks are accessible from outside network. Thus information exchanged inside the air-gapped network is considered secure. Fig. 3.7 shows an example of an air-gapped network. In the network, all devices, phones, laptops, desktop computers are connected via the internet; a laptop and printer are connected to the Internet through wi-fi. The computer is being isolated from the outside-world internet and is said to be air-gapped.

The air-gapped network has been widely used in military defense systems, financial systems, industrial control systems, and many other places, where protecting sensitive information from breaching is crucial. This air gapping is employed as a solution to assure security; however, it is not a perfect solution against cyber-attacks. Hackers or outsiders can breach the data through unprotected or weakly protected IoT devices. For instance, the heat from the computer and sound inside the network can be used for exfiltrating the data. The paper [36] shows how an LCD screen is used for data leakage. The paper [37] introduces how data leakage occurs by controlling the air conditioning system in an organization that employed air-gapping. In this proposal, we will use the light to exfiltrate the data.

3.2.2 Smart Bulbs

An Internet-connected LED smart light allows the user to customize the light intensity in a remotely controllable manner. A smart bulb consists of a logic board, power supply, and LED board. The smart bulb can be controlled using a mobile app, through an automation hub, or by programming a bulb that is connected to the internet to control other bulbs. The bulb's various functionalities that can be controlled including brightness, hue, turning on/off switch. Smart bulbs could be applied in IoT with three different architecture.

1. Smartphone Centric Architecture: Fig. 3.2 shows the smartphone-centric architecture. In this architecture, the smart bulb is controlled by a cell phone via Bluetooth. There is no direct connection from the smart bulb to the internet. The mobile phone

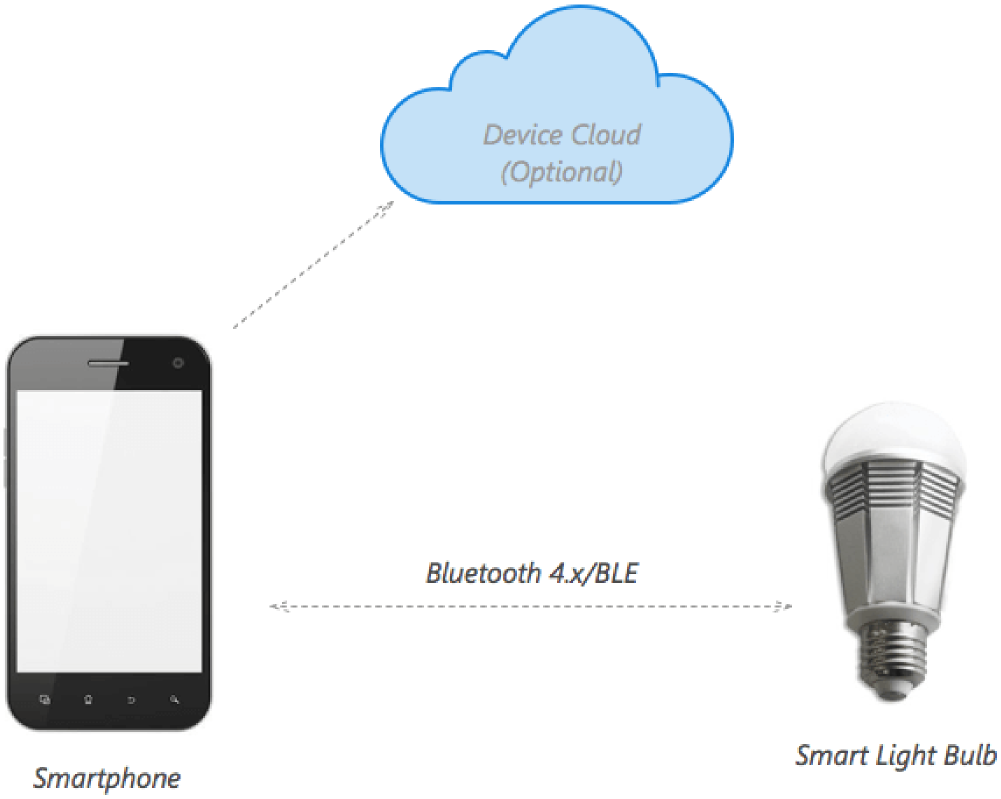


FIGURE 3.2: Smart phone centric architecture [38]

is the only gateway for the light bulb to reach a data cloud. If an adversary needs to compromise the mobile phone first to control the bulb.

2. Hub centric architecture: Fig. 3.3 depicts hub centric architecture. Different than the smartphone-centric architecture, the smart bulb in the hub-centric architecture is connected to the hub first, before it reaches to other IoT devices and data cloud. This architecture hub acts as an intermediary. Often time, the hub device is equipped with a firewall or sophisticated authentication protocols, making the light bulb less vulnerable from security attacks.

3. Cloud-centric architecture: Fig. 3.4 presents a cloud-centric architecture. The smart

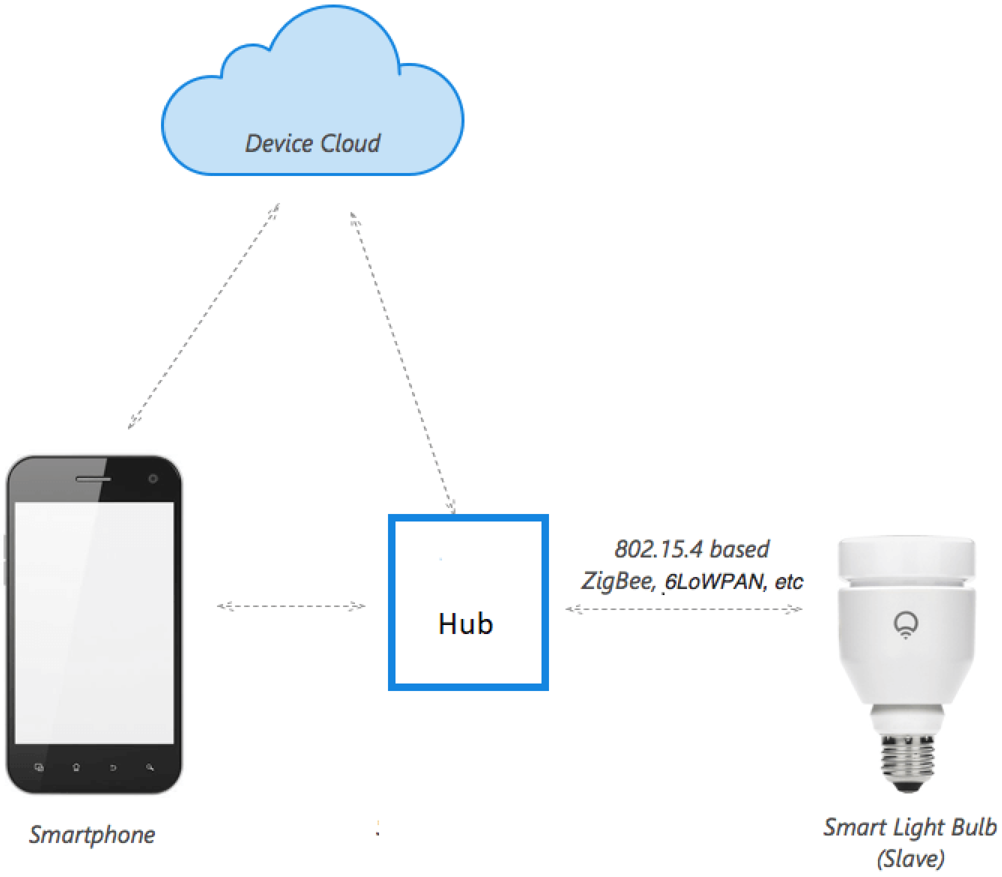


FIGURE 3.3: Hub centric architecture [38]

device is directly connected to a Wi-Fi network and acts as a hub. In this scenario, the smart bulb connected to the internet acts as a master and the other smart bulbs operate as slaves. The master bulb is responsible for all communication with smart slave bulbs and the device.

In this proposal, we will implement the covert data channel in the smartphone-centric architecture since it is equipped with the lowest protection.

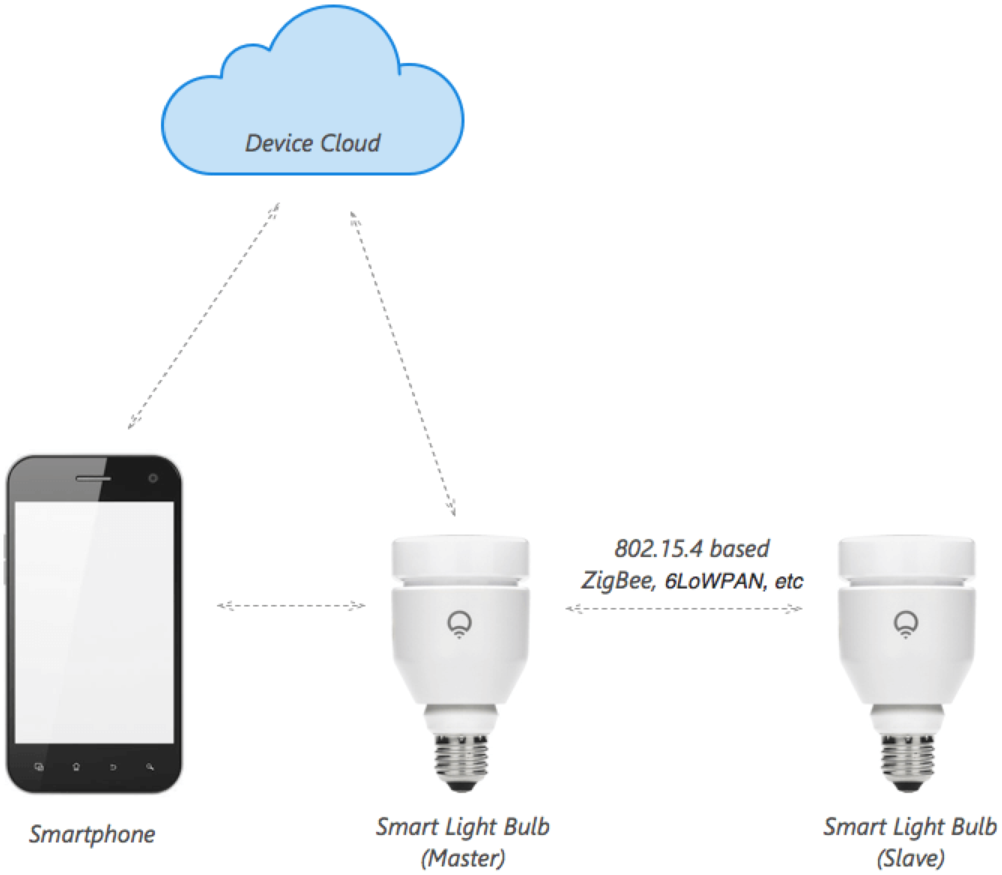


FIGURE 3.4: Cloud centric architecture [38]

3.2.3 Covert Data Exfiltration

Data exfiltration refers to extracting information from computers or electronic devices unauthorizedly. To realize the covert data exfiltration, we need first to implement a covert channel, a hidden channel, from the system. Many types of covert channels exist to breach the air-gapped network, including thermal channel, electromagnetic channel, physical media channel and mechanical channel. Electromagnetic channels are further divided into radio frequency channels, light channel and magnetic channels. Mechanical channels are also classified as acoustic and seismic channels. These classifications of channels

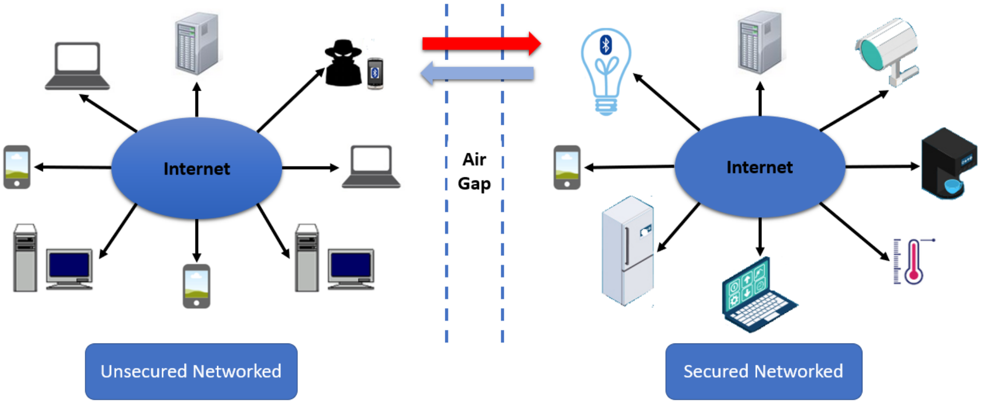


FIGURE 3.5: Attack model for Bluetooth data extraction

are done based on the property we use for communication. For example, if we exploit a characteristic of light, such as frequency of light as a bridge for communication, then the channel is said to be a covert optical channel. The work [36] illustrates how an outsider or hacker can communicate with the IoT device using a covert thermal channel by attacking the central air conditioning system. The paper [37] shows how they used covert optical channels for leakage of sensitive data through the emanation of visible light via a standard LCD screen. The authors in [39] present how they used a covert thermal channel to exfiltrate data. The work [40] how covert acoustic channel is used for exfiltrating data. We will use an optical channel or radio frequency channel for exfiltrating data by exploiting smart lights.

3.3 Threat Model

3.3.1 Attack Condition

Many companies employ an air-gapped network as a security measure to secure the data exchange inside the company's internal network. Unfortunately, these air-gapped networks are not as secure as a thought in the old days, since attackers can find a loop to bridge the air-gapped network and get the data out from the air-gapped network. For instance, the attackers can use IoT as a source to exfiltrate data. Many companies are incorporating smart devices in their offices by changing the infrastructure. In the competition, Lean Enterprise Advanced Knowledge Solutions (LEAKS) has installed smart light bulbs in their offices and connected them to their internal network. We will use this smart bulb to exfiltrate the data from LEAKS. The attack is executed at hardware, firmware, software, network, or application level for covert data exfiltration. In the proposed model, we will modify the firmware for the smart bulb to implement the attack.

3.3.2 Smart Light Bulb Attack Model

In our attack, we will hack the Bluetooth of the smart bulb and connect it to the internal system Bluetooth device for exfiltrating the data. Fig. 3.5 shows the attack model. The air-gapped network shown in the figure consists of smart devices and systems. This network is connected to the internal server. All the data to be shared between these networks are stored in the server. The adversary intervenes by establishing communication with the internal network by exploiting any of the IoT device features in the internal network for exfiltrating the data. We assume that the smart bulb is connected to the internal network

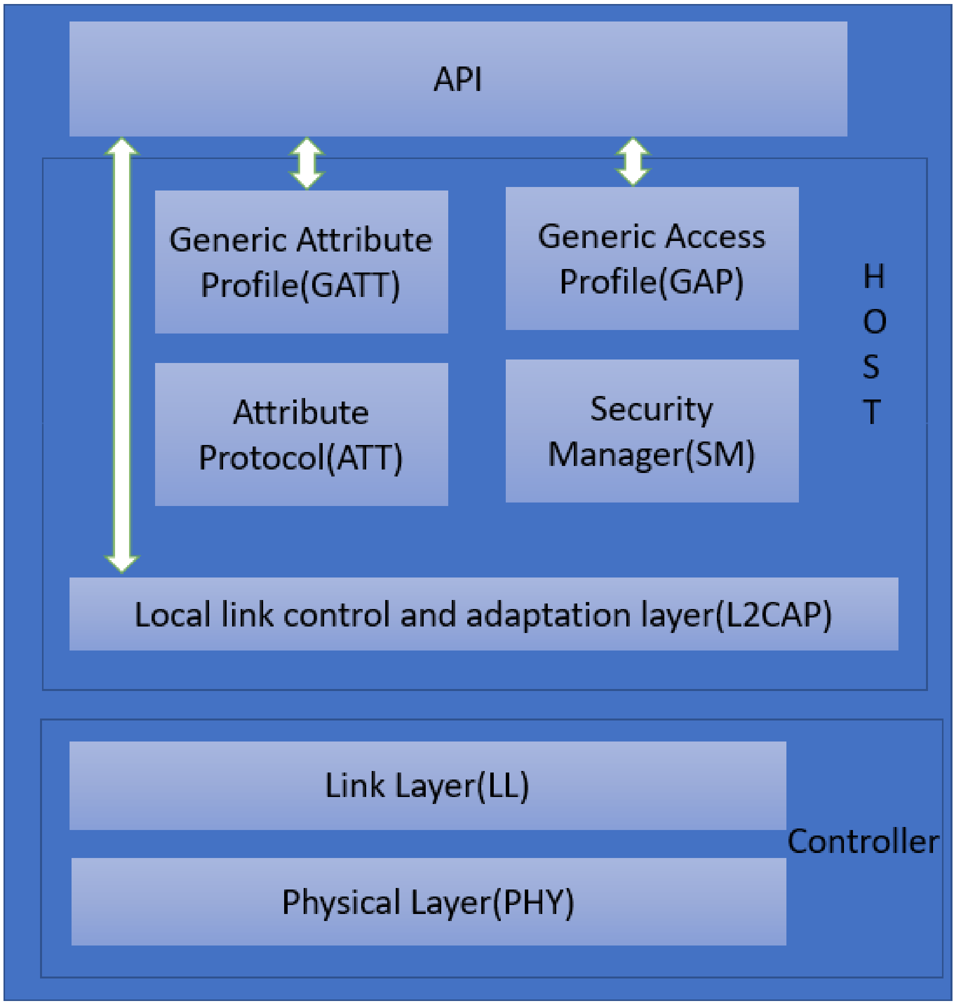

API
Generic Attribute Profile(GATT)
Generic Access Profile(GAP)
H O S T
Attribute Protocol(ATT)
Security Manager(SM)
Local link control and adaptation layer(L2CAP)
Link Layer(LL)
Controller
Physical Layer(PHY)

FIGURE 3.6: Bluetooth low energy device protocol stack.

using Bluetooth. Low band radio frequency signals are used as a bridge in this proposed attack. Reverse-engineering the smart bulb's Bluetooth is one possible way to investigate the vulnerabilities of the protocol.

3.4 Proposed Data Exfiltration

3.4.1 Exploiting Weaknesses of Bluetooth Protocol Stack to Develop Attack Surface

The Bluetooth low energy (BLE) device stack consists of a separate host and controller. The diagram of the BLE device stack is shown in Fig. 3.6. Host-Control Interface (HCI) provides communication between the host and controller using a standard interface. The logical link control and adaptation layer protocol (L2CAP) layer provides data encapsulation services to the upper layers and allows logical end-to-end data communication. The authentication mechanism between the two devices is taken care of by the security manager.

All the applications are present in the General Attribute Profile (GATT) and General Access Profile (GAP) layer of the protocol stack. The attribute protocol (ATT) layer allows specific data of a device visible to other devices. In this attack, we will exploit the ATT layer to control the equipment and implement covert data-channel. By examining the GATT layer, a specific address of the device can be known. The information from the bulb is observed after we exploit the API of the bulb.

1. General Access Profile (GAP): To establish Bluetooth connection, some devices broadcast data, and others observe data. The device which broadcasts the data is a peripheral device; the other device is a central device. Broadcasting of data is called as advertising. The function of the General Access Profile (GAP) is to make the broadcasts visible to central devices though they are not connected.

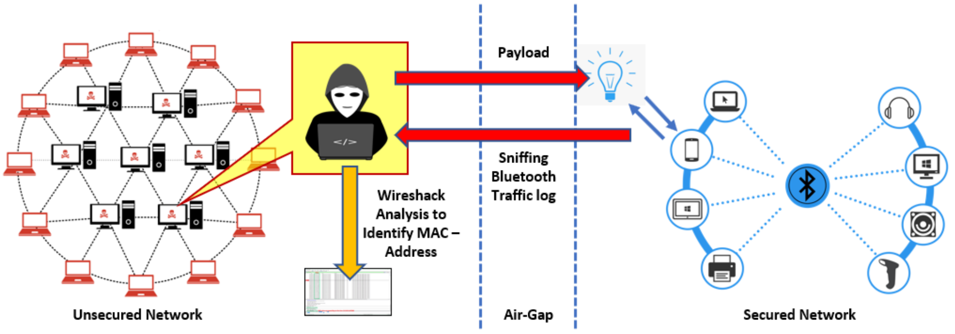


FIGURE 3.7: Covert data extraction and launch an attack in the air-gap network.

2. General Attribute Profile (GATT): The General Attribute profile defines the data transfer between the two connected BLE devices. The data transaction between devices in this layer is classified as profile, services, and characteristics. The profile is a collection of services defined by peripheral users. For example, in the heart rate profile, there have two services, namely, heart rate service and device information service. Services come below profile. Each service has one or more. Characteristics. Each service is given a unique ID (UUID) defined by the developer. For example, the magic blue bulb from this competition offers three services 0xFFE0, 0xFFE5, 0xFFF0. Each characteristic is also given a unique ID. BLE peripheral can be interacted using Characteristics. In our bulb, we will use Characteristic value 0xFFE9 to write a value and thus change the color. In this project, we exfiltrated the GATT transactions using nrf-sniffer and analyzed the data using Wireshark.

3.4.2 Plans for Data Exfiltration

In the attack model, we assume that a smart bulb is connected to the internal network using Bluetooth shown in Fig. 3.7. To communicate with the internal network, we will exploit the Bluetooth connection of the smart bulb. By connecting the smart bulb to our device and using some of the mobile apps' advanced options, we will know about the Bluetooth packet log. Those log files could reveal the MAC address of the smart bulb. A close analysis will allow us to find the entry to take control of the smart bulb. Assuming the internal user controls the light by playing music. By examining the data received in the log, then the song played by the internal user can be guessed. Controlling the light to annoy the user for disconnecting from the network and comparing the MAC address will help know the number of users in a room. Based on the Bluetooth traffic observation, the log the file can be extracted by connecting the device to the smart the bulb with the Bluetooth. After analyzing the Bluetooth packet, the payload in the tool Wireshark allows us to extract the MAC address of the bulb. In the next sections, we demonstrate how to implement two covert data channels to obtain the information regarding (1) the song played inside the organization, and (2) the number of people in the organization protected by the air-gapped network.

3.5 Implementation Covert Data Channel 1

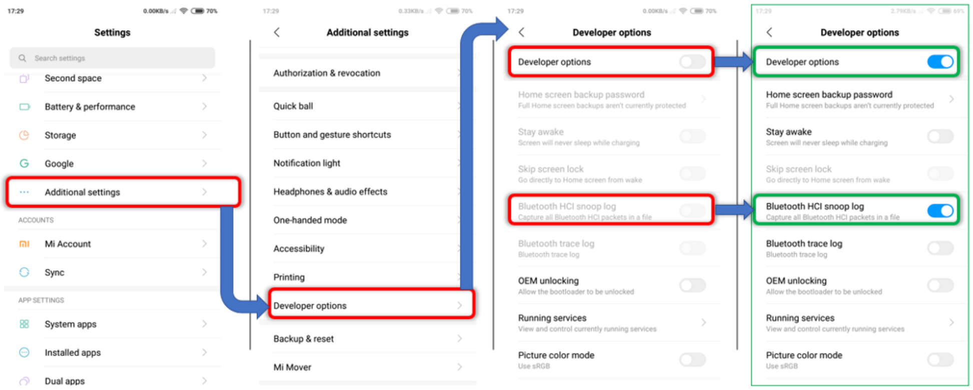
3.5.1 Assumption, Tools, and Setup

In this attack model, we assumed that the user operates the bulb by playing songs from the music library. To communicate with the internal network, we will exploit the Bluetooth connection using a smart bulb. Once we connect to the smart bulb and select the Android phone's advanced options, we will get the Bluetooth packet log. By close analysis of this log file using Wireshark, we will know the handle used to change the bulb's color.

3.5.2 Experiment

The smart bulb changes the color according to the musical beat change in the song. Using Nrf-sniffer, we can sniff the data between two Bluetooth connected devices. Analyzing the sniffed data, by looking at the handle we found, gives us the value of the color change. The time for every change in the value of the color for a song is captured. After comparing the time values captured with reference time values of the songs, we can guess the tune played by the user. In the rest of the subsection, we introduce three necessary execution steps in detail.

- Step 1: Creating a reference library Five different pitch level songs were selected for the test. The magic bulb was operated using all these songs. The Bluetooth log file was generated by turning one advanced option in an Android-based phone, as shown in Fig. 3.8. Analyzing the log file gives the details of color value and associated time. The time for every change in the color of the bulb is captured for



17:29 0.00KB/s 70%

Settings

Search settings
Second space
Battery & performance
Storage
Google
Additional settings
ACCOUNTS
Mi Account
Sync
APP SETTINGS
System apps
Installed apps
Dual apps

17:29 0.33KB/s 70%

Additional settings

Authorization & revocation
Quick ball
Button and gesture shortcuts
Notification light
Headphones & audio effects
One-handed mode
Accessibility
Printing
Developer options
Backup & reset
Mi Mover

17:29 0.00KB/s 70%

Developer options

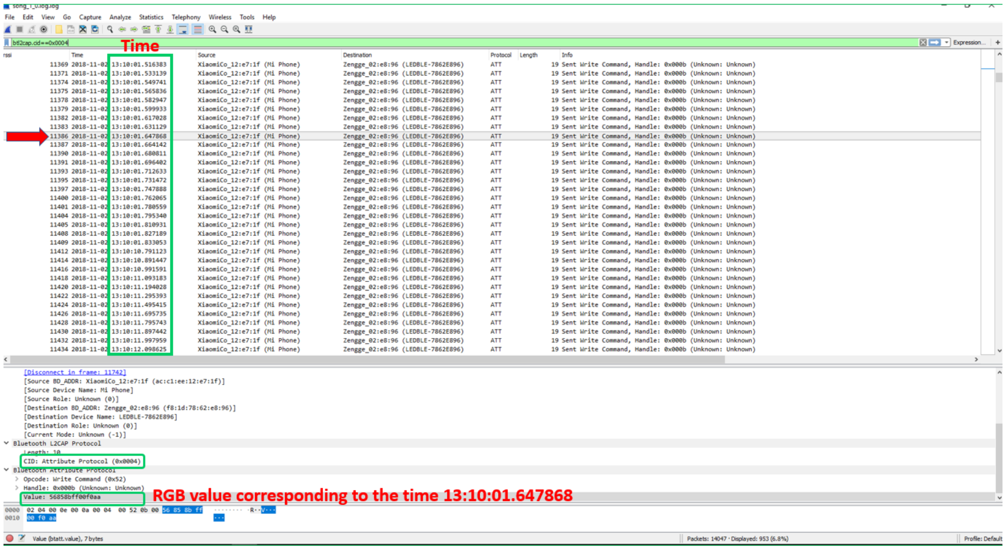
Developer options ○
Home screen backup password
Full Home screen backups aren't currently protected
Stay awake ○
Screen will never sleep while charging
Skip screen lock ○
Go directly to Home screen from wake
Bluetooth HCI snoop log ○
Capture all Bluetooth HCI packets in a file
Bluetooth trace log ○
Bluetooth trace log
OEM unlocking ○
Allow the bootloader to be unlocked
Running services
View and control currently running services
Picture color mode ○
Use sRGB

17:29 2.79KB/s 69%

Developer options

Developer options ●
Home screen backup password
Full Home screen backups aren't currently protected
Stay awake ○
Screen will never sleep while charging
Skip screen lock ○
Go directly to Home screen from wake
Bluetooth HCI snoop log ●
Capture all Bluetooth HCI packets in a file
Bluetooth trace log ○
Bluetooth trace log
OEM unlocking ○
Allow the bootloader to be unlocked
Running services
View and control currently running services
Picture color mode ○
Use sRGB

FIGURE 3.8: Steps to enable log file creation in android phone.



song_1_0.log.log
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Expression...
Time

btl2cap.cid==0x0004

```
rssi  Time                          Source                           Destination                          Protocol Length Info
11369 2018-11-02 13:10:01.516383   XiaomiCo_12:e7:1f (Mi Phone)    Zengge_02:e8:96 (LEDBLE-7862E896)    ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
11371 2018-11-02 13:10:01.533139   XiaomiCo_12:e7:1f (Mi Phone)    Zengge_02:e8:96 (LEDBLE-7862E896)    ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
11374 2018-11-02 13:10:01.549741   XiaomiCo_12:e7:1f (Mi Phone)    Zengge_02:e8:96 (LEDBLE-7862E896)    ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
11375 2018-11-02 13:10:01.565836   XiaomiCo_12:e7:1f (Mi Phone)    Zengge_02:e8:96 (LEDBLE-7862E896)    ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
11378 2018-11-02 13:10:01.582947   XiaomiCo_12:e7:1f (Mi Phone)    Zengge_02:e8:96 (LEDBLE-7862E896)    ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
11379 2018-11-02 13:10:01.599933   XiaomiCo_12:e7:1f (Mi Phone)    Zengge_02:e8:96 (LEDBLE-7862E896)    ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
11382 2018-11-02 13:10:01.617028   XiaomiCo_12:e7:1f (Mi Phone)    Zengge_02:e8:96 (LEDBLE-7862E896)    ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
11383 2018-11-02 13:10:01.631129   XiaomiCo_12:e7:1f (Mi Phone)    Zengge_02:e8:96 (LEDBLE-7862E896)    ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
11386 2018-11-02 13:10:01.647868   XiaomiCo_12:e7:1f (Mi Phone)    Zengge_02:e8:96 (LEDBLE-7862E896)    ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
11387 2018-11-02 13:10:01.664142   XiaomiCo_12:e7:1f (Mi Phone)    Zengge_02:e8:96 (LEDBLE-7862E896)    ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
11390 2018-11-02 13:10:01.680811   XiaomiCo_12:e7:1f (Mi Phone)    Zengge_02:e8:96 (LEDBLE-7862E896)    ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
11391 2018-11-02 13:10:01.696402   XiaomiCo_12:e7:1f (Mi Phone)    Zengge_02:e8:96 (LEDBLE-7862E896)    ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
11393 2018-11-02 13:10:01.712633   XiaomiCo_12:e7:1f (Mi Phone)    Zengge_02:e8:96 (LEDBLE-7862E896)    ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
11395 2018-11-02 13:10:01.731472   XiaomiCo_12:e7:1f (Mi Phone)    Zengge_02:e8:96 (LEDBLE-7862E896)    ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
11397 2018-11-02 13:10:01.747888   XiaomiCo_12:e7:1f (Mi Phone)    Zengge_02:e8:96 (LEDBLE-7862E896)    ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
11400 2018-11-02 13:10:01.762065   XiaomiCo_12:e7:1f (Mi Phone)    Zengge_02:e8:96 (LEDBLE-7862E896)    ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
11401 2018-11-02 13:10:01.780559   XiaomiCo_12:e7:1f (Mi Phone)    Zengge_02:e8:96 (LEDBLE-7862E896)    ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
11404 2018-11-02 13:10:01.795340   XiaomiCo_12:e7:1f (Mi Phone)    Zengge_02:e8:96 (LEDBLE-7862E896)    ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
11405 2018-11-02 13:10:01.810931   XiaomiCo_12:e7:1f (Mi Phone)    Zengge_02:e8:96 (LEDBLE-7862E896)    ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
11408 2018-11-02 13:10:01.827189   XiaomiCo_12:e7:1f (Mi Phone)    Zengge_02:e8:96 (LEDBLE-7862E896)    ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
11409 2018-11-02 13:10:01.833053   XiaomiCo_12:e7:1f (Mi Phone)    Zengge_02:e8:96 (LEDBLE-7862E896)    ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
11412 2018-11-02 13:10:10.791123   XiaomiCo_12:e7:1f (Mi Phone)    Zengge_02:e8:96 (LEDBLE-7862E896)    ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
11414 2018-11-02 13:10:10.891447   XiaomiCo_12:e7:1f (Mi Phone)    Zengge_02:e8:96 (LEDBLE-7862E896)    ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
11416 2018-11-02 13:10:10.991591   XiaomiCo_12:e7:1f (Mi Phone)    Zengge_02:e8:96 (LEDBLE-7862E896)    ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
11418 2018-11-02 13:10:11.093183   XiaomiCo_12:e7:1f (Mi Phone)    Zengge_02:e8:96 (LEDBLE-7862E896)    ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
11420 2018-11-02 13:10:11.194028   XiaomiCo_12:e7:1f (Mi Phone)    Zengge_02:e8:96 (LEDBLE-7862E896)    ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
11422 2018-11-02 13:10:11.295393   XiaomiCo_12:e7:1f (Mi Phone)    Zengge_02:e8:96 (LEDBLE-7862E896)    ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
11424 2018-11-02 13:10:11.495415   XiaomiCo_12:e7:1f (Mi Phone)    Zengge_02:e8:96 (LEDBLE-7862E896)    ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
11426 2018-11-02 13:10:11.695735   XiaomiCo_12:e7:1f (Mi Phone)    Zengge_02:e8:96 (LEDBLE-7862E896)    ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
11428 2018-11-02 13:10:11.795743   XiaomiCo_12:e7:1f (Mi Phone)    Zengge_02:e8:96 (LEDBLE-7862E896)    ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
11430 2018-11-02 13:10:11.897442   XiaomiCo_12:e7:1f (Mi Phone)    Zengge_02:e8:96 (LEDBLE-7862E896)    ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
11432 2018-11-02 13:10:11.997959   XiaomiCo_12:e7:1f (Mi Phone)    Zengge_02:e8:96 (LEDBLE-7862E896)    ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
11434 2018-11-02 13:10:12.098625   XiaomiCo_12:e7:1f (Mi Phone)    Zengge_02:e8:96 (LEDBLE-7862E896)    ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
```

```
    [Disconnect in frame: 11742]
    [Source BD_ADDR: XiaomiCo_12:e7:1f (ac:c1:ee:12:e7:1f)]
    [Source Device Name: Mi Phone]
    [Source Role: Unknown (0)]
    [Destination BD_ADDR: Zengge_02:e8:96 (f8:1d:78:62:e8:96)]
    [Destination Device Name: LEDBLE-7862E896]
    [Destination Role: Unknown (0)]
    [Current Mode: Unknown (-1)]
v Bluetooth L2CAP Protocol
    Length: 10
    CID: Attribute Protocol (0x0004)
v Bluetooth Attribute Protocol
  > Opcode: Write Command (0x52)
  > Handle: 0x000b (Unknown: Unknown)
    Value: 56858bff00f0aa
0000  02 04 00 0e 00 0a 00 04  00 52 0b 00 56 85 8b ff   ·········R··V···
0010  00 f0 aa                                           ···
```

RGB value corresponding to the time 13:10:01.647868

Value (btatt.value), 7 bytes | Packets: 14047 · Displayed: 953 (6.8%) | Profile: Default

FIGURE 3.9: Song bluetooth packets.

all the five songs. This timing information for all the five songs acts as the reference library.

- Step 2: Sniffing actual network Nrf-sniffer scans for the BLE connections. Once the required device is paired, the Bluetooth traffic can be captured in Wireshark using Nrf-sniffer. The captured data is exported to a file. The corresponding timing value

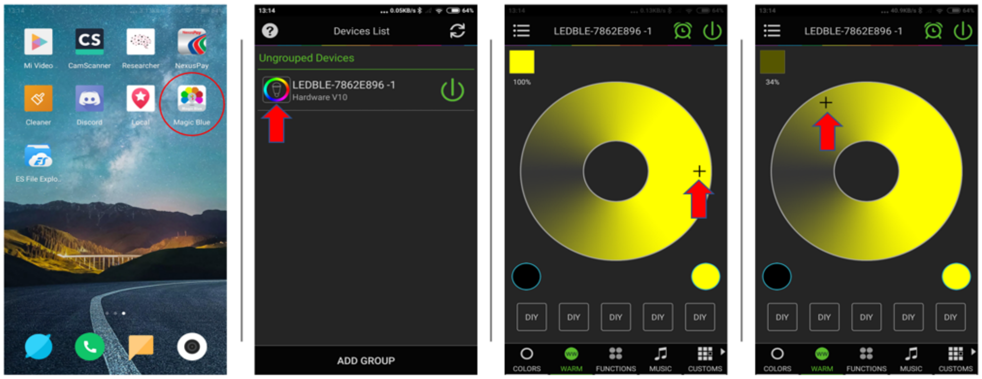


FIGURE 3.10: Steps in android phone to connect with smart bulb and change the color of the bulb to record sample in the log file.

for the color change is noted.

- Step 3: Comparing with reference library timing file captured from step 2 is compared with all the reference libraries' timing patterns. The matching reference pattern is identified, and the corresponding song can be guessed. Fig. 3.9 shows the sniffed log in Wireshark. The time for value is highlighted in the picture. Similarly, we get all the timing information for different colors in one song. All the color values and corresponding times for a song are exported to a file. From the list, the time for every change in color value is noted to obtain the pattern. The change in values is searched from the list, and corresponding timing is noted to get the pattern. The same procedure is followed for the rest of the tunes.

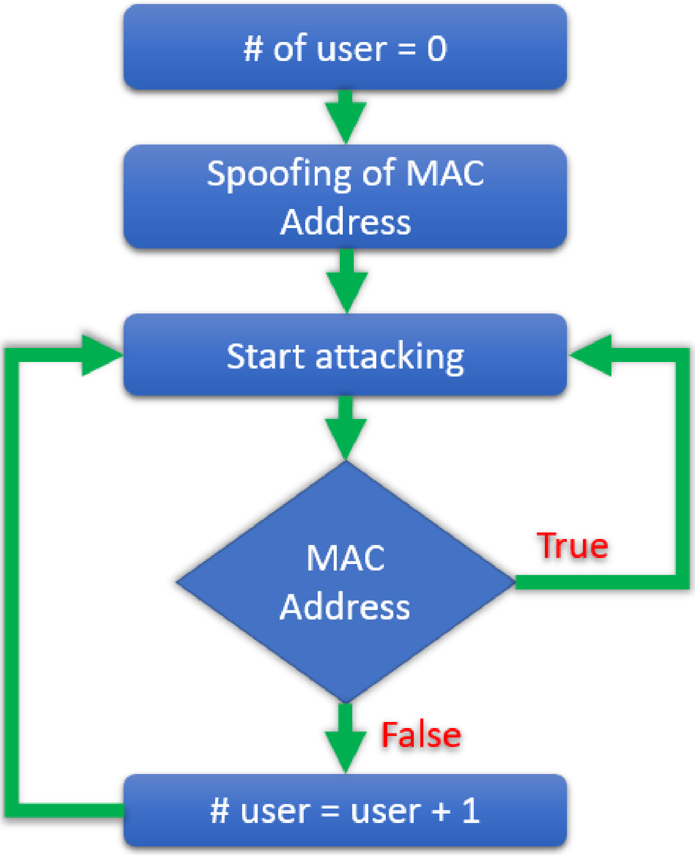
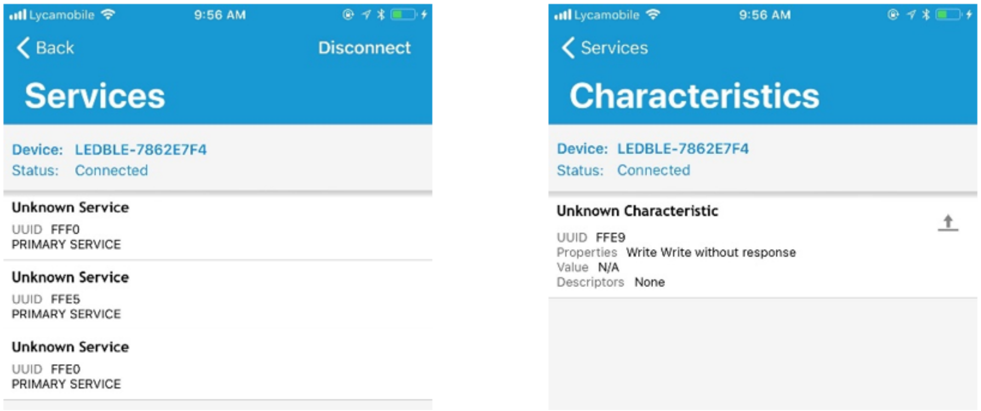


FIGURE 3.11: Flow chart of attack scenario 2.

3.6 Implementation Covert Data Channel 2

3.6.1 Assumption, Tools and Setup

In this attack model, we assumed that everyone in the network has the Magic Blue app in their handheld devices. Further assuming there is only one smart bulb in the room. To communicate with an internal network, we will exploit the Bluetooth connection using the smart bulb. Once we connect to the smart bulb and select the advanced options on



Services

Device: LEDBLE-7862E7F4
Status: Connected

Unknown Service
UUID FFF0
PRIMARY SERVICE

Unknown Service
UUID FFE5
PRIMARY SERVICE

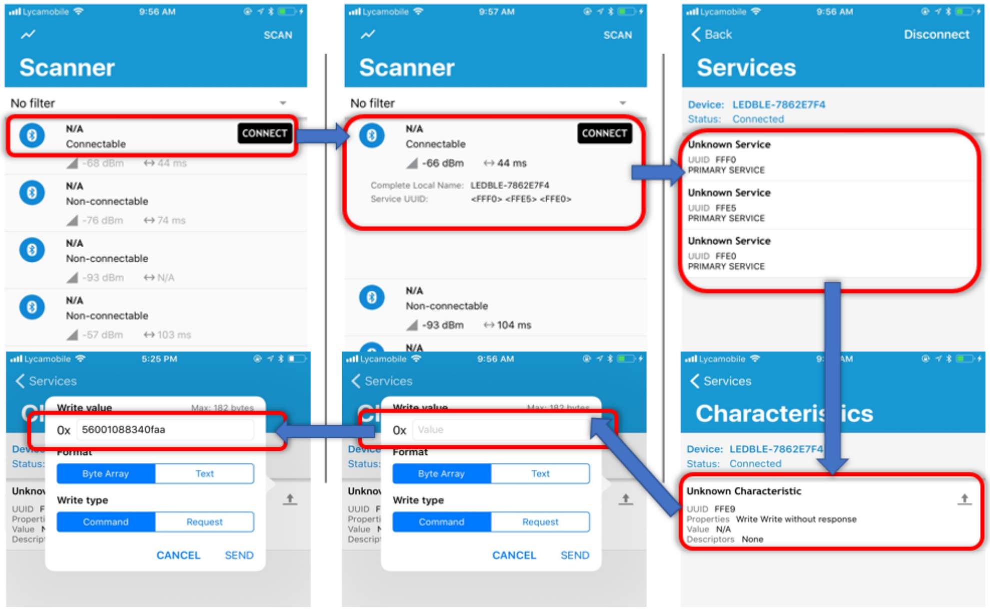
Unknown Service
UUID FFE0
PRIMARY SERVICE

Characteristics

Device: LEDBLE-7862E7F4
Status: Connected

Unknown Characteristic
UUID FFE9
Properties Write Write without response
Value N/A
Descriptors None

FIGURE 3.12: LED BLE services and characteristics shown by nrf connect (Left) and writable characteristic of the bulb (Right).



Scanner

No filter

N/A
Connectable
-68 dBm ↔ 44 ms

N/A
Non-connectable
-76 dBm ↔ 74 ms

N/A
Non-connectable
-93 dBm ↔ N/A

N/A
Non-connectable
-57 dBm ↔ 103 ms

N/A
Connectable
-66 dBm ↔ 44 ms
Complete Local Name: LEDBLE-7862E7F4
Service UUID: <FFF0> <FFE5> <FFE0>

N/A
Non-connectable
-93 dBm ↔ 104 ms

Services

Device: LEDBLE-7862E7F4
Status: Connected

Unknown Service
UUID FFF0
PRIMARY SERVICE

Unknown Service
UUID FFE5
PRIMARY SERVICE

Unknown Service
UUID FFE0
PRIMARY SERVICE

Characteristics

Device: LEDBLE-7862E7F4
Status: Connected

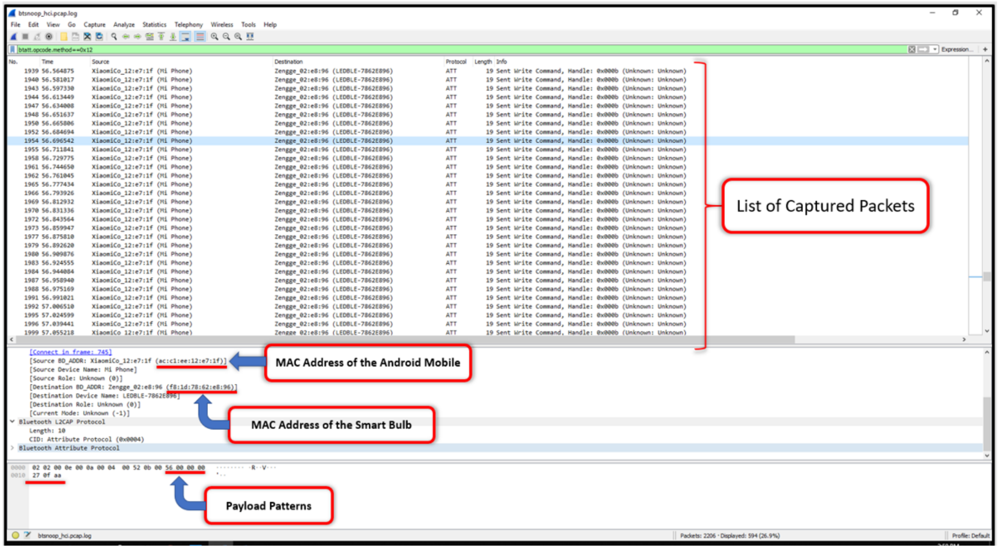
Unknown Characteristic
UUID FFE9
Properties Write Write without response
Value N/A
Descriptors None

Write value Max: 182 bytes
0x 56001088340faa
Format: Byte Array
Write type: Command

Write value Max: 182 bytes
0x Value
Format: Byte Array
Write type: Command

FIGURE 3.13: Steps to write to the characteristic number to change the color of bulb using nrf connect.

the Android phone, we will get the Bluetooth packet log. By close analysis of this log file using Wireshark, we will know the handle used to change the color of the bulb. MAC address of the connected devices can also be identified from the Bluetooth log.


btsnoop_hci.pcap.log
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
btatt.opcode.method==0x12
Expression…
List of Captured Packets
MAC Address of the Android Mobile
MAC Address of the Smart Bulb
Payload Patterns
btsnoop_hci.pcap.log
Packets: 2206 · Displayed: 394 (26.9%)
Profile: Default

```
No.  Time      Source                           Destination                             Protocol Length Info
1939 56.564875 XiaomiCo_12:e7:1f (Mi Phone)     Zengge_02:e8:96 (LEDBLE-7862E896)       ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
1940 56.581017 XiaomiCo_12:e7:1f (Mi Phone)     Zengge_02:e8:96 (LEDBLE-7862E896)       ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
1943 56.597330 XiaomiCo_12:e7:1f (Mi Phone)     Zengge_02:e8:96 (LEDBLE-7862E896)       ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
1944 56.613449 XiaomiCo_12:e7:1f (Mi Phone)     Zengge_02:e8:96 (LEDBLE-7862E896)       ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
1947 56.634008 XiaomiCo_12:e7:1f (Mi Phone)     Zengge_02:e8:96 (LEDBLE-7862E896)       ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
1948 56.651637 XiaomiCo_12:e7:1f (Mi Phone)     Zengge_02:e8:96 (LEDBLE-7862E896)       ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
1950 56.665806 XiaomiCo_12:e7:1f (Mi Phone)     Zengge_02:e8:96 (LEDBLE-7862E896)       ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
1952 56.684694 XiaomiCo_12:e7:1f (Mi Phone)     Zengge_02:e8:96 (LEDBLE-7862E896)       ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
1954 56.696542 XiaomiCo_12:e7:1f (Mi Phone)     Zengge_02:e8:96 (LEDBLE-7862E896)       ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
1955 56.711841 XiaomiCo_12:e7:1f (Mi Phone)     Zengge_02:e8:96 (LEDBLE-7862E896)       ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
1958 56.729775 XiaomiCo_12:e7:1f (Mi Phone)     Zengge_02:e8:96 (LEDBLE-7862E896)       ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
1961 56.744650 XiaomiCo_12:e7:1f (Mi Phone)     Zengge_02:e8:96 (LEDBLE-7862E896)       ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
1962 56.761045 XiaomiCo_12:e7:1f (Mi Phone)     Zengge_02:e8:96 (LEDBLE-7862E896)       ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
1965 56.777434 XiaomiCo_12:e7:1f (Mi Phone)     Zengge_02:e8:96 (LEDBLE-7862E896)       ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
1966 56.793926 XiaomiCo_12:e7:1f (Mi Phone)     Zengge_02:e8:96 (LEDBLE-7862E896)       ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
1969 56.812932 XiaomiCo_12:e7:1f (Mi Phone)     Zengge_02:e8:96 (LEDBLE-7862E896)       ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
1970 56.831336 XiaomiCo_12:e7:1f (Mi Phone)     Zengge_02:e8:96 (LEDBLE-7862E896)       ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
1972 56.843564 XiaomiCo_12:e7:1f (Mi Phone)     Zengge_02:e8:96 (LEDBLE-7862E896)       ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
1973 56.859947 XiaomiCo_12:e7:1f (Mi Phone)     Zengge_02:e8:96 (LEDBLE-7862E896)       ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
1977 56.875810 XiaomiCo_12:e7:1f (Mi Phone)     Zengge_02:e8:96 (LEDBLE-7862E896)       ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
1979 56.892620 XiaomiCo_12:e7:1f (Mi Phone)     Zengge_02:e8:96 (LEDBLE-7862E896)       ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
1980 56.909876 XiaomiCo_12:e7:1f (Mi Phone)     Zengge_02:e8:96 (LEDBLE-7862E896)       ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
1983 56.924555 XiaomiCo_12:e7:1f (Mi Phone)     Zengge_02:e8:96 (LEDBLE-7862E896)       ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
1984 56.944084 XiaomiCo_12:e7:1f (Mi Phone)     Zengge_02:e8:96 (LEDBLE-7862E896)       ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
1987 56.958940 XiaomiCo_12:e7:1f (Mi Phone)     Zengge_02:e8:96 (LEDBLE-7862E896)       ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
1988 56.975169 XiaomiCo_12:e7:1f (Mi Phone)     Zengge_02:e8:96 (LEDBLE-7862E896)       ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
1991 56.991021 XiaomiCo_12:e7:1f (Mi Phone)     Zengge_02:e8:96 (LEDBLE-7862E896)       ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
1992 57.006510 XiaomiCo_12:e7:1f (Mi Phone)     Zengge_02:e8:96 (LEDBLE-7862E896)       ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
1995 57.024599 XiaomiCo_12:e7:1f (Mi Phone)     Zengge_02:e8:96 (LEDBLE-7862E896)       ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
1996 57.039441 XiaomiCo_12:e7:1f (Mi Phone)     Zengge_02:e8:96 (LEDBLE-7862E896)       ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)
1999 57.055218 XiaomiCo_12:e7:1f (Mi Phone)     Zengge_02:e8:96 (LEDBLE-7862E896)       ATT      19 Sent Write Command, Handle: 0x000b (Unknown: Unknown)

    [Connect in frame: 745]
    [Source BD_ADDR: XiaomiCo_12:e7:1f (ac:c1:ee:12:e7:1f)]
    [Source Device Name: Mi Phone]
    [Source Role: Unknown (0)]
    [Destination BD_ADDR: Zengge_02:e8:96 (f8:1d:78:62:e8:96)]
    [Destination Device Name: LEDBLE-7862E896]
    [Destination Role: Unknown (0)]
    [Current Mode: Unknown (-1)]
v Bluetooth L2CAP Protocol
    Length: 10
    CID: Attribute Protocol (0x0004)
> Bluetooth Attribute Protocol

0000  02 02 00 0e 00 0a 00 04  00 52 0b 00 56 00 00 00   ········ ·R··V···
0010  27 0f aa                                           '··
```

FIGURE 3.14: Wireshark software tools for payload patterns analysis.

3.6.2 Experiment

The smart bulb is connected to one of the users in the internal network. The screenshot of the bulb app in a mobile device is shown in the Fig. 3.10. Using Nrf-sniffer, we can sniff the data between two Bluetooth connected devices. The MAC address of the user device is noted. Examining the connections with bulb and comparing the MAC address of the users, the number of people present in a room can be guessed. The Fig. 3.11 shows the flowchart of the attack scenario. The rest of the subsection gives detailed information about three necessary execution steps as in the flowchart.

- Step 1: Reverse-Engineering the smart bulb Initially, Nrf connect was used to scan the Bluetooth-enabled devices. The Nrf connect displays all the services and characteristics offered by the Bluetooth enabled devices. The smart bulb is detected, as shown in the Fig. 3.12. From the Fig. 3.12, we can see that bulb has three services FFF0, FFE5, FFE0. By further analysis, we found that the service FFE5 provides a writable characteristic FFE9, as shown in Fig. 3.13. After identifying the pattern of the payload for the bulb, we have to overwrite the specific number. The android phone was used to get the Bluetooth log file. We were enabling some advanced options, as in Fig. 3.8 saves the Bluetooth traffic. The Bluetooth traffic log file is analyzed using Wireshark. Analyzing the packets in Wireshark, the payload 56 00 00 00 27 0f aa was identified as shown in Fig. 3.14. Further analysis and testing revealed the value 00 00 27 correspond to RGB and 56 00 RR GG BB 0f aa is the payload pattern used. Changing the RGB values of the payload pattern in nrf connect the bulb color can be changed. The observed pattern is used to write an automated script to change the bulb color.

- Step 2: Collect the MAC address of the users Using Nrf sniffer to sniff the Bluetooth traffic between the bulb and connected user. The sniffed log is analyzed to get the MAC address. The MAC address obtained is updated in a file.

- Step 3: Counting the number of users in a room. Initially, the number of users is assumed zero. Nrf connect is used to see if the smart bulb is connectable. When a connection is made by one device, the smart bulb becomes unconnectable. If the smart bulb is not connectable, we will sniff the data using Nrf Sniffer. The MAC address of the connected device is analyzed and compared with the MAC address obtained in Step 2. If the MAC address is matching with the list obtained in step 2 procedure, the user had already connected to the device. We annoy the user by controlling the bulb as in Step 1 and assume the user gets disconnected from the smart bulb. If the MAC address obtained doesn't match with the MAC address obtained in Step 2, the user number count is incremented by 1. This process is

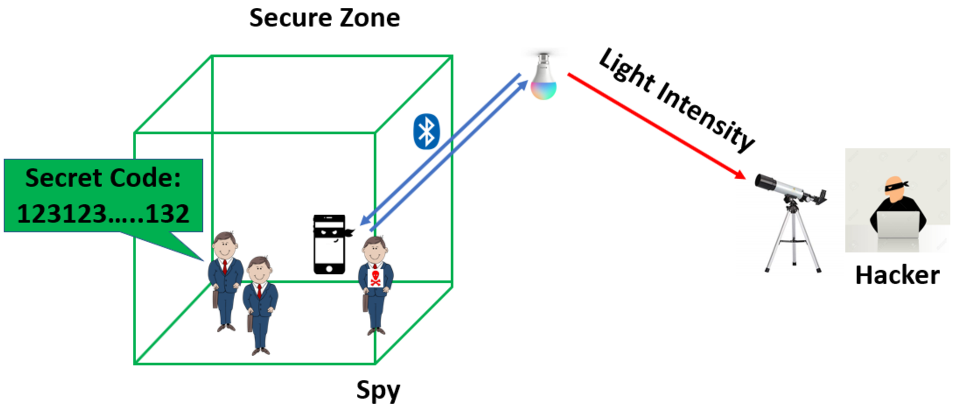


FIGURE 3.15: Attack model with light sensor.

repeated continuously to get the number of users present in the network.

3.7 Implementation Covert Data Channel 3

3.7.1 Assumption, Tools and Setup

IoT devices are communicating with the network through many wireless technologies (Bluetooth and Zigbee, etc.). Due to wireless nature, most of the information is unsecured and are vulnerable to attacks. In Fig. 3.15, an adversary could breach the connection between IoT devices and leak critical information.

Once the adversary is inside a secure zone and activates the Bluetooth and smart bulb app to stream real-time information to a smart bulb outside of the secure zone. A mobile signal jammer is necessary to execute a critical condition for the attack model. The cellular frequency range is usually 700MHz to 2000MHz. For WLAN frequency range can be 2.4GHz to 2.5GHz. In this attack model, the smart bulb is outside of the secure zone.

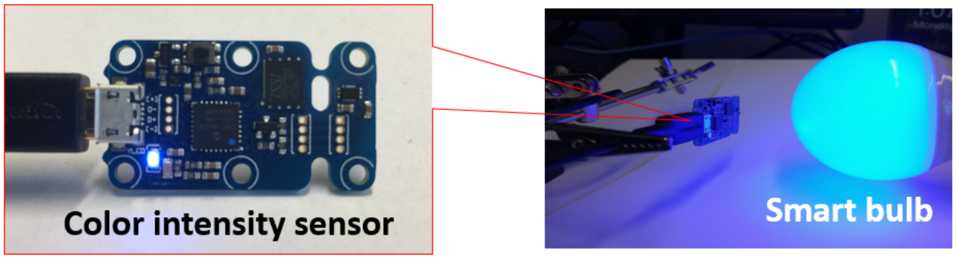


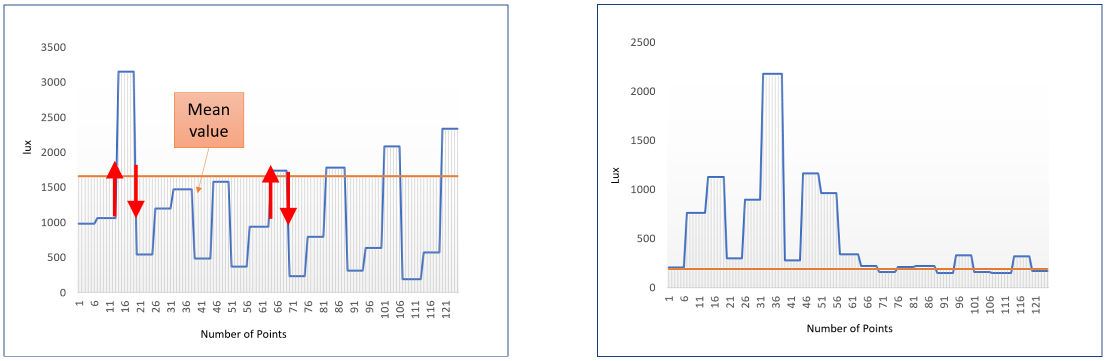
FIGURE 3.16: Measurement of light intensity of smart bulb

Once the adversary will execute the attack, light intensity changes from a distance. The hacker can encrypt the light intensity information by analyzing the waveform.

3.7.2 Experiment

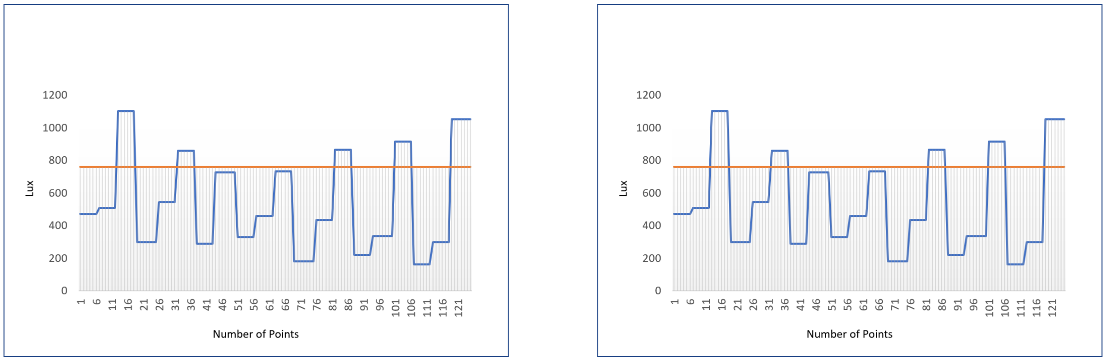
In this attack model, we assume that one of the adversaries activated the Bluetooth and mobile applications to capture the light intensity data for transmission. The light intensity depends on the voice or speech nearby human. The transmission executes in real-time through the mobile application and controls the smart bulb light intensity. Another adversary will be outside the secret room to observe the light intensity. To capture the smart bulb's electrical fluctuation, the adversary will use a light sensor to measure the light luminosity. Once the captured data is collected, it can be analyzed to decode light intensity match the patterns. Our experimental setup was shown in the Fig. 3.16.

In Fig. 3.17, two different samples (song) were selected with different pitch or tones though the smart bulb app. Both of sample was run for an extended period. For simplicity, only the first 5 seconds was analyzed. The captured light intensity value was further investigated for the first five-second. In Fig. 3.16 a) and Fig. 3.16 b) shows the first trial of both samples. Further analysis shows that sample 1 has ten transitions crosses to the



(A) (B)

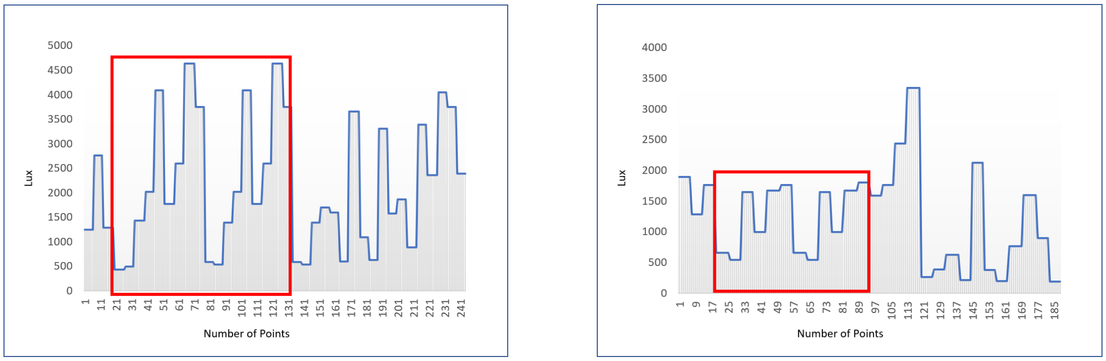
FIGURE 3.17: Trial 1: Number of changes for (a) sample 1 and (b) sample 2 songs to mean value in first 5 seconds.



(A) (B)

FIGURE 3.18: Trail 2: Number of changes for (a) sample 1 and (b) sample 2 songs to mean value in first 5 seconds.

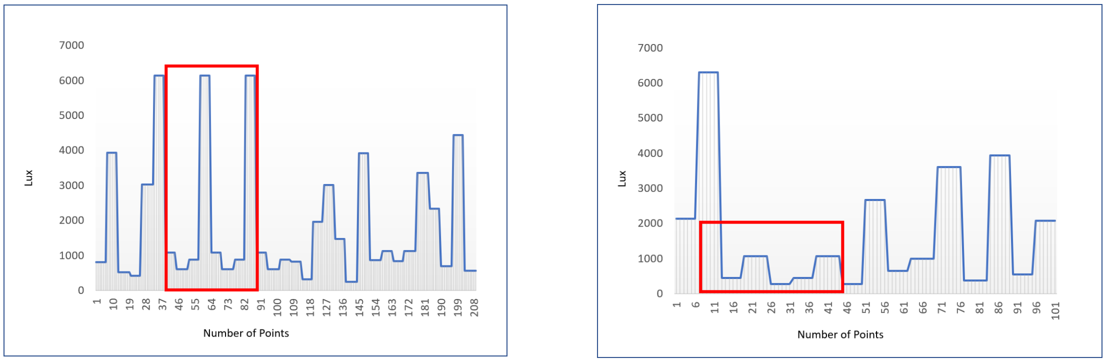
mean value, and sample 2 exhibited ten transition crossing to the mean value. The second trial is shown in Fig. 3.17 a) and Fig. 3.17 b) corresponds to sample 1 and sample 2 for the same time interval, respectively. A close analysis shows sample 1 and sample 2 both exhibited the same number of cross transitioning. For sample 1, the second trial had one less transition compare to the first trial. For sample 2, it was ten times in the first trial,



(A)

(B)

FIGURE 3.19: Stable patterns on cyan color measurement.



(A)

(B)

FIGURE 3.20: Stable patterns on yellow color measurement.

and for the second trial, it was 11 times. Also, both of the samples followed similar patterns in the second trial. It can be concluded; if the adversary has an extensive library or dictionary to compare, then this will easily decrypt the words just by observing the light intensity transmitted by the smart bulb.

Another experiment was set up to identify numerical numbers by observing the smart bulb light intensity. Our goal is to identify a similar light intensity pattern for a fixed numerical number in this experiment. The sound of the different numerical values were transmitted through the smart bulb app. Fig. 3.18 a) and Fig. 3.18 b) are corresponding repetitive patterns of number eight and number five, respectively, in cyan color. A pattern of light intensity was observed for both numerical values in cyan color. Similarly, Fig. 3.19 a) and Fig. 3.19 b) are corresponding repetitive patterns of number eight and number five, respectively, in yellow color. This concludes that the repeat pattern can be observed in constant color. We can conclude that the adversary will be able to identify the numerical value in constant color. This work demonstrates the potential security threats of IoT devices. Transition patterns and color intensity are a feasible side-channel signal to leak information in IoT devices.

3.8 Conclusion

IoT devices have inherent vulnerabilities due to limited computational power and weak authentication protocols. In this project, we use a smart bulb to demonstrate the BLE protocol vulnerability and how conventional app can be used for leaking information. This smart bulb uses the Bluetooth protocol stack to communicate with the user device. The protocol has several weaknesses. In our experiment, we exploited GATT and ATT layers of Bluetooth protocol to control the bulb.

Chapter 4

Physical Attack Implementation on Communication Protocols

4.1 Introduction

The IoT devices are vulnerable due to a lack of security protocols. Lack of security creates an opportunity for the adversary to implement an attack and leak critical information. The wireless network is always connected and continuously communicating between different nodes within the network. Any node in the network can leak vital information and can lead to compromising the network. Our work demonstrates attack any of the nodes can lead security threats to IoT devices.

4.2 Background

4.2.1 Sensor Network

The sensor network is becoming more integrated with IoT devices. Driverless vehicle relays heavily on wireless module for vehicle-to-vehicle (V2V) communication to maintain the safe gap between the vehicles [41]. Any attack falsification of the data or eavesdropping can lead to serious consequences. Attack on temperature and humidity sensor networks in the fabrication process can lead to low yield wafer production. An authentication system for verification can secure communication between the host and the sensor nodes and improve the security protocols. However, the authentication process using pre-shared keys can slow down network communication and complicate network management [42]. The key verification process can cause overhead on the IoT nodes [43]. The authentication not only involves crypto key sharing but also transfers data both way from source to destination. This creates an opportunity for the adversary to implement a physical attack [44]. The nodes are not monitored enough to mitigate any internal or external attack. The standard type of attacks in the wireless network is physical, network, software, and encryption attack. Our attack falls under the physical attack with node tampering by implementing a man-in-the-middle attack. The goal of the attack is to demonstrate the stealthy of the attack and leak or capture information.

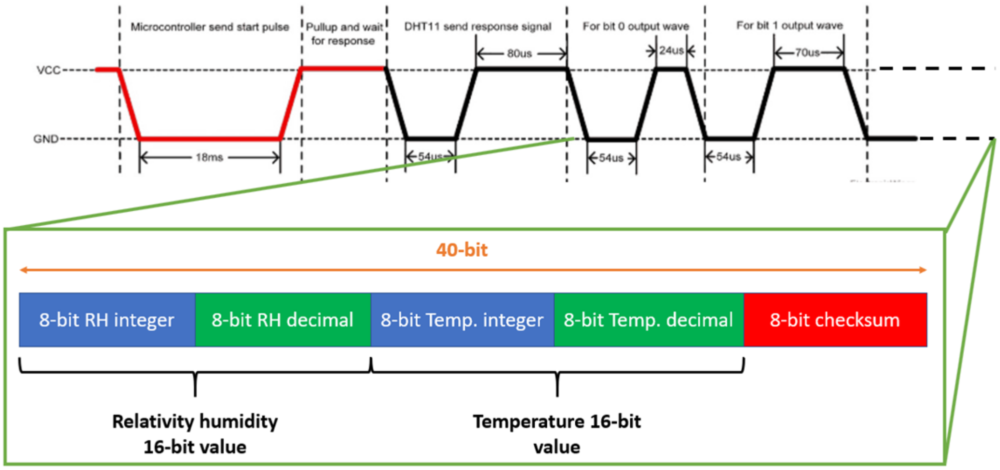


FIGURE 4.1: Hand shaking process between microcontroller and sensor in Single wire protocol.
[45]

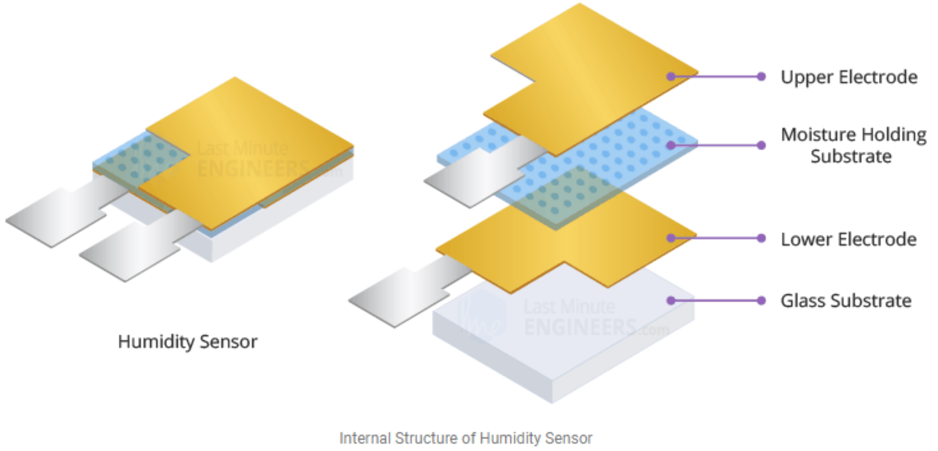

Upper Electrode
Moisture Holding Substrate
Lower Electrode
Glass Substrate
Humidity Sensor
Internal Structure of Humidity Sensor

FIGURE 4.2: Internal structure of humidity sensor.
[46]

4.2.2 Temperature Sensor

The sensor consists of an NTC temperature sensor/thermistor to measure temperature. Negative Temperature Coefficient (NTC) means that the resistance decreases with an increase in the temperature. The sensor also consists of an 8-bit SOIC-14 packaged IC [46].

This IC measures and processes the measured analog signal. Before transmitting the signal to the raspberry pi, It converts the analog signal into a 40-bit digital value. The first 16-bit and next 16-bit represent the relativity humidity and temperature value, respectively.

DHT11 is a part of the DHTXX series of relative humidity and temperature sensor. Inside the sensor, a moisture-holding substrate is a sandwich between the upper and lower electrode. The two electrodes with moisture holding substrate cause the ions to release as water vapor is absorbed by the intermediate substrate, which then increases the conductivity between the two electrodes shown in Fig. 4.2.

The host microcontroller receives the signal shown in Fig. 4.1, and once the handshake is completed, it begins to collect 40-bit data. Start pulse (Request) starts communication with DHT11. First, we should send the start pulse to the DHT11 sensor. After getting a start pulse from, the DHT11 sensor sends the response pulse, which indicates that DHT11 has received a start pulse. The response pulse is low for 54us and then goes high for 80us as shown in Fig. 4.1.

4.3 Methodology

4.3.1 Attack Scenario 1

To implement the attack in the sensor network, we are proposing to use an ultralow-power microcontroller (MSP340) as a man middle attack between sensor and node. The sensor is an ultra-low-cost digital temperature and humidity sensor (DHT11) and connecting to the node single-board computer (Raspberry Pi 3 B+). For wireless communication low power RF module (Digi XBee) was used to data transmission and receiving.

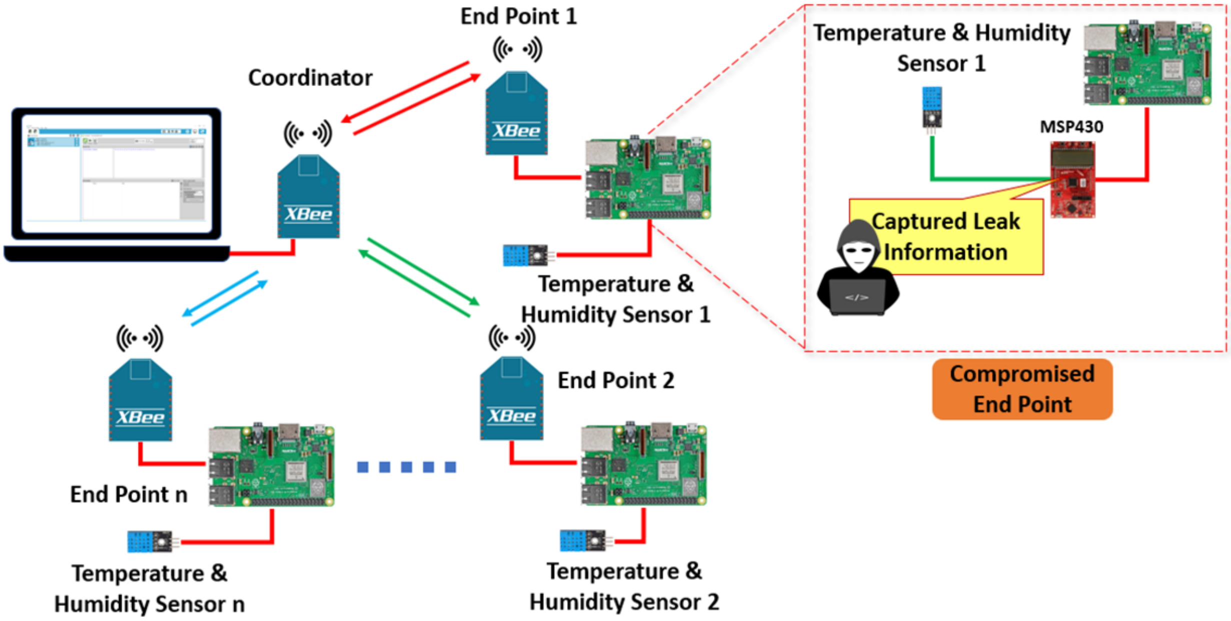

End Point 1
Coordinator
XBee
XBee
Temperature & Humidity Sensor 1
End Point 2
XBee
XBee
End Point n
Temperature & Humidity Sensor n
Temperature & Humidity Sensor 2
Temperature & Humidity Sensor 1
MSP430
Captured Leak Information
Compromised End Point

FIGURE 4.3: Man-in-the-middle attack in wireless network.

Fig. 4.3 shows the full implantation of the sensor network. The coordinator controls all the nodes at the endpoint. Let us consider the endpoint one is under attack by an adversary. The MSP340 receives the request signal from the raspberry pi and relays the request to the sensor. After the successful handshaking between the sensor and raspberry pi, the sensor starts to transmit data to MSP340. The MSP340 will save the data in memory and then relay the data to the raspberry pi.

4.3.2 Attack Scenario 2

The adversary can introduce an analog trigger attack with a passive electronic such as a capacitor. In this attack, the capacitor value of 0.1uf can be connected to the data communication path of the DHT11 as shown in Fig. 4.4. As a consequence, the capacitor will cause glitch or fault injection. The fault injection will cause the voltage to glitch. This will

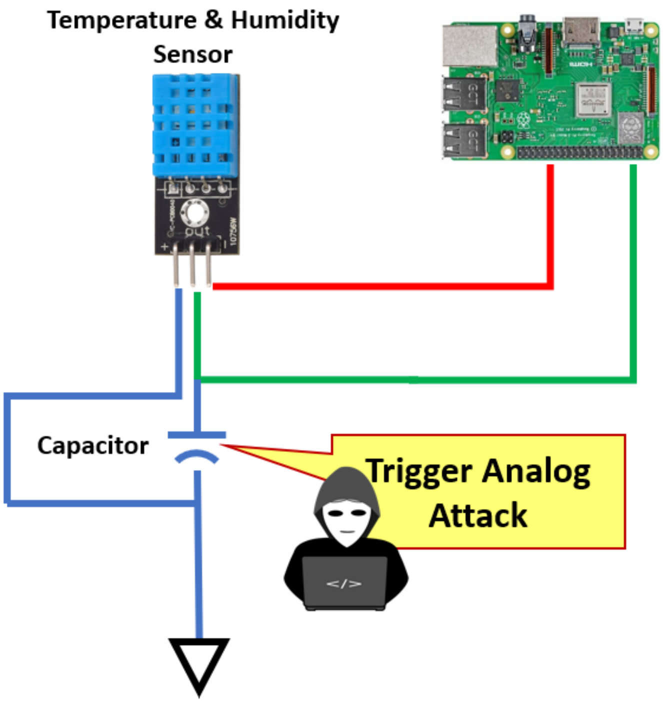


FIGURE 4.4: Analog hardware Trojan Trigger in wireless network.

lead to a device malfunction and data corruption. Most of the sensors are low power and energy-efficient; these sensors lack any advanced crypto protection.

4.3.3 Experimental Setup

For implementing the attack, we used is a microcontroller to a man-in-the-middle (MITM) attack. Utilizing an MSP430 microcontroller and intercept data between the DHT11 sensor and the host requesting data. The MSP430 periodically requires data from the DHT11 once every 640ms; data requests take 520ms resulting in an overall optimal data acquisition rate of 0.86Hz. Once the MSP430 has received the data from the DHT11, it then performs data manipulation on the sensor. The data manipulation can be any bitwise

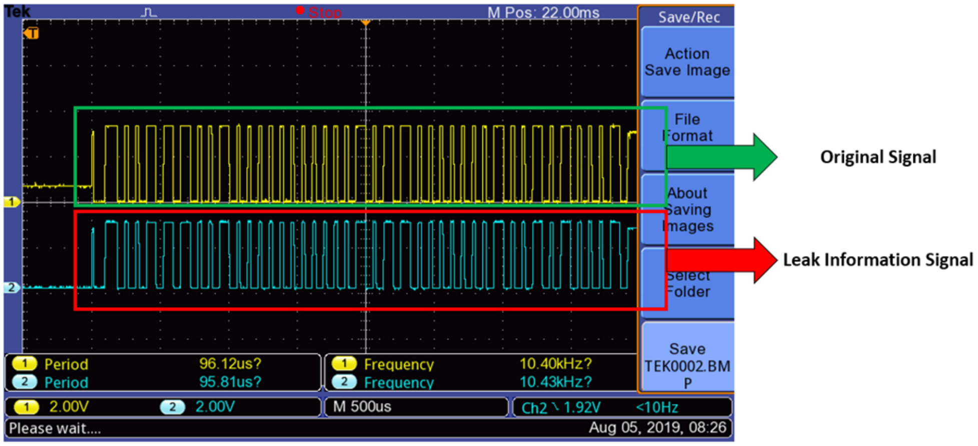


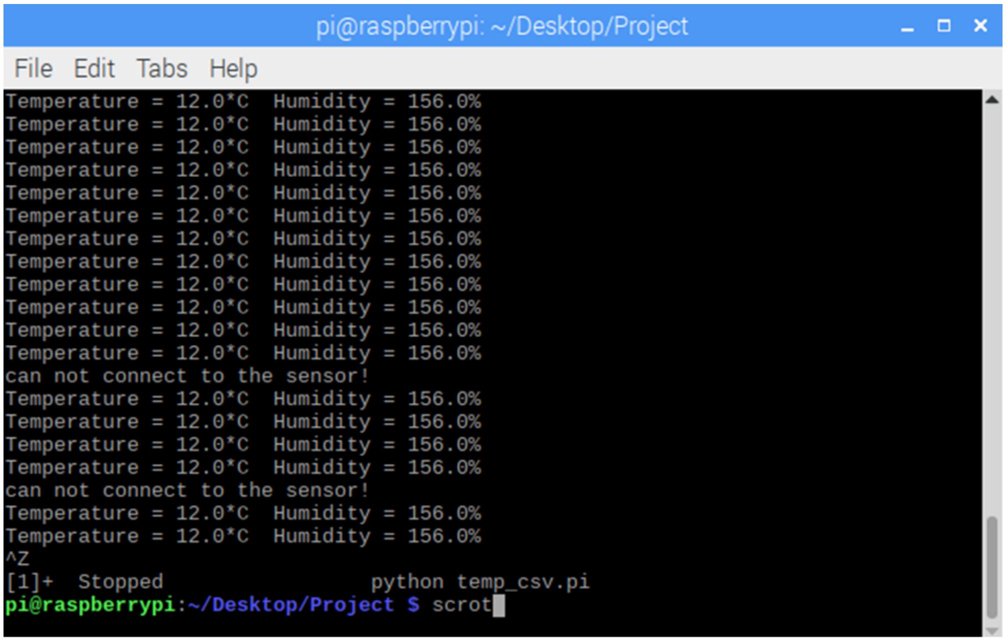
FIGURE 4.5: Comparison between the original and relay signal

operation, bounding, or addition or subtraction. For this paper, a bitwise XOR is used to invert the third to the least significant bit of both temperature and pressure. After modifying the data, a new checksum byte is generated by adding the temperature and pressure bytes. Whenever the master of the sensor requests data, the MSP430 returns the most recent modified data, mimicking the data transmission format of the DHT11.

4.4 Experiment Results

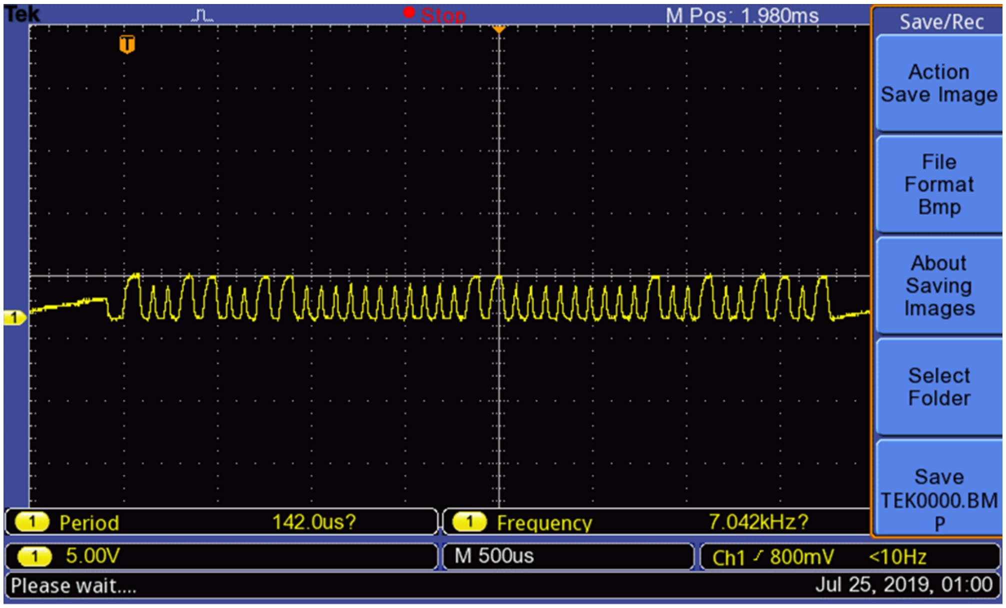
In Fig. 4.5 42-bits, data sent by the sensor is received by the MSP340. The MSP340 function as a buffer stage between the sensor and raspberry pi. Both temperature and humidity values are sent over 32-bits. The adversary can easily alter single bits and control the data displayed at the coordinator side (host).

In the experiment, we can alter the transmitted value from the sensor node. First, The value is converted into a 64-bit floating-point before transmission from the endpoint (sensor) to the coordinator (host). By forcing some particular bits to be zero or set bits can



```
Temperature = 12.0*C  Humidity = 156.0%
Temperature = 12.0*C  Humidity = 156.0%
Temperature = 12.0*C  Humidity = 156.0%
Temperature = 12.0*C  Humidity = 156.0%
Temperature = 12.0*C  Humidity = 156.0%
Temperature = 12.0*C  Humidity = 156.0%
Temperature = 12.0*C  Humidity = 156.0%
Temperature = 12.0*C  Humidity = 156.0%
Temperature = 12.0*C  Humidity = 156.0%
Temperature = 12.0*C  Humidity = 156.0%
Temperature = 12.0*C  Humidity = 156.0%
Temperature = 12.0*C  Humidity = 156.0%
can not connect to the sensor!
Temperature = 12.0*C  Humidity = 156.0%
Temperature = 12.0*C  Humidity = 156.0%
Temperature = 12.0*C  Humidity = 156.0%
Temperature = 12.0*C  Humidity = 156.0%
can not connect to the sensor!
Temperature = 12.0*C  Humidity = 156.0%
Temperature = 12.0*C  Humidity = 156.0%
^Z
[1]+  Stopped                 python temp_csv.pi
pi@raspberrypi:~/Desktop/Project $ scrot
```

FIGURE 4.6: Temperature and Humidity value after capacitor trigger attack



1 Period 142.0us?
1 Frequency 7.042kHz?
1 5.00V
M 500us
Ch1 ⁄ 800mV <10Hz
Please wait....
Jul 25, 2019, 01:00

FIGURE 4.7: Voltage glitch due to capacitor trigger attack

alter the original values. For example, 0x00040F0000000000 is a 64-bits floating number will be used for unsetting the bit position of the original value. In 64-bits floating number system, the 14th, 21st, 22nd, 23rd, and 24th bits will be affected and set to zero. The alter value was effected for relative humidity measurement. Any measured relative humidity values above 55% will be altered with the activation of the attack. Before storing the value or transmitted to other nodes, the detection method generates a 16-bits key of parity bits from a 64-bits value by selecting each byte and shuffling the bits in each byte splitting the byte into nibbles. Then generating an even parity bit from each nibble and assembling the parity bits into a 16-bits Key. The 16-bits key will be created before transmitting from one node, and the receiving node will generate a 16-bits key for comparison. The detection method has a 100% accuracy rate.

An analog hardware Trojan with a 0.1uf capacitor value was connected to the sensor network as a payload circuit. This leads to inaccurate data transmission from the sensor to the microcontroller (host). The capacitor causes temperature and humidity to change to an abnormal value of 12^{0}C and 156%. Manipulated value are shown in Fig. 4.6. The capacitor connects to the data line causes the voltage glitch. The voltage glitch can be seen in Fig. 4.7. This observation helped us to come up with our final implementation of analog hardware trojan. Further details will be discussed in chapter 5.

4.5 Conclusion

MITM is always effective in changing the intercepted data. An adversary can eavesdrop on the critical information. The coordinator node is unable to verify if the data sent by the endpoint. In the next chapter, we will be focusing more on improving the defense and detection mechanism of IoT protocols.

Chapter 5

Analog Trojan

5.1 Introduction

Fabless design and outsourcing fabrication have become a prevalent business model to maximize semiconductor companies' profits. However, the globalized business model raises a serious concern on the trustworthiness of outsourced electronic devices. For instance, malicious modification, a.k.a hardware Trojan, could be placed in the original design to alter logic function or leak information [47] at various phases of the long circuits and systems supply chain. Although there are comprehensive surveys on digital Trojans [48], the investigation of analog Trojans is still in its youth. Indeed, the Trojan war does occur not only in the digital domain; it is sprawling to the analog domain, too. Recent literature [49–52] calls attention to the Trojans in analog/RF circuits or analog-circuit triggered Trojans. Along the line of that call, this work studies hardware Trojans crossing the digital and analog domains.

Different from common digital Trojans, analog Trojans use analog triggering mechanisms or/and analog circuit based payloads [50]. Because of the small size and analog characteristics, it is challenging to detect analog Trojans by scrutinizing hardware footprint or side-channel signals. The work [53] uses composite a logic ring oscillator and a multi-purpose controller to detect A2 analog Trojans [51]. The sensor sensitivity mostly depends on the Trojan switching frequency. The work [54] proposes to reduce the supply voltage and increase the source voltage for the ring oscillator to increase the sensor's sensitivity of the sensor. The run-time Trojan detection R2D2 method presented in [52] identifies a set of concerning signals and then initiates a hardware interrupt when there are abnormally toggling on these guarded signals.

Complementary to the general studies on analog Trojans, this work investigates and then mitigates the security threats on the communication interface between digital master and analog slave devices shown in Fig. 5.1 [55]. More specifically, Inter-integrated Circuit (I^{2}C) interface is adopted as our study subject. Since the clock line plays a critical role in I^{2}C communication, it is imperative to mitigate attacks on the clock line and improve the attack resilience of the data line against analog Trojans. The related work [56] introduces a frequency sensor to detect the tampered clock period. However, that method is effective only when attacks mute the clock signal for a period of time, rather than a single clock cycle. In addition, no evidence is provided in [56] to demonstrate if the frequency sensor is able to handle clock split attacks. Once the additive detection module is known in public, attackers could compensate clock cycles to disguise the attack.

Aiming for the hardware Trojans across Analog and Digital domains, we propose an obfuscated Trojan attack detection method named *ADobf* to provide a high sensitivity on the compromised clock cycles and protect the detection unit itself with an obfuscation key. Our method strengthens I^{2}C communication channels with minor overhead.

Other analog Trojan, IP piracy issues are also attracting researchers' attention. The

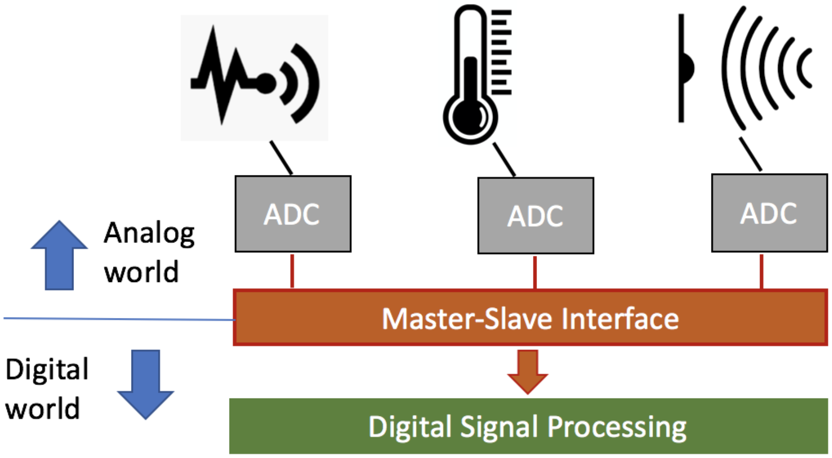


FIGURE 5.1: Attack surface between analog and digital worlds. [55]

work [57] uses key bits to select one transistor from an array of transistors such that the critical parameters in the analog block can be configured and the gain of an amplifier, cut-off frequency of filters, and the operating frequency of a PLL can be obfuscated. The key obfuscation is further applied to a mesh topology to thwart IP piracy attacks on voltage-controlled oscillators [58].

5.2 Attacks Crossing Analog and Digital Domains

Attacks that implemented on the boundary of analog and digital worlds are different than those in digital domain with regard to adversary, attack means and cost, visibility, effective period, and challenges posed on attack detection. In this work, we use I^{2}C interface as an example to analyze the attack crossing analog and digital domains.

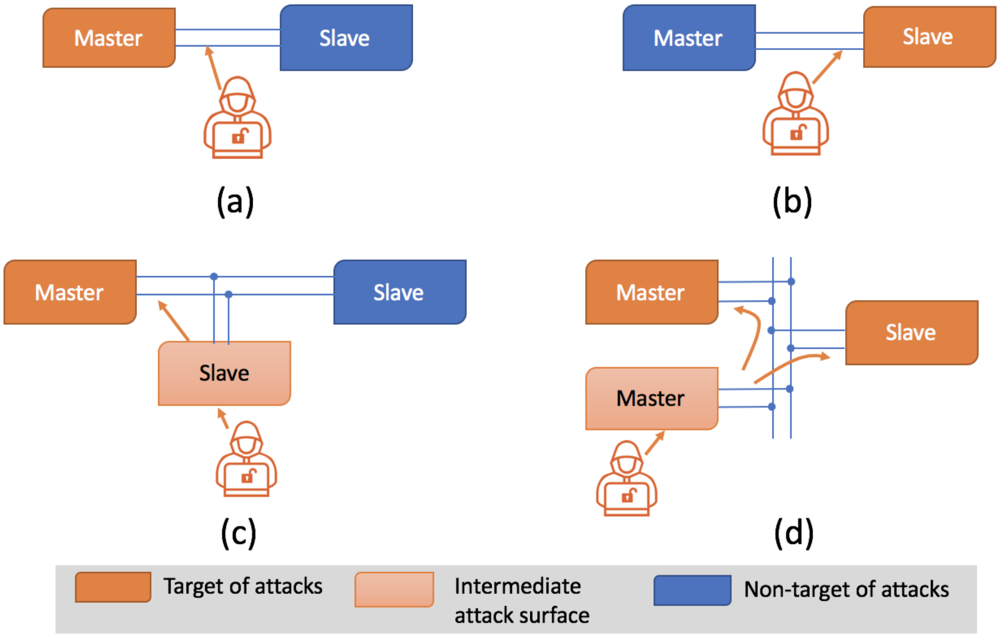


FIGURE 5.2: Attack models on master-slave communication channel. (a) direct attack on master, (b) direct attack on slave, (c) attack on master via a compromised slave, and (d) attack on master and slave via another compromised master. [55]

5.2.1 Attack Models

The goal of crossing domain attacks is either manipulating the signal from the slave (in analog domain) to disturb the master (in digital domain), or impersonating the master to mislead the slave. Fig. 5.2 depicts four scenarios. The top two in Fig. 5.2 are direct attacks, which cause the signal on the master-salve interface to lose its integrity. In contrast, the bottom two cases in Fig. 5.2 show an indirect (sophisticate) attack, where adversary first compromises another slave or master and then impersonates that device to inject malicious signals. The case is shown in Fig. 5.2(d) indicates that an indirect attack will potentially affect more devices than a direct attack.

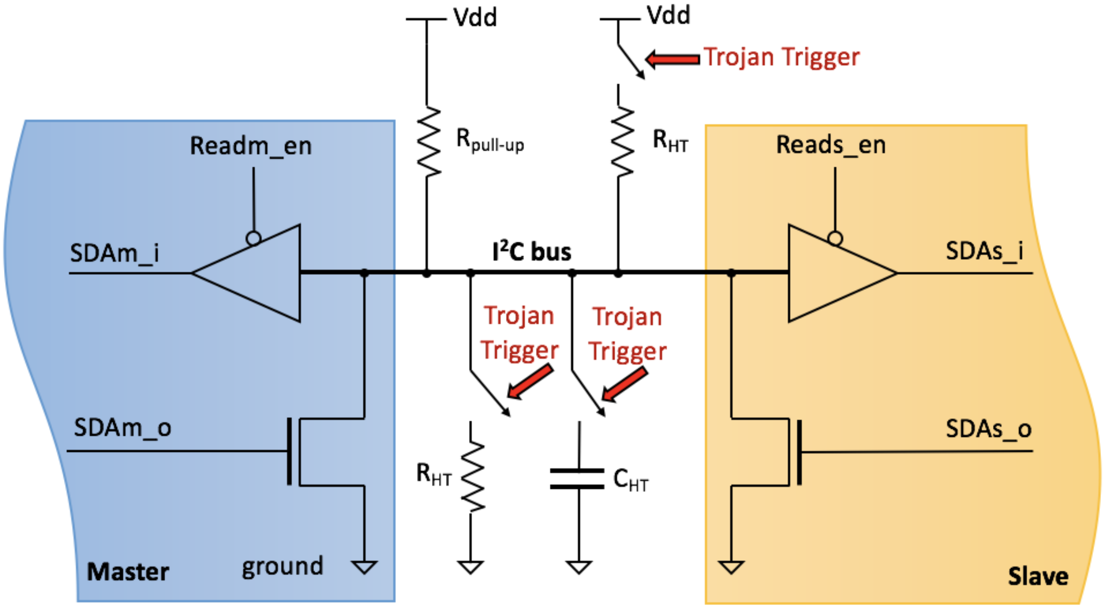
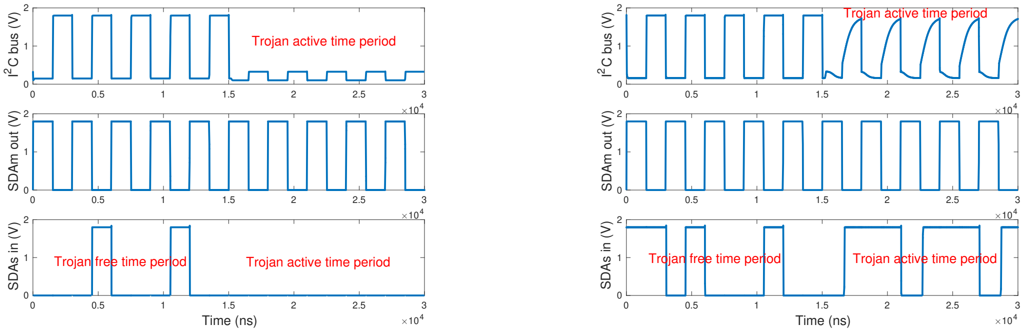


FIGURE 5.3: Trojan attack in I^2C communication channel. [55]



(A)

(B)

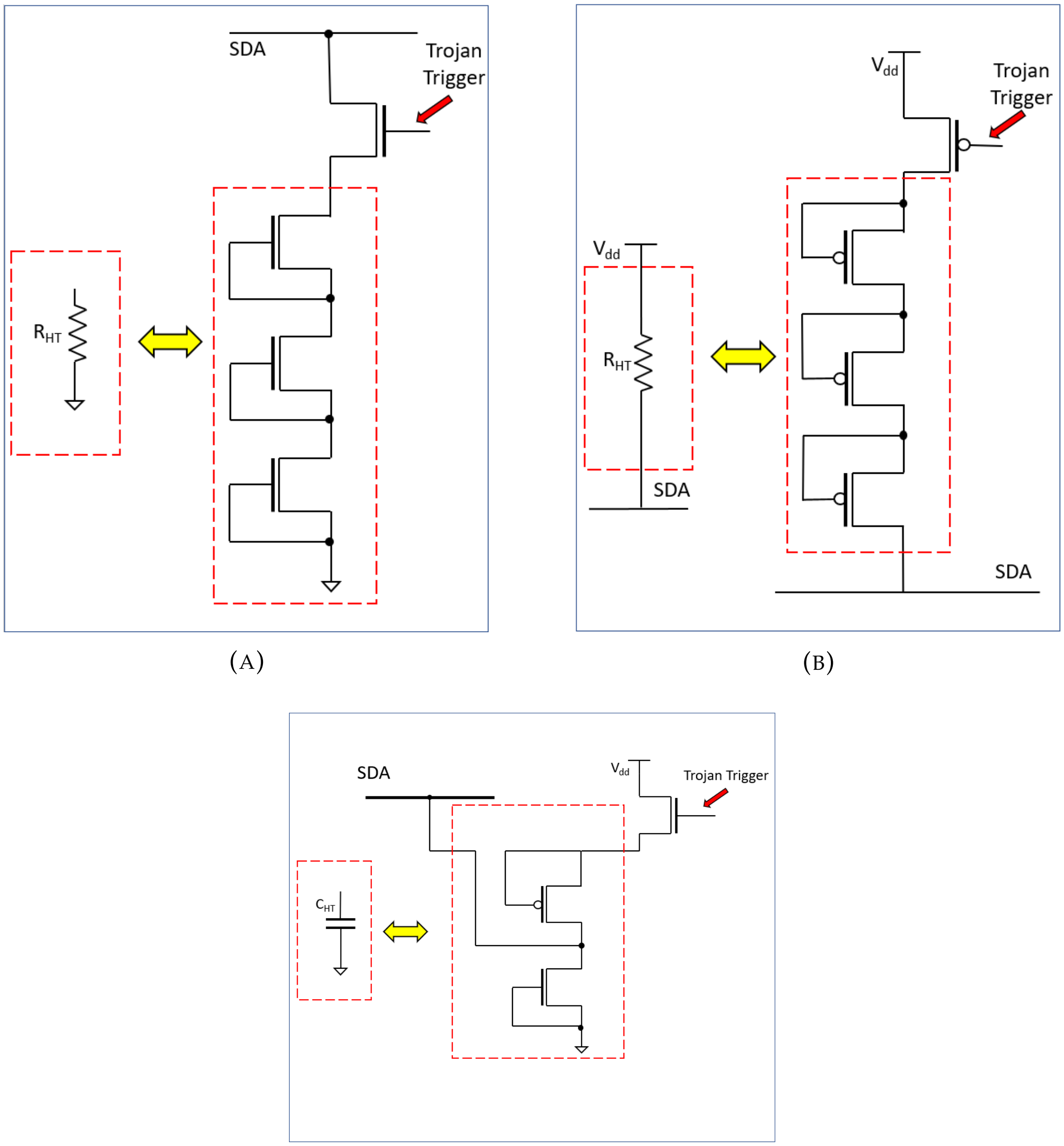
FIGURE 5.4: Impact of (a) resistor and (b) capacitor based analog Trojans on the I^2C data line. [55]

5.2.2 Demonstration of Attack Examples

1. Attack on I^{2}C Data Line SDA: We continue to use the I^{2}C master-slave interface to demonstrate practical attacks. As shown in Fig. 5.3, three analog Trojans are inserted

on the data link between a master and a slave. The Trojan is implemented in a format of pull-up or pull-down resistor R_{HT} and capacitor C_{HT}. The Trojan will be activated by an internal or external trigger signal. A simple pull-down resistor R_{HT} could mute the valid bit 'high', as shown in Fig. 5.4(a). In contrast, the capacitor based analog Trojan could lead the valid bit 'low' to go 'high', as shown in Fig. 5.4(b). Both Trojan cases in Fig. 5.4 demonstrate the analog characteristics of Trojans appeared in the I^2C communication channel.

2. Attack on I^2C Clock Line SCL: As the clock line plays a critical role in the master-slave communication protocol, it could be a primary target of analog Trojans. Since the slave is an off-chip device, the clock line spans both digital and analog domains. If there is no specific clock regulation available on board, the master-slave interface is vulnerable to clock attacks, including clock mute and clock split. A simple resistor or capacitor shown in Fig. 5.3 will be sufficient to execute clock attacks. We will introduce three different clock attacks based on the connection of the analog Trojan to ground and power source. Fig. 5.5 three different analog Trojan. Fig. 5.5 a) shows, an equivalent resistance can be represented by three nmos shorted to ground. Fig. 5.5 b) shows an equivalent resistance can be equal to the pmos shorted to a power source. The last Fig. 5.5 c) shows an equivalent capacitor to nmos and pmos combinational circuit shorted to the ground. As I^2C data transmission heavily relies on the clock line, a clock split attack will sabotage the data frame as shown in Fig. 5.6. If carefully crafted, the compromised data frame will be acknowledged as if normal. It is easy to implement a clock mute attack, as well. Fig. 5.7 illustrates the result of a clock mute attack performed in the master-slave interface between a Xilinx FPGA chip and an off-chip temperature sensor. The muted clock cycle successfully leads to the



(A)

(B)

(C)

FIGURE 5.5: Impact of (a) resistor connected to on the I^2C data and (b) resistor connected to power source and (c) capacitor based analog Trojans on the I^2C data line.

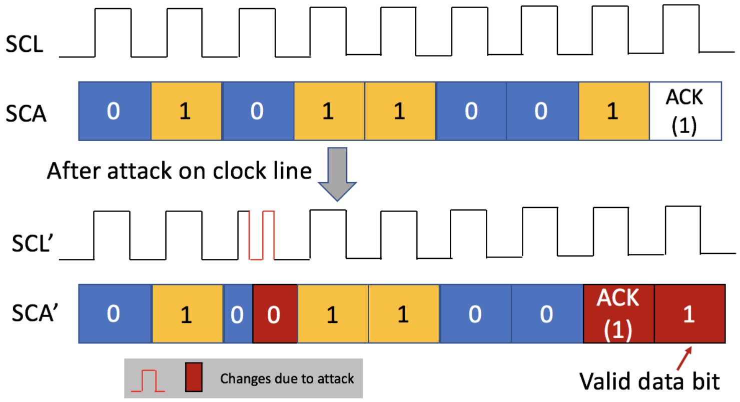


FIGURE 5.6: Impact of a clock split attack on data frames in I^{2}C communication [55].

modification of the most significant bit of a data frame, which represents the sensed ambient temperature.

5.2.3 Challenges on Analog Attack Detection

Recent literature [50] reports that an analog Trojan formed with few transistors is powerful enough to alter the system priority. Analog Trojans also appear in a phase locked loop and sensor controllers [57]. Compared with digital attacks, analog attacks can be performed on more surfaces and cost less hardware. As a result, analog Trojans pose more challenges on attack detection than digital Trojans. The main reasons are as follows: (1) more trigger mechanisms could be exploited to activate analog Trojans, (2) analog threshold for the attack determination makes the detection less reliable, (3) more ambient parameters would cause false positive, and (4) more asynchronous signals involved in the system would need more precise sampling for the process of attack detection. Overall, the balance of false positive/negative and the sensitivity of attack detection is extremely challenging for the analog world.

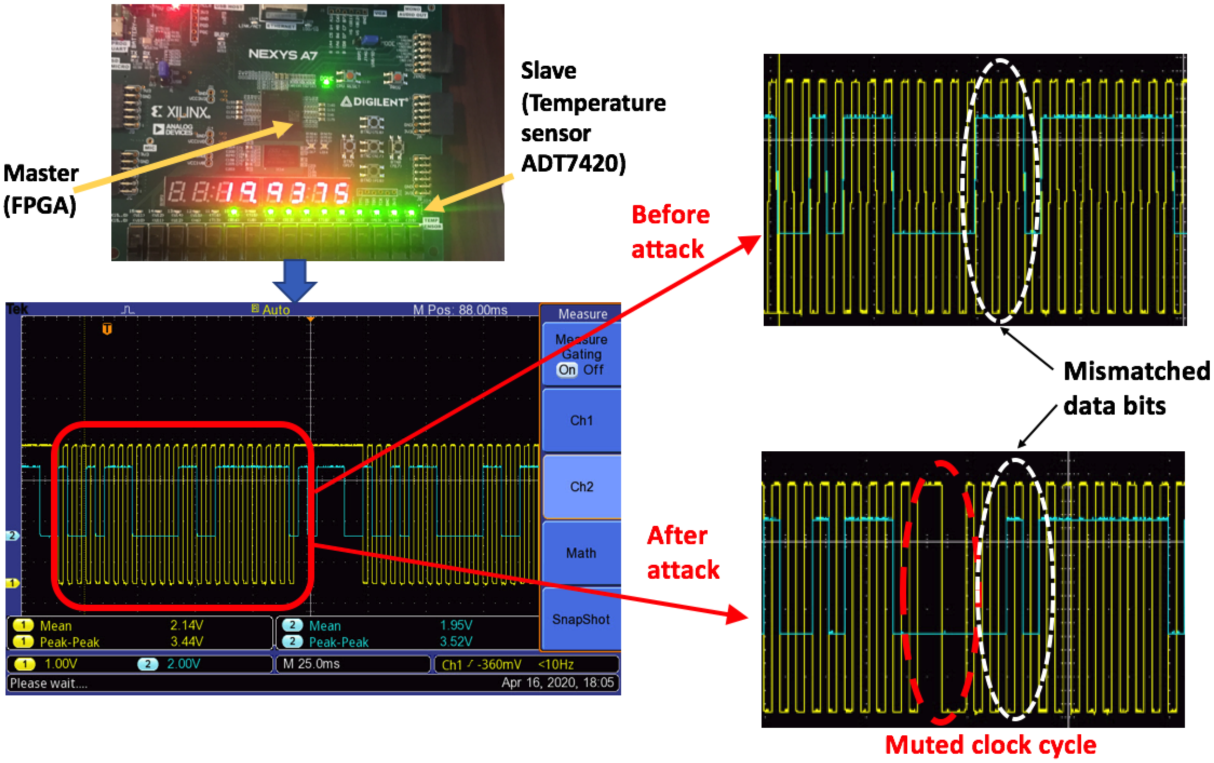


FIGURE 5.7: Oscilloscope reading of data and clock lines for an I^{2}C interface in the scenario of clock mute attack. [55]

5.3 Proposed Method for Analog Trojan Detection

We propose an obfuscated Trojan detection method, *ADobf*, to thwart clock attacks in I^{2}C communication. Our ADobf method provides a high sensitivity to the compromised clock signal and meanwhile protects the attack detection module itself with an obfuscation key. The architectural diagram of our key idea is depicted in Fig. 5.8. The proposed ADobf based attack detection method compares the clock signal from the clock generator in the master device with the voltage from the I^{2}C clock line. Due to the switching delay induced by the open-drain transistor, there will be one (and only one) voltage glitch if the

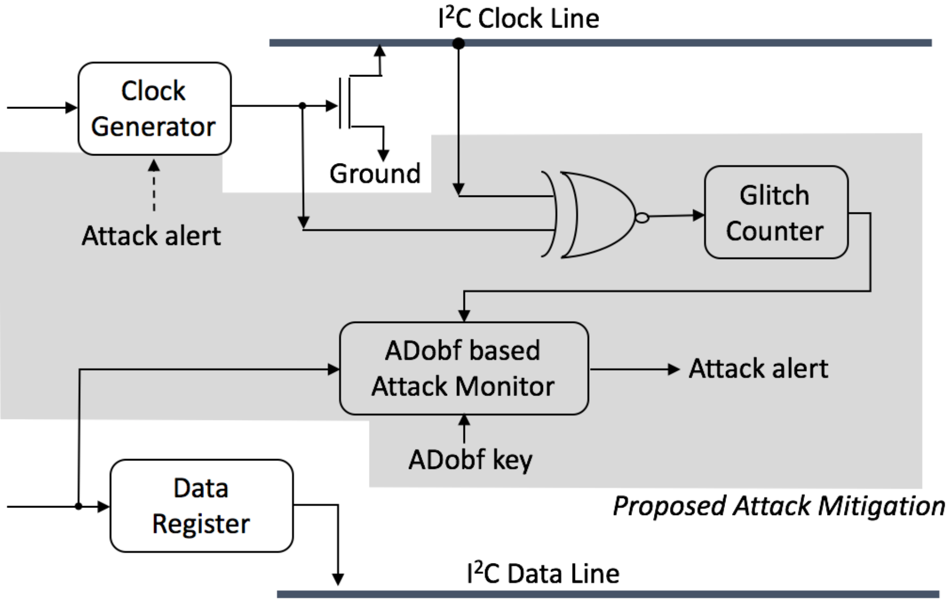


FIGURE 5.8: Proposed architecture diagram. [55]

clock signal is propagated normally. The *Glitch Counter* collects the voltage glitches continuously and assists the *ADobf based Attack Monitor* to determine the presence of attacks. An attack alert will notify the clock generator and prevent the data line from accepting malicious data frames.

More details of the ADobf based attack monitor are described in Algorithm 1. On each rising edge of the clock glitch, the glitch counter increases by 1. The counter content is compared with the clock threshold *Obf.threshold*, which is the correct number of clock glitches without any attacks. The fact that the glitch counter captures fewer glitches than *Obf.threshold* indicates some clock cycles being muted. On the contrary, if more glitches are obtained, a clock split attack could happen in the past measurement period. The value of *Obf.threshold* is a run-time and obfuscated parameter, which is not available for attackers who do not have the obfuscation key *obf.key*. Although attackers could calculate the number of clock cycles that each data frame takes to transfer over the I^{2}C interface,

Algorithm 1: Proposed ADobf detection method against attacks on master-slave interface.

```
Data: clock line, data line, cmd instruction, obfkey
Result: Attack alert
$CLK_{glitch}$ = $CLK_{gen}$ XNOR $CLK_{I2Cbus}$;
Obf.threshold = I2C_cmd_decode (cmd.instr, Obf.key, cmd.start, cmd.stop);
Initialize Glitch.counter;
while $CLK_{glitch}$ rising edge ↱ do
    Glitch.counter++;
    if cmd.stop then
        if (Glitch.counter < Obf.threshold) then
            Clock muted detected;
        else
            if (Glitch.counter > Obf.threshold) then
                Clock split detected;
            else
                Normal operation;
            end
        end
        Reset Glitch.counter;
    else
        Glitch.counter++;
    end
end
function I2C_cmd_decode(*);
  Calculate no. clock cycles (cmd.instr, cmd.start, cmd.stop) → NumCycle;
  NumCycle & Barrel_shifter(obf.key) → Obf.threshold;
  return Obf.threshold;
```

they will not be able to predict the key-dependent *Obf.threshold*. For simplicity, we use a barrel shifter to rotate the obfuscation key and then perform a bitwise AND logic with the number of clock cycles *NumCycle* in the attack-free scenario. Since only the master user knows the obfuscation key, the proposed ADobf attack detection method is capable of resisting clock attacks originated from someone having access to the I^{2}C bus.

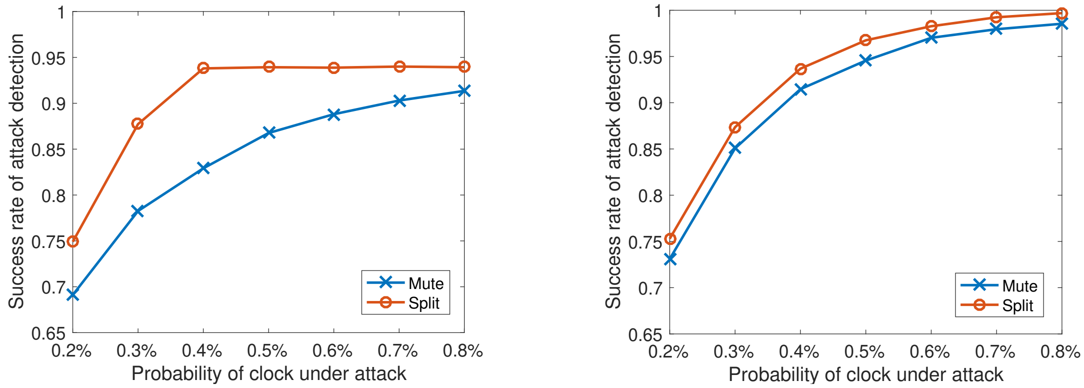
5.4 Experimental Results

We evaluated the proposed ADobf method in the context of I^{2}C interface. The attack detection method was implemented in Verilog HDL and the master module was synthesized with a 45nm FreePDK technology.

5.4.1 Effectiveness of Obfuscated Attack Detection

1. Success Rate of Detection: We transmitted 1000 data bits from an I^2C slave to a master. We set 7 and 8 respectively to the number of the address bits for the slave device and the size of each data frame. The clock line for the I^2C channel randomly tampered so that some clock cycles are muted or split. Each compromised clock cycle led to the loss of one valid data bit. We varied the number of clock cycles under attack from 2 to 8. As shown in Fig. 5.9, the success rate of attack detection of the proposed method increases with the number of clock cycles under attack and the size of the obfuscation key. When the probability of clock tampering is 0.2%, our detection against the clock mute attacks is around 0.7 (0.69 and 0.73 for the key size of 4 and 16, respectively). As the clock line was attacked with a higher frequency (0.8%), our attack detection method achieves a detection rate of above 0.95. Fig. 5.9 (a) and (b) also indicate that it is easier to detect clock split attack then clock mute attack. A longer obfuscation key will further improve the success rate of clock attack detection.

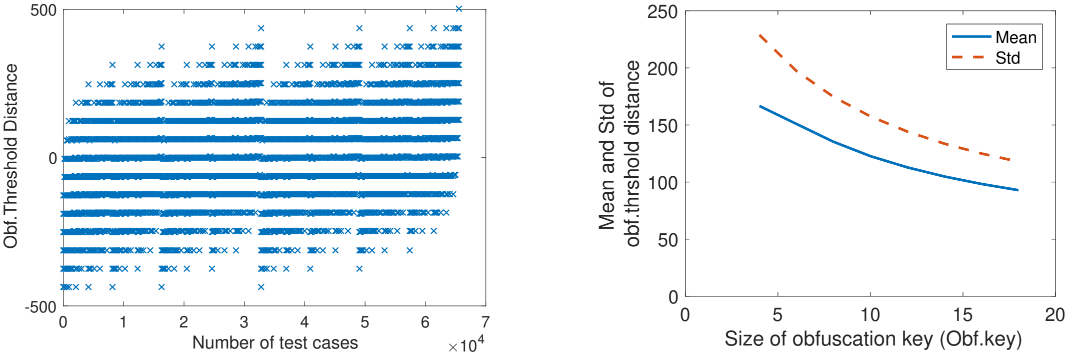
2. Unpredictability on Obfuscation threshold: The strength of our ADobf method relies on the unpredictability of obf.threshold. If the comparison threshold is known, adversary may find a way to bypass the countermeasure. In this subsection, we assess the unpredictability of obf.threshold. Given 16-bit obfuscation key, we randomly selected one key vector and used 1000 clock cycles to generate the obf.threshold for all possible wrong keys. Fig. 5.10 (a) shows that the difference between the obf.threshold obtained from correct and incorrect key cases is in a wide range of -500 to 500. Fig. 5.10 (b) further confirms that the average distance and the standard



(A)

(B)

FIGURE 5.9: The success rate of proposed ADobf attack detection with the key size of (a) 4 and (b) 16 [55]

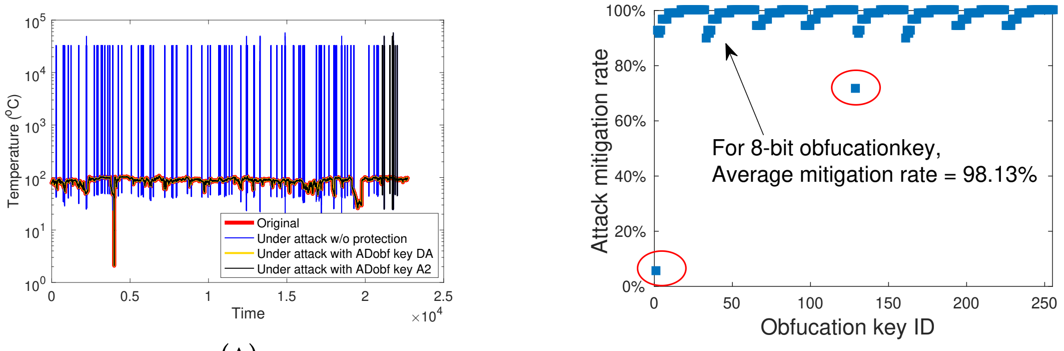


(A)

(B)

FIGURE 5.10: Unpredictability of proposed ADobf method. (a) Obf.threshold distance between the 8-bit correct key and wrong keys. (b) Statistics of obf.threshold for various key sizes. [55]

deviation are large for different key sizes, which validates that ADobf is capable of providing a high unpredictability and thus thwarts the attack from reverse engineering on the proposed countermeasure.



(A)

(B)

FIGURE 5.11: Attack mitigation achieved by proposed method. (a) A temperature benchmark from NAB transmitted over a compromised I^{2}C interface, and (b) attack mitigation rate for different obfuscation keys. [55]

3. Attack Mitigation Rate: Next, we used the I^{2}C master-slave interface to transmit the Numenta Anomaly Benchmark (NAB) [59], *machine temperature system failure*, in which 22695 floating numbers were reported. Each floating number for the sensed temperature was represented by 32 bits, 16 for integer and another 16 for fraction. We introduced 100 clock glitches in the NAB transmission. As shown in Fig. 5.11 (a), the original NAB carries one abnormal data point (below 10 degrees); however, the clock mute attack increases the number of abnormal temperatures to close 100. Our proposed ADobf method significantly alleviates the clock mute attack. The mitigation effect varies with different obfuscation keys. We examined the mitigation rate for all possible 8-bit obfuscation keys. As shown in Fig. 5.11 (b), our method achieves an average attack mitigation rate of 98%. There are only two cases out of 256 below the mitigation rate of 80%.

TABLE 5.1: Comparison of Performance and Hardware Overhead.

Metric	Area (um^2)	Dynamic Power (uw)	Leakage Power (uw)	Critical Path Delay(ns)
Baseline Master	2320.22	12.39	11.20	0.93
ADobf Master 4	2525.77	14.88	12.23	0.94
ADobf Master 8	2696.12	17.99	13.41	0.94

5.5 Hardware Cost

The proposed attack detection and mitigation method bring in moderate hardware overhead as reported in Table 5.1. The ADobf module costs 8.9% and 16.2% more area for 4-bit and 8-bit keys, respectively. The power consumption increases accordingly. As our method runs in parallel with normal I^2C operation, the critical path delay is only increased by 1%.

5.6 Conclusion

Complementary to the studies on general digital and analog Trojans, this work focuses on the investigation and mitigation of the security threats on the I^2C interface between digital master and analog slave devices. Unlike the sensor-based abnormal clock detection, the proposed ADobf algorithm achieves a high attack detection (close to 1) against 0.8% attack injection rate. Our case study with NAB benchmark shows that the proposed method can mitigate over 98% clock mute attacks.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

IoT devices have increased the network connectivity and computing capability of the embedded system. Large scale deployment of IoT have affected our lives significantly. Study shows IoT device played an import role in the application of civil, industrial and might expand extensively into military use in the future. However, IoT devices lack protected and secured protocols both in the software and hardware side, which might work as an entry point for the adversary to launch malicious attacks. Most of the existing monitoring capability of the IoT device is currently unnoticed by the developer side. As a result, Hardware Trojan (HT) can lead to potentially hardware protocol vulnerability. This thesis exploited the protocol vulnerability and gave a solution for analog base HT malicious activity.

In Chapter 3, we exploit the smart bulb's Bluetooth low energy protocols to covert exfiltration data in conversational secure air-gapped networks. This section briefly introduces the security threats in the air-gapped network and potential vulnerabilities of smart light bulbs. The handshaking procedure and authentication mechanism between host and client occur in the General Attribute Profile (GATT) and General Access Profile layer of the protocol stack. We exploited the Attribute protocol layer to control the device and implement covert data-channel. Nrf-sniffer application sniffed the data between two Bluetooth connected devices. The captured Bluetooth data is then exported to log files, which revealed the smart bulb's MAC address. A close analysis allowed us to find the entry to take control of the smart bulb. By examining the GATT layer, the characteristic address of the bulb is identified. We used Characteristic value 0xFFE9 to write a value and thus change the color GATT transactions using nrf-sniffer. To overwrite the characteristic number, first, the pattern of the payload is determined for the bulb. Analyzing the packets in Wireshark, the payload 56 00 00 00 27 0f aa was identified and revealed the value 00 00 27 corresponds to RGB and 56 00 RR GG BB 0f aa is the payload pattern used. Changing the RGB values of the payload pattern in nrf connect the bulb color can be changed without any issues. The observed pattern is used to write an automated script to change the bulb color. The protocol has several weaknesses. In our experiment, we exploited GATT and ATT layers of Bluetooth protocol to control the bulb.

In Chapter 4, we introduce the proposed MITM in the single wire protocols. The MSP340 acts as an HT circuit. It receives and request signal from the raspberry pi and relays the request to the dht11 sensor. The msp430 performs data manipulation with a bitwise operation. For our attack, we used a bitwise XOR operation to invert the third to the least significant bit of both temperature and pressure data. In the inactive stage, the msp430 returns the most recent modified data to raspberry pi and mimicking the data transmission from the dht11 sensor. Once the trigger is activated, the sensor data is then converted into a 64-bit floating-point and forcing some bits in the fraction to be zero or set bits. Hence, 0x00040F0000000000 is a 64-bits floating number will be used for unsetting the bit position of the original value. In 64-bits floating number system, the 14th, 21st, 22nd, 23rd, and 24th bits will be affected and set to zero. We set the value in such a way that the relative humidity will only be affected if the value crosses 55%. So, 55% is our trigger value for MSP340. Due to the unmonitored and unsecured single wire protocol between the sensor and the host and an adversary can easily eavesdrop on the critical information without any detection.

In Chapter 5, we used the I^{2}C master-slave interface to demonstrate an analog HT practical implementation. The master-slave interface of the I^{2}C is vulnerable to clock mute and clock split. We implement a clock mute attack resulting in the master-slave interface between a Xilinx FPGA chip and an off-chip temperature sensor. The clock line for the I^{2}C channel has randomly tampered, so that compromised clock cycle led to the loss of valid data bit. The attack detection rate depends on the number of clock cycles under attack and the obfuscation key's size. The clock tampering probability is 0.2% and our detection against the clock mute attacks is around 0.69 and 0.73 for the key size of 4 and 16, respectively. The detection rate of above 0.95 for higher frequency 0.8% clock line attack. The success rate of clock attack detection heavily depends on a longer obfuscation key. The proposed detection mechanism ADobf based attack depends on the technique of comparing the clock signal with the clock generator in the master device side. The standard deviation is large enough for different key sizes to validate the ADobf capability of mitigating attack from reverse engineering on the proposed countermeasure. In our experiment, we introduced 100 clock glitches in the NAB transmission. The original NAB carries one abnormal data point below 10 degrees. The clock mute attack increases with the number of unusual temperatures close to 100. As we mentioned earlier, the mitigation effect varies with different obfuscation keys, and our method achieves an average attack mitigation rate of 98%.

6.2 Future Work

Based on the results and simulation, our technique justifies the improvement of data transmission reliability in I^{2}C protocols. The proposed method detects a malicious attack, which is novel and effective. Our final solution is based on the I^{2}C master-slave interface. We want to extend our detection mechanism to other synchronous Serial Peripheral Interface (SPI) in future work. We plan to carry out a comparison experiment on the dynamic power, leakage power, critical path delay, and other overheads. We also plan to extend our work in the asynchronous protocol, such as Universal asynchronous receiver-transmitter (UART). Most of the chip operation relay on the clock signal, and mostly all glitch attacks use the clock signal. We also want to explore other types of fault injection, such as timing glitch and electromagnetic glitch. This gave an insight into how our detection mechanism will handle other glitch attacks. As IoT devices generate an abundance of data and identifying any anomalies in the detection method can be a changeling. In the NAB benchmark, our detection method shows 98% success rate. In the future, we will run an experiment in both the Etsy Skyline and Twitter ADVec benchmark. As the detection method process data in real-time, and it is a continuously learning and prediction process. Identification of any anomalies present in the detection method and anomaly detection effectiveness can be investigated during the benchmark comparison experiment.

References

[1] L. Columbus, "Roundup of internet of things forecasts and market estimates, 2016", *Forbes*, 2016, https://www.forbes.com/sites/louiscolumbus/2016/11/27/roundup-of-internet-of-things-forecasts-and-market-estimates-2016/#2b7dc31292d5.

[2] L. Dignan, "Iot devices to generate 79.4zb of data in 2025, says idc", *ZDnet*, 2019, https://www.zdnet.com/article/iot-devices-to-generate-79-4zb-of-data-in-2025-says-idc.

[3] J. R. Anna Gerber, *Connecting all the things in the internet of things*, https://developer.ibm.com/technologies/iot/articles/iot-lp101-connectivity-network-protocols/, 2020.

[4] *Internet of things world forum (iotwf) leaders announce new iot reference model and iotwf talent consortium*, https://telecomreseller.com/2014/10/14/internet-of-things-world-forum-iotwf-leaders-announce-new-iot-reference-model-and-iotwf-talent-consortium/.

[5] S. Farahani, "Chapter 3 - zigbee and ieee 802.15.4 protocol layers", in *ZigBee Wireless Networks and Transceivers*, S. Farahani, Ed., Burlington: Newnes, 2008, pp. 33 –135, ISBN: 978-0-7506-8393-7. DOI: https://doi.org/10.1016/B978-0-7506-8393-7.00003-0. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780750683937000030.

[6] N. Vidgren, K. Haataja, J. L. Patiño-Andres, J. J. Ramírez-Sanchis, and P. Toivanen, "Security threats in zigbee-enabled systems: Vulnerability evaluation, practical experiments, countermeasures, and lessons learned", in *2013 46th Hawaii International Conference on System Sciences*, 2013, pp. 5132–5138.

[7] P. Morgner, S. Mattejat, and Z. Benenson, "All your bulbs are belong to us: Investigating the current state of security in connected lighting systems", *CoRR*, vol. abs/1608.03732, 2016. arXiv: `1608.03732`. [Online]. Available: `http://arxiv.org/abs/1608.03732`.

[8] R. Roman, C. Alcaraz, J. Lopez, and N. Sklavos, "Key management systems for sensor networks in the context of the internet of things", *Comput. Electr. Eng.*, vol. 37, no. 2, 147–159, Mar. 2011, ISSN: 0045-7906. DOI: `10.1016/j.compeleceng.2011.01.009`. [Online]. Available: `https://doi.org/10.1016/j.compeleceng.2011.01.009`.

[9] M. A. Simplício, M. V. M. Silva, R. C. A. Alves, and T. K. C. Shibata, "Lightweight and escrow-less authenticated key agreement for the internet of things", *Comput. Commun.*, vol. 98, pp. 43–51, 2017.

[10] J. Czyz, M. J. Luckie, M. Allman, and M. Bailey, "Don't forget to lock the back door! a characterization of ipv6 network security policy", in *NDSS*, 2016.

[11] K. Angrishi, "Turning internet of things(iot) into internet of vulnerabilities (iov) : Iot botnets", *CoRR*, vol. abs/1702.03681, 2017. arXiv: `1702.03681`. [Online]. Available: `http://arxiv.org/abs/1702.03681`.

[12] C. Wueest, *Targeted attacks against the energy sector*, https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/targeted_attacks_against_the_energy_sector.pdf, 2014.

[13] *What is series (1): What is the osi reference model ?*, https://nicolaswindpassinger.com/osi-reference-model, 2018.

[14] A. Pand, *Nuclear power plants around the world unprepared for cyberattacks, warns new report*, https://www.ibtimes.com/nuclear-power-plants-around-world-unprepared-cyberattacks-warns-new-report-2126456, 2015.

[15] A. Chapman, *Hacking into internet connected light bulbs*, https://www.contextis.com/en/blog/hacking-into-internet-connected-light-bulbs, 2014.

[16] B. Rodrigues, *Arris cable modem has a backdoor in the backdoor*, https://w00tsec.blogspot.com/2015/11/arris-cable-modem-has-backdoor-in.html, 2015.

[17] Jmaxxz, *Backdooring the frontdoor*, DEF CON, https://doi.org/10.5446/36251 $Lastaccessed : 10Jul2020$, 2016.

[18] E. Fernandes, J. Jung, and A. Prakash, "Security analysis of emerging smart home applications", in *2016 IEEE Symposium on Security and Privacy (SP)*, 2016, pp. 636–654.

[19] E. Fernandes, J. Paupore, A. Rahmati, D. Simionato, M. Conti, and A. Prakash, "Flowfence: Practical data protection for emerging iot application frameworks", in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016, pp. 531–548.

[20] E. Fernandes, A. Rahmati, J. Jung, and A. Prakash, "Security implications of permission models in smart-home application frameworks", *IEEE Security & Privacy*, vol. 15, no. 2, pp. 24–30, 2017.

[21] A. Costin, J. Zaddach, A. Francillon, and D. Balzarotti, "A large-scale analysis of the security of embedded firmwares", in *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, 2014, pp. 95–110.

[22] V. Stoffer, *Outdated computers and operating systems*, `https://commons.lbl.gov/display/cpp/Outdated+Computers+and+Operating+Systems`, 2013.

[23] L. Markowsky and G. Markowsky, "Scanning for vulnerable devices in the internet of things", in *2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, vol. 1, 2015, pp. 463–467.

[24] A. Costin, J. Zaddach, A. Francillon, and y. i. a. S. p. u. h. p. U. m. a. Davide title = A Large-Scale Analysis of the Security of Embedded Firmwares booktitle = 23rd USENIX Security Symposium (USENIX Security 14).

[25] J. Stone, "A flaw in amazon's ring doorbells leaked customers' wi-fi credentials", *Cyberscoop*, 2019, `https://www.cyberscoop.com/ring-doorbell-wi-fi-flaw/`.

[26] B. Read, "Terrifying videos show men hacking into home security cameras", *The Cut*, 2019, `https://www.thecut.com/2019/12/amazon-ring-security-cameras-hacked-in-peoples-homes.html`.

[27] X. Lei, G. Tu, A. X. Liu, C. Li, and T. Xie, "The insecurity of home digital voice assistants - vulnerabilities, attacks and countermeasures", in *2018 IEEE Conference*

on Communications and Network Security (CNS), 2018, pp. 1–9. DOI: 10.1109/CNS.2018.8433167.

[28] D. Garcia, "Universal plug and play (upnp) mapping attacks", *DEFCON-19*, 2011.

[29] N. Kayastha, D. Niyato, E. Hossain, and Z. Han, "Smart grid sensor data collection, communication, and networking: A tutorial", *Wireless Communications and Mobile Computing*, vol. 14, no. 11, pp. 1055–1087, 2014. DOI: 10.1002/wcm.2258. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/wcm.2258. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/wcm.2258.

[30] A. Hansen, J. Staggs, and S. Shenoi, "Security analysis of an advanced metering infrastructure", *International Journal of Critical Infrastructure Protection*, vol. 18, pp. 3 –19, 2017, ISSN: 1874-5482. DOI: https://doi.org/10.1016/j.ijcip.2017.03.004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1874548217300495.

[31] X. Wang, H. Salmari, M. Tehranipoor, and J. Plusquellic, *Hardware trojan detection and isolation using current integration and localized current analysis*, 2008.

[32] R. Spreitzer, V. Moonsamy, T. Korak, and S. Mangard, "Systematic classification of side-channel attacks: A case study for mobile devices", *IEEE Communications Surveys Tutorials*, vol. 20, no. 1, pp. 465–488, 2018.

[33] A. A. Pammu, K. Chong, W. Ho, and B. Gwee, "Interceptive side channel attack on aes-128 wireless communications for iot applications", in *2016 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, 2016, pp. 650–653.

[34] D. Genkin, A. Shamir, and E. Tromer, "Rsa key extraction via low-bandwidth acoustic cryptanalysis", in *Advances in Cryptology – CRYPTO 2014*, J. A. Garay and R. Gennaro, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 444–461, ISBN: 978-3-662-44371-2.

[35] *Bridgeware: The air-gap malware*, https://cacm.acm.org/magazines/2018/4/226377-bridgeware/fulltext?mobile=false.

[36] M. Guri, O. Hasson, G. Kedma, and Y. Elovici, "An optical covert-channel to leak data through an air-gap", in *2016 14th Annual Conference on Privacy, Security and Trust (PST)*, 2016, pp. 642–649. DOI: 10.1109/PST.2016.7906933.

[37] Y. Mirsky, M. Guri, and Y. Elovici, "Hvacker: Bridging the air-gap by attacking the air conditioning system", *CoRR*, vol. abs/1703.10454, 2017. arXiv: 1703.10454. [Online]. Available: http://arxiv.org/abs/1703.10454.

[38] *3 types of software architecture for internet of things devices*, https://stanfy.com/blog/3-types-of-software-architecture-for-connected-devices [Accessed 9/20/2018].

[39] S. Zander, P. Branch, and G. Armitage, "Capacity of temperature-based covert channels", *IEEE Communications Letters*, vol. 15, no. 1, pp. 82–84, 2011, ISSN: 1089-7798. DOI: 10.1109/LCOMM.2010.110310.101334.

[40] M. Guri, Y. A. Solewicz, A. Daidakulov, and Y. Elovici, "Fansmitter: Acoustic data exfiltration from (speakerless) air-gapped computers", *CoRR*, vol. abs/1606.05915, 2016. arXiv: 1606.05915. [Online]. Available: http://arxiv.org/abs/1606.05915.

[41] M. Amoozadeh, A. Raghuramu, C. Chuah, D. Ghosal, H. M. Zhang, J. Rowe, and K. Levitt, "Security vulnerabilities of connected vehicle streams and their impact

on cooperative driving", *IEEE Communications Magazine*, vol. 53, no. 6, pp. 126–132, 2015.

[42] Z. Zhang, M. C. Y. Cho, C. Wang, C. Hsu, C. Chen, and S. Shieh, "Iot security: Ongoing challenges and research opportunities", in *2014 IEEE 7th International Conference on Service-Oriented Computing and Applications*, 2014, pp. 230–234.

[43] Y. Yang, H. Cai, Z. Wei, H. Lu, and K.-K. R. Choo, "Towards lightweight anonymous entity authentication for iot applications", in *Information Security and Privacy*, J. K. Liu and R. Steinfeld, Eds., Cham: Springer International Publishing, 2016, pp. 265–280, ISBN: 978-3-319-40253-6.

[44] T. Xu, J. B. Wendt, and M. Potkonjak, "Security of iot systems: Design challenges and opportunities", in *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2014, pp. 417–423.

[45] *Dht11*. [Online]. Available: `https://www.electronicwings.com/sensors-modules/dht11`.

[46] *How dht11 dht22 sensors work interface with arduino*. [Online]. Available: `https://lastminuteengineers.com/dht11-dht22-arduino-tutorial/`.

[47] J. Dofe and Q. Yu, "Novel dynamic state-deflection method for gate-level design obfuscation", *IEEE Transactions on CAD*, vol. 37, no. 2, pp. 273–285, 2018.

[48] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection", *IEEE Design Test of Computers*, vol. 27, no. 1, pp. 10–25, 2010, ISSN: 1558-1918. DOI: `10.1109/MDT.2010.7`.

[49] K. Subramani, G. Volanis, M. Bidmeshki, A. Antonopoulos, and Y. Makris, "Trusted and secure design of analog/rf ics: Recent developments", in *Proc. IOLTS*, 2019, pp. 125–128.

[50] K. Yang, M. Hicks, Q. Dong, T. Austin, and D. Sylvester, "A2: Analog malicious hardware", in *Proc. IEEE Symposium on SP*, 2016, pp. 18–37. DOI: `10.1109/SP.2016.10`.

[51] Y. Hou, H. He, K. Shamsi, Y. Jin, D. Wu, and H. Wu, "On-chip analog trojan detection framework for microprocessor trustworthiness", *IEEE Trans. on CAD*, vol. 38, no. 10, pp. 1820–1830, 2019, ISSN: 1937-4151. DOI: `10.1109/TCAD.2018.2864246`.

[52] Y. Hou, H. He, K. Shamsi, Y. Jin], D. Wu, and H. Wu, "R2d2: Runtime reassurance and detection of a2 trojan", *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 195–200, 2018.

[53] D. Deng, Y. Wang, and Y. Guo, "Novel design strategy towards a2 trojan detection based on built-in acceleration structure", *IEEE Transactions on CAD*, pp. 1–1, 2020, ISSN: 1937-4151. DOI: `10.1109/TCAD.2020.2977069`.

[54] S. K. X. Z. M. Tehranipoor and et al, "Detecting hardware trojans using on-chip sensors in an asic design", *Journal of Electronic Testing*, vol. 31, 11–26, 2015.

[55] M. R. Monjur, S. Sunkavilli, and Q. Yu, "Adobf: Obfuscated detection method against analog trojans on i2c master-slave interface", *IEEE 63rd International Midwest Symposium on Circuits Systems*, 2020.

[56] R. Jiménez-Naharro, F. Gómez-Bravo, J. Medina-García, M. Sánchez-Raya, and J. A. Gómez-Galán, "A smart sensor for defending against clock glitching attacks on the i2c protocol in robotic applications", *Sensors*, vol. 17, no. 4, p. 677, 2017.

[57] V. V. Rao and I. Savidis, “Protecting analog circuits with parameter biasing obfuscation”, in *Proc. LATS*, 2017, pp. 1–6.

[58] V. Rao and I. Savdis, “Mesh based obfuscation of analog circuit properties”, *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, 2019.

[59] *The numenta anomaly benchmark*. [Online]. Available: `https://github.com/numenta/NAB`.