Secure Data Acquisition for Physics-based Side Channel Cybersecurity

by

Aaron Propp

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2023

© Aaron Propp 2023

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

Arthur Grisel-Davy assisted in understanding what attacks can be detected using power-consumption physics-based cybersecurity, detailed in Section 3.3. Aaron Propp authored the remainder of the background and conducted the security analysis, implementation, and experiments.

Abstract

Cyber-physical systems (CPSs), such as planes and water treatment plants, are increasingly connected to the Internet. Connecting CPSs to the Internet allows for greater productivity but also increases the attack surface of these devices. Therefore, ensuring the security of CPSs is critical. One way to detect intrusions is through monitoring side-channel information, such as power consumption data.

This thesis proposes a data acquisition unit (DAQ) to monitor and securely transmit power consumption data to the cloud for security applications. Anomaly detectors can then use this data to raise security-related alarms. As the DAQ is connected to the Internet, it is essential to consider security threats against it. To this end, this thesis conducts a security analysis using the Canadian government's Harmonized Threat & Risk Assessment. In contrast to prior approaches, which have the detectors run locally, the detectors using the DAQ's data run in the cloud. This is enabled by the DAQ streaming the power consumption data to the cloud. Since the security-related anomaly detectors run in the cloud, they have access to more computational resources. Streaming has the additional advantage of scaling across multiple monitored devices. The DAQ is also significantly less expensive and more compact than an oscilloscope, which several prior methods use.

Acknowledgements

I would like to thank Professor Sebastian Fischmeister for his mentorship. The projects I participated were engaging and interesting and his grammar tips have improved my writing style. I would also like to thank Murray Dunne and Carlos Moreno for being great sounding boards whenever I needed it.

Dedication

This thesis is dedicated to my parents Debby Shapero Propp and David Propp. Thanks for all your love and support. I hope this thesis sits on your bookshelf and gets read.

Table of Contents

List of Figures ix

List of Tables x

List of Abbreviations xi

1 Introduction 1

1.1 Problem Statement 2

1.2 Related Work 2

1.3 Contribution 4

1.4 Thesis Organization 4

2 Background 6

2.1 Safety of Embedded Devices 6

2.2 Security for Safety-Critical Embedded Systems 7

2.2.1 Definitions 8

2.2.2 Case Studies 8

3 Physics-based Cybersecurity using Power Consumption 10

3.1 Overview 10

3.2 Physics-based Cybersecurity vs. Machine Condition Monitoring 12

3.3 Power-based Physics-based Cybersecurity 13

3.4 Architecture 15

4 Secure Data Acquisition for Power-Consumption-based Physics-based Cybersecurity **17**

4.1 Attacker Profile and Access 17

4.2 Security Assessment using Threat and Risk Assessment 18

4.3 Overview of Security Standards 26

4.4 Security Model 27

4.4.1 NIST Capabilities 27

4.4.2 Auxiliary Controls 32

4.5 Vulnerability and Residual Risk Assessments 34

4.5.1 Vulnerability Assessment 34

4.5.2 Residual Risk Assessment 38

5 Realizing Secure Data Acquisition **40**

5.1 Implementation Overview 40

5.2 Functionality 42

5.2.1 Transmitting a Power Packet 42

5.2.2 Updating the Firmware Over-the-Air 43

5.2.3 DAQ Device Twin 44

6 Experiments **46**

6.1 Qualitative Evaluation 46

6.2 Quantitative Evaluation 50

6.3 Discussion 58

6.4 Lessons Learned 60

7 Conclusion **62**

7.1 Future Work 63

References **64**

List of Figures

3.1 A general overview of capturing power consumption data via the DAQ. . . 15

5.1 Specific implementation of the DAQ. 40
5.2 The process for transmitting a power packet. 42
5.3 The secure update process. 43

6.1 Distribution of latency for different sampling frequencies. 52
6.2 Timing jitter for different sampling frequencies. 53
6.3 Dropped packet rate for different sampling frequencies. 54
6.4 Time to restart the DAQ. 55
6.5 Time to complete a remote update and resume communication. 57

List of Tables

1.1 Sampling rates of power consumption sensors. 3

4.1 Summary of asset values. 21
4.2 Summary of threat levels. 26
4.3 Vulnerability assessment table. 35
4.4 Summary of vulnerability assessment. 38
4.5 Residual risk assessment. 39

6.1 ETSI requirement fulfillment. 49
6.2 Throughput for various frequencies. 50
6.3 Levene's test for 5-minute vs 24-hour captures. 50
6.4 Welch's t-test for 5-minute vs 24-hour captures. 51
6.5 Usage of program and data memory. 56
6.6 Maximum stack usage compared to allocated stack size. 56
6.7 Encryption/decryption times. 58
6.8 Percent difference for different cryptographic operations. 60

List of Abbreviations

2FA Two Factor Authentication 34

AAMI Association for the Advancement of Medical Instrumentation 9

ADC Analog-to-Digital Converter 49

ADU Azure Device Update 18, 19, 40, 41, 47

AES Advanced Encryption Standard 20, 21, 28, 31, 33, 36, 44, 47, 48, 57

APT Advanced Persistent Threat 17

ARM Advanced RISC Machine 28, 33, 47

ARP Address Resolution Protocol 25

ASIL Automotive Safety Integrity Level 7

ATT Adversary Threat Level 17

CAN Controller Area Network 4, 41, 42, 44, 56, 59

CAPEC Common Attack Pattern Enumeration and Classification 14, 15, 22–25, 35

CBC Cipher Block Chaining 28, 31, 57

CIA Confidentiality, Integrity, and Availability 1

CLS Cybersecurity Labelling Scheme 46, 47

CPU Central Processing Unit 11

CWE Common Weakness Enumeration 35–37

DAQ Data Acquisition Unit 2–4, 14–16, 18–37, 39, 40, 42, 44, 46–49, 51–54, 58–63

DC Direct Current 13

DDoS Distributed Denial of Service 18, 22, 23

DFD Data Flow Diagram 19

DHCP Dynamic Host Configuration Protocol 41, 48, 56, 59

DNN Deep Neural Network 11

DNS Domain Name System 24, 37, 39, 48

DoS Denial of Service 14

DUT Device Under Test 3, 15, 16

ECC Elliptical Curve Cryptography 20, 21, 44, 47

ECDSA Elliptic Curve Digital Signature Algorithm 30

EDM Electrical Discharge Machining 4

EM Electromagnetic 11

ENISA European Union Agency for Cybersecurity 26, 27, 29, 30, 32, 33

ETSI European Telecommunications Standards Institute 47–49

FPGA Field Programmable Gate Array 18–20, 23, 29, 36, 40, 41, 49

GCC GNU Compiler Collection 55

GMAC Gigabit Media Access Controller 61

GP-TM Good Practice-Technical Measure 27, 30–33

HDBSCAN Hierarchical Density-based Spatial Clustering of Applications with Noise 11

HTTP Hypertext Transfer Protocol 24, 31

IAEA International Atomic Energy Agency 9

ICMP Internet Control Message Protocol 29, 37, 48

IDE Integrated Development Environment 55, 59

IDS Intrusion Detection System 1, 2, 11, 12, 14, 15, 23

IEC International Electrotechnical Commission 7

IoT Internet of Things 5, 11, 21, 22, 26–31, 46–48

IoT-SCM-PP IOT Secure Communications Module Protection Profile 27, 29, 34

IoT-SE-PP IoT Secure Element Protection Profile 27, 33, 34

IP Internet Protocol 24, 25, 34, 41, 56

ISO International Organization for Standardization 7, 9

ISR Interrupt Service Routine 42

JTAG Joint Test Action Group 29, 36, 48

LED Light Emitting Diode 41, 44, 56

LSTM Long Short-Term Memory Network 11

MAC Media Access Control 27

MCAS Maneuvering Characteristics Augmentation System 7

MCM Machine Condition Monitoring 10, 12

MCU Microcontroller Unit 18–20, 27, 47–49

ML Machine Learning 2

MPU Memory Protection Unit 48

MQTT Message Queue Telemetry Transport 28, 29, 34, 41, 43, 53

NAS National Aerospace Standard 9

NIST National Institute of Standards and Technology 30–32, 60

NISTIR NIST Interagency Report 26, 27

NTP Network Time Protocol 34, 48, 60, 61

OCTAVE Operationally Critical, Threat, Asset and Vulnerability Evaluation 60

OTA over-the-air 2, 23, 30, 36, 43, 46, 47, 60, 62

PBC Physics-Based Cybersecurity 2, 4, 10–13

PCB Printed Circuit Board 18

PLC Programmable Logic Controller 12

PoE Power over Ethernet 54

RAM Random Access Memory 11, 41, 43, 54, 55, 59, 61

RSA Rivest-Shamir-Adleman 28

RTOS Real-time Operating System 6, 28, 48, 57

SAS Shared Access Secret 20, 21, 28, 29, 31, 46, 47

SHA2 Secure Hash Algorithm 2 28, 33

SNTP Simple Network Time Protocol 42, 56

SOUP Software of Unknown Provenance 7

STRIDE Spoofing, Tampering, Repudiation, Information Disclosure, Elevation of Privilege 19

SWD Serial Wire Debug 29, 36, 48

TARA Threat Assessment & Remediation Analysis 60

TCP Transmission Control Protocol 25, 63

TLS Transport Layer Security 3, 28, 29, 34, 36, 37

TRA Threat and Risk Assessment 17–19, 21, 25, 26, 34, 62

UDP User Datagram Protocol 63

URL Uniform Resource Locator 43

USB Universal Serial Bus 9

Chapter 1

Introduction

An embedded device is any device that has a computer working within a more extensive physical system. Embedded devices, including cars, ships, factories, and planes, are everywhere. Kitchen appliances such as a stove or a refrigerator have embedded devices.

All embedded systems are subject to faults. A fault is the suspected cause of a system error, which can lead to system failure [9]. Most work ensuring failures do not happen in embedded systems has been in ensuring the system's safety. Safety measures focus on avoiding hazardous situations than can lead to catastrophic failure that could damage the environment, capital infrastructure, or life [43]. Quantifying the safety level often takes the form of guaranteeing a maximum number of unintentional failures per hour of operation [36]. The Therac-25 is a canonical example [52] of a safety-critical system failing to avoid a hazardous situation. Due to race conditions when accessing shared variables, massive overdoses of radiation, 75-100 times greater than they should have been, were given to patients causing three people to die.

In recent years, security has become more of a concern as an increasing number of devices are connected to the public Internet for convenience and ease of use. Security focuses on preserving the Confidentiality, Integrity, and Availability (CIA) triad. In other words, the goal is to preserve confidentiality and integrity while maintaining availability. A more detailed discussion of security is found in Section 2.2. Devices being connected to the public Internet opens them up to additional avenues of attack. One of the most prominent examples is the 2015 attack on a Jeep Cherokee by Miller and Valasek [63]. The authors controlled the steering and brakes by infiltrating the vehicle via the infotainment system.

One way to detect attacks is by using an Intrusion Detection System (IDS) designed to detect attacks on the system. Many IDSs focus on capturing network information such

as Ethernet packets or host-based information such as system logs [41]. These information sources are subject to being spoofed by intelligent viruses, leaving IDSs using them potentially ineffective.

Physics-Based Cybersecurity (PBC) is an alternative approach to detecting intrusions which relies on capturing physics phenomena and properties from the system. These phenomena include power consumption, temperature, vibration, pressure, and volume. It is not subject to spoofing as the physical phenomena cannot be hidden. A key enabler for ensuring PBC works smoothly is capturing good data that has been measured accurately and precisely. A more detailed discussion can be found in Section 3.1.

As PBC becomes more widely available [22], the security of the data acquisition is paramount. If the data acquisition is vulnerable to eavesdropping or tampering attacks, nobody will use it, preventing the development of sophisticated methods to detect intrusions, such as Machine Learning (ML) or statistics-based methods. The goal of this thesis is to develop a data acquisition system that captures power consumption data while being both stable and secure.

1.1 Problem Statement

The problem statement for this thesis is as follows: given a cyber-physical system which consumes power, the Data Acquisition Unit (DAQ) should securely and reliably monitor the power consumption. Updating the DAQ's firmware should be done securely over-the-air (OTA). The power consumption data captured should be made available to security-related systems designed to detect or respond to intrusions.

1.2 Related Work

Kocher et al. [44] first proposed side-channel analysis for extracting secret keys. Side-channel analysis has since been applied to intrusion detection [16, 38, 53, 55, 65, 71], program tracing [57, 64], fuzzing [73], and firmware disassembly [10, 19].

All authors except [38, 55, 64, 71] use either a commercial DAQ or an oscilloscope. While oscilloscopes often have a high sampling rate, they are expensive and bulky. The high cost and large size limit an oscilloscope's ability to be used outside a lab setting and deployed at scale. By contrast, using a current sensor is relatively unobtrusive and

Approach	Rate
Clark et. al. [16]	250 Ks/s
Li et. al. [53]	250 Ks/s
Liang et. al. [55]	12 Ks/s
Rott [71]	does not mention a sampling rate
Jung et. al. [38]	1.7 Ks/s
Moreno et. al. [64]	192 Ks/s

Table 1.1: Sampling rates of power consumption sensors.

inexpensive. From here on in, a commercial DAQ refers to a commercially available DAQ, whereas the term DAQ refers to the data acquisition unit discussed in this thesis.

Liang et al., Jung et al., and Rott all use current sensors to acquire data about power consumption. Moreno et al. read in power consumption via a commercially available sound card instead. While novel, it is also impractical to deploy at scale. The approach of this thesis is to use current sensors for most use cases and shunt resistors for high-voltage use cases.

Oscilloscope-based approaches generally have a much higher sampling rate than the other approaches, with the maximum sampling rate being 10 Gs/s [10]. All oscilloscope-based approaches have a sampling rate on the order of Gs/s. The other approaches use the sampling rates found in Table 1.1. The approach of this thesis uses a configurable sampling rate which ranges from 1 Ks/s to 2 Ms/s. This sampling rate range exceeds all the non-oscilloscope-based methods, including the commercial DAQ methods, while at the same time having a smaller form factor than a commercial DAQ.

In contrast to all prior work, the DAQ is meant to be part of a distributed network of devices, enabling two essential traits. The first is that detectors can now run in the cloud, allowing access to more computing resources than available near the Device Under Test (DUT). The second is that multiple DUTs can be monitored simultaneously in a scalable manner.

However, this approach also introduces two downsides. The first is additional latency before the power consumption information reaches the detector (compared to if the detector was running locally). The second is that the security of data transmission becomes a critical factor as it now has to transmit over the public Internet. Previously the side-channel information never left the local machine the detector was running on. The security of data transmission can be mitigated using proper encryption (i.e., Transport Layer Security (TLS)).

Commercial power monitoring is a related area which tries to profile power usage of equipment. The aim of commercial power monitoring is to determine which equipment is running and how efficiently it is running, rather than detecting intrusions [37]. Itoh et. al. aim to profile equipment in convenience stores such as refrigerators to identify potentially deteriorating equipment and improve energy efficiency [37]. Chacko and Deshmukh monitor distribution transformers to determine power usage throughout the day [14]. Kim et. al. propose Viridiscope which aims to track power consumption in residential environments [42]. Tristo et. al. aim to quantify the energy usage of Electrical Discharge Machining (EDM) milling [78].

Commercial power monitoring solutions would not be suitable for side-channel analysis for two main reasons. The first is scale. All the approaches mentioned are measuring voltages on the scale of hundreds of volts [37, 14, 42, 78]. By contrast the DAQ is measuring on the scale of volts. Second, commercial power monitoring approaches are designed to detect large changes such as if the refrigerator has been turn on or not, which require lower sampling rates. Tristo et. al. for example use a sampling rate of 0.85 to 1.15 Ks/s [78]. The DAQ that this thesis proposes has a range from 1 Ks/s to 2 Ms/s and thus has a much higher sampling rate than can be provided by commercial power monitoring.

1.3 Contribution

This thesis makes the following contributions:

- Present an approach for capturing power consumption and Controller Area Network (CAN) information to enable physics-based cybersecurity research and solutions.
- Apply a security analysis framework to the approach and identify compliance and residual risks.
- Demonstrate the technical feasibility and viability of the approach through an implementation and experimental quantification of key metrics such as latency, jitter, and drop rates.

1.4 Thesis Organization

The remainder of this thesis is organized as follows: Chapter 2 provides an overview of embedded system safety and security. Chapter 3 discusses PBC and provides an outline

of the data acquisition architecture. Chapter 4 conducts a security assessment of the data acquisition system and reviews several relevant Internet of Things (IoT) security standards. Chapter 5 examines aspects of the implementation. Chapter 6 demonstrates both the stability of the data transmission and the security of the cryptographic routines used. Finally, Chapter 7 concludes the thesis and details avenues for future work.

Chapter 2

Background

Safety and security are two key properties to preserve in embedded systems. Both of these concepts are essential to keep in mind when developing an embedded system and help ensure its overall functioning.

2.1 Safety of Embedded Devices

Safety-critical systems are systems where if a failure were to occur, it would lead to unacceptable consequences including loss of life, significant property damage, or damage to the environment [62]. The goal is to minimize the risk of unintentional faults, whether random or systemic. Safety-critical systems will often include a fail-safe mechanism, where the system will shut down gracefully if a failure occurs.

An excellent example of a safety failure is the patriot missile defense system incident in 1991. The missile defense system failed to intercept an incoming Scud missile during the Gulf War. The scud missile hit the barracks and killed 28 soldiers. The system failed due to clock skew as the external time reported drifted the longer the missile defense system was up [46].

Another example of a safety failure is the sudden unintended acceleration in Toyota vehicles between 2005 and 2011 [45]. Initially blamed on incorrect floor mats, Koopman's investigation found many potential software bugs. These include incorrect use of the watchdog, concurrency issues, use of recursion, and using an Real-time Operating System (RTOS) that was not safety certified. Eighty-nine people died as a result.

A final example is the Boeing 737 MAX-8 jet. It used an Maneuvering Characteristics Augmentation System (MCAS) to prevent the plane from stalling [76]. However, the MCAS overrode pilot input and only read values from one of the two sensors it was supposed to use. The faulty MCAS led to two planes crashing, killing hundreds of people before the fleet was grounded. All these examples show why safety is paramount: lives are at stake, and if there are bugs, it can lead to needless deaths.

However, one can never get rid of all risks [62]. The amount of risk that remains after implementing safety controls is often called residual risk. Residual risk is the amount of risk society is willing to tolerate in safety-critical systems.

Depending on the domain, there are different standards to help ensure functional safety. For example, International Organization for Standardization (ISO) 26262 concerns road vehicle safety [36]. There are four Automotive Safety Integrity Level (ASIL)s, with ASIL-A being the lowest and ASIL-D being the highest. ASIL-D only tolerates $< 10^{-8}$ random failures per hour of operation. International Electrotechnical Commission (IEC) 62304 governs the development of medical device software [32]. The standard focuses on isolating Software of Unknown Provenance (SOUP) from software specifically developed for medical devices. There are three classes, with class C being the most severe, as these devices can cause death if they fail.

IEC 60880 [30] covers safety systems in nuclear power plants, whereas IEC 62138 [31] covers safety-related systems in nuclear power plants. DO-178C is the safety standard used to certify software that goes in commercial airliners [18]. The highest software level is catastrophic, which means deaths and potentially the losing the plane.

2.2 Security for Safety-Critical Embedded Systems

Computer security focuses on preserving three fundamental properties: confidentiality, integrity, and availability [79]. Confidentiality means only authorized agents can access the data or system (whether hardware or software) being protected. Integrity ensures that data has not been tampered with by a third party. Finally, availability means that the service or data is accessible whenever one wants to access it. Although most security research addresses confidentiality and integrity, all three properties are critical to preserve.

2.2.1 Definitions

In any security discussion, it is necessary to define a few key terms. *Assets* are the object one is trying to protect, whether hardware, software, or data [79]. *Vulnerabilities* are the weaknesses inherent to the system that an attacker may be able to exploit. MITRE, a non-profit focused on cybersecurity, maintains a whole library of known software vulnerabilities [1]. *Threats* are the potential damage that could happen to a system. Often it is helpful to create a threat model of what particular consequences we are trying to protect against. An *attack* exploits a vulnerability to realize a threat. Informally, an attack is often called an exploit. Finally, a *control* mitigates a threat by protecting the vulnerability.

It is important to distinguish between vulnerabilities and exploits. Not every vulnerability is exploitable. Many static code analysis tools return a very high false positive rate of vulnerabilities [66]. Nadeem et. al. found FindBugs, a static code analysis tool for Java, to have a false positive rate of 45.7% [66]. An example is the Log4j vulrnebility [28]. Software may be bundled with the vulnerable version of Log4j. But if the software does not accept external input or have a shell running in its environment, it is not exploitable. However, every exploit needs a vulnerability. Without a vulnerability in the software, there is nothing to exploit [79].

As an example of these concepts consider a buffer overflow attack. The vulnerability is copying strings of unknown lengths into a buffer. The threat is that the code could be pointed to execute from an untrusted place. The attack would be to craft a string so the return address is overwritten with the attacker's desired location. The control which protects the vulnerability is to only copy a certain number of bytes regardless of the input size. However, if no external input is received, this buffer overflow vulnerability cannot be exploited.

In summary, security is about protecting against intentional faults introduced by the attacker, while safety focuses on random (e.g., bit flip) or systemic (e.g., a bug) faults. Even though safety may appear to be a more significant concern, it is still necessary to protect against attacks. Below are a few successful attacks against embedded devices that have caused considerable harm.

2.2.2 Case Studies

Many people use insulin pumps daily, including 350,000 people in the United States alone [59]. Radcliffe [70] demonstrates two attacks against an insulin pump to put the victim into a hypoglycemic state. Both attacks rely on being in close physical proximity to

the device (100-200 feet). The first attack involves spoofing sensor data to show a higher sugar reading than exists in the patient. The second attack involves changing constants that control how quickly insulin is dispensed. Both of these attacks demonstrate that the security of medical devices matters.

Stuxnet was a worm that destroyed many centrifuges at the Iranian Natanz enrichment plant [48]. It did this by slightly changing the rotation speed of the centrifuges while replaying back normal behaviour to the monitoring computers. Initially spread via Universal Serial Bus (USB) keys, it shows the importance of not only protecting the network but physical security as well.

In 2014, a German steel mill was subject to an email phishing attack [50]. As a result of the phishing attack, the attacker gained access to the corporate network, which was used to gain access to the plant network. Once in the plant network, it caused critical process components to become unregulated. Ultimately, this led to severe physical damage to the plant. Other than Stuxnet, this is the only other attack to destroy industrial control systems in the wild.

There are several cybersecurity standards specific to embedded system domains. Association for the Advancement of Medical Instrumentation (AAMI) TIR57 [6] focuses on the cybersecurity of medical device software, particularly where threats are also a safety risk. ISO 21434 [35] is the standard for automotive cybersecurity, focusing on cybersecurity risk management. International Atomic Energy Agency (IAEA) Nuclear Security Series No. 42-G [29] is the standard for cybersecurity in nuclear energy and covers both digital information and plant security. National Aerospace Standard (NAS) 9933 [67] covers cybersecurity in the aerospace and defense sectors and aims to provide recommended controls to preempt threats.

Chapter 3

Physics-based Cybersecurity using Power Consumption

PBC uses multiple different physical phenomena, not just power. 3.1 discusses different approaches and PBC as a whole. Machine Condition Monitoring (MCM) is similar to PBC, with different goals in mind. MCM helps to perform predictive maintenance, while PBC focuses on maintaining security.

3.1 Overview

PBC uses physical side-channel information to determine whether an attack is occurring. Other than power consumption, examples of other side channels are acoustic [54], temperature [56, 80], electromagnetic emissions [39, 72], vibration [75], and clock crystals [26, 84].

- Acoustic: Acoustic sensors can be placed near the monitored device to pick up acoustic signals, which can then be analyzed by comparing them to known behaviour. If a deviation is detected, that may be a reason to trigger an alarm. A significant challenge with this approach is filtering out background noise coming from other sources.
- Temperature: Microcontrollers consume energy during operation and as a result generate heat as a byproduct. A thermal sensor can then measure the device's temperature which often correlates with compute load. Change from the typical execution pattern can indicate the presence of an attack.

- Electromagnetic (EM) emissions: Computers will emit EM emissions at run time. Using EM sensing equipment, regular system operation can be profiled. If there is a significant deviation, this may indicate an ongoing attack.

- Vibration: A physical device may vibrate during operation due to the movement of mechanical components. A vibration sensor can capture this side channel during regular operation. Changes can then be monitored by a detector for attacks.

- Crystal clocks: All clocks embedded in microcontrollers are driven using crystals. These crystals can have minor imperfections and defects, giving each crystal a unique signature. If the change in signature is recognized by a detector, a change in the signature may indicate an impersonation attempt. An impersonation attempt is when an inauthentic entity (e.g., a device, user, etc.) is trying to present itself as an authentic entity [79].

Most approaches for PBC use a machine learning-based technique. Li et al. [54] and Wang et al. [80] both use an Long Short-Term Memory Network (LSTM)-based architecture. Tran et al. [75] use a Deep Neural Network (DNN)-based approach. Khan et al. [39] and Sehatbakhsh et al. [72] both use clustering approaches, with threshold-based clustering and Hierarchical Density-based Spatial Clustering of Applications with Noise (HDBSCAN) algorithms respectively.

Some of the works still use statistical methods. Liu et al. [56] use the Levenberg-Marquardt algorithm to analyze temperature time-series data collected by IoT sensors in a farm setting. Halder et al. [26] use Sequential Change-Point Detection for fingerprinting clock crystals. Zhao et al. [84] fingerprint clock crystals with the Empirical Rule and Dynamic Time Warping.

PBC differs from traditional cybersecurity mainly in two key ways: the source of information and instrumentation. Traditional cybersecurity relies on input data from the device, whether it is network traffic data, operating system logs, or system resource usage (e.g., Random Access Memory (RAM), Central Processing Unit (CPU) time). All these metrics, however, can be spoofed by clever malware such that the IDS will not trigger an alarm. Even Stuxnet could spoof the motor speed, so the operators did not notice anything unusual [48]. In contrast, all the physical emanations discussed, including power consumption, are much more difficult to spoof. If malware is causing a device to consume more power, the malware code must change before the device starts consuming less power.

Traditional cybersecurity often involves modifying the system itself [41]. The modification of the system can take many forms such as installing an anti-virus. PBC, by contrast,

requires little modification to the system, if at all. The information's reliability and the need for little instrumentation make PBC more suited to security for cyber-physical systems.

3.2 Physics-based Cybersecurity vs. Machine Condition Monitoring

MCM monitors various metrics from a machine, such as vibration and temperature, to determine the machine's health [49]. MCM aims to support a predictive maintenance scheme where machines can be fixed before they fail. Predictive maintenance provides two main benefits. The first is that it prevents unplanned downtime when a machine needs to be fixed because it failed unexpectedly. The second is that over-maintenance can be prevented by fixing the machine when it needs to be fixed.

MCM is often used to monitor equipment in industrial settings, including Programmable Logic Controller (PLC)-based machines, pumps, and motors. MCM uses metrics such as vibration, ball bearing wear, machine load, torque, acceleration, spindle speed, and temperature. All of these metrics can help support a predictive maintenance scheme. For example, an increase in a machine's temperature may indicate the machine's impending failure.

PBC will want more sensitive and frequent measurements for the side channels it uses. Since IDSs using PBC are trying to identify attacks, and some attacks have subtle signatures [16], PBC needs more granular data more frequently. By contrast, MCM might suffice with a measurement once every five minutes [58]. A good example is temperature. Wang et al. [80] are trying to measure the temperature of a microcontroller to correlate it with tasks, whereas MCM is trying to determine if a machine is overheating and in danger of becoming a fire hazard. The former requires more frequent granular data, and the latter could work with a measurement once a minute.

In summary, MCM is trying to identify when machine failure is imminent or machine performance is degrading. PBC is trying to identify an anomaly in the signal pattern, which may indicate an attack. Although both use sensors to measure similar metrics, in many use cases PBC needs data at a higher sample to be functional.

3.3 Power-based Physics-based Cybersecurity

Power is the amount of energy transferred per unit of time. In this context, Watt's Law can be used to calculate electrical power. Watt's Law is:

$$P = IV \tag{3.1}$$

Where P is power measured in watts (W), I is current measured in amperes (A), and V is the voltage measured in volts (V). For the monitored systems, the power consumed is primarily Direct Current (DC), but it varies as a function of computational usage.

Ohm's Law relates current to voltage over a conductor:

$$V = IR \tag{3.2}$$

Where R is resistance measured in ohms (Ω). If the load over which power is being measured is purely resistive, then Equation 3.1 and 3.2 can be combined to derive:

$$P = I^2R \tag{3.3}$$

Power consumption-based PBC relies on the fact that different computations require different amounts of power. This was demonstrated by Kocher et al. [44] in their work conducted to extract keys from side-channel information. If enough power traces are gathered during normal activity, a behavioural profile can be generated using statistical techniques or some form of machine learning.

When the device being monitored is specific enough, such as the medical compounder mentioned in Clark et al. [16], it would be highly unusual for it to be doing something outside its intended purpose. A violation of the behavioural profile can then trigger an alarm, alerting the user that something anomalous is happening. Notifying the user of anomalous behaviour is commonly implemented with a binary classifier, which classifies runtime behaviour as either normal or abnormal.

Several attacks can be identified by a power consumption-based PBC approach. Li et al. [53] detect attacks on a distribution terminal unit, including a distribution attack, an information gathering attack, and an overload attack. Clark et al. [16] detect several common malware examples including ramnit, delf, and dorkbot. Both Rott [71] and Liang et al. [55] focus on detecting attacks against additive manufacturing. The attacks are

modifications to the G-Code 3D printing file, including changing the scale, printing speed, and layer height.

Further, the following attacks are detectable by IDSs using the DAQ's data, classified according to MITRE Common Attack Pattern Enumeration and Classification (CAPEC). MITRE CAPEC is a framework that identifies attack patterns against hardware and software [1].

- CAPEC-112 Brute Force: The attacker attempts to try all possible values for a secret protecting an asset. If the login routine has a power consumption pattern, an IDS can detect repeated attempts by observing the pattern multiple times.
- CAPEC-115 Authentication Bypass: The attacker gains access to the system by evading the authentication mechanism. Similarly, an IDS can detect the absence of the login routine's power consumption pattern if an attacker bypasses the authentication mechanism.
- CAPEC-125 Flooding: The attacker engages in excessive interactions with the target over a short period, affecting its availability. An IDS can detect flooding by observing a specific power consumption pattern or the high power usage that flooding can cause.
- CAPEC-130 Excessive Allocation: The attacker aims to have the target allocate excessive resources, such as memory, to service its requests. This attack is detectable if it causes a high computing load, which would cause a correspondingly high power consumption, or is part of a Denial of Service (DoS) attempt. If part of a DoS attempt, the rationale is similar to that of a flooding attack.
- CAPEC-16 Dictionary-based Password Attack: The attack tries entries from a dictionary to guess the password. This attack would cause repeated login attempts, which would be detectable by observing the login routine's power consumption repeating.
- CAPEC-165 File Manipulation: The attacker modifies file contents or attributes to cause parsing errors. This attack is detectable during ransomware attacks or if it manipulates many files. Manipulating many files could cause increased power usage, and an IDS could detect ransomware attacks if the ransomware attacks have unique power consumption patterns.
- CAPEC-166 Force the System to Reset Value: The attacker forces the target to reset to a previous state. This attack is detectable during firmware downgrade. Different versions of firmware have different power consumption patterns [25].

- CAPEC-175 Code Inclusion: The attacker forces the target to execute locally stored code or retrieve and execute remotely stored code. This attack is detectable if the executed code affects power consumption.
- CAPEC-20 Encryption Brute Forcing: The attacker has access to the ciphertext and encryption algorithm used. They exhaustively search the key space to find the key that decrypts the ciphertext. An IDS could detect repetitions of the encryption routine's power consumption pattern.
- CAPEC-268 Audit Log Manipulation: The attacker modifies, deletes, or adds entries to the log file. Suppose log events are present in the log file but not in the power consumption data. Then the attacker inserted an entry in the log file, which could be detected by not finding the corresponding power consumption pattern. If events were missing from the log file but present in the power consumption data, then the attacker deleted a log entry which an IDS could detect by finding a power consumption pattern without its corresponding log entry.

3.4 Architecture

Below in Figure 3.1 is a general view of how data is captured from a DUT. The DUT is any device whose power consumption information is being collected, from personal computers to medical compounders. Ideally, no additional instrumentation is required on the device, and any power sensor being used is configured externally.

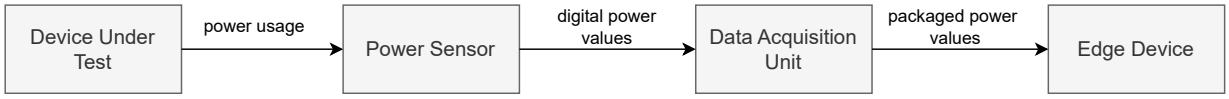

Figure 3.1: A general overview of capturing power consumption data via the DAQ.

This power consumption data is then fed through a sensor that either captures the data directly (e.g., an oscilloscope) or information that can be used to derive power consumption (e.g., shunt resistor and current sensor). The power values are then digitized into discrete values and sent to the DAQ.

The DAQ then calibrates, packages, and transmits these values to an edge device. An edge device stands in between the individual DAQ and the data store. It is responsible for multiple units and is often a fully-fledged computer compared to the microcontroller the

DAQ is based on. The edge device then bundles and transmits the data to the data store. Detectors can query the data store and raise alarms for potential attacks.

There are three main challenges with the current architecture for acquiring data. The first is decentralized data collection, with multiple DUTs in different locations. In other words, detectors cannot operate locally. This decentralization necessitates the edge device and data stores as described previously. The second challenge is that the data is streamed over the public Internet. This channel must be secured, or an attacker can eavesdrop and tamper with the data being streamed. This topic is discussed further in Chapter 4. Finally, the data is high resolution peaking at 2 Ms/s. The high resolution impacts how the firmware is developed and how the edge software is implemented downstream. How the issue of high resolution is handled will be discussed further in Chapter 5.

Chapter 4

Secure Data Acquisition for Power-Consumption-based Physics-based Cybersecurity

In any security analysis, it is critical to define an attacker profile. After defining the attacker profile, a security analysis is conducted, including enumerating assets and threats, implementing controls, identifying vulnerabilities, and calculating residual risk scores. The security analysis framework used is the Threat and Risk Assessment (TRA), designed by the Canadian government.

4.1 Attacker Profile and Access

Before conducting a security analysis, it is vital to identify agents that may want to attack the system. The United States Joint Chiefs of Staff recently declassified their cyber survivability implementation guide [5]. The guide lays out five different Adversary Threat Level (ATT)s: Extreme, Advanced, Moderate, Limited, and Nascent. The highest two levels, Extreme and Advanced, are nation-state actors. Examples of ATT Extreme are the Russian Foreign Intelligence Service and Advanced Persistent Threat (APT)-29. Examples of ATT Advanced are Russian APT-28 and Chinese APT-41. Examples of ATT Moderate are terrorists and state-sponsored networks [12]. Examples of ATT Limited are organized crime and insiders. Lastly, examples of ATT Nascent are hacktivists and script kiddies.

The threat actors at the levels ATT Limited and ATT Nascent are not state-level actors.

Disgruntled employees, an example of insiders, are a dangerous threat, especially since they may have access to the source code and cryptographic keys. They aim to make the DAQ inaccessible out of spite. A bot-network operator, an example of organized crime, sees the DAQ as another device in their network with which they can later launch Distributed Denial of Service (DDoS) attacks. The bot-network operator aims to remotely load his malicious code during an update so that the device responds to his controls. Script kiddies are motivated by fun and showcasing their capabilities, whereas hacktivists act in support of a cause.

The attacker's access to the system is also important to define. Figure 5.1 shows a block diagram of the DAQ. The attack model is that the attacker has access to all external communication. In other words, the attack can both eavesdrop and tamper with the DAQ's communication with the Edge device and the Azure Device Update (ADU) server. The attacker also has physical access. Physical access allows the attacker to tamper with the shunt resistor board. However, since the Field Programmable Gate Array (FPGA) and Microcontroller Unit (MCU) are on the same Printed Circuit Board (PCB), the attacker only has indirect access to the communication between the firmware and FPGA.

4.2 Security Assessment using Threat and Risk Assessment

The TRA is a framework designed by the Canadian government to manage risks to assets, employees, and services [4]. The TRA covers assets from the scale of the DAQ to nuclear reactors. There are six phases in total: identifying the scope, identifying and evaluating the assets, evaluating the threats, assessing current vulnerability levels, calculating residual risks, and recommending new safeguards.

All TRA assessments [4] define a scope. The scope of this assessment is confined to the physical DAQ board and the firmware running on it. The scope excludes protecting the source code for the firmware and FPGA from attackers and considering safeguards once the power samples reach the Edge Device. Any residual risks ranked Medium or higher should be addressed.

The TRA specifies five levels of asset values: Very High, High, Medium, Low, and Very Low. The levels are defined by the amount and type of damage in the event of compromise. The three types of damage considered are physical harm to people, psychological harm to people, and financial impact. When assessing an asset's value, all three harms should be considered, and the highest value should be chosen.

The five levels of asset values are:

- Very High: widespread loss of life, widespread psychological trauma, or an impact of greater than $1 000 000 000.
- High: potential loss of life for some or serious illness for many, serious embarrassment for many, or an impact greater than $10 000 000.
- Medium: serious illness to some or serious discomfort to many, serious embarrassment for some or serious inconvenience for many, or an impact of greater than $100 000.
- Low: serious discomfort for some or minor discomfort for many, serious inconvenience for some or minor inconvenience for many, or an impact greater than $1 000.
- Very Low: minor discomfort for some or negligible, minor inconvenience for some or negligible, or an impact of less than $1 000.

An asset can be assigned up to three values: one for each of confidentiality, integrity, and availability. As an example, consider the record of an employment for an employee. If the file were leaked, this would violate confidentiality. The leaked file would cause the employee a minor inconvenience, so the confidentiality value would be Very Low. If the file were destroyed or stolen, this would violate availability. In the worst case replacing the file would cost more than $1 000 but certainly less than $100 000, giving availability a value of Low. A similar argument could be made if the file was tampered with, giving Low value for integrity as well.

Khan et al. [40] propose an asset identification process in the context of a Spoofing, Tampering, Repudiation, Information Disclosure, Elevation of Privilege (STRIDE)-based framework for cyber-physical systems. Khan et al. suggest constructing a Data Flow Diagram (DFD) and, afterwards, identifying assets from the DFD. Figure 5.1 presents the DFD for the DAQ. Going through the DFD, one can identify all assets by examining each component for hardware and software assets, in addition to cryptographic keys.

The components in the scope of this TRA assessment are the shunt resistor board, the FPGA, the bootloader, and the firmware. The edge device and theADU server are external to the DAQ and, therefore, outside the scope. The shunt resistor board itself is a physical asset. There is no software or cryptographic information present. The FPGA hardware and software are both assets. No cryptographic information resides on the FPGA. The compiled firmware image consists of both the bootloader and the application firmware. The MCU the image resides on is also an asset. Finally, the bootloader's software contains

an Elliptical Curve Cryptography (ECC) authentication key pair. Also residing on the MCU but not part of the compiled image are the Azure SAS and remote update Advanced Encryption Standard (AES) keys. All hardware components fall into the physical device asset, defined below.

Summarizing the above process, the assets that need protection are as follows:

- Compiled firmware: The firmware image running on the DAQ board. Several threats might seek to render the firmware non-operational, which would be a threat to availability. An attacker may also seek to read the compiled firmware and reverse engineer it, a threat to confidentiality.
- FPGA binary: The design running on the FPGA. The main threat is against its confidentiality, as an attacker can try to read the binary and reverse engineer it.
- Physical device: The DAQ board itself, including any sensors and enclosures. The main threats against this asset are availability-based because if the device is stolen it can no longer provide its service.
- Azure Shared Access Secret (SAS) key: This symmetric key secures communication between the firmware and the Edge device. As with any key, the primary attacks will be trying to steal the key somehow, a threat to confidentiality.
- Remote update AES key: This symmetric key encrypts firmware images before sending them to the bootloader. The main threat is if the key gets stolen, which is a confidentiality concern.
- ECC authentication key pair: This is the key pair used to sign and verify new images. Again, the main threat is an attacker stealing the key, specifically the private (or signing) key, which is a confidentiality concern.

Reverse engineering the compiled firmware would have an impact of greater than $1 000, giving a confidentiality value of Low. If the firmware stopped running due to the DAQ being stolen or the firmware being tampered with, this would be a serious embarrassment for some as the attack would have to be explained to the deployment site, assigning a value of Medium to both integrity and availability. A similar argument can be made for the FPGA binary's asset values.

The physical device would have a replacement cost of less than $1 000 if stolen. However, it would seriously inconvenience the person who had to drive out to replace the device,

Asset	Confidentiality	Integrity	Availability
Compiled Firmware	Medium	Medium	Medium
FPGA Binary	Low	Medium	Medium
Physical Device	n/a	Low	Low
Azure SAS key	Medium	Medium	Medium
Remote update AES key	Medium	Low	Low
ECC authentication key pair	Medium	Medium	Medium

Table 4.1: Summary of asset values.

giving a Low value to its availability. If the device were tampered with instead of stolen, a replacement would still need to be issued, giving an Integrity value of Low. Confidentiality is not applicable.

Any of the keys leaking would be a serious embarrassment for some as the attack would have to be explained to the deployment site, giving a confidentiality value of Medium. If the Azure SAS key or the ECC authentication key pair were deleted or modified, this would be a serious embarrassment as they are both crucial for the device's operation, giving integrity and availability values of Medium. On the other hand, the remote update AES key can always be regenerated giving it integrity and availability values of Low. Table 4.1 summarizes the asset values.

Threats in the TRA are assessed on two dimensions: the threat likelihood and the threat gravity. Threat likelihood is divided into six frequencies: daily, 1-10 days, 10-100 days, 100-1,000 days, 1,000-10,000 days, and greater than 10,000 days.

These frequencies are mapped to likelihood levels via asset-location pairs. The most serious is the same location and similar assets. For example, if the threat you were assessing was the theft of office supplies, one would first look to see if any office supplies were stolen at one's specific office location. The second most serious is a different location but similar assets or the same location but different assets. Continuing the example, a different location but similar assets would be looking at the past frequency of office supply thefts at other offices. Different assets but the same location would be looking at the past frequency of computer thefts at the same office. Finally, the least serious is a different location and different assets, which would be the past frequency of computer thefts at other offices.

Given that none of the threats have been used as an attack against the DAQ, one can look at the frequencies against IoT devices and their keys (similar assets but different locations). With that in mind, the frequencies map to likelihood levels as follows: High (1-10 days), Medium (10-100 days), Low (100-1,000 days), and Very Low ($>$ 1,000 days).

As per step 5d, if the likelihood of a deliberate threat is rated Low or Medium, a second threat level should be computed based on a High likelihood. The justification given is that threat agent intentions can change far faster than the ability to devise and implement new controls.

All threats below are deliberate. IoT is a relatively recent area, and many practitioners are embarrassed to reveal when they have been the subject of successful cyber-attacks, skewing any publicly available past frequencies. It is reasonable to assume that each threat below will occur more than once every 1,000 days. Because of all these reasons and out of an abundance of caution, all threat likelihoods will be assessed as High initially.

The attacker's knowledge/skill and resources form the basis of threat gravity assessment for deliberate threats. High-threat gravity is both extensive knowledge/skill and extensive resources. Medium-threat gravity is limited knowledge/skill and extensive resources, moderate knowledge/skill and moderate resources, or extensive knowledge/skill and limited resources. Low-threat gravity is limited knowledge/skill and limited resources. For example, limited resources mean access to a consumer laptop, moderate resources mean access to more computational equipment than just a laptop, and extensive resources mean access to a supercomputer. Limited knowledge/skill means the average person, moderate knowledge/skill means the skill of a script kiddie, and extensive knowledge/skill means a nation-state actor's skill.

Given that all threat likelihoods are High, the threat level will be Very High if the threat gravity is High, High if the threat gravity is Medium, and Medium if the threat gravity is Low.

Threats, as explained in Section 2.2.1, are the potential for damage to the system. Below is a list of threats relevant to IoT devices and an explanation of their impacts. The list is a result of consulting several surveys on security and IoT, each of which proposes taxonomies of threats to IoT infrastructure [7, 15, 27, 82, 47]. The threats of interest to this security assessment are ones that affect the DAQ or its connections. Threats to elements higher up the stack, such as a DDoS attack against the Edge device, are critical but irrelevant. A CAPEC identifier [1] identifies each threat to allow others to quickly identify the threat and aid in forming the taxonomy.

- Theft of the DAQ (CAPEC-507) [47]: An attacker steals the DAQ. From a security perspective, it is difficult to distinguish between theft and the device being removed for repairs. Regardless if the DAQ is stolen, no power packets can be provided. This threat, when actualized, is often a stepping stone to the next threat. As an average person can perform this attack, the threat gravity is Low.

- Physically tampering with the DAQ (CAPEC-440) [7, 82]: Instead of stealing the DAQ, the attacker tampers with it. The attacker may seek to swap out the shunt resistor board with a defective one or damage other electronic components present. This attack requires some knowledge about electronic components, making the threat gravity Medium.

- Tampering with the calibration values (CAPEC-176) [82]: The attacker tampers with the parameters used to calibrate the power samples. The power samples then are incorrect, affecting downstream IDSs. This attack requires knowledge of the calibration parameters and where they are stored, making the threat gravity Medium.

- Replacing the shunt resistor board (CAPEC-531) [47]: The attacker replaces the shunt resistor board with an inauthentic copy. This attack seeks to inject malicious data from the sensor end, thus changing the power samples. It differs from physically tampering with the DAQ, as the goal is to affect the integrity of the power samples, not the availability of the DAQ. This attack requires some knowledge of electronic components, making the threat gravity Medium.

- Reading the firmware and FPGA binaries (CAPEC-121) [7]: If the attacker has physical access to the device, they can attempt to read out the contents of the FPGA and the microcontroller's flash. Access to the flash contents can allow them to reverse engineer the binaries and gain access to the keys stored on the device. Since this requires more than just a laptop (e.g., a programmer), the threat gravity is Medium.

- Node impersonation (CAPEC-151) [15, 27, 47]: This threat is pretending that a maliciously inserted device is an authentic device, often using keys and information gained from the previous threat. This threat violates both availability, as the actual device is no longer operational, and integrity, as the device's contents have been changed. This threat requires some technical skill to execute, so the threat gravity is Medium.

- Malicious code injection via OTA updates (CAPEC-186) [15, 27]: This threat focuses on exploiting the update process. If an attacker can load a different firmware onto the DAQ, it can later be used as part of a DDoS attack, like the MIRAI botnet [8] or for another purpose the attacker wants. This threat requires some technical skill to execute, so the threat gravity is Medium.

- False data injection (CAPEC-148) [27]: Based on the node impersonation threat, false data injection aims to have the DAQ run normally except for providing incorrect data

samples. Providing incorrect data samples aims to confuse any detectors relying on the integrity of the power samples transmitted. This threat requires some technical skill to execute, so the threat gravity is Medium.

- Side-channel attack on the cryptographic routines (CAPEC-189) [27, 82, 47]: As demonstrated by [44] and others, observing side-channel information such as power and timing on systems running cryptographic routines can leak the keys. This threat then aims to observe the DAQ in order to deduce the keys. Time measurements do not require physical access but power measurements do. This threat requires a great deal of knowledge about cryptographic routines, even if the computational resources needed are limited, making the threat gravity Medium.

- Sniffing attack (CAPEC-157) [15, 27, 47]: This attack aims to eavesdrop on the communication between the DAQ and Edge device. By observing what is sent, the attacker can understand the device's state and even steal power samples for training their own models. If a remote update is in progress, they can also gain access to the firmware binary. This threat requires knowledge of communication protocols like Hypertext Transfer Protocol (HTTP) to execute, making the threat gravity Medium.

- Adversary in the middle (CAPEC-94) [7, 15, 27, 47]: The adversary attempts to insert themselves between the DAQ and edge device. The attacker can try to change the power samples being transmitted, the commands being sent to the DAQ, and perform a malicious code injection as described above. The adversary may also choose to drop or record the message. This threat requires knowledge of the Internet Protocol (IP) stack, making the threat gravity Medium.

- Replay attack (CAPEC-60) [15, 82]: This attack does not require knowledge of the keys. Instead, the attacker captures packets from the Edge device and observes their effect on the DAQ. The attacker then replays these command packets to put the DAQ in an unauthorized state. This threat requires knowledge of the IP stack, making the threat gravity Medium.

- Sinkhole attack (CAPEC-598) [15, 7, 27, 82]: This attack aims to get the DAQ to communicate with a counterfeit Edge device. By broadcasting fraudulent information when the DAQ attempts to resolve the Edge's hostname (by setting up another Domain Name System (DNS) server), the DAQ can be tricked into communicating with the counterfeit Edge device instead. Unless the attacker also had access to the Edge's certificates, this would make the DAQ unavailable. This attack requires some skill and access to a computer to act as a fake DNS server, making the threat gravity Medium.

- Ping flood attack (CAPEC-125) [15, 7, 82, 47]: This attack aims to flood the DAQ device with ping requests in an attempt to bring the DAQ offline by overwhelming the DAQ's networking stack. More generally, this is a denial of service attack and can be done with other protocols such as Address Resolution Protocol (ARP). This attack requires some skill and access to a bot-net to execute, making the threat gravity Medium.

- Battery draining attack (CAPEC-391) [15, 27, 82, 47]: This attack also aims to bring the DAQ offline. Instead of overwhelming its networking stack, it seeks to drain the DAQ's battery by issuing many requests or loading a malicious image with a continuous loop of taxing calculations. As a cable and not a battery powers the DAQ, the equivalent would be unplugging but not stealing the DAQ. This attack only requires unplugging the DAQ, making the threat gravity Low.

- Buffer overflow attack (CAPEC-100) [15]: This attack tries to get the DAQ to execute malicious code or else get stuck by providing malformed input as commands. When the buffer the command is stored in overflows, it can overwrite the stack, corrupting it to execute code. More likely, the return address would invalid, crashing the DAQ. This attack requires extensive knowledge about crafting malicious input while requiring limited resources making the threat gravity Medium.

- Acknowledgement spoofing (CAPEC-594) [7]: The adversary seeks to spoof acknowledgements to messages the DAQ sent. By injecting this traffic, the adversary convinces the DAQ the Edge received its message, causing packets to be lost. This threat requires knowledge of the IP stack, making the threat gravity Medium.

- De-synchronization (CAPEC-220) [7]: The adversary sends messages to the DAQ requesting that it re-transmit frames. If the adversary claims enough transmission errors, the connection between the DAQ and the edge device will fall out of sync, destroying the connection. This attack requires knowledge about Transmission Control Protocol (TCP) to execute, making the threat gravity Medium.

Table 4.2 lists all threats and their threat levels for each of confidentiality (C), integrity (I), and availability (A).

Asset and threat evaluation is essential to the TRA process. The assets are varied and fall into three categories: hardware, software and cryptographic information. No asset is valued above a medium, as even if an attacker stole the cryptographic information, this would only cause serious embarrassment for some. Overall most threats target the

Threat	C	I	A
Theft of the DAQ	n/a	n/a	Medium
Tampering with the DAQ	n/a	High	High
Tampering with the calibration values	n/a	High	n/a
Replacing the shunt resistor board	n/a	High	n/a
Reading the binaries	High	n/a	n/a
Node impersonation	n/a	High	High
Malicious code injection	n/a	High	High
False data injection	n/a	High	High
Side-channel attack	High	n/a	n/a
Sniffing attack	High	n/a	n/a
Adversary in the middle	High	High	High
Replay attack	n/a	High	n/a
Sinkhole attack	n/a	n/a	High
Ping flood attack	n/a	n/a	High
Battery draining attack	n/a	n/a	Medium
Buffer overflow attack	n/a	High	High
Acknowledgement spoofing	n/a	n/a	High
De-synchronization	n/a	n/a	High

Table 4.2: Summary of threat levels.

availability of the DAQ or its integrity. Attacks on the confidentiality of the DAQ focus on reverse engineering the firmware or eavesdropping on its communications. After asset and threat evaluation, the next step in the TRA process is identifying relevant controls.

4.3 Overview of Security Standards

There are many security standards available for IoT devices. As the DAQ can be classified as an IoT device, these standards are relevant for the thesis. Security standards provide recommendations for controls to implement. NIST Interagency Report (NISTIR) 8259A provides general recommendations for IoT device cybersecurity [68]. It also references other security standards for IoT devices including the European Union Agency for Cybersecurity (ENISA) Baseline Security Recommendations for IoT [20]. The ENISA standard is more detailed and includes an asset taxonomy, a threat taxonomy, and a list of security measures.

The Common Criteria is an attempt to make a universal list of security practices which could be tailored to individual scenarios via profiles. With regards to IoT devices, there are two standards: IoT Secure Element Protection Profile (IoT-SE-PP) [34] and the IOT Secure Communications Module Protection Profile (IoT-SCM-PP) [33]. The IoT-SE-PP focuses on the device itself and the IoT-SCM-PP focuses on establishing a trusted channel.

4.4 Security Model

Controls are necessary to implement to mitigate vulnerabilities. Section 4.4.1 presents controls that fall into the core capabilities identified by NISTIR 8259A. Section 4.4.2 covers controls that are relevant and fall outside of NISTIR 8259A's core capabilities. As a matter of terminology, ENISA controls can also be referred to as Good Practice-Technical Measure (GP-TM), along with a number. For example, GP-TM-18 discusses secure software updates.

4.4.1 NIST Capabilities

NISTIR 8259A identifies six core capabilities for IoT devices: device identification, device configuration, data protection, controlling logical access to devices, secure software updates, and monitoring the device's cybersecurity state.

Device Identification

Unique physical and logical identifiers are essential because they enable asset management [68]. The unique logical identifier supports automated device management. The physical identifier is meant to be used when the logical identifier is not accessible. The DAQ fulfills both requirements. The logical identifier is the device name in Azure IoT Hub. The physical identifier is the MCU serial number and Media Access Control (MAC) address located on the sticker inside the enclosure.

Device Configuration

Device configuration supports future cybersecurity practices and increases the device's usability [68]. The device should be configurable, only authorized entities should be allowed to change the configuration, and authorized entities should be able to restore the device

to a secure configuration. Being accessible only to authorized entities supports access management and security of the configuration itself. Finally, restoring the device to a secure configuration can help contain an in-progress breach.

This capability is fulfilled as the DAQ's configuration can be changed via a device twin. Only those authorized to modify the device twin can change its configuration. Authorized entities can also restore the device twin to a secure configuration, defined as the default configuration.

Data Protection

Data protection is critical in preventing many eavesdropping attacks [68]. The device should use standardized cryptographic algorithms to protect data both at rest and in transit, authorized entities should be able to prevent access to all data on the device, and it should be possible to configure the use of cryptography (including key length) via the device configuration capability. Using standardized cryptographic algorithms is important in preserving the confidentiality and integrity of the communication channel. It is also important to have a fail-safe to prevent access to all data on the DAQ in case of a compromise. Finally, being able to change the cryptographic algorithms and key sizes can allow the device to remain relevant in the face of new attacks.

All transmission occur over Message Queue Telemetry Transport (MQTT) via TLS 1.2, using the cipher suite TLS_RSA _WITH_AES_128_CBC_SHA_256. The cipher suite specifies Rivest-Shamir-Adleman (RSA) to be used as the key exchange and authentication method, AES-128 Cipher Block Chaining (CBC) mode as the symmetric cipher, and Secure Hash Algorithm 2 (SHA2)-256 as the hash function. The program and data are stored in flash unencrypted. However, they are not accessible via the debugger as all flash reads and writes are disabled via the Advanced RISC Machine (ARM) security bit [74]. As the security bit is already set, all access to data stored on the device is disabled by default.

Unfortunately, Azure IoT Hub only supports configuring the type of security used, either SAS keys or X509 certificates, when creating the device [24]. The DAQ uses an SAS key which defaults to 264-bit, and the length is only configurable at creation time. Consequently, the key sizes are not configurable for the device's lifetime. Azure RTOS only ships with one cipher suite included. Other cipher suites would have to be added manually. Therefore, neither the cipher suite nor the key length are configurable via the device configuration.

ENISA extends the above protections by recommending enabling all security features by default and removing all unused features by default [20]. The more security features enabled, the harder it is for the attacker to launch a successful attack. Similarly, removing unused functionality reduces the attack surface. For this project, this is done by enabling TLS by default and disabling Internet Control Message Protocol (ICMP) support. ENISA also recommends using strong, unique default passwords. ENISA's password policy prevents an attacker from guessing the default password and assuming it is the same across multiple devices, making it harder for botnet recruitment. While not a password, the SAS key is unique per device and, being 264-bit, is very difficult to brute-force.

The IoT-SCM-PP recommends allowing the key to be updated [33]. The longer a key is used, the more likely it is to be exposed. By allowing the device's keys to be updated, you can replace compromised keys and even preempt a key from being compromised. The current architecture does not support updating a key. However, utilizing a backup key could be considered in the future.

Logical Access to Devices

Controlling logical access to devices is crucial as it reduces the opportunity for the attacker [68]. It should be possible to disable any local or network interface not necessary for the device's functioning, networking interfaces should only be available to authorized entities, and the ability to lock or disable devices with too many failed authentication attempts should be available. It is common practice to lock down an account if too many failed login attempts are made. Similarly, if there are too many failed authentication attempts, locking down the networking interface could prevent an attack that is in progress.

The Serial Wire Debug (SWD) interface used for debugging the microcontroller and the Joint Test Action Group (JTAG) interface used for debugging the FPGA are both disabled, reducing the attack surface. All communication over the networking interface takes place over MQTT via TLS. The networking interface is, therefore, only available to authorized entities. The last recommendation makes sense in the context of consumer IoT devices, as usernames and passwords are used to access the devices. In this context, with an SAS key being used, login attempts do not have the same meaning. In other words, an SAS key cannot be forgotten as a password can be since it is stored on the DAQ. Therefore this recommendation is inapplicable to the DAQ.

Secure Software Updates

A critical capability is updating the software running on the IoT device securely [68]. The IoT device must be able to be updated either remotely or locally, the update must be authenticated before installing it, authorized entities should be able to roll the software back, updating should be restricted to authorized entities only, and the ability to enable or disable updates should be available. Updates should also be configurable via device configuration, namely whether updates are automatic or manually initiated and if the device's users are notified when an update is completed.

Updating IoT devices allows developers to mitigate vulnerabilities leading to a more secure device. The update, however, must be authenticated, or an attacker can maliciously load their image onto the device. Ideally, the image is verified before it is installed, as the attacker may want to overwrite an already working image. Rolling the software back is critical as a bug may be introduced in a new release, in which case the device should be restored to the previous release until it is fixed. Updating devices should be a separate permission, following the principle of least privilege. If an attack is imminent, it is prudent to disable all updates in the meantime. Finally, depending on the business case, updates may or may not require notifications and should be automatically or manually initiated.

The DAQ can update remotely via OTA updates. All updates are authenticated via Elliptic Curve Digital Signature Algorithm (ECDSA) curve secp256k1, but only after installation, due to insufficient flash space for a second image on the Q19 model. Only 64 Kb of free space is left, but the image size is 256 Kb. As the goal is to maintain compatibility with the Q19 model, the authentication has to happen after installation. It is possible to roll back the software, but only if given a version number is greater than the one already installed. For example, version 1.3.0 is currently installed, but you want to roll back to version 1.2.0. Version 1.2.0 would be re-labelled to version 1.3.1 and sent out as an update. This process prevents the attacker from reverting to a previous vulnerable version. The ability to update software is a separate permission from managing the device, according to the principle of least privilege.

In Azure IoT Hub, all devices are updated manually, and devices can always be updated (i.e., there is no option to disable updates). Therefore the requirement to be able to enable/disable updates still needs to be fulfilled, in addition to configuring whether automatic updates are enabled or not. No notification system currently exists for the users of the DAQ, but there is a dashboard showing how many devices have been updated for a given rollout.

Beyond what National Institute of Standards and Technology (NIST) recommends, ENISA control GP-TM-18 states that the update server should be secure, the image should

be transmitted securely, and the image should not contain any credentials [20]. These measures exist to preserve the confidentiality of the firmware image and any credentials a device may use. The update server where the image is stored is the Azure containers service, and only authorized entities can access it. The image is transmitted over HTTP. HTTP is the primary protocol that the Azure Device Update service uses to transmit images. However it is encrypted in advance via AES-128 in CBC mode and is therefore secured even if the channel is transparent. Finally, the firmware is written not to contain any hard-coded credentials or data, including the device name, SAS key, and server hostname.

GP-TM-20 states that firmware updates should be backwards compatible. While not directly a security measure, it makes firmware updates easier to roll out, supporting vulnerability mitigation. When updating the DAQ, the policy is that no settings are touched, and only the program is updated.

Cybersecurity State Awareness

Being aware of the cybersecurity state is the final NIST capability [68]. The IoT device must be able to report its cybersecurity state, it should be clear when the device is in a good cybersecurity state versus about to enter a degraded cybersecurity state, only authorized entities should be able to view the state indicator, a subset of the authorized entities should be able to modify the state indicator, and the state information should be able to be made available on another service.

A cybersecurity state is anything that indicates the current functionality of the device, whether that be stack usage, error counters, or any other metrics. It is essential that these metrics be reported so that the health of the devices can be monitored. It is also critical that these metrics can distinguish between a good state and a degraded state. Otherwise, they are of little value since there is no way to tell if the device is in a degraded state or not. Controlling access to the state indicator follows the principle of least privilege and makes it less likely that someone accidentally changes the state indicator. Finally, being able to use the state indicator on other services allows multiple devices to be monitored at once.

The status thread serves this purpose and transmits various metrics, including stack usage, networking error counters, time since the last reset, and the total number of resets for the device's lifetime. The power packets can also be monitored to see when the most recent power packet was transmitted. If the total number of resets or the networking error counters are high, this may indicate a lack of network stability and, therefore, a degraded state. However, currently, these state indicators are not being monitored. As a result,

controlling access to the state is inapplicable since it is not stored. Since the state is not stored, they cannot be sent to another service. Nonetheless, a monitoring solution could be built off these state indicators if so desired.

ENISA also recommends logging various events, including regular functioning of the device to local storage, and that the logs should be retrievable on demand [20]. Logging can help identify intrusions as they are happening and serve as a useful tool when identifying what went wrong in the event of a successful attack. While logging does occur in the form of sending status information as discussed, it is not stored locally. There are two main reasons for this. The first reason is that there is only 64 Kb left of flash which is reserved for future use. The second reason is that in the event of an attack, an attacker may erase the space where logs are stored, obviating their usefulness.

ENISA also recommends monitoring the DAQ and auditing the security model on a regular basis [20]. GP-TM-56 states that the device's behaviour should be regularly monitored, which can help detect attacks if unusual behaviour is detected. Monitoring device behaviour is a worthwhile control for detection and could be enabled on the Edge device. Unfortunately, it is not enabled on the agreggator and is not a control that can be implemented directly on the DAQ. GP-TM-57 recommends periodically auditing to ensure security measures continue to be effective and to conduct penetration testing at least twice a year. While not directly a control, auditing helps ensure the rest of the controls remain effective. Currently, no regular auditing occurs with the DAQ. Regular audits could be explored later if more funding becomes available.

4.4.2 Auxiliary Controls

In addition to the baseline NIST controls, ENISA and Common Criteria recommend several additional security measures, including hardware security and device administration.

Hardware Security

ENISA recommends implementing hardware security measures, specifically an immutable hardware root of trust [20]. A hardware root of trust ensures that every program running on the device has been verified before it runs. Unfortunately, the Atmel SAME70 family does not provide a hardware root of trust. The bootloader is trusted unconditionally, but it does verify the firmware every time before running. ENISA also recommends employing a hardware security module to protect the keys and device identity. Using a hardware security module ensures the keys are physically isolated from the rest of the program. An

example is ARM TrustZone [3], a common feature built into many chips. Unfortunately, the Atmel SAME70 family only provides a security co-processor capable of handling some AES modes and SHA2 hashing.

System Safety and Reliability

Ensuring that the system is safe and reliable is a significant focus of ENISA [20]. GP-TM-15 recommends that the system should not cause physical injury or excessive property damage. Although not a security measure, devices should be safe. Given that the DAQ is simply monitoring power consumption, the worst thing that could happen is it overheats. However, it is in an enclosure which would absorb any excess heat. GP-TM-16 states that the device should be able to recover after entering a malfunctioning state. This control helps ensure availability, as even if the device temporarily becomes unavailable, it will be able to resume regular operation. The DAQ utilizes a hardware watchdog for this. If no packets have been sent in the past 16 seconds, the device will automatically reset to recover from its error state. GP-TM-17 states that standalone operation without network connectivity should be possible. Reducing reliance on network connectivity can increase the device's reliability, improving availability. In the case of the DAQ, its functionality depends on network connectivity, so this recommendation is unmet. However, as soon as network connectivity is restored, functionality resumes as usual.

Secure Input and Output Handling

Sanitizing input and output is important as it prevents classes of attacks like buffer overflows which rely on the program accepting untrusted output at face value. GP-TM-54 mandates this [20]. All input data is only copied up to a specified number of bytes and is automatically null-terminated. Outputted data follows a strict format for both power samples and status information.

Device Administration

The IoT-SE-PP recommends having a device administrator and separate security roles for different users [33]. Role separation aligns with the principle of least privilege and ensures that users have access to the least information possible, mitigating what a disgruntled employee can do. Role separation is practiced for administering the DAQ, with fine-grained permissions for accessing the devices, Edge devices, and update servers.

The IoT-SE-PP recommends identifying all users before allowing access [33]. In addition, the device version information should be queried beforehand to speed up access to the information. Identifying all users protects the confidentiality of the device configuration. All users are authenticated before accessing the DAQ device twins. Tag information is fetched before identification which can include version information. In addition to being identified, all users should be authenticated before being given access. Authenticating users supports confidentiality as it ensures the right users are accessing the device information, not just that they say they are the right users. All users log in via Two Factor Authentication (2FA), which ensures they are authenticated before accessing device information.

The IoT-SE-PP recommends that version information be queryable per device, supporting asset management [33]. That way, it is easy to identify which devices are missing functionality and potential security fixes. All DAQs have version information which can be queried.

Network Protection

The IoT-SCM-PP recommends having a list of allowed external network devices to check against before connecting, preventing connections to rogue devices [34]. This recommendation is not fulfilled as the firmware application uses MQTT over TLS, which does not support filtering, unlike raw IP packets [61]. However, this is a minor concern because the device only connects to the Edge device for communication and public Network Time Protocol (NTP) servers for timekeeping.

4.5 Vulnerability and Residual Risk Assessments

After identifying the threats and currently implemented safeguards, the TRA then identifies how vulnerable the DAQ currently is via the vulnerability assessment. The residual risk assessment then combines the asset, threat, and vulnerability tuple to compute each tuple's overall residual risk score.

4.5.1 Vulnerability Assessment

The TRA views controls through two lenses: prevention and detection. For prevention, there are three levels: High meaning probability of compromise is >75%, Medium meaning probability of compromise is 25-75%, and Low meaning probability of compromise is <25%.

Detection	Prevention		
	Low (N/A)	**Medium**	**High**
High	Medium	High	Very High
Medium	Low	Medium	High
Low (N/A)	Very Low	Low	Medium

Table 4.3: Vulnerability assessment table.

Regarding detection, there are three levels: High meaning assets are exposed to extensive injury, Medium meaning assets are exposed to moderate injury, and Low meaning assets are exposed to limited injury. If a control addresses either only prevention or only detection, the other lens is marked as not applicable.

As an example of what the prevention levels mean, take the case of a safe. If there is no lock on the safe, the probability of compromise is almost a 100% giving it a High vulnerability level. If the safe is locked with a simple combination lock that can be easily cut, the probability of compromise lowers to 25-75%. If a proper deadbolt is used instead, the probability of compromise falls well under 25%.

For detection levels, consider a web server. If intruders are not detected for several days, this exposes the web server to great harm. If instead there is a routine job that checks for intrusions once a day, this reduces the harm the web server is exposed to. If there is real-time malware checking, this exposes the web server to very little harm.

These two values can be combined to give an overall vulnerability level as shown in Table 4.3.

As explained in Section 2.2.1, a vulnerability is a weakness in the asset that allows a threat to be realized into an attack if exploitable. Below is a list of vulnerabilities that, if left alone, enable the threats defined Section 4.2. The CAPEC identifier identifies the appropriate weakness in the MITRE Common Weakness Enumeration (CWE) framework where possible [1, 2]. CWE does not cover physical or supply chain security in the same way as CAPEC, so those vulnerabilities do not have a CWE identifier.

- DAQ is not physically secured: If the DAQ is not physically secured, it becomes easier to steal it. No controls mitigate this vulnerability, so the vulnerability level is Very High.

- Power cable is not physically secured: If the power cable can be unplugged easily, this opens up the DAQ to being denied power. No controls mitigate this vulnerability, so the vulnerability level is Very High.

- Debugging ports are not sealed (CWE-284): If the JTAG and SWD interfaces are still available, an attacker can easily read both the FPGA and firmware binaries from the DAQ. Both debugging ports are sealed, a preventative measure with a probability of compromise approaching 0%. No detection measure is relevant, so the overall vulnerability level is Very Low.

- External control of calibration values (CWE-15): If the calibration values are accessible via the JTAG interface, an attacker can easily modify them. As the JTAG port is sealed, the calibration values are not accessible and therefore the probability of compromise approaches 0%. No detection measure is relevant, so the overall vulnerability level is Very Low.

- Insecure OTA updates (CWE-494): If remote updates are not secured, an attacker can easily obtain the firmware image or inject their own code into the DAQ. Remote updates are secured by several controls, including encryption and firmware signing, both preventative measures with a probability of compromise approaching 0%. No detection measure is relevant, so the overall vulnerability level is Very Low.

- Side-channel vulnerable cryptographic routines (CWE-208): If the cryptographic routines do not take a constant amount of time, the keys can be obtained, eliminating the protection provided by encryption. The cryptographic routines are assumed to operate in constant time, which will be verified in Section 6. No detection measure is relevant, so the overall vulnerability level is Very Low.

- Unencrypted communication channels (CWE-311): If no encryption is used to protect the communication channels, it becomes straightforward to tamper with and eavesdrop on the communication channel. As discussed, TLS is used, bringing the probability of compromise far below 25%. No detection measure is relevant, so the overall vulnerability level is Very Low.

- No sequence number or session key (CWE-294): If a sequence number is not used for commands or if the key used to encrypt the communication session is not changed periodically, replay attacks become much easier to execute. The attack can still be executed even if the attacker cannot break the encryption. Because TLS is used, a new AES key is negotiated per session. In addition, all device twin updates come with a version number (which acts as a sequence number). These controls bring the

probability of compromise far below 25%. No detection measure is relevant, so the overall vulnerability level is Very Low.

- Channel accessible by non-endpoint (CWE-300): If the channel is accessible by a party other than the DAQ or the Edge device, an attacker acting in the middle becomes straightforward. The communication channel uses TLS for encryption, bringing the probability of compromise far below 25%. No detection measure is relevant, so the overall vulnerability level is Very Low.

- Relying on the closest DNS server: If the DAQ always uses the nearest DNS server, it may be subject to a sinkhole attack if an attacker maliciously sets up a DNS server that is closer than the actual DNS server. There are no controls to prevent this vulnerability from being exploited or to detect when an attack is in progress, meaning the overall vulnerability level is Very High.

- Responding to ping messages (CWE-770): If the DAQ responds to ping messages instead of ignoring them, this will have a greater chance of overwhelming the networking stack. As a result, the DAQ will be more vulnerable to a ping flood attack. This vulnerability is mitigated as ICMP is disabled on the DAQ. No detection measure is relevant, making the overall vulnerability level Very Low.

- External inputs are not sanitized (CWE-20): If external inputs are trusted, this opens up the DAQ to buffer overflow attacks. Proper sanitization of input data must therefore take place. Input data are processed in buffers of a known size, and only the appropriate amount of bytes are copied. These measures prevents buffer overflow attacks. The same technique could be used for detection, making the overall vulnerability level Very Low.

- Improper verification of source of communication (CWE-940): If the source of an acknowledgement is not verified, the attacker can easily spoof acknowledgements. Failure to verify the source of acknowledgements causes the DAQ to be unaware that the Edge device is not receiving its messages. As communication occurs over TLS, the source of communication, and as a consequence acknowledgements, is verified. Using TLS brings the probability of compromise far below 25%. No detection measure is relevant, so the overall vulnerability level is Very Low.

Table 4.4 lists the vulnerabilities along with the vulnerability level. The controls mitigate most of the vulnerabilities. However, the lack of physical security for the DAQ and a rogue DNS server remain pressing vulnerabilities.

Vulnerability	Level
DAQ not physically secured	Very High
Power cable not physically secured	Very High
Debugging ports not sealed	Very Low
External control of calibration values	Very Low
Insecure OTA updates	Very Low
Side-channel vulnerable	Very Low
Unencrpyted communication channels	Very Low
No sequence number or session key	Very Low
Relying on the closest DNS server	Very High
Channel accessible by non-endpoint	Very Low
Responding to ping messages	Very Low
External inputs are not sanitized	Very Low
Improper verification of source of communication	Very Low

Table 4.4: Summary of vulnerability assessment.

4.5.2 Residual Risk Assessment

The residual risk assessment is straightforward. The first step is to list all asset-threat-vulnerability tuples. If a threat or asset has multiple facets that can be compromised (e.g., availability and integrity), they are listed as separate items. After, all qualitative scores are converted to numbers from 1-5, with one being Very Low and five being Very High. The three scores are then multiplied to arrive at the residual risk score. This residual risk score can be interpreted as follows: 1-5 is Very Low, 5-12 is Low, 15-32 is Medium, 36-75 is High, and 80-125 is Very High. Anything Medium or higher should be re-examined to find relevant controls to mitigate the threat. For brevity's sake, only tuples that result in a score of Medium or higher are listed in Table 4.5 below. These concerns will be addressed in the discussion.

Asset (Aspect)	Threat	Vulnerability	Residual Risk
Physical DAQ (A)	Theft	DAQ not secured	$2 \times 3 \times 5 = 30$ (Medium)
Physical DAQ (A)	Tampering with the DAQ	DAQ not secured	$2 \times 4 \times 5 = 40$ (High)
Physical DAQ (A)	Shunt resistor board replacement	DAQ not secured	$2 \times 4 \times 5 = 40$ (High)
Physical DAQ (A)	Unplugging	Power cable not secured	$2 \times 3 \times 5 = 30$ (Medium)
Compiled Firmware (A)	Sinkhole attack	Using closest DNS server	$3 \times 4 \times 5 = 60$ (High)

Table 4.5: Residual risk assessment.

Chapter 5

Realizing Secure Data Acquisition

Section 5.1 provides an overview of the specific implementation. Section 5.2 highlights core functionality: transmitting the power traces, updating the firmware, and the command interface.

5.1 Implementation Overview

The shunt resistor board, FPGA, and bootloader & firmware all physically reside in the DAQ enclosure. The edge device and ADU are both servers which help support the DAQ. The specific implementation is shown below in Figure 5.1.

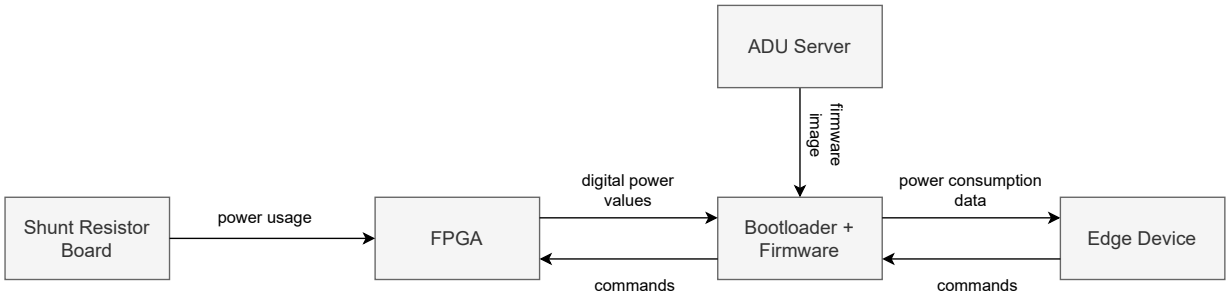

Figure 5.1: Specific implementation of the DAQ.

The shunt resistor board physically measures the power usage and is custom hardware developed by Palitronica. The FPGA relays the information to the firmware by storing the

data in shared memory and triggering an interrupt in the firmware. The firmware can issue commands telling the FPGA to start and stop capturing data, in addition to changing the frequency at which power consumption data is collected. It is implemented on a Lattice LFXP2-8E.

Both the bootloader and firmware are implemented primarily on the Atmel SAME70 Q20B model. However, due to chip shortages during the pandemic, support for the Atmel SAMV71 Q19B model was added. This model comes with less flash (512Kb vs. 1 Mb) and RAM (252 Kb vs. 384 Kb). As such, not all functionality can be supported, including transmitting CAN packets.

The bootloader's role is to check if any updates are pending, and if so, download them from the ADU server and apply them. Afterwards, it calls into the firmware. The firmware's primary role is to transmit power consumption data to the edge device. It can also receive commands from the edge device.

Many threads are running within the firmware. The following is a list in alphabetical order with a brief description of each thread and its role:

- Azure thread: This thread is responsible for processing all commands from the edge. It also transmits all the packets the various threads generate via the cloud thread. To ensure constant connectivity, it kicks the watchdog every time a packet is transmitted.
- CAN thread: This thread receives interrupts from the FPGA and packages the data into CAN packets which are then sent to the Azure thread for transmission.
- Cloud thread: The cloud thread helps send and receive all MQTT messages.
- Dynamic Host Configuration Protocol (DHCP) thread: The DHCP thread is responsible for executing all DHCP operations, including acquiring and renewing an IP address.
- Heartbeat thread: This thread drives the heartbeat Light Emitting Diode (LED) to blink every second.
- IP thread: The IP thread is responsible for helping deliver and receive IP packets once they reach the IP layer.
- Power thread: This thread receives interrupts from the FPGA and packages the data into power consumption packets which are then sent to the Azure thread for transmission.

- Simple Network Time Protocol (SNTP) client thread: This thread is the SNTP client responsible for keeping the clock up to date by receiving periodic updates from SNTP servers.
- Status pin thread: This thread toggles a test pin every second as a debugging aid when a logic analyzer is connected.
- Status thread: This thread prepares a status packet once every ten seconds to transmit via the Azure thread. It includes various debugging information and serves as a keep-alive in case power and CAN packets are both disabled.

5.2 Functionality

The DAQ has three core aspects to its functionality: transmitting power packets, updating its firmware remotely, and responding to commands issued via the device twin.

5.2.1 Transmitting a Power Packet

The process for transmitting a power packet is shown below in Figure 5.2. Before the first power packet is transmitted, the Interrupt Service Routine (ISR) must be registered with the appropriate port.

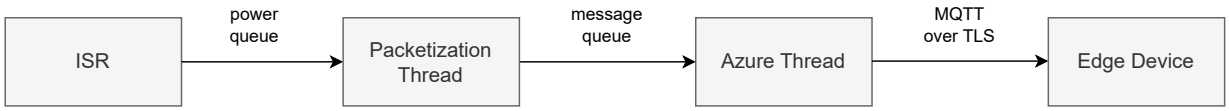

Figure 5.2: The process for transmitting a power packet.

The ISR adds a sequence number, a timestamp in nanoseconds, and the packet's length before sending a pointer to the packet on the power queue. The packetization thread reads from this queue and adds the message type to the packet. Optionally, it calibrates the power samples with a quadratic model. The model is $ax^2 + bx + c$, where a, b, and c are all predetermined constants unique to the electromagnetic sensor.

A pointer to the re-formatted and calibrated packet is then placed on a message queue. The message queue is the entry point for all threads that need to transmit a packet, including the status and CAN threads. The Azure thread reads the message type and

creates a packet with that as the message id. This labelling helps the Edge device receiving the packets differentiate the processing of the packets and adds no additional information. The packet itself is added as the payload for the MQTT message. The Azure thread then publishes the MQTT message and the listening Edge device handles the packet from there.

5.2.2 Updating the Firmware Over-the-Air

The OTA update process is shown below in Figure 5.3. First, the firmware resets the board after placing the update Uniform Resource Locator (URL) in RAM. Resetting allows the bootloader to take control. The first step is to persist the URL to flash. Persisting the URL ensures if a power outage were to occur during the update, the bootloader could make another attempt at updating the firmware.

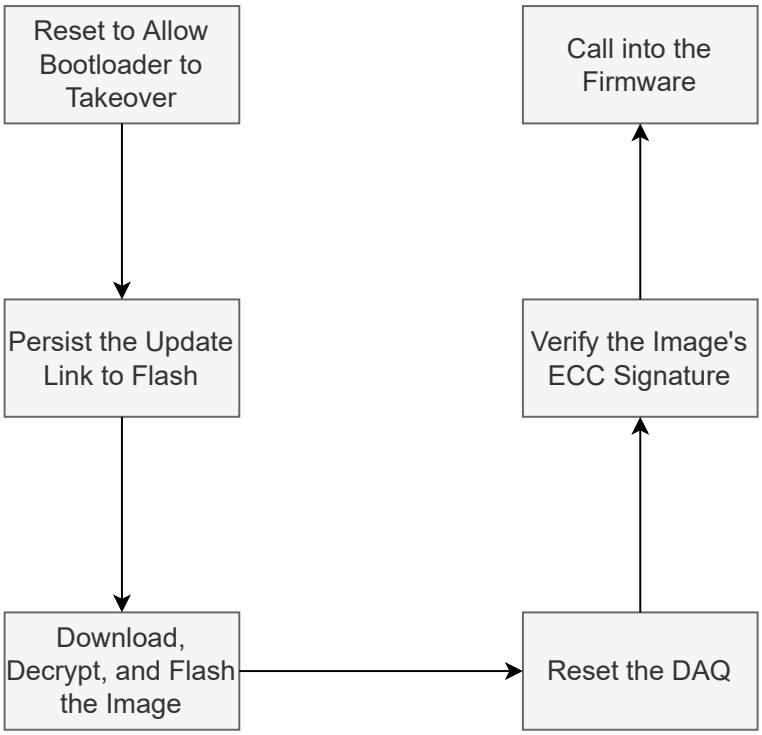

Figure 5.3: The secure update process.

The second step is to download the image and then decrypt and flash it. This step is done in parts, as the whole image cannot be held in RAM. In other words, part of the

image is downloaded, then decrypted, and finally flashed until the whole image has been flashed. As the request for the image returns an arbitrary number of bytes, but flashing can only be done a page (512 bytes) at a time an algorithm for chunking is required as shown in Algorithm 1. The DAQ is then reset. The bootloader verifies the image's ECC signature to ensure it is authorized before finally running the image.

5.2.3 DAQ Device Twin

The DAQ is controlled via a device twin. All changes to properties take place as device twin changes, where the new value for the property is modified in the requested properties section and then updated in the reported properties section once the device makes the change. As all device twin changes have a version number, there is no threat of a replay attack. The properties that can be modified are:

- statusLedPeriodms: the period of the status LED thread in milliseconds
- samplingFrequencykHz: the sampling frequency to use for power samples in kHz
- enableTransmission: whether power samples should be transmitted or not
- enableCANTransmission: whether CAN packets should be transmitted or not

In addition, a one-off change can be requested via a direct method. The direct methods are:

- Reset: performs a software reset on the DAQ.
- updateHostname: updates the hostname to use for the Edge device.
- updateAesKey: updates the AES key for decrypting firmware updates.

Algorithm 1 Chunking algorithm for flashing a firmware image

```
fw_buffer_offset ← 0
is_aes_init ← 0
while image is not exhausted do
    if is_aes_init = 0 then
        set IV as first 16 bytes of the packet
    end if
    firmware_data_size ← ending address − starting address
    if firmware_data_size ≠ 0 then
        copy_size ← firmware_data_size
        if fw_buffer_offset + copy_size ≥ 512 × X then
            copy_size ← 512 × X − fw_buffer_offset
        end if
        Copy copy_size bytes from the packet into the buffer at fw_buffer_offset
        fw_buffer_offset ← fw_buffer_offset + copy_size
        if fw_buffer_offset = 512 × X then
            Flash and decrypt the contents of the buffer
            flashed_size ← flashed_size + fw_buffer_offset
            fw_buffer_offset ← 0
            clear the buffer
            if firmware_data_size − copy_size > 0 then
                copy the remainder of the packet
                fw_buffer_offset ← firmware_data_size − copy_size
            end if
        end if
    end if
    if is_aes_init = 0 then
        is_aes_init ← 1
    end if
end while
```

Chapter 6

Experiments

In a security evaluation it is important to use internationally recognized programs when evaluating the system, not just lists of controls. The Cybersecurity Labelling Scheme (CLS) is one such program run by the Singapore government [17]. Evaluating the performance of the DAQ using key metrics such as latency and time to restart is also important in showing its suitability for capturing power consumption data. The evaluations performed are both qualitative and quantitative.

6.1 Qualitative Evaluation

The CLS aims to improve IoT security and give customers of smart devices an informed choice when purchasing [17]. The CLS consists of four stages: fulfilling security baseline requirements, adherence to international standards, lifecycle requirements and software binary analysis, and black box penetration testing. The first two are attestation-based, while the last two require laboratory testing. The qualitative evaluation will focus on assessing the DAQ's fulfillment of the first two stages since assessing the last two stages is infeasible.

There are three baseline requirements. First, avoid using universal default passwords (e.g., admin). The DAQ avoids this by giving each device a unique Azure SAS key. The second requirement is supporting OTA firmware updates. Section 5.2.2 explains how the DAQ supports OTA firmware updates. The last requirement is having a vulnerability reporting policy to facilitate vulnerability management. A vulnerability reporting policy is an organizational policy and not a control that can be fulfilled on the DAQ, making it inapplicable.

The second stage is fulfilling the European Telecommunications Standards Institute (ETSI) EN 303 645 standard for cybersecurity of IoT [21]. The standard consists of thirteen recommendations for IoT devices. Section 4.4 discusses many of these recommendations. This section will only discuss new recommendations in depth to keep it concise.

The first ETSI requirement is that devices do not use universal default passwords. This requirement is already a condition of CLS baseline requirements, and the DAQ fulfills it. The Azure SAS key is sufficiently randomized and is not generated incrementally (i.e., password1, password2, etc.).

The second requirement is to have a vulnerability management process in place. Currently, the DAQ does not have a vulnerability management process in place. Therefore the DAQ does not fulfill this requirement.

ETSI requires all devices to keep their software updated. Section 5.2.2 describes how the DAQ supports OTA updates. Provision 5.3-5 dictates that devices should check for security updates periodically. Devices checking for updates is not the model the DAQ uses. Instead, the ADU server pushes updates to the devices. Pushing updates enables greater control over the update process and supports rollbacks if necessary. Provision 5.3-6 says that users should be able to postpone updates. Again, with the pushing update model, users cannot control whether the update is applied. Provisions 5.3-7 to 5.3-10 are technical, with Section 4.4.1 discussing them in detail.

The remainder of the provisions for keeping the software updated are policies rather than technical measures. Provision 5.3-11 requires that release notes accompany all security updates explaining what risks are mitigated by the update. There is no process to disclose release notes in place. Provision 5.3-12 requires the ADU server to inform the user when device downtime will occur due to an update. Figure 5.3 shows the downtime caused by an update. Every update causes downtime, and therefore informing the user would be redundant. Provision 5.3-13 specifies that there should be a defined support period. As the DAQ is new, there is currently no specified support period. Provisions 5.3-14 and 5.3-15 deal with constrained devices, which the DAQ is not. Provision 5.3-16 requires that the DAQ have a label with a device model, which it is (see Section 4.4.1). In summary, the requirement to keep the software updated is partially met, with some of the policy requirements needing to be fulfilled by the DAQ.

ETSI also requires that devices securely store sensitive security parameters. Unfortunately, the MCU the DAQ uses as its platform is the Atmel SAME70. The MCU does not support a hardware security module for key storage, such as ARM TrustZone [3]. However, there is hard-coding of private keys in the firmware. The public key of the ECC authentication keypair is hard-coded, but this is acceptable as it is a public key. The AES

update key is unique per update, not per device. This lack of granularity is a limitation of the model for deploying updates. In conclusion, the DAQ partially fulfills this requirement due to the lack of hardware support from the MCU.

Requirement 5.5 is to communicate securely. To ensure secure communication, the DAQ uses the cipher suite TLS_RSA_WITH_AES_128_CBC_SHA_256. All cryptographic primitives are from Azure-provided libraries or using the hardware cryptographic accelerator. Except for the hardware accelerator, all cryptographic primitives are updatable. Except for DHCP, DNS, and NTP, any communication with the DAQ is authenticated. The only critical security parameter transmitted to the DAQ, the AES update key, is only transmitted after a secure, authenticated channel is in place. Therefore, the DAQ meets the requirement to communicate securely.

Requirement 5.6 is to minimize attack surfaces. The JTAG and SWD interfaces are disabled via software. All unused software services, such as ICMP, are disabled. There is no unused code present in the compiled binary. All threads run with root privileges. Azure RTOS has an extension Azure RTOS ThreadX Modules that distinguishes between root and user privileges, but it is not a core part of the RTOS [60]. An Memory Protection Unit (MPU) controls access to sections of memory. The implemented controls minimize most attack surfaces except for threads operating with root privileges.

Requirement 5.7 is to ensure software integrity. ETSI recommends using a hardware root of trust, which is unfortunately not supported in the Atmel SAME70. The DAQ verifies the integrity of the firmware before executing it but can only accomplish this at the software level. The DAQ also cannot alert the Edge device if it detects a change in the firmware since only there is only room for one copy of the firmware in the Q19 model. In other words, if the sole copy of the firmware becomes corrupted, there is no way the bootloader can send an alert to the Edge device. Given the hardware constraints, the DAQ makes its best effort to ensure the authenticity of the firmware before executing it.

Requirements 5.8 and 5.11 mandate secure storage of personal data and allowing users to delete their data quickly, respectively. Since the DAQ does not capture personal data, it fulfills both requirements.

Requirement 5.9 is to make devices resilient to outages, both network and power. If either power or network connectivity is lost, the DAQ will keep restarting until network connectivity is restored. As the DAQ's main functionality is to transmit packets, it cannot maintain service without network connectivity. The DAQ is resilient and meets this requirement.

Requirement 5.10 is to examine system telemetry data if collected. System telemetry data can often reveal security anomalies. Microsoft Defender for IoT [23] is one such

Requirement	State
No universal default passwords	Met
Communicate securely	Met
Secure personal data storage	Met
System resilient to outages	Met
Ease of deleting user data	Met
Ease of installation	Met
Keep software updated	Partially met
Secure key storage	Partially met
Minimize attack surfaces	Partially met
Ensure software integrity	Partially met
Examine telemetry data	Partially met
Validate input data	Partially met
Vulnerability management system	Not met

Table 6.1: ETSI requirement fulfillment.

system. However, when the DAQ used it, the network instability significantly increased, so it was disabled. The status packets also contain telemetry data but are not analyzed. Examining the telemetry data must be implemented on the Edge device, so it is outside the scope of this thesis. In conclusion, telemetry data is gathered but not examined.

Requirement 5.12 is to make the installation and maintenance of devices easy. The DAQ is delivered ready to connect and is, by default, securely configured. There is no human effort in setting it up once delivered other than plugging it into an Ethernet connection. The DAQ therefore fulfills this requirement.

The last requirement is to validate input data. All input data the DAQ receives is through the device twin, which only authenticated users can update. Further, the DAQ ensures data received through the device twin is validated. However, the data from the shunt resistor is not validated and is trusted as is. The DAQ cannot distinguish between the maximum value tolerated by the Analog-to-Digital Converter (ADC) and the value being genuinely out-of-range. Making this distinction and reporting out-of-range values to the MCU could be a new feature implemented on the FPGA. The DAQ partially fulfills this requirement.

Table 6.1 summarizes the requirements of the ETSI standard. Most of the requirements that are partially met are due to either hardware or software limitations.

Sampling Frequency (kHz)	Throughput (kbps)	Expected Throughput (kbps)
1	19	19.63
10	196	196.32
100	1810	1963.20

Table 6.2: Throughput for various frequencies.

Levene statistic	p-value
5.50	0.02

Table 6.3: Levene's test for 5-minute vs 24-hour captures.

6.2 Quantitative Evaluation

Throughput measures the amount of data that can pass through a network connection at any given time. Throughput is measured in kbps (thousands of bits per second). Throughput provides an understanding of network requirements when deploying it at scale. The ideal level for throughput is that the measured throughput should match the theoretical throughput.

Table 6.2 shows the throughput for various sampling frequencies. Data was captured via Wireshark and analyzed using Wireshark's conversations feature. For the low sampling frequency (1 kHz), 40 minutes of data was used. For the medium and high frequencies, 10 and 100 kHz respectively, Wireshark stopped capturing data after approximately 7 hours.

For the latency, jitter, and dropped packed rate experiments, it is necessary to determine if the behaviour of the DAQ differs statistically in a 5-minute capture compared to a 24-hour capture. Ten trials were run for the 5-minute captures at 10 kHz, and five trials were run for the 24-hour captures at 10 kHz. Levene's test determines if the two groups (i.e., 5-minute and 24-hour) have equal variances [51]. Table 6.3 shows the results of Levene's test.

As the p-value for Levene's test is below 0.05, one accepts the hypothesis that the variances of the two groups are statistically different. Therefore Welch's T-test must be applied instead [81]. Table 6.4 shows the results of Welch's t-test.

As the p-value is significantly higher than 0.05, one accepts the hypothesis that there is no statistically significant difference between the two groups. Therefore, 5-minute captures

t-statistic	degrees of freedom	p-value
0.95	62093.82	0.34

Table 6.4: Welch's t-test for 5-minute vs 24-hour captures.

are appropriate for the latency, jitter, and dropped packet rate experiments.

Latency measures the time it takes for data to go through a pipeline. In this case, the time of measuring the power sample till it is transmitted over the network will be measured. As an equation:

$$\Delta t = t_{transmit} - t_{sample} \tag{6.1}$$

Latency is measured in seconds. Latency is an important indicator of how fast a system is; the less latency, the better.

The data was captured using a logic analyzer (Saleae Logic Pro 16), running at 6.25 MS/s, with a test pin set upon receiving the power data interrupt and another test pin set when the packet was transmitted. For all frequencies, 10 trials each 5 minutes in length were conducted. Figure 6.1 shows the result of latency measurements for low (1 kHz), medium (10 kHz), and high (100 kHz) frequencies.

Jitter is a second-order measurement based on latency measured in seconds. The average jitter is defined as the mean of the difference of consecutive latency measurements. Formally:

$$AvgJitter = \frac{\sum_{i=1}^{n} |\Delta t_i - \Delta t_{i-1}|}{n} \tag{6.2}$$

Jitter is a measure of a system's variability; therefore, the ideal jitter is zero, meaning all latency values are the same. The same captures were used as in the latency experiment. Jitter is shown in Figure 6.2.

The dropped packet rate measures as a proportion how many packets failed to make it to the destination. How many packets are dropped is important because missing data could impact detectors adversely. Quantifying the drop rate indicates how reliable the power consumption data gathered is. By increasing the sampling rate within the range, one can also see how well the DAQ can transmit at higher frequencies. The ideal value is no packets are dropped, but in general, the lower the proportion of packets getting lost, the better.

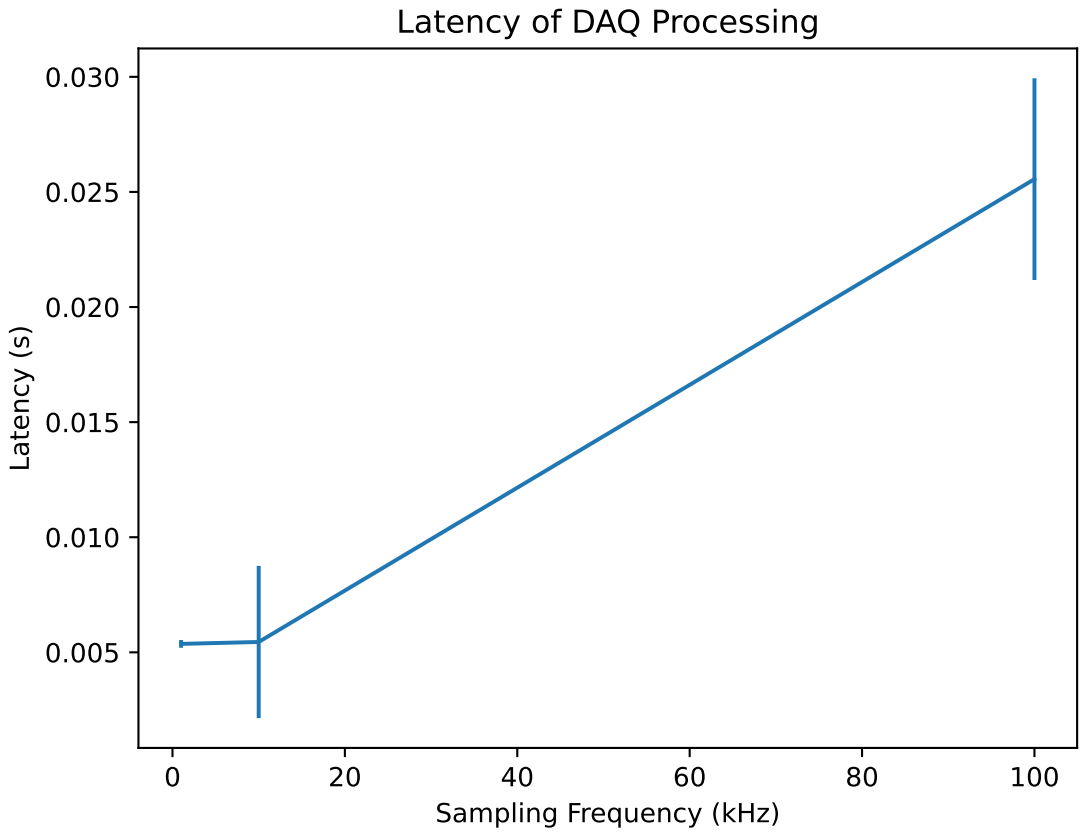

Figure 6.1: Distribution of latency for different sampling frequencies.

The experimental setup to determine the dropped packet rate is identical to that used for the latency measurements. A packet is considered dropped if a test pin was raised for the power data interrupt, but no corresponding test pin is present for sending the packet. The same captures were used as in the latency experiment. Figure 6.3 shows the dropped packet rate for different frequencies.

The amount of time the DAQ takes to restart after losing power is also a relevant metric. The less time it takes to restart, the sooner the DAQ can detect attacks again, but ideally, it's on the order of seconds. Restart time will be measured from the time power is provided again until the communication channel is re-established. This metric indicates the robustness of the DAQ to adverse effects.

This time was measured using a logic analyzer (Saleae Logic Pro 16), running at 6.25

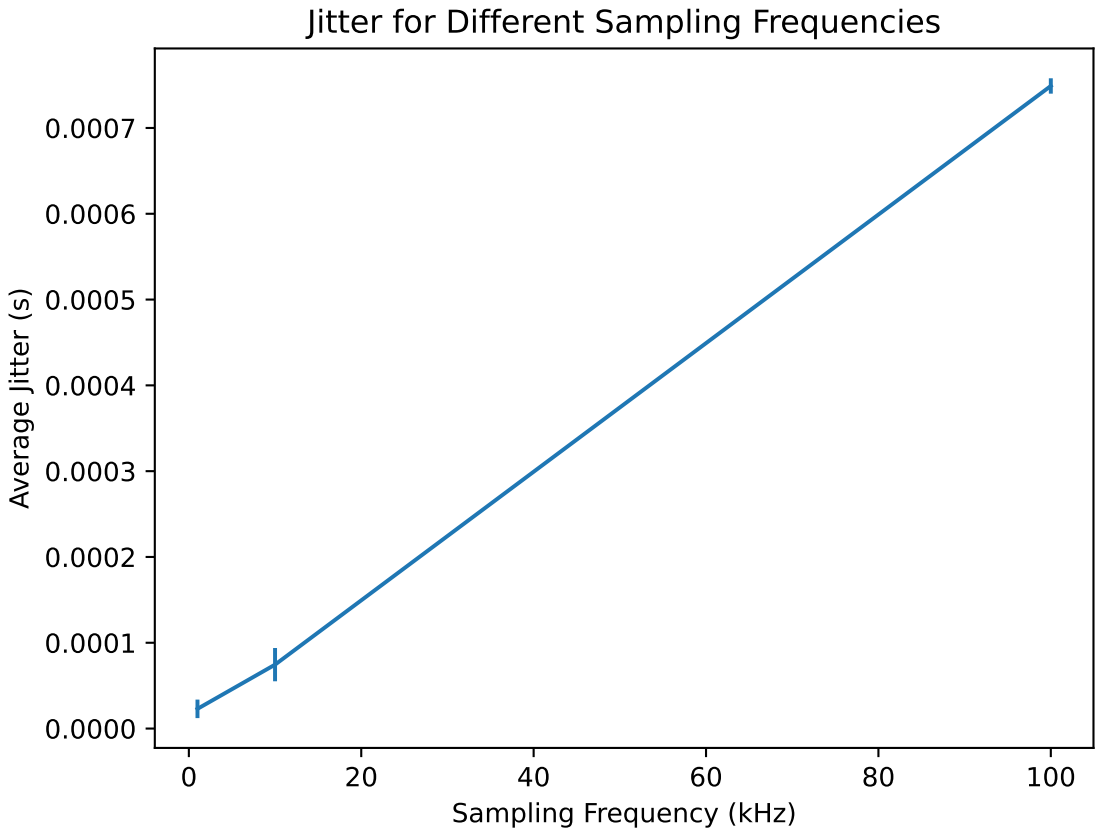

Figure 6.2: Timing jitter for different sampling frequencies.

MS/s. All test pins are briefly set high on boot. The difference between that and transmitting the first packet is the time it takes to restart the board. This process was repeated for ten trials. Figure 6.4 shows the result.

Periodically, devices will need to be updated. This process should be as unobtrusive as possible and minimize the time it takes for the device to update. By minimizing the time updates take, the DAQ can start detecting attacks sooner. The total time will be measured from the receipt of the update command till the communication channel is re-established. It is critical to minimize the time updating takes as it shows that the DAQ is responsive and will minimize missing power traces.

This time was measured using Wireshark captures. By looking at the timestamps for the last MQTT packet before updating and the first MQTT packet after updating, the time

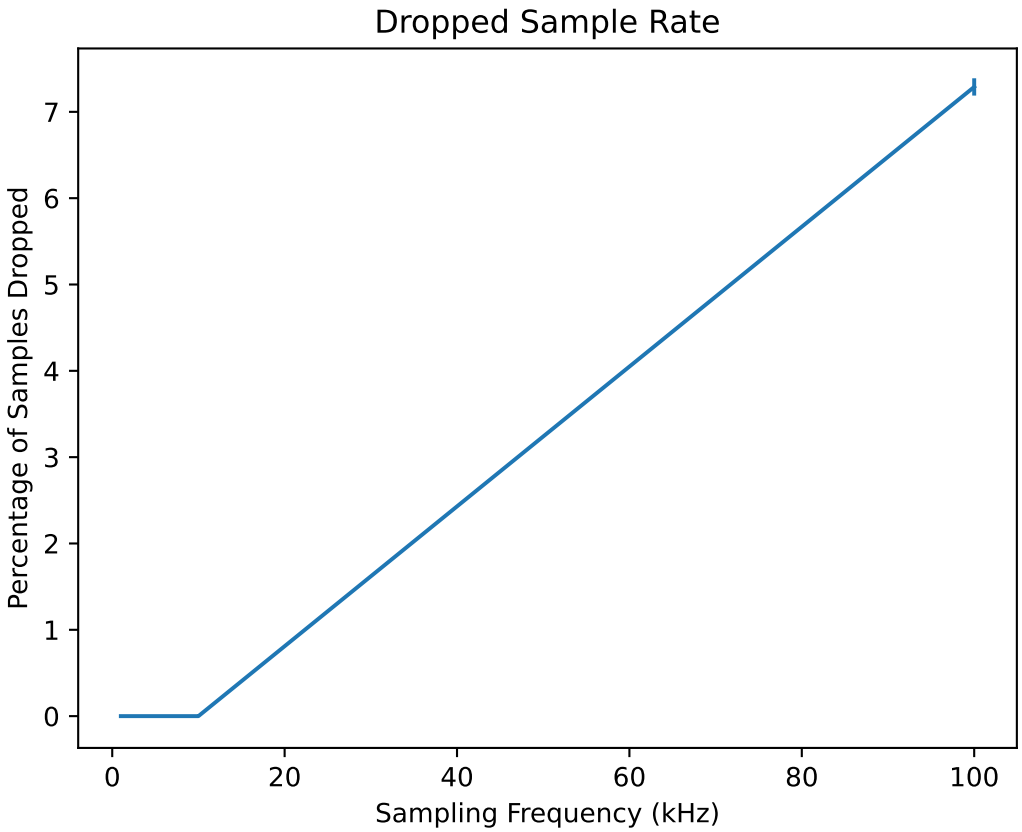

Figure 6.3: Dropped packet rate for different sampling frequencies.

to update and resume communication can be determined. This experiment was conducted for ten trials. Figure 6.5 shows the result.

The power consumption of the DAQ itself is also important to measure. Power consumption is often measured in units of W (Watts). Measuring this determines the additional power required at a deployment site. Ideally, power consumption should be as little as possible so it adds less demand for power when deployed. Actual power consumption was difficult to measure as the power supply could not drive the Power over Ethernet (PoE) injector. However, the mean well model used SPBW03G-05 has a maximum power usage of 3 W.

Memory requirements (both program and RAM memory) are important to quantify for an embedded application. Both are measured in bytes, and in an embedded context, Kb is

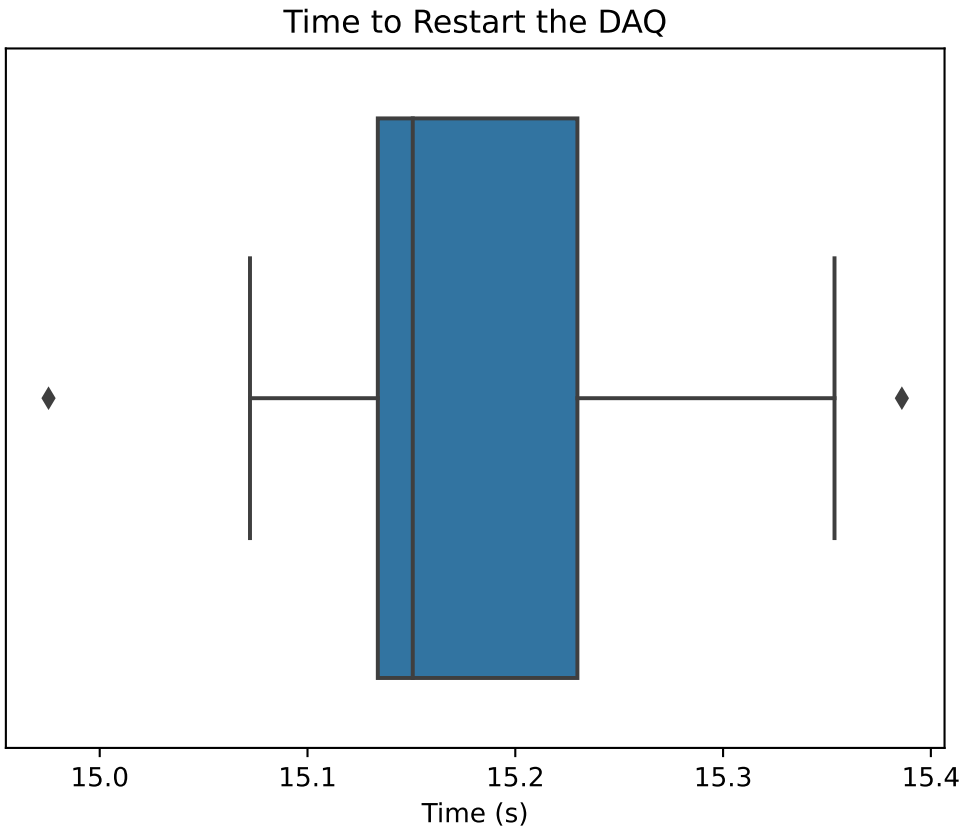

Figure 6.4: Time to restart the DAQ.

used. Knowing these values sets requirements in the minimum flash and RAM required by a new microchip. So long as these values do not exceed what is provided by a microchip, there is no ideal value. Table 6.5 shows the amount of program and data memory used, obtained from the Integrated Development Environment (IDE)'s summary statistics.

As well, each thread is given a stack in RAM. A thread's maximum amount of memory is called the high watermark. Measuring the high watermark for each thread shows how much buffer there is before the thread crashes and can allow the stack sizes to be further tuned as needed. A good rule of thumb is that the high watermark should not exceed 80% of the stack size when compiled with GNU Compiler Collection (GCC) optimization level -O1. Table 6.6 shows the usage of the individual threads. The results were obtained by enabling ThreadX run-time stack checking. High watermarks were then checked after five minutes of run-time via the debugger.

	Program Memory (Kb)	Data Memory (Kb)
Bootloader	128	221.35
Firmware (Q19)	256	205.23
Firmware (Q20)	256	214.64

Table 6.5: Usage of program and data memory.

	Max. Usage (bytes)	Stack Size (bytes)	Percent Stack Used
Azure Thread	2512	4096	61.33
CAN Thread	948	1024	92.58
Cloud Thread	2928	4096	71.48
DHCP Thread	3820	4096	93.26
Heartbeat Thread	536	688	77.91
IP Thread	1268	2048	61.91
Power Thread	828	1024	80.86
SNTP Client Thread	1468	2048	71.68
Status LED Thread	384	592	64.86
Status Thread	668	848	78.77

Table 6.6: Maximum stack usage compared to allocated stack size.

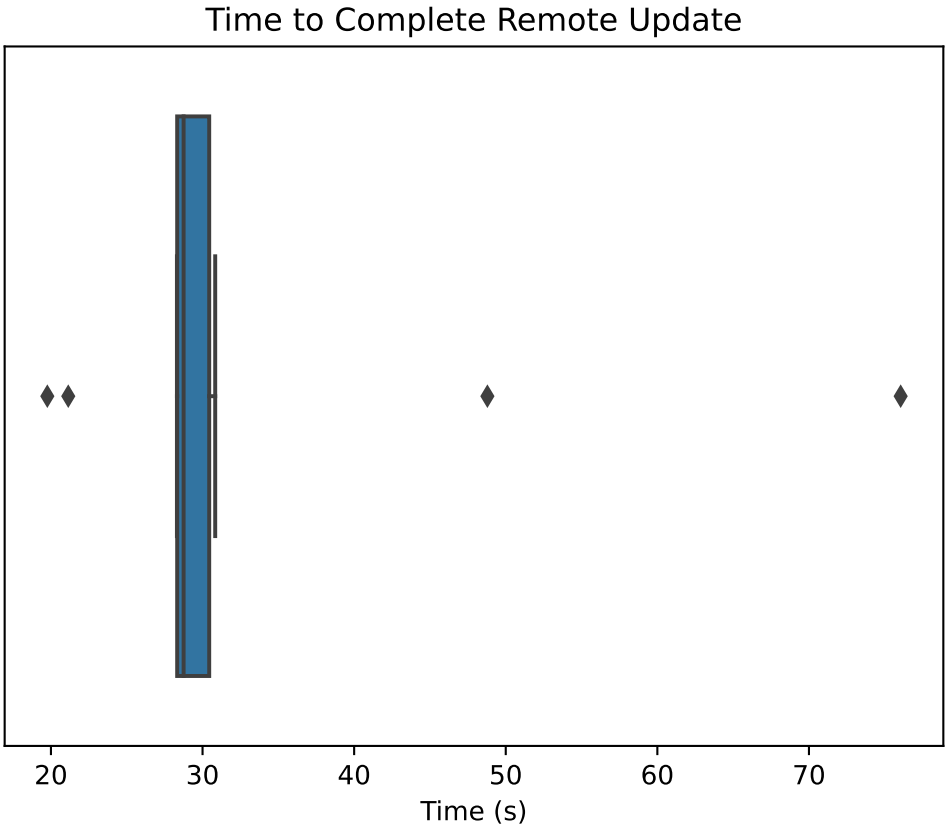

Figure 6.5: Time to complete a remote update and resume communication.

Security being an important emphasis of the thesis, ensuring that the cryptographic routines are not subject to side-channel attacks, including timing is important [44]. Two implementations of AES-128 are used in the project: a hardware-based module provided by the microchip and a software-based approach in the Azure libraries. Both will be run with different keys while timing measurements are taken. The ideal scenario is that the same amount of time is required to perform the operation. However, as long as all measurements are within a margin of error of 0.05%, that will be acceptable.

The Azure RTOS and SAME70 hardware AES-128 CBC implementations were tasked with encrypting a zero-filled plaintext using different keys and decrypting the corresponding ciphertext. The keys and the plaintext were chosen from Bassham 2002 Appendix C.1 [11]. Each key was used to complete these operations repeatedly over several seconds. Since an individual en/decryption takes microseconds, this is sufficient to calculate average times.

Key	SW Encryption	SW Decryption	HW Encryption	HW Decryption
1	167.47	212.01	29.55	29.47
2	167.47	212.01	29.55	29.47
3	167.47	212.01	29.55	29.47
4	167.47	212.01	29.55	29.47
5	167.47	212.01	29.55	29.47
6	167.47	211.92	29.55	29.47
7	167.47	212.01	29.55	29.47
8	167.47	212.01	29.55	29.47
9	167.47	212.01	29.55	29.47
10	167.47	212.01	29.55	29.47

Table 6.7: Encryption/decryption times.

A logic analyzer (Saleae Logic Pro 16) running at 500 MS/s captured the timing data. Table 6.7 shows the mean times in microseconds rounded to two decimal places to complete the operation using the chosen implementation and key.

6.3 Discussion

For the low and medium frequencies, throughput matches expected throughput reasonably closely. For the high frequency, the throughput is 7.8% lower than expected, as seen in Table 6.2. The throughput demands are per DAQ. For low and medium frequencies, this should not impose a significant demand on the local network, even for modest connections. For the high frequency, modest connections (e.g., 10 Mbps) may be overwhelmed depending on the number of DAQs deployed. Modern connections (e.g., fibre optics) should handle the traffic.

Latency centers around 5.43 ms for the low and medium frequencies, as seen in Figure 6.1. For the high frequency, this is almost 5 times at 25.49 ms. The rate at which packets are generated at the high frequency is once every 5 ms. Since it takes just over 5 ms to transmit each packet, the packets will queue up. The queue has five elements, explaining why the latency is almost 5 times greater. Each packet will have to wait for the four packets ahead of it to transmit, in addition to its own latency. For the low and medium frequencies, the rate at which packets are generated is once every 500 ms and once every 50 ms, respectively.

During the medium frequency, several packets were dropped due to network issues. The medium latency reported amongst these packets, excluded from the latency analysis, was 0.38 ms. Most of the time is spent transmitting the packet and not processing it. This fact opens up the question of how to improve the transmission time, given that it all takes place in an Azure-provided library.

The jitter and dropped packet rate both reflect the latency data. The jitter is very low for both the low and medium frequencies and spikes with the high frequency in Figure 6.2. The spike at the high frequency is explained by the wider spread between the first and third quartile in Figure 6.1. Similarly, almost no packets are dropped at the lower two frequencies, but 7.3% are dropped at the high frequency, as shown in Figure 6.3. These dropped packets are explained primarily by packets failing to make it into the queue.

The time to restart the DAQ is approximately 15.2 s, as shown in Figure 6.4. This time shows that the DAQ is responsive and consistent in connecting within 20 seconds.

The time to complete a remote update from initiation to the resumption of service is approximately 30 seconds, as shown in Figure 6.5. This time includes the restart time of 15.2 seconds, leaving just under 15 seconds to connect to the update server, receive and flash the image, and finally verify the image. The amount of time it takes is quite reasonable and shows that there is minimal downtime during an update. The outlier (76 s) in Figure 6.5 experienced several network interruptions while connecting to the update server. However, this shows the update process's resiliency, as it eventually successfully updated even if it took longer.

The DAQ uses a maximum of 3 W. Compared to an average laptop which consumes approximately 60 W [77], the power requirements are minimal and can easily be supported in a deployment environment.

The memory usage amounts output by the IDE can be found in Table 6.5. The Q20 has a program memory of 1024 Kb and a RAM size of 384 Kb, whereas the Q19 has a program memory of 512 Kb and a RAM size of 256 Kb. Note that program memory usage is additive, as both the bootloader and firmware must be in flash. However, only the maximum data memory matters, as only one program runs in RAM at a time. Therefore the program memory needed is 384 Kb, and the data memory needed is 221.35 Kb, both of which are less than the capacity present in the Q19. Further, there is some room present for expansion on the Q19, in case new features are required.

Except for three threads, all others leave enough space for increased usage during runtime. However, the CAN, power, and DHCP threads are close to a stack overflow happening, as their stack utilizations are all higher than 80%. These threads should have their stack sizes increased accordingly.

Operation	Percent Difference (%)
SW Encryption	0.0022
SW Decryption	0.0449
HW Encryption	0.0039
HW Decryption	0.0020

Table 6.8: Percent difference for different cryptographic operations.

The main point for the cryptographic timing analysis is that the same operation should take a similar amount of time given different keys. Table 6.8 shows the percent difference between the highest and lowest times per operation. As can be seen, all times per operation are within 0.05% of each other, meaning that extracting the key being used via a timing attack will be difficult.

On an interesting note, none of NIST Special Publication 800-30 [69], Operationally Critical, Threat, Asset and Vulnerability Evaluation (OCTAVE) Allegro [13], or MITRE Threat Assessment & Remediation Analysis (TARA) [83] consider the threats probability of compromising the asset as a function of time. NIST Special Publication 800-30 rates the probability of compromise on a scale from Very Low to Very High without considering time. OCTAVE Allegro only has three ratings for the probability of compromise: High, Neutral, or Low and also does not consider time. MITRE TARA does not use the probability of compromise when scoring threats.

6.4 Lessons Learned

Throughout the development of the DAQ, different problems were encountered. Three of the main issues were significant and unpredictable drift in the NTP client, maintaining compatibility with the Q19, and inconsistent writes to flash when downloading an OTA update. Each problem was solved in its own way and imparts lessons for future development.

Higher than normal drift in timestamps went unnoticed until recently. The timestamps were assumed correct but, on further inspection, proved to be unreliable. The issue was that the calculations were a combination of the ThreadX system time, with a resolution in milliseconds, and using the system ticks, with a resolution in microseconds. The time stored had a resolution of milliseconds, with system ticks being used to provide microsecond resolution on demand. The problem was solved by switching how time was kept. Instead

of keeping track of time with variables, the Gigabit Media Access Controller (GMAC) clock was used. The GMAC clock only required updates when an NTP server was polled. Otherwise, it incremented itself, eliminating the drift introduced by calculations.

The lesson learned from fixing the drift is always use hardware modules when provided. Hardware solutions are often less error-prone than custom software solutions. It is also essential to question thought patterns. The Azure NTP client sample code introduced the thought pattern of using calculations to keep track of time. However, only seconds were recorded. Trying to follow the thought pattern and achieve sub-millisecond precision is how the problem was introduced.

The DAQ firmware must also be compatible with the Q19 model. This requirement was introduced during the pandemic when the Q20 model was in short supply. The need to maintain compatibility was caused by the chip shortage and the relatively high prices for Q20 microchips at the time. As mentioned, the amount of RAM available was now effectively 256 Kb, down from 384 Kb. As a result, any time a new feature was introduced, it kept bumping up against this limit. The solution was found serendipitously, when expanding a struct used for defining a specific section of flash. Enlarging the struct caused there to be no more available RAM, even though the struct itself was never used as a variable type. By eliminating the struct and using constants to represent the addresses of the flash section instead, 32 Kb of RAM was saved for future use.

The lesson learned from saving RAM usage is to always question assumptions. It was assumed that the struct would not use any RAM, since it was not used as a variable type. However, the compiler decided to store the definition in memory anyways. Questioning this assumption led to the solution. It is also important to remain agile in the face of unforeseen circumstances. By maintaining compatibility with the Q19 model, the project was allowed to continue.

While developing the bootloader, writing the same image to flash gave inconsistent results. These intermittent failures caused the verification process to fail. More importantly, the image would have been non-functional, and the DAQ would have been bricked. The reason for this issue was discovered to be not the code written for the bootloader, but the flash driver provided by Atmel. By adding several Data Barriers around writing to flash, the problem was solved and writes to flash were consistent and predictable.

The lesson learned from the faulty flash driver is the importance of testing. Packets were inspected, and writes were immediately read back to ensure the content did not change. Hashes were compared to ensure the same image received is what was written. These tests helped narrow down the problem to the faulty flash driver. Questioning the assumption that the flash driver was correct also helped solve the problem.

Chapter 7

Conclusion

Physics-based cybersecurity is an ongoing area of research. The main advantage compared to conventional cybersecurity is that it cannot easily be spoofed. The technique proposed in this thesis is portable, as it does not require an oscilloscope which most methods mentioned in the related work use.

Using the TRA, a comprehensive security assessment was conducted. Controls were selected from several standards under the limitations of the current hardware and firmware being used. The remaining threats are theft, unplugging the DAQ, and a sophisticated attacker deploying a sinkhole attack. A sinkhole attack, however, would require the attacker to gain control over the network, in which case the availability of the DAQ becomes a low-priority concern.

Two key distinguishing features of the DAQ are the use of a device twin and secure OTA firmware updates. The device twin allows for remote configuration of the DAQ and more features to be introduced over time. Secure OTA firmware updates allow for features and bugs to be fixed, while protecting the image from being intercepted or unauthentic images from being flashed instead.

As explained in Section 6, the data transmission is relatively stable for the 10 kHz sampling rate with very few packets dropped. The DAQ has a small power footprint, only using 3 W. Also, the cryptography used is not reasonably subject to a timing analysis attack.

7.1 Future Work

There are several areas for future work, including achieving stability at higher frequencies, capturing other physical phenomena, and enhancing security further.

While capturing power data at 10 kHz is quite stable, 100 kHz is less stable, with higher jitter and a higher dropped packet rate. The lack of stability at higher frequencies is due to a few factors, including using TCP instead of User Datagram Protocol (UDP) and software-based encryption instead of hardware-based encryption. Switching to using the hardware accelerator for encryption and decryption when communicating with the Edge could reduce the time it takes to send a packet below the 5 ms threshold.

While power is currently supported as the only physical phenomenon, other physical phenomena could also be captured. Acoustic, temperature, and vibration are all physical phenomena that have been used for physics-based cybersecurity. Supporting other phenomena could take the form of having multiple sensors on board concurrently or enabling the DAQ firmware to support different phenomena and selecting which sensor to use as needed.

A key reason several controls could not be implemented is due to hardware limitations. If support for the Q19 is dropped, two copies of the firmware can be present in flash simultaneously, allowing for verifying the image before flashing it to the correct firmware location. Other controls, such as secure hardware storage for keys, are unavailable on the microcontroller. Switching to a new hardware platform would enable the use of these hardware security measures.

References

[1] Mitre capec. https://capec.mitre.org/.

[2] Mitre cwe. https://cwe.mitre.org/.

[3] Trustzone for cortex-m. https://www.arm.com/technologies/trustzone-for-cortex-m.

[4] Harmonized tra methodology (tra-1). https://www.cyber.gc.ca/en/tools-services/harmonized-tra-methodology, Oct 2007.

[5] Cyber survivability endorsement (cse) implementation guide. https://events.afcea.org/afceacyber23/Custom/Handout/Speaker0_Session10259_1.pdf, Jul 2022.

[6] Aami tir57 principles for medical device security - risk management. Standard, Association for the Advancement of Medical Instrumentation, Arlington, US, 2016.

[7] Vipindev Adat and Brij B Gupta. Security in internet of things: issues, challenges, taxonomy, and architecture. *Telecommunication Systems*, 67:423–441, 2018.

[8] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J Alex Halderman, Luca Invernizzi, Michalis Kallitsis, et al. Understanding the mirai botnet. In *26th {USENIX} security symposium ({USENIX} Security 17)*, pages 1093–1110, 2017.

[9] Algirdas Avižienis, Jean-Claude Laprie, and Brian Randell. Dependability and its threats: a taxonomy. In *Building the Information Society: IFIP 18th World Computer Congress Topical Sessions 22–27 August 2004 Toulouse, France*, pages 91–120. Springer, 2004.

[10] Daehyeon Bae and Jaecheol Ha. Implementation of disassembler on microcontroller using side-channel power consumption leakage. *Sensors*, 22(15):5900, 2022.

[11] Lawrence E. Bassham III. The advanced encryption standard algorithm validation suite (aesavs), Nov 2002.

[12] M de Bruijne, M van Eeten, C Hernández Gañán, and Wolter Pieters. Towards a new cyber threat actor typology. 2017.

[13] Richard A. Caralli, James F. Stevens, Lisa R. Young, and William R. Wilson. Introducing octave allegro: Improving the information security risk assessment process. Standard, Carnegie Mellon University, Pittsburgh, US, 2007.

[14] Saji T Chacko and Vikas Deshmukh. Iot based online power consumption monitoring of a distribution transformer feeding domestic/commercial consumer loads. In *2019 4th International Conference on Information Systems and Computer Networks (ISCON)*, pages 441–445. IEEE, 2019.

[15] Kejun Chen, Shuai Zhang, Zhikun Li, Yi Zhang, Qingxu Deng, Sandip Ray, and Yier Jin. Internet-of-things security and vulnerabilities: Taxonomy, challenges, and practice. *Journal of Hardware and Systems Security*, 2:97–110, 2018.

[16] Shane S Clark, Benjamin Ransford, Amir Rahmati, Shane Guineau, Jacob Sorber, Wenyuan Xu, and Kevin Fu. {WattsUpDoc}: Power side channels to nonintrusively discover untargeted malware on embedded medical devices. In *2013 USENIX Workshop on Health Information Technologies (HealthTech 13)*, 2013.

[17] Cybersecurity labelling scheme (cls) publication no. 2. Standard, Cybersecurity Agency of Singapore, Singapore, SG, 2022.

[18] Do-178c software considerations in airborne systems and equipment certification. Standard, Radio Technical Commission for Aeronautics, Washington D.C., US, 2012.

[19] Thomas Eisenbarth, Christof Paar, and Björn Weghenkel. Building a side channel based disassembler. In *Transactions on computational science X*, pages 78–99. Springer, 2010.

[20] Enisa: Baseline security recommendations for iot in the context of critical information infrastructures. Standard, European Union Agency for Network and Information Security, Athens, Greece, 2017.

[21] Etsi en 303 645 - cyber security for consumer internet of things: Baseline requirements. Standard, European Telecommunications Standards Institute, Sophia Antipolis, FR, 2020.

[22] Jairo Giraldo, David Urbina, Alvaro Cardenas, Junia Valente, Mustafa Faisal, Justin Ruths, Nils Ole Tippenhauer, Henrik Sandberg, and Richard Candell. A survey of physics-based attack detection in cyber-physical systems. *ACM Computing Surveys (CSUR)*, 51(4):1–36, 2018.

[23] Batami Gold, Ariel Saghiv, Shereen Bhar, and Lillian Jackson. System architecture for ot/iot monitoring - microsoft defender for iot. https://learn.microsoft.com/en-us/azure/defender-for-iot/organizations/architecture, Jun 2023.

[24] Kelly Gremban, Dennis Kennedy, Jimaco Brannian, and Diana Richards. Use the azure portal to create an iot hub. https://learn.microsoft.com/en-us/azure/iot-hub/iot-hub-create-through-portal, Mar 2023.

[25] Arthur Grisel-Davy, Amrita Milan Bhogayata, Srijan Pabbi, Apurva Narayan, and Sebastian Fischmeister. Work-in-progress: Boot sequence integrity verification with power analysis. In *2022 International Conference on Embedded Software (EMSOFT)*, pages 3–4. IEEE, 2022.

[26] Subir Halder, Mauro Conti, and Sajal K Das. Coids: A clock offset based intrusion detection system for controller area networks. In *Proceedings of the 21st International Conference on Distributed Computing and Networking*, pages 1–10, 2020.

[27] Vikas Hassija, Vinay Chamola, Vikas Saxena, Divyansh Jain, Pranav Goyal, and Biplab Sikdar. A survey on iot security: application areas, security threats, and solution architectures. *IEEE Access*, 7:82721–82743, 2019.

[28] Raphael Hiesgen, Marcin Nawrocki, Thomas C Schmidt, and Matthias Wählisch. The race to the vulnerable: Measuring the log4j shell incident. *arXiv preprint arXiv:2205.02544*, 2022.

[29] Iaea nuclear security series no. 42-g: Computer security for nuclear security. Standard, International Atomic Energy Agency, Vienna, AT, 2021.

[30] Iec 60880: Nuclear power plants - instrumentation and control systems important to safety - software aspects for computer-based systems performing category a functions. Standard, International Electrotechnical Commission, Geneva, CH, 2006.

[31] Iec 62138 nuclear power plants - instrumentation and control systems important to safety - software aspects for computer-based systems performing category b or c functions. Standard, International Electrotechnical Commission, Geneva, CH, 2018.

[32] Iec 62304: Medical device software - software lifecycle processes. Standard, International Electrotechnical Commission, Geneva, CH, 2006.

[33] Iot secure communcations module protection profile. Standard, IoT Protection Profile Working Group, Bonn, Germany, 2019.

[34] Iot secure element protection profile. Standard, IoT Protection Profile Working Group, Bonn, Germany, 2019.

[35] Iso 21434: Road vehicles - cybersecurity engineering. Standard, International Organization for Standardization, Geneva, CH, 2021.

[36] Iso 26262: Road vehicles - functional safety. Standard, International Organization for Standardization, Geneva, CH, 2011.

[37] Toshihiro Itoh, Jun Fujimoto, and Ryutaro Maeda. Application of wireless sensor nodes to commercial power consumption monitoring. In *2011 Symposium on Design, Test, Integration & Packaging of MEMS/MOEMS (DTIP)*, pages 227–230. IEEE, 2011.

[38] Woosub Jung, Yizhou Feng, Sabbir A Khan, Chunsheng Xin, Danella Zhao, and Gang Zhou. Demo abstract: A distributed power side-channel auditing system for online iot intrusion detection.

[39] Haider Adnan Khan, Nader Sehatbakhsh, Luong N Nguyen, Robert L Callan, Arie Yeredor, Milos Prvulovic, and Alenka Zajić. Idea: Intrusion detection through electromagnetic-signal analysis for critical embedded and cyber-physical systems. *IEEE Transactions on Dependable and Secure Computing*, 18(3):1150–1163, 2019.

[40] Rafiullah Khan, Kieran McLaughlin, David Laverty, and Sakir Sezer. Stride-based threat modeling for cyber-physical systems. In *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, pages 1–6. IEEE, 2017.

[41] Ansam Khraisat, Iqbal Gondal, Peter Vamplew, and Joarder Kamruzzaman. Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, 2(1):1–22, 2019.

[42] Younghun Kim, Thomas Schmid, Zainul M Charbiwala, and Mani B Srivastava. Viridiscope: design and implementation of a fine grained power monitoring system for homes. In *Proceedings of the 11th international conference on Ubiquitous computing*, pages 245–254, 2009.

[43] John C Knight. Safety critical systems: challenges and directions. In *Proceedings of the 24th international conference on software engineering*, pages 547–550, 2002.

[44] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Annual international cryptology conference*, pages 388–397. Springer, 1999.

[45] Phil Koopman. A case study of toyota unintended acceleration and software safety. *Presentation. Sept*, 2014.

[46] Danny Kopec and Suzanne Tamang. Failures in complex systems: case studies, causes, and possible remedies. *ACM SIGCSE Bulletin*, 39(2):180–184, 2007.

[47] Ritika Raj Krishna, Aanchal Priyadarshini, Amitkumar V Jha, Bhargav Appasani, Avireni Srinivasulu, and Nicu Bizon. State-of-the-art review on iot threats and attacks: Taxonomy, challenges and solutions. *Sustainability*, 13(16):9463, 2021.

[48] Ralph Langner. Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Security & Privacy*, 9(3):49–51, 2011.

[49] Jacob Lauzier. Machine condition monitoring: Improving machine health & oee. https://www.machinemetrics.com/blog/machine-condition-monitoring, Mar 2021.

[50] Robert M Lee, Michael J Assante, and Tim Conway. German steel mill cyber attack. *Industrial Control Systems*, 30(62):1–15, 2014.

[51] Howard Levene et al. Contributions to probability and statistics. *Essays in honor of Harold Hotelling*, 278:292, 1960.

[52] Nancy G Leveson and Clark S Turner. An investigation of the therac-25 accidents. *Computer*, 26(7):18–41, 1993.

[53] Yanjie Li, Ruiwen He, Xiaoyu Ji, and Wenyuan Xu. Using power side-channel to implement anomaly-based intrusion detection on smart grid terminals. In *2019 IEEE 3rd Conference on Energy Internet and Energy System Integration (EI2)*, pages 2669–2674. IEEE, 2019.

[54] Zhongqi Li, Jianwei Zhang, Maoning Wang, Yuzhong Zhong, and Fei Peng. Fiber distributed acoustic sensing using convolutional long short-term memory network: a field test on high-speed railway intrusion detection. *Optics express*, 28(3):2925–2938, 2020.

[55] Sizhuang Liang, Xirui Peng, H Jerry Qi, Saman Zonouz, and Raheem Beyah. A practical side-channel based intrusion detection system for additive manufacturing systems. In *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, pages 1075–1087. IEEE, 2021.

[56] Xuefei Liu, Chao Zhang, Pingzeng Liu, Maoling Yan, Baojia Wang, Jianyong Zhang, and Russell Higgs. Application of temperature prediction based on neural network in intrusion detection of iot. *security and communication networks*, 2018:1–10, 2018.

[57] Yannan Liu, Lingxiao Wei, Zhe Zhou, Kehuan Zhang, Wenyuan Xu, and Qiang Xu. On code execution tracking via power side-channel. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 1019–1031, 2016.

[58] Drew Mackley and Brian Overton. Four condition monitoring best practices to turn data into action. https://www.processingmagazine.com/maintenance-safety/condition-monitoring/article/21265481/four-condition-monitoring-best-practices-to-turn-data-into-action, May 2022.

[59] Brooke H McAdams and Ali A Rizvi. An overview of insulin pumps and glucose sensors for the generalist. *Journal of clinical medicine*, 5(1):5, 2016.

[60] Phil Meadows, Tiejun Zhou, and David Conger. Azure rtos threadx modules: Chapter 1 - overview. https://learn.microsoft.com/en-us/azure/rtos/threadx-modules/chapter1, Apr 2023.

[61] Phil Meadows, Tiejun Zhou, David Conger, and Sarah Vilaysom. Chapter 4 - description of azure rtos netx duo services. https://learn.microsoft.com/en-us/azure/rtos/netx-duo/chapter4, Feb 2022.

[62] Melissa Mendias, Sneha Lele, and Ashish Arora. Functional safety & safety critical systems-an overview. 2021.

[63] Charlie Miller and Chris Valasek. Remote exploitation of an unaltered passenger vehicle. *Black Hat USA*, 2015(S 91), 2015.

[64] Carlos Moreno, Sebastian Fischmeister, and M Anwar Hasan. Non-intrusive program tracing and debugging of deployed embedded systems through side-channel analysis. In *Proceedings of the 14th ACM SIGPLAN/SIGBED conference on Languages, compilers and tools for embedded systems*, pages 77–88, 2013.

[65] Mehari Msgna, Konstantinos Markantonakis, and Keith Mayes. The b-side of side channel leakage: Control flow security in embedded systems. In *International Conference on Security and Privacy in Communication Systems*, pages 288–304. Springer, 2013.

[66] Muhammad Nadeem, Byron J Williams, and Edward B Allen. High false positive detection of security vulnerabilities: a case study. In *Proceedings of the 50th Annual Southeast Regional Conference*, pages 359–360, 2012.

[67] Nas 9933: Critical security controls for effective capability in cyber defense. Standard, Aerospace Industry Association, Arlington, US, 2018.

[68] Nistir 8259a: Iot device cybersecurity capability core baseline. Standard, National Institute for Standards and Technology, Gaithersburg, US, 2020.

[69] Nist special publication 800-30: Guide for conducting risk assessments. Standard, National Institute for Standards and Technology, Gaithersburg, US, 2012.

[70] Jerome Radcliffe. Hacking medical devices for fun and insulin: Breaking the human scada system. In *Black Hat Conference presentation slides*, volume 2011, 2011.

[71] Michael Rott. *Intrusion detection for additive manufacturing: A side-channel analysis approach*. PhD thesis, Wichita State University, 2020.

[72] Nader Sehatbakhsh, Alireza Nazari, Monjur Alam, Frank Werner, Yuanda Zhu, Alenka Zajic, and Milos Prvulovic. Remote: Robust external malware detection framework by using electromagnetic signals. *IEEE Transactions on Computers*, 69(3):312–326, 2019.

[73] Philip Sperl and Konstantin Böttinger. Side-channel aware fuzzing. In *European Symposium on Research in Computer Security*, pages 259–278. Springer, 2019.

[74] Erich Styger. Preventing reverse engineering: Enabling flash security. https://mcuoneclipse.com/2014/06/22/preventing-reverse-engineering-enabling-flash-security/, Jun 2014.

[75] Minh-Quang Tran, Mahmoud Elsisi, Meng-Kun Liu, Viet Q Vu, Karar Mahmoud, Mohamed MF Darwish, Almoataz Y Abdelaziz, and Matti Lehtonen. Reliable deep learning and iot-based monitoring system for secure computer numerical control machines against cyber-attacks with experimental verification. *IEEE Access*, 10:23186–23197, 2022.

[76] Gregory Travis. How the boeing 737 max disaster looks to a software developer. *IEEE Spectrum*, 18, 2019.

[77] Kenny Trinh. How many watts does a laptop use. https://www.netbooknews.com/tips/how-many-watts-laptop-use/, Jan 2023.

[78] Gianluca Tristo, Giuliano Bissacco, Andrej Lebar, and Joško Valentinčič. Real time power consumption monitoring for energy efficiency analysis in micro edm milling. *The International Journal of Advanced Manufacturing Technology*, 78:1511–1521, 2015.

[79] Paul C. Van Oorschot. *Computer Security and the Internet: Tools and Jewels*. Springer, 2020.

[80] Tingting Wang, Kai Fang, Wei Wei, Jinyu Tian, Yuanyuan Pan, and Jianqing Li. Microcontroller unit chip temperature fingerprint informed machine learning for iiot intrusion detection. *IEEE Transactions on Industrial Informatics*, 19(2):2219–2227, 2022.

[81] Bernard L Welch. The generalization of 'student's'problem when several different population varlances are involved. *Biometrika*, 34(1-2):28–35, 1947.

[82] Lars Wüstrich, Marc-Oliver Pahl, and Stefan Liebald. Towards an extensible iot security taxonomy. In *2020 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6. IEEE, 2020.

[83] Jackson Wynn, Joseph Whitmore, Geoff Upton, Lindsay Spriggs, Dan McKinnon, Richard McInnes, Richard Graubart, and Lauren Clausen. Mitre threat assessment & remediation analysis. Standard, MITRE, Bedford, US, 2011.

[84] Yilin Zhao, Yijie Xun, and Jiajia Liu. Clockids: A real-time vehicle intrusion detection system based on clock skew. *IEEE Internet of Things Journal*, 9(17):15593–15606, 2022.