Cybersecurity for the Internet of Things: A Micro Moving Target IPv6 Defense

Kimberly A. Zeitz

Dissertation submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computer Engineering

Joseph G. Tront, Chair

Randolph C. Marchany, Co-Chair

Jaime A. Camelio

Scott F. Midkiff

David R. Raymond

Patrick R. Schaumont

May 30, 2019

Blacksburg, Virginia

Keywords: Internet of Things, IPv6 Security, Embedded Systems, Moving Target Defense, Wireless Sensor Networks

Cybersecurity for the Internet of Things: A Micro Moving Target IPv6 Defense

Kimberly A. Zeitz

(ABSTRACT)

As the use of low-power and low-resource embedded devices continues to increase dramatically with the introduction of new Internet of Things (IoT) devices, security techniques are necessary which are compatible with these devices. This research advances the knowledge in the area of cybersecurity for the IoT through the exploration of a moving target defense to limit the time attackers may conduct reconnaissance on embedded systems while considering the challenges presented from IoT devices such as resource and performance constraints. We introduce the design and optimizations for μMT6D, a Micro-Moving Target IPv6 Defense, including a description of the modes of operation and use of lightweight hash algorithms. Through simulations and experiments μMT6D is shown to be viable for use on low power and low resource embedded devices in terms of footprint, power consumption, and energy consumption increases in comparison to the given security benefits. Finally, this provides information on other future considerations and possible avenues of further experimentation and research.

Cybersecurity for the Internet of Things: A Micro Moving Target IPv6 Defense

Kimberly A. Zeitz

(GENERAL AUDIENCE ABSTRACT)

This research aims to advance knowledge in the area of cybersecurity for the Internet of Things through the exploration and validation of a moving target defense to apply for limiting the time attackers may conduct reconnaissance on low powered embedded system devices considering the challenges presented from IoT devices such as resource and performance constraints. When an attack is carried out against a network, reconnaissance is utilized to identify the target machine or device. Limiting the time for reconnaissance, therefore has a direct impact on the ability of an adversary to carry out an attack. Many of the security techniques utilized today do not fit the IoT constraints. Research in this area is just beginning and security is often not considered. Sensors collecting and sending information can be compromised both through the network and access to the physical devices. How can these devices securely send information? How can these devices withstand attacks aiming to stop their functionality or to gain information? There are many aspects which need to be investigated to understand security vulnerabilities and potential defenses. As our

technologies evolve our security defenses need to evolve as well. My research aims to further the understanding of the security of the IoT devices which have quickly become pervasive in our society. This research will expand the knowledge of the ability to safe guard connected devices from cyber-attacks and provide insight into the space and performance requirements of a technique previously only used on large scale systems. By designing, implementing experimental prototypes, and conducting simulations and experiments this research assesses the viable use of a Micro Moving Target IPv6 Defense (μMT6D).

Grant

This research was sponsored by the Office of Naval Research funded Naval Surface Warfare Center Dahlgren Division In-House Laboratory Independent Research Program.

Acknowledgments

A Virginia Tech Feature Dissertation

CAST

Author	Kimberly A. Zeitz
Committee Chair	Joseph G. Tront
Committee Co-Chair	Randolph C. Marchany
Committee Member	Jaime A. Camelio
Committee Member	Scott F. Midkiff
Committee Member	David R. Raymond
Committee Member	Patrick R. Schaumont

CREW

Location Manager/Food Stylist	Ann B. Zeitz
Caffeination/Gang Boss	Gary Zeitz
Dialog Coach	Mary L. Bane "Gamaw"
Location Scout	Jessica Z. Self
Sound Mixer	Nathan W. Self
Body Double/Style Coordinator	Rebecca A. Z. Bowie
Puppy Wrangler/Driver for Mrs. Bowie	Christopher T. Bowie
Best Boy	Henry Z. Self
Data Wrangler/ Digital Imaging Technician	L. Michael Cantrell II

SPECIAL THANKS

NSWCDD ITAP Panel . . . ONR . . . H52 . . . Server and GUI Applications Group (SAGA)

. . . VT ITSL . . . VT Graduate School . . . VT Bradley Department of ECE

Contents

List of Figures

List of Tables

List of Abbreviations

μMT6D Micro Moving Target IPv6 Defense

6LoWPAN IPv6 over Low Power Wireless Personal Area Networks

DAG Directed Acyclic Graph

DDoS Distributed Denial of Service

DTLS Datagram Transport Layer Security

ESNs Environmental Sensor Networks

IID Interface Identifier

IoT Internet of Things

IPv6 IP version 6

MAC Media Access Control

MT6D Moving Target IPv6 Defense

RPL Routing Protocol for Low-Power and Lossy Networks

SLAAC Stateless Address Auto Configuration

TLS Transport Layer Security

UDP User Datagram Protocol

WSNs Wireless Sensor Networks

Chapter 1

Introduction

It is thought that the phrase "Internet of Things" (IoT) first appeared as the title of a presentation made by Kevin Ashton in 1999 at Procter & Gamble (P&G) connecting the idea for use of Radio-Frequency Identification (RFID) technology and the internet. Ashton has described his original and standing belief that the Internet and computers are reliant on human input and are therefore held back from gathering information and being used for more physical tasks and interactions with things not just information. Object tracking, repairing, and analysis of physical objects are just a few examples of tasks possible with a computer with an awareness of physical things. RFID is one method allowing for computers to observe, identify, and understand the world and the presence and pervasiveness of RFID and other technologies in our society is quickly increasing. Years after his presentation for P&G, Ashton has written that "The Internet of Things has the potential to change the world, just as the Internet did. Maybe even more so [9]."

The impact of IoT devices can already be seen and continues to evolve. There is a paradigm shift occurring in the way society interacts with technology [3]. The Internet of Things can be described as groups of wireless devices, often sensors and actuators, communicating and

connected via the Internet. These are low power, low resource, and often unattended devices that exchange data and allow for a service. These services can vary from home automation such as motion lights and unlocking doors to manufacturing uses such as equipment controls. This is an integration of the physical world with the information world [76] made possible with the use of major technologies such as RFID, Wireless Sensor Networks (WSN), Mobile communications, and the existing Local Area/Wide Area Networks (LAN/WAN) [3].

As the possibilities and services provided by the IoT grow, so does the risk of cyber-attacks which can now have more of a physical threat thanks to the connection of everyday and new physical devices to the Internet. Protecting the information and the sensors and remote-controlled objects themselves is a challenge that needs to be addressed. Security and privacy are considered to be the top priority for IoT applications and further emphasis has been placed on the need for performance, reliability and management as well [4]. Emerging technologies are often introduced before standards and security concerns are fully assessed. Ubiquitous computing and the pervasiveness of IoT devices pose a significant threat to personal privacy [25].

The current atmosphere surrounding the release of new Internet of Things (IoT) applications is both one of excitement and great concern. The small embedded systems making up the sensors and actuators that are becoming a part of smart home, industrial, and even military applications and systems are seen everywhere and have many uses. They also are characterized as being low-power and low-resource by nature of their small size and this leads to challenges for security. The existence of the IoT has in part been made possible by IPv6.

Overall, IPv6 has many benefits including the performance and addition of a great number more addresses to be utilized, but the security of IPv6 has become a point of concern and has slowed the acceptance of IPv6 and spurred new research areas [96]. The Stateless Address Auto Configuration (SLAAC), for example, allows for a host machine to generate its own IPv6 address, but by using the host MAC address leaves the machine as a target with a static and predictable address. Researchers have compared IPv4 and IPv6 and highlighted the security concerns for IPv6 [104], but with the ever growing use of Internet of Things (IoT) devices, the transition to IPv6 needs to happen and a transition to secure practices and solutions for the IoT needs to also be transitioned to at the same time. In addition to the concerns with IPv6, the applications utilized on small embedded systems that comprise IoT devices are often deployed on constrained resource devices and security is an afterthought due to the limited space on the device and quick development and deployment of applications and products. We need to be able to predict potential threats for these applications before they emerge and have unique security practices catered to these devices [118]. Large scale security methodologies need to be adapted and new procedures produced for the IoT.

As we move forward researchers are exploring different security techniques and algorithms for the IoT. Security algorithms are being evaluated for performance on IoT devices [70], models for the use of Blockchain, such as those used by Bitcoin, with IoT and smart home devices. This is based on the characteristics of blockchaining technology as private, distributed, and secure and leveraging this to gain a lightweight and scalable system for security and privacy for the IoT [36]. Security algorithms are also being analyzed for performance in terms of

processing cycles and execution time on the Raspberry Pi [70]. These are larger than many embedded devices utilized within the IoT sphere, but often, wireless sensor networks and IoT or smart home environments have more powerful devices that they connect and communicate with. Security defenses and policy enforcement is also being looked at such as with IoT SENTINEL [81]. This is an automated device-type identification system developed for adding security to IoT environments by detecting connected devices and enforcing communications based on rules in an effort to minimize the risk of compromising the devices on the network [81].

This research aims to add to the knowledge base of security practices suitable for the IoT. The Micro Moving Target IPv6 defense (μMT6D) is a security techniques based on the algorithm of the large scale version (MT6D) [43] , but optimized and designed for use with IoT devices [113]. We present simulations comparing the use of this security technique on a WiSMote with versions including different lightweight hash functions utilized by the mote to rotate its own address. This address change algorithm has been shown to prevent an attacker from targeting a device and using reconnaissance to gather information and ultimately conducting an attack [40].

This has outlined a Micro-Moving Target IPv6 Defense, μMT6D. The goal of this research is to experiment with, analyze, and assess the viability of the use of μMT6D to protect IoT low-powered embedded devices by limiting the time an attacker may conduct reconnaissance and therefore preventing them from being able to target a device. The remaining chapters describe the motivation behind this research in Chapter 2, further background information

in Chapters 3, 4, and 5 the design and optimizations in Chapter 7, and a description of the testing and validation in Chapter 9. Background on the selection of Lightweight Hash Algorithms used to determine those initially placed in the Hash Library is given in Chapter 8 followed by the details and a discussion on the host based simulations, experiments, and analysis for each in Chapters 10, 11, 12, and 13. The border based simulations and scaling experiments are given in Chapters 14 and 15. The future direction this research could be taken is Chapter 16 and finally concluding remarks are in Chapter 17.

Chapter 2

Motivation

There are several reasons why security is a major area of concern for the IoT. Traditional methods and techniques do not directly apply and may not be sufficient for use with IoT devices [7]. Since most of the communications are wireless, this can also make them targets for eavesdropping. Since the devices can be unattended for prolonged periods of time, this can leave them vulnerable to physical attacks. Further, IoT devices limited in energy and computing power do not allow for the implementation and use of complex security schemes possible on other devices [2, 10]. Current encryption techniques often require more processing power and memory than is found on most embedded devices [89]. Methodologies for secure communications need to be adapted because they require more storage and power [15], and the distributed nature and large number of devices introduces authentication challenges [76]. Often, to perform authentication, IoT devices rely on a gateway such as a mobile device or wireless access point [117].

The data that is collected by these devices is an ever growing collection of potentially sensitive information from these sensors including private health, movement, environmental, or other types of metrics. Finally, the addition of these devices which traditionally were not

networked, or are being used for new applications, may lead to the discovery of new vulnerabilities and security risks not previously considered. Standardization efforts are underway, but new attacks and vulnerabilities increase with the release of new IoT applications and technologies [68, 71]. The large number of devices alone increases the potential of an attacker being able to find a weakness or vulnerability [2]. It is expected that 50 billion devices will be interconnected by 2020, and this number is further expected to reach a trillion [66]. In general, more devices means more attack points and safeguarding these devices is therefore vital to the security of the data and applications for which the devices exist.

We need to look at advancing the state of the art of security techniques to include methodologies and defenses suited for these low-power and low-resource IoT devices. There are many security questions which have not been answered in relation to the use of embedded devices for these new applications and research findings and metrics may not yet even exist for comparison or reference. Just a few of the open questions which exist for IoT devices include:

How can IoT devices withstand attacks aiming to ...

- halt functionality?
- consume resources?
- gain information?
- send false information?

These are very general and even more questions remain unexplored. Although the use of small embedded systems is not a new area, the rate at which they are being used and their application for use in new areas such as smart homes and everyday physical objects has led to these devices existing in many more places they were not previously. The advances in securing these devices need to progress along with the applications.

Each of these open questions deals with an attack with a different goal. Battery exhaustion attacks and denial of service (DoS) attacks attempt to stop the functionality and disrupt the service of a device. Eavesdropping is just one passive attack with the purpose of gathering information. Finally, man-in-the middle attacks, as well as, physically tampering with sensors exist for the purpose of sending false information to devices and systems. In the context of IoT, imagine a smart thermostat being "tricked" into reading that an area is much hotter through the use of a heat lamp under the sensor. The thermostat may then run the air conditioning and result in a high bill for the building owner. This may seem costly although relatively harmless, but many embedded devices are used for services which are considered to be much more high-risk. Health devices responsible for human lives, machinery equipment where malfunctions could lead to explosions or chemical leakages, and finally weapon or procedure sensors and actuators responsible for system notifications or deployments are just a few examples of devices for which security is a concern to prevent possibly devastating consequences from targeted attacks. These security vulnerabilities do not just apply to household IoT products. Research has been conducted to show how this shift toward IoT devices relates to defense strategies and systems. A "Military Internet

of Things" (MIOT) [111] has been explained as an outcome of information based modern warfare. The battlefield/zone will become an information network filled with systems and devices all in communication to aid information sharing, decision making, and weapons control. The MIOT is a system allowing for the sharing of information with regard to military people, equipment, and operational systems [111]. The IoT is quickly becoming interwoven in many different aspects and operations of our society. These open questions and concerns all demonstrate the need for more security strategies and defenses and an analysis of their use with IoT devices within different domains and applications.

Chapter 3

Background

3.1 IoT Related Work

As mentioned in Chapter 2, security techniques must be adapted or developed to secure the communications and sensitive data of new IoT devices and applications [15, 66, 89]. Research and experiments for the IoT have varied from Secure Multi-Hop Routing Protocols [29] to testing large scale simulator applications [24]. Applications vary widely from healthcare [115], green architecture, environmental monitoring, to smart transportation [78] and security remains a major topic of concern. Since low-powered and low-resource devices cannot make use of some traditional authentication methods or intrusion detection systems, new models and techniques are needed [76, 117]. Many authentication methods, encryption forms, hardware approaches, and protocols have been explored for use [19, 23, 50, 109]. Much research has focused on the Transport Security Layer [46, 92, 105]. However, implementing encryption and authentication methodologies do not obscure the addresses of the devices, leaving an opening for adversaries to find and target these devices.

There are many security concerns related to the use of new IoT devices and applications.

As mentioned, being left unattended, possibly in a hostile environment, can lead to concerns such as harsh conditions including temperature, radiation, vibration and motion, etc. In addition, internal failures, interference from other devices, and user misuse could all lead to security vulnerabilities [28]. There are also concerns stemming from the transition to IPv6. Denial-of-Service and many other attacks utilized by attackers with IPv4 are still threats in IPv6. Detection methods and new security tools are being researched that apply to both IPv6 and these new IoT devices [67]. In terms of device addressing, the introduction of IPv6 has both positive and negative ramifications. Unfortunately, the StateLess Address AutoConfiguration, SLAAC, allowing devices to create their host portion of the IPv6 address, (the interface identifier (IID) often comprised of the MAC address), allows for malicious users to identify device location and monitor and/or track the communications of those devices [41]. Tracking and monitoring an address allows an adversary plenty of time to gather information in this reconnaissance stage, and ultimately plan out an attack strategy. This is the scenario moving target defenses aim to prevent.

3.2 Moving Target Defense Related Work

Attackers utilize reconnaissance to identify a target machine or device for conducting an attack. A defense to this involves limiting the time and possibility of selecting a target to directly impact the ability of an adversary to carry out an attack. This is the motivation and underlying concept behind a moving target defense security strategy.

The concept of a moving target defense has been a topic of research for many years and involves many different focuses and areas. Okhravi et al. did a survey in 2013 of over 100 references filled with concepts, designs, analysis, and applications [87], Green et al. has publications focusing on MTD performance [51] and network characterizations [52]. The effectiveness and evaluation of MTD systems is also an ongoing topic [112, 119, 122] including work on theory and framework [120]. MTD can be categorized as both passive and active defense systems, some implementing the use of attack indicators to aid decisions and system responses [121].

One moving target defense approach has been explored for improving the security of MANETs by hiding the actual topology of the network. In this case, legitimate nodes on the network change their identity referred to as a "virtual identity" and securely relay this change. The nodes themselves have one real identity and several virtual identities. It is suggested that the use of hash chains would allow for the generation of a pool of identities that would be difficult for an attacker to predict. A translation service maps the virtual identities to the real identities. One potential drawback is the use of a pool of virtual identities which may be repeated or switched between nodes. This has the potential to be vulnerable to traffic pattern analysis [6]. Another approach has been adapted for protection against worms that have a hitlist of precomputed target IP addresses. This Network Address Space Randomization (NASR) in a basic form would configure the DHCP server to expire at intervals thought suitable for randomization. A key challenge with this is avoiding address changes for hosts and disrupting services and having to examine network traffic with service fingerprinting for

address changing decisions makes this extremely difficult to scale [8].

Large scale testing of a network of over 300 nodes has been utilized to evaluate network and application randomization at Sandia National Laboratories. The goal presented is enhancing control system security through the conversion of existing systems into moving targets [27]. A scalable MTD has also been explored using software-defined networking (SDN) [75]. Research has also been conducted in the area with Mobile IPv6 aimed to also obscure the subnet addresses of packets [56], OpenFlow Random Host Mutation (OF-RHM), which utilizes virtual IP addresses in both IPv4 and IPv6 [60], and an architecture called Mutable Networks (or MUTE), which allows networks to change configurations based on information from a central device [5]. A central device in is also utilized in [110] as an "Adaption Engine" to control a network MTD.

Other MTD systems are based on the use of decoys in order to obscure real nodes from attackers in a smaller network setting. The work focuses on the issue of optimizing when to randomize IP addresses in order to minimize network disruption and minimizing the detection probability of the real nodes [31] and the game-theoretic aspect of the interaction between the external adversary and the network decoy nodes [30]. Decoy-Enhanced Seamless IP Randomization (DESIR) combines IP randomization and decoy techniques to be able to provide service availability and also security by changing the IP addresses of both real servers and decoy nodes [100].

MOTAG utilizes dynamic packet indirection proxies to relay data traffic between clients and protected servers and moves secret proxies to new network locations periodically [63]. Cloud-

enabled defenses also exist such as a cloud-based MTD for client-to-server re-assignment involving protected server "shuffling [64]." Dynamic Network Address Translation (Dynat) is a type of MTD which changes network addresses and protocols and was developed with different implementation varieties or "Dynat types" to apply to different use cases [80].

Researchers at the Virginia Polytechnic Institute and State University have developed MT6D, a Moving Target IPv6 Defense, which has been shown successful in thwarting targeted attacks, host tracking, and eavesdropping by obscuring the network and transport layer addresses and avoiding static defenses which allow adversaries unlimited time to conduct attacks [44]. MT6D provides privacy and prevents targeted network attacks by obscuring the communication of two devices through address rotation [42, 43]. Optimizations were first done to this moving target defense by translating the implementation code from Python to C [55]. Further work has involved simulation and analysis of an IPv6 network and MT6D [32], evaluating effectiveness against Distributed Denial of Service (DDoS) Attacks [35], and assessing and aiding the rigorousness of MT6D with previous address analyses, intrusion monitoring, and the concept of honeypots [17]. Research has continued and MT6D has been furthered to include a network layer client-server implementation utilizing a Distributed Hash Table (DHT) Blind Rendezvous for session establishment [82, 83, 84].

This can be applied to the IoT. A method has been developed for the dynamic address change on low-powered devices [90], [97]. This research takes previous research in this area further and presents a design and optimizations for fully implementing μMT6D and describes the base simulation setup and future hardware testing for further experimentation,

assessment, and validation. By considering the challenges presented by IoT devices such as resource and performance constraints, this design, optimizations, and validation testing and experimentation will allow for the assessment of the use of a Micro-Moving Target IPv6 Defense, μMT6D. As an overall goal to lead the direction for this work, we present the question:

Is a moving target defense methodology a viable defense for limiting reconnaissance time and thwarting attacks for IoT devices?

The next sections describe the concept overview and the design and optimizations for μMT6D, including potential lightweight hash algorithms, the possible testing and validation configurations for experimentation, the simulation base and hardware testbed needed along with a discussion of metrics and validation. This is followed by a series of experiments and results. Finally, we then outline a few other future works and directions for this research.

Chapter 4

A Timeline of MT6D

An overview of the background of MT6D was highlighted in Chapter 3. This section further breaks down the goals and efforts of some of the related PhD and Master's works that came previous to this research.

4.1 The MT6D Beginning: 2009-2012

The dissertation "Achieving Security and Privacy in the Internet Protocol Version 6 Through the Use of Dynamically Obscured Addresses" [40] by Matt Dunlop, Ph.D. was published in 2012 and was the first work on A Moving Target IPv6 Defense (MT6D). A breakdown of the topics covered includes:

1. An overview of the vulnerabilities in the formation of IPv6 Addresses and demonstrated on a production IPv6 network

2. The information attackers could gain by exploit these vulnerabilities

3. A method and proof of concept for dynamically rotating addresses (MT6D) and how

it prevents these vulnerabilities

The origination of a subnet management solution called Stateless Address Auto Configuration (SLAAC), was meant to solve the problem of subnet management. IPv6 uses 64 bit addresses compared to 32 bit addresses of IPv4. This gives a much larger address space of 2^{128} or 340,282,366,920,938,463,463,374,607,431,768,211,456 IPv6 addresses compared to 2^{32} or 4,294,967,296 IPv4 addresses. The address space must be managed and manually doing so would be both time consuming and complex. The use of Dynamic Host Configuration Protocol for IPv6 (DHCPv6) would place a huge burden on network managers because as hosts connect to the network addresses would be issued by a DHCP server. In contrast, SLAAC allows for the configuration of the network configuration by the administrator and an auto configuration of the host portion, or Interface Identifier (IID). The auto configuration of the IID is often done according to the Extended Unique Identifier (EUI)-64 standard by taking the 48-bit Media Access Control (MAC) address and extending it to a 64-bit number with the insertion of the 16-bit hex value 0xFFFE between the 24-bit Organizationally Unique Identifier (OUI) and the Network Interface Controller (NIC). This can be seen in RFC 4291 [57]. The MAC address is first separated into two 24-bits, with one being OUI and the other being NIC specific. The 16-bit 0xFFFE is then inserted between these two 24-bits for the 64-bit EUI address. Finally, the universal/local (U/L) bit which is the seventh bit from the left is inverted.

Attackers can utilize this information to track users in multiple ways. The host portion of the SLAAC address remains static allowing for attackers to track users regardless of the network

they are connected to. It also advertises the MAC address which can allow attackers to glean information about the device itself. Dunlop demonstrated both geotemporal tracking and traffic analysis with the use of the IID. In one case, six Virginia Tech subnets were pinged for the IID and a the movement of a cell phone was tracked. In another case, a sensor to sniff IPv6 traffic was placed at the network border to record all network travelling over the network. Google searches, YouTube queries, and Gmail data, and Twitter tweets, often unencrypted due to the bandwidth and processing overhead of TLS, were all be gathered and connected to a user by IID. Futher, computer and operating system types were identified by the OUI also present in these addresses. Finally, it was explained how an attacker could use Spoofing an IID to falsely incriminate someone. Some user tracking could be seen as positive, many applications aim to utilize "converged security" or the combination of the physical and technological to come up with security solutions. Two examples given included locating victims in emergency response situations and noting when unauthorized users enter physical spaces. These can be seen as "good" tracking, but still infringe upon user privacy. In summary, the IPv6 address formation and the fact that is remains static pose a huge privacy risk to users.

Dynamic address obscuration techniques protect hosts from targeted attacks as well as privacy-related attacks [40]. MT6D provides many advantages over other similar techniques. MT6D allows for the address rotation interval to be set to small increments such as five seconds. Other techniques such as privacy extensions may allow addresses to remain static for a day or up to a week. It is more difficult for an attacker to locate and target a device

the more frequently it changes. MT6D provides anonymity. Tracking and correlation are prevented with MT6D because the packet payload is encrypted along with the header information obscuring both the source and destination network and transport layer addresses. The MT6D address calculation includes a timestamp (a changing nonce) instead of a rotating session key allowing for hosts to communicate without repeated authentication, which saves overhead and latency. Finally, some solutions such as IPsec in tunnel mode, do not provide any address protection from an attacker inside the sender's or receiver's subnets, since obscuration occurs at gateways. MT6D provides this protection.

Dunlop explains the three components of the obscured IIDs used for the dynamic addressing of MT6D:

1. A value specific to an individual host such as a MAC address

2. A secret shared by the sender and receiver such as a symmetric key

3. A changing value known by both parties such as time

The algorithm will be described in Chapter 7. Dunlop did experiments focusing on transfer speed and packet loss of MT6D on full scale computers and with his results showed that this is a viable security technique. These core insights led to further research to establish the concepts of and enhance MT6D.

4.2 A Micro MT6D: 2013-2015

As mentioned in the Introduction, securing the ever growing IoT devices being utilized in many different applications is a major concern. In his Master's thesis "Micro-Moving Target IPv6 Defense for 6LoWPAN and the Internet of Things", Matt Sherburne made strides that opened the door for research in designing a Micro Moving Target IPv6 Defense suitable for IoT devices [98]. He demonstrated that there is a way to support address hopping on resource constrained embedded devices. His paper along with one he co-authored with Tanner Preiss, [90, 97], introduced a technique for forcing the embedded operating system, Contiki OS, to change the generated IPv6 address on the embedded host device. He looked at the route addition success rate for RPL DAG with address rotation intervals from one to ten. The ending proof of concept required the hosts to send the address change messages immediately and to disable the delay timer meant to decrease contention. His work looked at current consumption and footprint of his proof of concept implementation.

4.3 An MT6D Server: 2013-2016

The next phase of research to highlight is the application of MT6D to a client-server network. This section outlines the work by Chris Morrell for his Ph.D. dissertation "Improving the Security, Privacy, and Anonymity of a Client-Server Network through the Application of a Moving Target Defense"[85]. Beyond the network based MT6D model, Morrell explored the need for a client server network which could support the application of IoT devices that need

to send data to a larger machine for aggregation, such as Sherburne introduced [98]. His works utilized a Distributed Hash Table (DHT) for a blind rendezvous to allow for the secure exchange of the MT6D configuration data. Dynamically generated configurations reduces issues from compromised passwords and replay attacks. It also allows for the ability to alter the defense configuration with fluctuations in the needed security level. This was in part inspired by the thesis work of Dileep Basam, "Strengthening MT6D Defenses with Darknet and Honeypot capabilities" [18]. His work explored the use of a honeypot and monitor network for machines utilizing MT6D. The results showed that a network administrator could monitor the IPv6 addresses which had previously been assigned with MT6D in order to look into the unpredictability of MT6D. Large usage of addresses by unexpected hosts or unexpected address occurrences could mean that the keying material could have been compromised and a new exchange of information is needed.

The scheme presented outlined major benefits:

1. The blind rendezvous scheme allows for secure private connection information sharing
2. It is distributed and dynamic
3. Cryptographic algorithms ensure the security and authenticity
4. The distributed hash table avoids a single point of failure
5. Forcing host connection information to expire allows for only current connection information to be available, allowing host roaming on the network

6. This client server method is scalable

The Client/Server MT6D was analyzed in regards to data transfer time overhead. This was shown to be negligible compared to the security provided, and this was also shown to be scalable with the percentage of data transfer time overhead diminishing from 75% with one client to 7.2% with ten concurrent clients. Other metrics explored included address generation and bind times with the increase in number of clients. The upper bounds of his implementation was at about 55000 actively bound IPv6 addresses. Past this boundary the binding time is longer than the three second rotation interval set for the experiments. This was paired with an experiment on data rates for packet transmission. The maximum supportable data rate was found to be 150000 packets per second when 10000 or less addresses are bound to the server. Combining the two insights, for this proof of concept illustration, it was found that there should be about 10000 to 15000 concurrent clients per server.

4.4 This Research An Enhanced Micro MT6D: 2014-2019

The need for security solutions for small embedded systems is still an area in high demand for research and exploration, as was explained in Chapter 2. Updates to the standards for IPv6 have been ongoing. In 2014, RFC 7217 updated the SLAAC standard to create an RID

or Random (But Stable) Identifier. This is done with the equation:

$$RID = F(Prefix, Net\_Iface, Network\_ID, DAD\_Counter, secret\_key) \quad (4.1)$$

In this equation they define F as A pseudorandom function (PRF) that must not be computable from the outside without knowledge of the secret key [48]. This alteration does not change the fact that a static IPv6 address can be vulnerable to targeted attacks.

This research picks up and build upon the already established research and methodologies for MT6D and Micro MT6D mentioned in this timeline. The contributions include:

1. A design including a lightweight hash library and experiments with four lightweight hash functions
2. An outline of host based and border based modes of operation
3. A customizable proof of concept μMT6D
4. Measurements on footprint and power and energy consumption to show viability for small embedded devices
5. Simulations and experiments looking at different address rotation intervals and applications with varying workloads
6. Host Based μMT6D metrics and analysis from over 50 simulations across four experiments

7. Initial metrics looking at scalability

8. Initial metrics for border router overhead savings

These contributions are elaborated further in the next chapters.

Chapter 5

Threat Model & Scope

5.0.1 Threat Model

For this research, we assume a remote attacker not within wireless range of the WSN or IoT devices. The attacks possible by these remote attackers can be classified into categories. Targeted Attacks are aimed at certain hosts or "nodes" and can be classified as either passive or active attacks. As mentioned in Chapter 2, a DoS attack is an example of an active attack which disrupts communications. A Man in the Middle (MITM) attack is another active attack in which communications are intercepted and can be altered. MT6D protects hosts from being targeted by attackers because if the address is targeted the attacker only has the period of time between address rotations to carry out such an attack.

Unspecified Attacks, such as described in [40], where the attacker does not target a host, but instead may block any discovered host from using network resources would not be protected from happening by MT6D or μMT6D, but could be limited. Dunlop explained how MT6D, and therefore μMT6D based on the same principles can prevent an attacker from knowing the density of the network and a network with a higher density decreases the probability of

a single node being effected [40].

Prior work outlined in Chapter 4 has shown the security benefits of MT6D and μMT6D. MT6D and μMT6D protect against targeting, tracking, and traffic correlation [40]. The Dynamic address obscuration technique protect hosts from targeted attacks as well as privacy-related attacks while providing anonymity [40]. It also can limit the amount of time attackers have to conduct reconnaissance and limits the possible damage inflicted on a targeted host. Large IPv6 subnets would take a while for an attacker to scan and target a host. Once targeted, the damage they can inflict is limited to the time between address rotations [40]. MT6D does not obscure the subnet from which a host communicates, but an attacker gains little in reconnaissance by knowing the subnet[40].

5.0.2 Scope

1. Other lightweight hash functions may be included in the hash library of this design. Simulations and experiments are run with prototype implementations including provided implementations of lightweight hash functions. These are a selection of possible hash functions for the hash library and other implementations or hash functions can be used depending on the needs of the application and preferences. Some implementations and hash functions may exist or may be developed in the future that could enhance the metrics looked at in this research.

2. Other research has supported the use of MT6D to prevent against targeted attacks.

This research has a goal to further previous research and show the viability for use on small embedded devices. The design, including the use of a hash library and the focus on host or border based implementations can also apply to systems of other scales, but metrics are collected for small embedded systems to show viability on the chosen devices.

3. The chosen metrics of footprint, power, and energy consumption were selected as the basis to show viability and low or acceptable overhead of μMT6D. There are many other metrics that can be explored further in simulations and experiments. Others are left to future work.

4. The work by Morrell in [85] established a technique and design for the inclusion of a key exchange between a client and server. This work relies on a pre-shared scenario where the key was agreed upon and provided to the host wishing to communicate beforehand and "out-of-band." Future work should include the integration of this research and that of the client server MT6D.

5. MT6D originally included the use of encryption as optional, because address tracking is not feasible and the protection for targeted attacks is still in place. This work did not focus on adding encryption, however, future work could look at possible encryption mechanisms to add to the security provided by μMT6D where message data may be sensitive. Other works look at the use of encryption for small embedded systems [77] and Sherburne outlined work on a limited version of Transport Layer Security

(TLS) and Datagram Transport Layer Security (DTLS) with methods for encrypting Transport Layer TCP and UDP datagrams, and a 6LoWPAN version of IPsec [98]. As he mentioned, neither of these solve the problem of a mote having a static IPv6 address, but the inclusion of these could bring more security benefits in the realm of small embedded devices, an area in which research is focusing as IoT devices and different applications and security needs increase.

Chapter 6

Design Criteria

The design criteria goal considered for μMT6D include size viability, low or acceptable power and energy overhead, platform agnosticism, customization, and novelty.

1. **Viable size** - For viability the concern is whether or not a Micro Moving Target IPv6 Defense can be designed, implemented, and run so that it meets the size requirements to fit on a representative small embedded device. For this research we chose the WiSMote.

2. **Low or acceptable power and energy overhead** - Given the benefits of μMT6D the power and energy overhead should be characterized as either:

   (a) **Low** - The overhead is negligible.

   (b) **Acceptable** - The overhead may be greater than what is considered negligible, but the benefit outweighs the added overhead.

3. **Platform agnostic** - μMT6D should be able to be applied to various systems.

4. **Customizable** - μMT6D should be able to be applied to different use cases and scenarios.

5. **Novel** - μMT6D should be novel but supported by known research. The novelty of this research is in the design for small embedded systems, but it utilizes an algorithm and concept backed by multiple research efforts showing the success of this security technique for full scale systems [25, 32, 32, 40, 41, 42, 43, 44, 53, 83, 84, 90, 97, 98]. These have been discussed in Chapter 3 and Chapter 4.

These design criteria can be matched to goals either explained and/or included in experimentation by the work:

1. **Viable size** - This is shown with footprint in kB given for the proof of concept implementations run for the experiments.

2. **Low or acceptable power and energy overhead** - This is shown with power and/or energy metrics given in milliwatts (mW) and Joules (J) respectively from the experiments.

3. **Platform agnostic** - The design outlined in Chapter 7 is platform independent and can be implemented and built for different hardware.

4. **Customizable** - The design outlined in Chapter 7 allows for selection of address rotation intervals and of lightweight hash function, making this a customizable security technique.

5. **Novel** - The novelty of the security technique is shown in the motivation in Chapter 2 and in the design outlined in Chapter 7. Chapter 3 and 4 show the vital and supporting

research leading up to this work.

6. **Secure** - As this research contributes towards the design and evaluation of a security technique, security is a design criteria. The security of μMT6D relates to the security of MT6D as it is based on the algorithm and the client/server version it may be connect to. It also relates to the security of the lightweight hash functions used.

Chapter 7

Design

A Micro-Moving Target IPv6 Defense, or μMT6D, was designed to operate within IPv6 over Low-power Wireless Personal Area Networks, or 6LoWPAN. As discussed, [97] and [90] showed the viability of frequently rotating the IPv6 address of a single 6LoWPAN device. In order to implement and assess μMT6D, we have designed the modes of operation and optimizations which will be needed.

7.1 Concept Overview

The overall concept of operations for μMT6D is that by changing the address of the IoT device for which this security mechanism is deployed, this will limit the amount of time an attacker has to conduct reconnaissance and therefore also limit the viability of such an attacker carrying out an attack. As mentioned in Chapter 3, and following the same principles of MT6D, μMT6D aims to obscure the communication of resource constrained devices through the use of address rotation to prevent targeted attacks. Figure 7.1 shows an overview of this concept. At the top is a depiction of three nodes, or resource constrained

IoT devices, communicating and utilizing μMT6D. This represents an example of an actual configuration. Below is an attacker's view of the configuration in which seemingly many more devices are communicating and also look to appear and disappear on random addresses. The attacker does not have an idea of which nodes are actually communicating and believes there are many more nodes due to the many IPv6 addresses observed. After having an understanding of the basic concept, it is important to detail the design for this concept. We present the design of two different modes of operation for μMT6D, Host-based and Border-based [113].

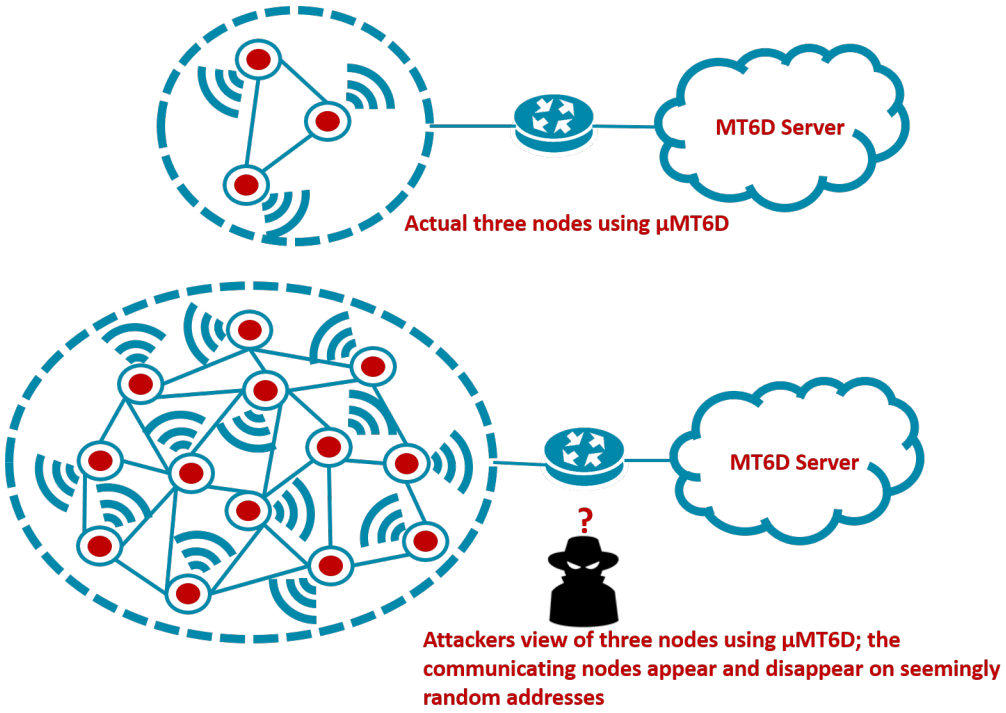

Figure 7.1: Concept Overview of μMT6D

7.1.1 Host-Based

For a host-based design of μMT6D, the implementation of the security technique would be present and carried out by the host devices themselves. These are our resource constrained IoT devices. In this mode of operation, the implementation would be optimized for use on low-power and low-memory devices so as to not interfere with the task for which the devices are intended. A flow diagram for the host based design can be seen in Figure 7.2.

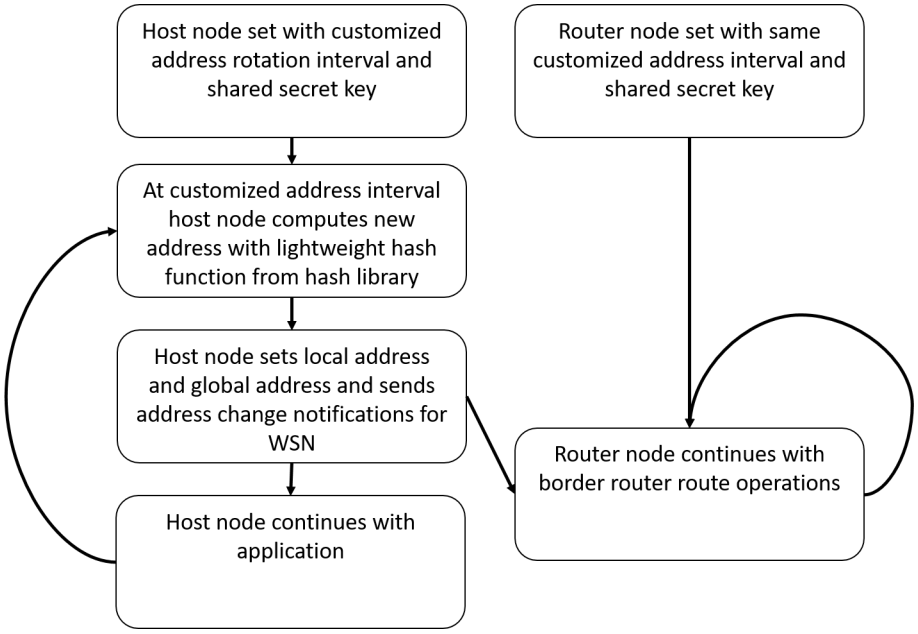

Figure 7.2: Host Based Design Flow Diagram of μMT6D

7.1.2 Border-Based

As mentioned in the introduction, the vast majority of IoT devices are low-powered and low-resource. Due to these limitations, these devices often have another device, or hub, which is more capable and less resource constrained with which to communicate [117]. The

individual nodes may be engineered to only have the resources needed to perform their given role in the IoT. In this scenario, a border device, such as a router or other gateway device, could be used for the address rotation. One security implication of this which would need to be explored and noted is the fact that there would be a single point of failure for this moving target defense. A flow diagram for the host based design can be seen in Figure 7.3.

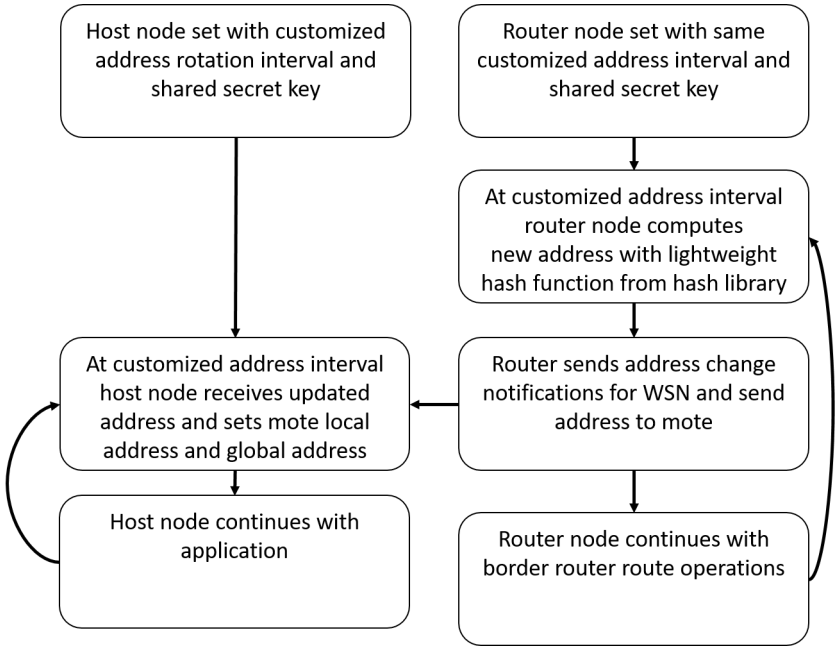

Figure 7.3: Border Based Design Flow Diagram of μMT6D

7.1.3 Address Hashing

The original principles of MT6D were used for the design of μMT6D for use with low power and low resource IoT devices and within WSNs. As mentioned in Chapter 3, μMT6D is a security technique that prevents targeted attacks with the use of an address rotation computed based on an algorithm utilizing hash functions. The original MT6D was provided

in [43] and used in [84] for a large scale server implementation. These utilize SHA256. For the address rotation, the IID, or initial Interface Identifier for the device, a shared session key, and a timestamp are concatenated together and then hashed. The first 64 bits of this are then concatenated with the 64-bit network address of the host yielding the IPv6 128-bit address. This is done also for the destination or communicating host so that both clients or a client and a server can have a source and destination address calculated. This can be seen with the equation:

$$IID'_{x(t_i)} = H[IID_x||K_S||t_i]_{0\rightarrow 63}$$

The μMT6D IID, IID is calculated with x representing the host and t_i the time at instance i. IID_x is the statically defined IID from the host x. K_S is the shared session key, and H is a cryptographically strong hashing algorithm which returns a result over 64 bits. MT6D uses the SHA256 hashing algorithm, μMT6D utilizes a hash library of lightweight hashing algorithms. This allows for comparison and customization and the ability for new lightweight hashing algorithms to be included later. A window of the previous and next addresses is also kept in case there are any inaccuracies in the network time used for the timestamp. The addition of changing source and destination ports can also be included.

Chapter 8

Lightweight Hash Algorithms

One optimization technique for μMT6D is the utilization of lightweight cryptographic hash algorithms. MT6D was first implemented with the use of the Secure Hash Algorithm, SHA-256. This is suited fine for use with the traditional computing devices, but the algorithm is not optimized for use with the resource constrained IoT devices. For this reason, this optimization of μMT6D will allow for the switching and selection of different lightweight hash algorithms. This will also allow for the use of future algorithms for inclusion within μMT6D as the state of the art advances.

Different hash algorithms have pros and cons concerning security, memory size, power, and throughput. In 2012, a comparison was done assessing various versions of hash functions such as PHOTON, QUARK, SPONGENT, and ARMADILLO [73]. Different algorithms were also implemented on an ATMEL AVR ATtiny45 8-bit microcontroller, and their performance was evaluated [16]. The area and throughput of the KECCAK, PHOTON, and SPONGENT hash functions have also been evaluated and implemented on Xilinx Spartan 6 FPGAs [65]. These and other works can be utilized to select different hash functions to include for use and evaluation with μMT6D. Many of the algorithms could also be further optimized depending

on the target IoT device. Ultimately, comparisons and experimentation would help to find which are best suited for what applications or modes of operation of μMT6D. An overview of the lightweight hash algorithms being considered for experimentation within μMT6D are below.

8.1 BLAKE

In 2008, BLAKE was submitted as a proposal to the NIST hash function competition. It was selected as one of five final candidates in the final round, but Keccak was selected as the winner and became the SHA-3 algorithm in 2012. BLAKE consists of three components, which were built upon historical components. The iteration mode HAFIA, improves the Merkle-Damgård paradigm. The internal structure is utilized in the BLAKE hash function and is a local wide-pipe structure. Finally, the compression algorithm is modified from the Bernstein stream cipher called ChaCha [12].

In 2010, the BLAKE Family of Hash Functions, including BLAKE-28, -32, -48, and -64, were presented with field-programmable gate array (FPGA) hardware implementations [99]. These hash functions produce message digests where their final sizes include 224-, 245-, 384, and 512-bits. The implementation results show the comparisons for the hash functions for F (MHz), Throughput (10xMbps), and Area (Slices) [99].

BLAKE2 was presented in 2013 as an improved version for BLAKE in terms of speed in software. Two types of BLAKE2 hash algorithms are given including BLAKE2b, for 64-

bit platforms, and BLAKE2s, for smaller architectures such as 8- to 32-bit platforms. The digests produced are between 1 and 64 bytes and 1 to 32 bytes for BLAKE2b and BLAKE2s. BLAKE2 was also designed to be faster than BLAKE with a reduced number of rounds. Additionally, it uses 32 percent less RAM. BLAKE2b, which would be useful for smaller embedded systems, requires 336 bytes of RAM [13].

The specification for BLAKE2 is RFC 7693 [93]. This provides a detailed overview of the algorithm, as well as, a link to the provided C implementation. BLAKE2 is described as being highly secure, at least as secure as SHA-3, as well as suited for both hardware and software on any platform [93]. BLAKE2, however, was optimized for speed in software [93].

8.2 SQUASH

Shamir presented SQUASH, which is short for "SQUare-hASH", as a hash algorithm designed for challenge-response MAC applications in resource constrained devices. It is presented as simple and efficient with a construction at least as secure as Rabin's public key encryption scheme for the same application. Drawing from the Rabin cryptosystem, SQUASH is based on the one-way function $c = m^2 mod n$. Here, collision resistance was not a priority, because with the challenge-response authentication a collision was not considered a security threat [95]. Collision resistance is a higher priority in hash functions utilized for digital signature schemes or other applications with a goal to prevent forgery. The FPGA implementation of the SQUASH algorithm at most was shown to require 6,000 gates which was viewed high for

constrained devices [49]. A C++ simulation of SQUASH showed the performance to scale linearly as the size of an RFID tag's register increases, but this simulation was only able to process a very small number of tags per second. The averages were drastically below those of [49] and this was attributed to the specialized hardware [72]. In 2009, Ouafi and Vaudenay showed the vulnerability of the linear mixing function previously utilized in SQUASH, but conclude that the security of SQUASH is still open with the use of non-linear mappings [88].

8.3 ARMADILLO

Another hash algorithm which was proposed for use with RFID tags in mind is ARMADILLO. This cryptographic hash function is hardware-dedicated and suited for providing data integrity, such as in digital signature schemes, and for providing message authentication [14]. An updated version, ARMADILLO2, was presented to add a compression function optimized to be more compact in hardware and also more secure [14]. Although, a key recovery attack was shown successful for ARMADILLO1, [94], as well as shown possible for ARMADILLO2 [1]. Further vulnerabilities and possible free-start and semi-free-start collision attacks were also shown for ARMADILLO2 [86]. These cryptanalysis papers demonstrate that ARMADILLO2 is not suited for use in security applications. A third version, ARMADILLO3, like the previous versions combines a substitution and permutation layer, but reduces the size requirements and address some of the attacks shown for the previous versions [101].

8.4 QUARK

The QUARK hash function family, including U-QUARK, D-QUARK, and S-QUARK, is a set of lightweight hash functions based on the sponge construction designed for use with low-power and low-energy devices [11]. The varying uses include message authentication as well as stream encryption, or authenticated encryption. The lightest version, U-QUARK, is described as providing at least 64-bit security against all attacks, including collisions, multicollisions, distinguishers, and more [11]. The design of QUARK was driven by a balance between the security-performance ratio and features and design aspects drew from the GRAIN family of stream ciphers and KATAN family of block ciphers.

8.5 LHASH

Another hash function, Lightweight Hash Function, LHash, was also designed to allow for the different trade-offs between security, speed, energy consumption, and implementation costs [108]. Similar to QUARK, PHOTON, and SPONGENT, LHash is based on an extended sponge functions framework to allow for the adjustment of parameters resulting in this flexibility. The permutation of LHash is composed of an extended variant of an improved generalized Feistel structure (GFS), and called Feistel-PG. In addition, the S-box and MDS linear layer for the internal permutation were designed specifically for hardware. It is claimed that the MDS linear layer is similar to that of PHOTON, but more compact [108].

8.6 SPONGENT

SPONGENT is a family of lightweight hash functions with hash sizes of 88-,128-,160-, 224-, and 256-bits. The permutation is similar to that of PRESENT and it is based on a sponge construction [21]. As is the trend, the algorithms allow for adjustments in implementation to accommodate trade-offs in speed and memory size. Thirteen variants of SPONGENT are described in [22] as having a smaller footprint than QUARK and an area comparable to PHOTON, but many times more compact. Further, SPONGENT allows for variable levels of security, as with QUARK and PHOTON, by allowing a reduced level of second preimage security, while maintaining collision resistance. Other SPONGENT variations maintain the standard preimage security, second preimage security, and collision security and still have lower area requirements than SHA-1, -2, and -3 [22].

The SPONGENT hash algorithm has many positive aspects, but also some negative ones. On the positive side, the truncated differential characteristics lower the probability of finding a path for collision attacks and the algorithm is resistant to pre-image attacks. This resistance is in the pre-computational stage only. Unfortunately, the weaknesses include the fact that the algorithm is silent on side channel attacks, and the algorithm, when used for higher security implementations with larger block sizes and larger numbers take longer and are more cpu intensive.

8.7 PHOTON

Some hash functions, including both SHA-1 and SHA-2, are not ideal for small embedded devices due to the fact that they are much too large. Others, have focused on optimizations for software, such as BLAKE2, and most of these output at least 256 bits [54]. The PHOTON family of lightweight hash functions were designed for use in extremely constrained devices [54]. Photon utilizes the sponge functions framework first introduced by Bertoni et al [20]. The different hash functions are suitable for instances where varying levels of security may be needed depending on the use case. PHOTON provides security ranging from 64-bit preimage resistance security to 128-bit collision resistance security [54]. The parameters, security level, and performance of PHOTON was compared to other lightweight hash functions. In general, PHOTON required less space and had similar preimage resistance and collision resistance to other hash functions. This family of hash functions was designed with consideration to area and throughput trade-offs and therefore is flexible for different needs on constrained devices [54]. A low cost FPGA implementation of PHOTON is presented in [86] showing the feasibility of a fast lightweight implementation.

Some of the strengths of PHOTON include the fact that the internal memory size is low when compared to other hash algorithms, that the sponge like extension is a protection against structural flaws and collisions, and that the algorithm is protected from linear or differential attacks. One weakness, however, is that the use of S-boxes decreases the hardware efficiency of the algorithm [73].

8.8 SHAT

SHAT, standing for sponge hash algorithm, is a hash function designed by sponge construction that provides a trade-off between power consumption and throughput. It generates 128-, 256-, and 384-bit hash values [116]. The area, delay, power, and throughput has been compared to that of other current hash functions [116], however, the security of the hash function is not discussed.

8.9 Hash Comparisons

All of these hash algorithms have pros and cons concerning security, memory size, power, and throughput. In 2012, a comparison was done assessing various versions of hash functions such as PHOTON, QUARK, SPONGENT, and ARMADILLO [73]. Three algorithms achieved only 33% or 1000 gate equivalents (GE), including SPONGENT-88, SPONGENT-128, and PHOTON-80. SPONGENT-128 was described as the lightest hash function. Different algorithms were also implemented on an ATMEL AVR ATtiny45 8-bit microcontroller, and their performance was evaluated [16]. The area and throughput of the KECCAK, PHOTON, and SPONGENT hash functions have also been evaluated and implemented on Xilinx Spartan 6 FPGAs [65].

Future work can involve the selection of even more of these hash functions and future ones introduced for comparisons and testing within the experiments to find which is best suited to

different applications. For this research, due to the promising performance and size, as well as, their design being suited for small embedded systems those selected include BLAKE2s, QUARK, SPONGENT, and SHA256, which is utilized in MT6D.

Chapter 9

Testing and Validation Setup

A combination of simulation and physical hardware testing was selected for the experimentation of μMT6D. Not only do physical testbed and network conditions provide more realistic experimental conditions, but the addition of simulation allows for the ability to control constants and conduct larger scale testing. As mentioned with the growing number of IoT connected devices and applications, the ability for a security technique to be suited for a single node and also scale is very important. In many cases smaller-scale testbeds are sufficient for WSN experiments, but further knowing the scaling limitations and having some larger-scale testing is also an important aspect [47].

9.1 Simulation Setup Design

As mentioned, Micro-Moving Target IPv6 Defense, μMT6D, is a moving target defense designed to be lightweight and operate efficiently on small embedded devices. The open source Contiki operating system was selected for the implementation since it supports fully standard IPv6 along with IPv4 and has an option of utilizing different low-power wireless standards

such as 6LoWPAN, RPL (Routing Protocol for Low-Power and Lossy Networks (LLNs)), and CoAP (Constrained Application Protocol) [37]. Contiki was the first system to introduce the concept of IP networking for low-power wireless systems and includes functionality and tools suited for networked embedded systems [38]. Our simulations run with the Cooja simulator included in the Contiki development environment which allows for the use of different simulated low-power and resource devices such as WiSmote and TMote Sky nodes. An RPL Border Router is used as the network edge device as this is the primary routing protocol for IPv6 LLNs [107].

Consistent with typical network architectures [34], our configuration includes the RPL border router integrating 6LoWPAN over IEEE 802.15.4 with the IP network. This design configuration of the WSN of motes running μMT6D connected to a host machine running MT6D or an MT6D server can be seen in Figure 9.1. The RPL Border Router is the root of the directed acyclic graph (DAG) formed by the connected nodes running μMT6D. The border router hosts a webpage allowing for a visual of the IPv6 addresses of the neighboring nodes and routes in the simulation. The simulated RPL network was connected with a bridge to the local machine utilizing the Tunslip6 utility. Tunslip6 creates a virtual network interface (tun0) on the host and uses the Serial Line Internet Protocol (SLIP) to encapsulate and pass the IP traffic. This configuration will allow for future testing and experimentation with μMT6D. Another possible configuration which may be considered for utilization is simulating both the wireless sensor network and the host machine or server and network to which it is connecting. Such a configuration can be seen in Figure 9.2.

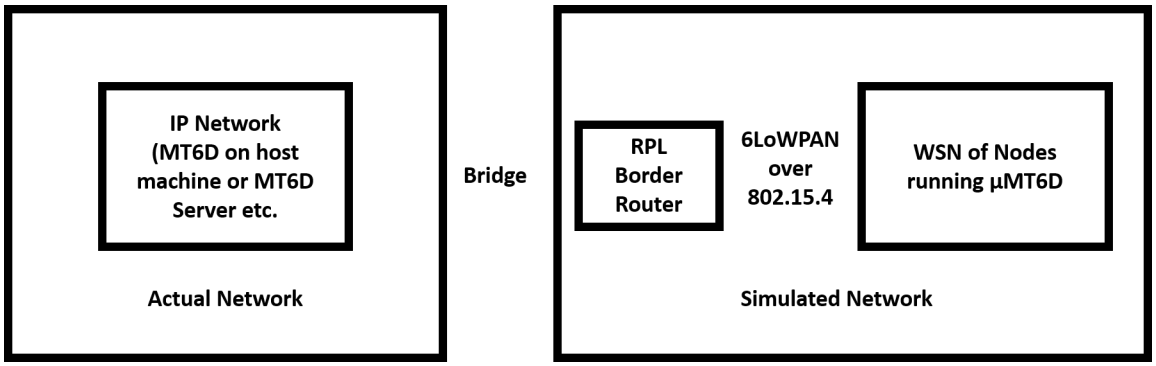

Figure 9.1: Simulation configuration of μMT6D

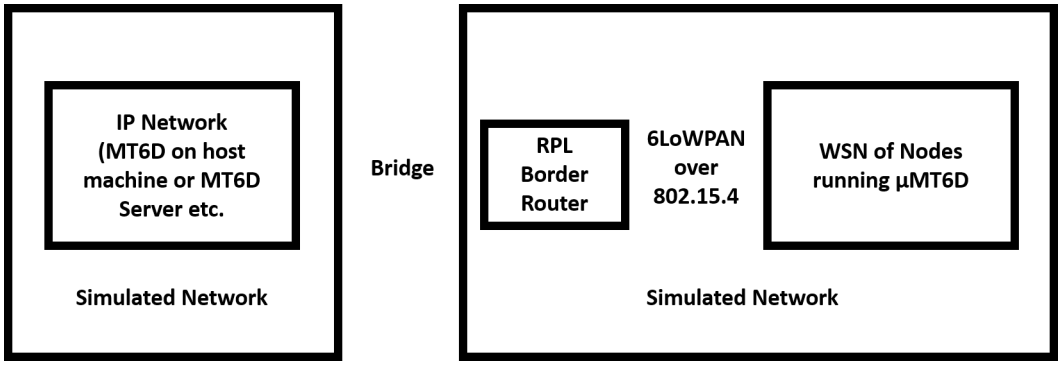

Figure 9.2: Simulation configuration of μMT6D plus network simulation

9.2 Hardware

Actual hardware experimentation and analysis was done in addition to simulation and testing. This configuration is similar to that of the simulation except a WSN is composed of physical motes. The used of both WiSMote and TMote Sky low-power wireless sensor devices allows for the implementation and testing of μMT6D for analysis in real network conditions. A Raspberry Pi is used as the border router. There is also the potential to combine simulation and real hardware for testing by the addition of having simulated network conditions being sent to the physical devices. Example configuration layouts can be seen in Figure 9.3 and Figure 9.4.

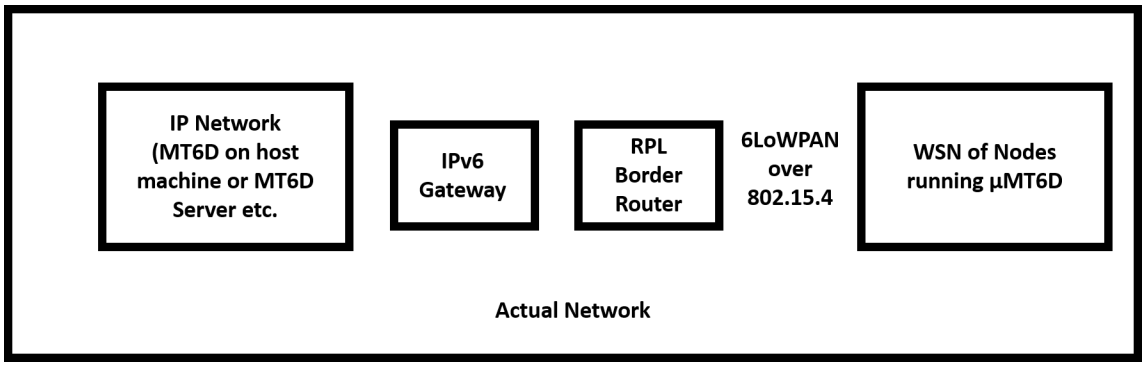

Figure 9.3: Physical hardware configuration of μMT6D testbed

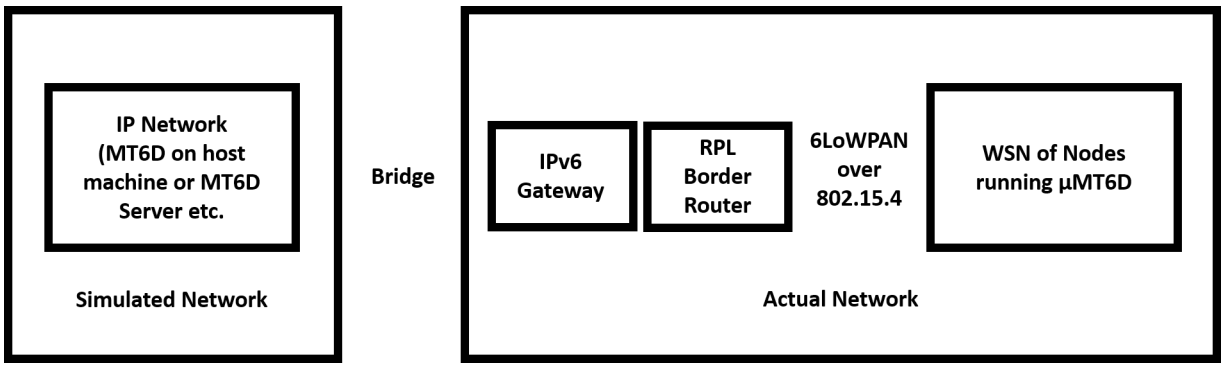

Figure 9.4: Physical hardware plus network simulation configuration of μMT6D testbed

9.3 Metrics and Validation

Whether through simulation or physical hardware testing, experiments conducted to analyze μMT6D were set up to be able to gain insights into the viability of μMT6D for resource constrained devices common in wireless sensor network applications. The percentage of overall resources utilized by a device running μMT6D in comparison to those running the same application without μMT6D was assessed in multiple experiments. The additional storage space needed, as well as, the additional power consumption and energy consumption needed by μMT6D should be as low as possible to enable the most resources to be available for the device application while still providing an appropriate level of security. This involves

analyzing the security technique from different angles.

We considered different hashing functions in terms of footprint, and power and energy consumption, and began to look at scaling to seek to find the applicability of this security technique as a part of a WSN running μMT6D. This demonstrated that μMT6D is a workable solution for a network containing embedded devices and can also lead to looking at further possibilities. Further, power and energy was explored. A security technique is of little use if the device cannot perform its intended functionality within the necessary footprint, power, and energy restrictions. Also, security requirements differ. A device with a non-critical application will have different requirements than one which is critical. For example, a temperature sensor sending data to a thermostat in a smart home will have little impact from a compromise except maybe a high utility bill in comparison to a sensor sending data to an emergency alarm to warn of equipment overheating that could lead to an impact such as an explosion and placing human lives at risk. Focusing on gathering data on the footprint, power consumption, and energy consumption of the devices running μMT6D will be necessary to show viability.

Chapter 10

Hardware & Calculations

The hardware utilized for the research was the WiSMote. The Contiki OS includes the WiSMote as a platform and also supports simulation of WiSMotes within the simulation tool Cooja. The WiSMote data sheet and schematics were used in setting up the calculations [58, 59, 102, 103].

This research looked at both power and energy metrics. The methods and equations used with the metrics from the Contiki OS Powertrace tool to report on power and energy results from the simulations are detailed below. The data collected included measures of power consumption of radio transmission time (TX), radio receiving/listening time (RX), full power CPU time (CPU), and reduced power CPU time (LPM) total power consumption in milliwatts. To compute the power in milliwatts the current consumption in milliamps (mA) was noted for each state from datasheets for the WiSMote. The WiSMote has a TI MSP430F5437x processor (MSP430X) and a [103]. The CPU and LPM and TX and RX states are mutually exclusive. If the mote is in the CPU, or full power computing state, it cannot be in the LPM, low power state. Similarly, if the mote is in the TX, transmitting, state, it cannot be in the RX, receiving, state. The motes are always in either CPU or LPM

states and may be in TX and RX states. So the total number of RTICKS for the period in which the Powertrace collected is CPU plus LPM. To compute the CPU Power in mW the Powertrace value is multiplied by the CPU current value in mA and then divided by the total of CPU plus LPM. This was done for every Powertrace value and then divided by the number of values to get the average power consumption in mW. This can be seen in equations 10.1, 10.2, 10.3, and 10.4. This was repeated for all the other types. The averages of the CPU, LPM, TX, and RX are then combined to get the total average power.

Next, when calculating energy you consider power over time. The energy calculations also used the metric outputs from Powertrace and multiplied the measurements by the amount of time which gave energy in terms of millijoules (mJ) which were converted to (J).

$$power\_cpu = \frac{cpu * current\_cpu}{rtimer\_ticks} \tag{10.1}$$

$$power\_lpm = \frac{lpm * current\_lpm}{rtimer\_ticks} \tag{10.2}$$

$$power\_tx = \frac{tx * current\_tx}{rtimer\_ticks} \tag{10.3}$$

$$power\_rx = \frac{rx * current\_rx}{rtimer\_ticks} \tag{10.4}$$

$$Power = \frac{Energy}{Time} \tag{10.5}$$

$$Energy = Power * Time \tag{10.6}$$

In order to verify these simulation metrics are similar to what would be expected, average expected power consumption metrics were calculated with information provided on the data sheets and different percentages for the different power states. The cases include Case 1, the address rotation interval of 5 seconds, Case 2, the address rotation interval of 1 minute, and Case 3, the address rotation interval of 1 hour. Expected power consumption in milliwatts with a radio duty cycle in which the radio is turned on and off to allow for power savings is given in Tables 10.1, 10.2, and 10.3. Expected power consumption in milliwatts with a radio duty cycle in which the radio is always on is given in Table 10.4. These were used to analyze and verify the results of simulations given later in the upcoming chapters.

Table 10.1: Expected Power Consumption (mW) by CPU Percentage Case 1

CPU	LPM	TX	RX	mW
0%	100%	20%	20%	0.001722
10%	90%	20%	20%	0.221553
20%	80%	20%	20%	0.441384
30%	70%	20%	20%	0.661215
40%	60%	20%	20%	0.881046
50%	50%	20%	20%	1.100877
60%	40%	20%	20%	1.320708
70%	30%	20%	20%	1.540539
80%	20%	20%	20%	1.760370
90%	10%	20%	20%	1.980201
100%	0%	20%	20%	2.200032

Table 10.2: Expected Power Consumption (mW) by CPU Percentage Case 2

CPU	LPM	TX	RX	mW
0%	100%	1.6667%	1.6667%	0.001693
10%	90%	1.6667%	1.6667%	0.221524
20%	80%	1.6667%	1.6667%	0.441355
30%	70%	1.6667%	1.6667%	0.661186
40%	60%	1.6667%	1.6667%	0.881017
50%	50%	1.6667%	1.6667%	1.100848
60%	40%	1.6667%	1.6667%	1.320679
70%	30%	1.6667%	1.6667%	1.540510
80%	20%	1.6667%	1.6667%	1.760341
90%	10%	1.6667%	1.6667%	1.980172
100%	0%	1.6667%	1.6667%	2.200003

Table 10.3: Expected Power Consumption (mW) by CPU Percentage Case 3

CPU	LPM	TX	RX	mW
0%	100%	0.0278%	0.0278%	0.001690
10%	90%	0.0278%	0.0278%	0.221521
20%	80%	0.0278%	0.0278%	0.441352
30%	70%	0.0278%	0.0278%	0.661183
40%	60%	0.0278%	0.0278%	0.881014
50%	50%	0.0278%	0.0278%	1.100845
60%	40%	0.0278%	0.0278%	1.320676
70%	30%	0.0278%	0.0278%	1.540507
80%	20%	0.0278%	0.0278%	1.760338
90%	10%	0.0278%	0.0278%	1.980169
100%	0%	0.0278%	0.0278%	2.200000

Table 10.4: Expected Power Consumption (mW) by CPU Percentage Radio Always On

CPU	LPM	TX	RX	mW
0%	100%	100%	100%	52.1017
10%	90%	100%	100%	52.3215
20%	80%	100%	100%	52.5414
30%	70%	100%	100%	52.7612
40%	60%	100%	100%	52.9810
50%	50%	100%	100%	53.2008
60%	40%	100%	100%	53.4207
70%	30%	100%	100%	53.6405
80%	20%	100%	100%	53.8603
90%	10%	100%	100%	54.0802
100%	0%	100%	100%	54.3000

Chapter 11

Host Based Initial Simulation Experiment

11.1 Simulation Introduction

This research explores the uses of a Micro Moving Target IPv6 Defense (μMT6D). In this initial experiment we assess the power consumption overhead and detail the overall security benefits gained from use of this technique with SHA256 [114]. Moving Target Defenses were seen as game changers in the security field and now we must change the game once more and look towards their application for IoT devices. With a need to provide privacy and a defense against targeted attacks on resource constrained devices that are a part of vital communication and control systems, μMT6D is a viable solution that we continue to develop and assess for future use.

11.2 Simulation Background

As mentioned in the introduction, MTDs were considered to be a "game-changing" theme in 2010 [62]. Different variants of MTDs were researched to discover their impact and assess their attack prevention capabilities [52, 87]. The effectiveness and evaluation of MTD systems is an ongoing topic [112, 119]. MTDs can be categorized as passive and active defense systems, some implementing the use of attack indicators to aid decisions and system responses [121] and other MTDs vary in the techniques used for obfuscation [60, 69]. A Moving Target IPv6 Defense (MT6D) was introduced as a viable security mechanism for preventing targeted attacks, host tracking, and eavesdropping [44]. It employs address rotation to obscure the communication of the devices [43]. MT6D was furthered to include a network layer client-server implementation utilizing a Distributed Hash Table (DHT) Blind Rendezvous for session establishment [82, 83, 84]. Most MTDs have been developed for full-scale systems and devices. However, there is a need for IoT security. Looking towards small embedded systems, the dynamic address change on low-powered devices was explored [90, 97] and then a design developed to make the entire MT6D viable for IoT devices as μMT6D [113]. The underlying algorithm used within μMT6D is based on the algorithm first included in MT6D. Anonymity is achieved through the rotating addresses and privacy due to an attacker not being able to track the rotating addresses [44]. Once a lightweight encryption is added to this implementation to encrypt packets before they are tunneled, μMT6D will also prevent traffic correlation and observation [44]. This paper shows the implementation and the results of an initial power analysis of μMT6D compared to a control application on

a medium sized IoT device.

11.3 Simulation Setup

This experiment consisted of an implementation of an application with no security mechanisms as our reference base, and the same application with the added security of μMT6D added to each WSN device and enabling them to change their own address. The setup consists of the architecture design given in [113], with the addition of the connection to the full scale MT6D server left to upcoming future work. This implementation includes a gateway device as the router operating over 802.15.4. The WSN mote is connected via the gateway device and a Serial Line Internet Protocol, SLIP, bridge provided from the Contiki Tunslip utility to an Ubuntu Virtual Machine on an external IPv6 network with an IPv6 router connected via the Hurricane Electric IPv6 Tunnel Broker. Future testing will include results when connected to the native operational IPv6 network of Virginia Tech. The selected lightweight operating system was Contiki 3.0 which is an open source operating system for the IoT. It provides the IPv6 network support including methods and evaluation techniques used in this research for our implementation and initial power testing [38]. The Contiki virtual machine also provided the means for simulation with Cooja, the Contiki network simulation tool. The WSN motes for the simulations and the physical test-bed are composed of WisMotes which are considered medium low power wireless motes complete with light and temperature sensors and are IEEE 802.15.4 compliant. The current gateway in

this implementation is also a WisMote. The IPv6 router was configured on a Raspberry Pi 3, Model B. The implementation will in the future be loaded on both simulated and physical hardware, but these initial tests were completed through the Cooja Simulator. In this implementation the WSN motes run a process in which they periodically send a udp packet. The gateway is an RPL, Routing Protocol for LLNs, Border Router. The RPL provided through the Contiki implementation is used for this research for routing the packets within the WSN.

11.4 Simulation Experiment SHA256

This simulation was the first power analysis for this research and involved one WSN mote sending a packet every minute and one WSN mote sending a packet every minute while it was also rotating its IPv6 address every five minutes with the μMT6D technique. Power measurements were taken for comparison of each. The Contiki ENERGEST and Collect View tools were used to collect the measurements every minute. Both the control application and μMT6D application sent a UDP packet every minute. The address rotation for μMT6D was set for every five minutes. Data was collected over a 24 hour period simulation for both. The data collected included measures of power consumption of radio transmission time (TX), radio receiving/listening time (RX), full power CPU time (CPU), and reduced power CPU time (LPM). These were then used to calculate the total power consumption in milliwatts. A small sample of data collected for each of the respective implementations for the motes

can be seen in Tables 11.1 and 11.2. The averages included in the analysis were from the entire simulations spanning 24 hours and totaling over 4000 values of data for each type. Confidence intervals are provided in Table 11.4.

Table 11.1: Sample Power Data Control

CPU	LPM	TX	RX	Total
0.022811746	0.162809311	0.0259469	59.89313	60.1047
0.019732352	0.162902548	0.01028	59.98838	60.1813
0.019155155	0.162920024	0.0079588	59.99101	60.18104
0.01941208	0.162912245	0.0081228	59.99082	60.18127
0.019311014	0.162915305	0.0076338	59.99137	60.18123
0.01860041	0.162936821	0.0053795	59.99392	60.18084
0.019358325	0.162913873	0.0076525	59.99135	60.18128
0.019361828	0.162913767	0.0068486	59.99226	60.18139
0.018870003	0.162928658	0.0070592	59.99202	60.18088
0.021601892	0.162845943	0.0139545	59.98423	60.18263

Table 11.2: Sample Power Data μMT6D

CPU	LPM	TX	RX	Total
0.058267544	0.161735788	0.1334499	59.84921	60.20266
0.047051261	0.162075392	0.0843805	59.90465	60.19816
0.046639202	0.162087869	0.0834725	59.90568	60.19788
0.046620038	0.162088449	0.082983	59.90623	60.19793
0.04733535	0.162066791	0.0865022	59.90226	60.19816
0.045130284	0.162133555	0.0768674	59.9143	60.19843
0.053393258	0.161883371	0.1065978	59.87955	60.20142
0.045960831	0.162108408	0.0797556	59.90811	60.19594
0.045772896	0.162114098	0.0794926	59.91018	60.19756
0.046634855	0.162088	0.0808337	59.90866	60.19822

11.5 Simulation Analysis

The Contiki Energest results of a one hour sample from the experiment can be seen in Figures 11.1, 11.2, 11.3, 11.4, and 11.5. The total averages can be seen in Table 11.3. This

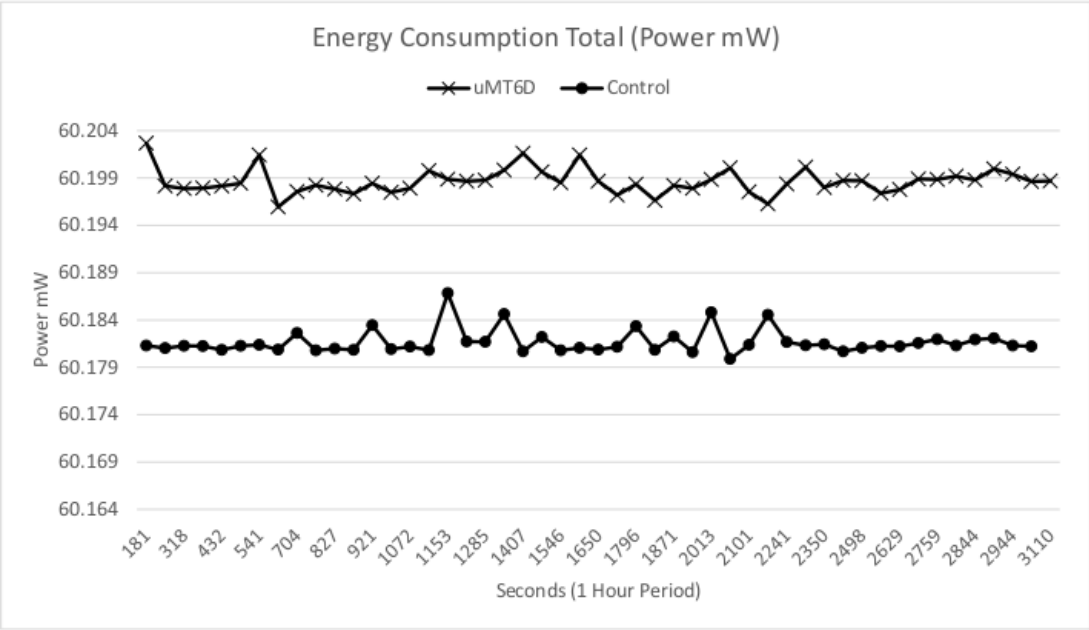

Figure 11.1: Total Power Consumption One Hour Sample (Note: The Y-axis begins at 60.164 mW)

Table 11.3: Average Power Consumption mW

Power in mW	Control	uMT6D
Total	60.182	60.199
CPU	0.019	0.046
LPM	0.163	0.162
RX	59.992	59.909
TX	0.008	0.018

initial experiment shows that there is some, but not an unmanageable amount of overhead when μMT6D is added to an IoT application. The increases in power consumption from our simulated experiment are seen in the total, CPU, and TX. This is expected because the WSN node is doing computation for the address rotations. The graphs show the natural variances in the power consumption. CPU and LPM modes are mutually exclusive and are RX and TX. This is why for LPM and RX the μMT6D power consumption is lower. Taken as a whole, there were no extreme peaks or increases. The peaks appear during times when

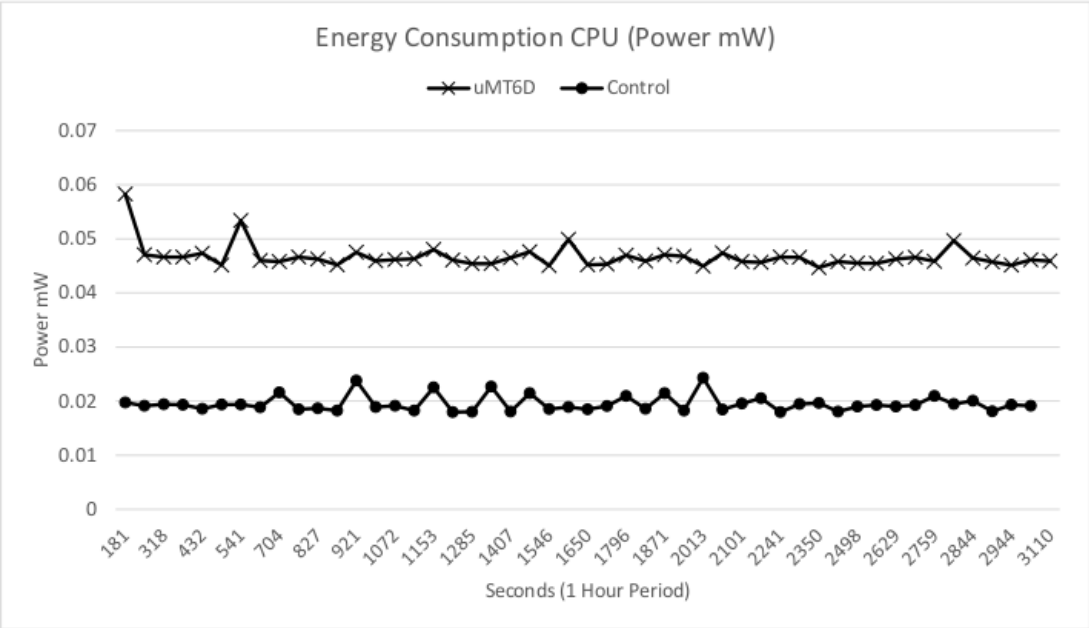

Figure 11.2: CPU Power Consumption One Hour Sample (Note: The Y-axis begins at 0 mW)

Table 11.4: 95 % Confidence Intervals in mW

CONTROL	±0.000126
μMT6D	±0.000080

the device is transmitting a packet. The μMT6D mote has the additional overhead of when it would compute a new address which keeps the average power consumption consistently higher than the control. It is worth noting that the radio was turned on for the entire experiment because the set radio duty cycle did not allow for the radio to be cycled off. This is addressed in Chapter 13. For the added security benefits provided by μMT6D, this overhead, although a challenge for some IoT applications, can certainly be worth accepting for use with critical applications and communications in which security is required to prevent target attacks.

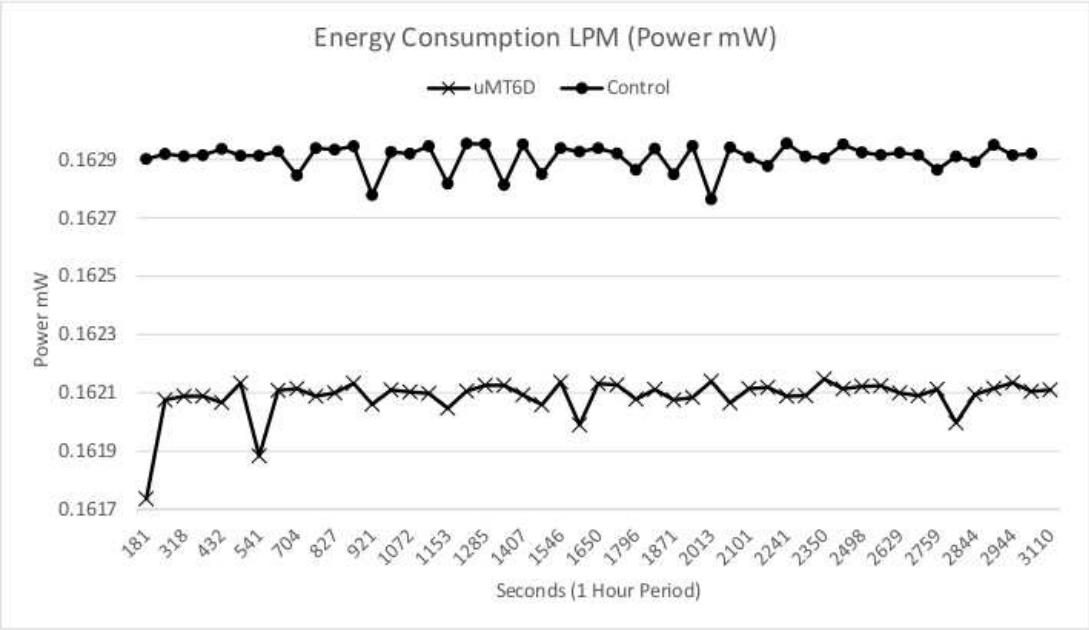

Figure 11.3: LPM Power Consumption One Hour Sample (Note: The Y-axis begins at 0.1617 mW)

11.6 Simulation Future Work & Conclusion

This initial implementation was the first step in a progression of planned optimizations and additions. On-going and future work includes the addition of a lightweight encryption scheme for packet encryption. To decrease power consumption overhead, the time interval for each address change can be altered to be more or less frequent. This trade-off could be explored further since affects the window of time that a device address remains static. Also, a hash library of lightweight hash algorithms has been incorporated for use with μMT6D and an assessment of the use of these different algorithms. This hash library allows for μMT6D to be updated based on new lightweight hashing algorithms as they are released. Lightweight hash functions such as PHOTON, Quark, and SPONGENT are designed to have

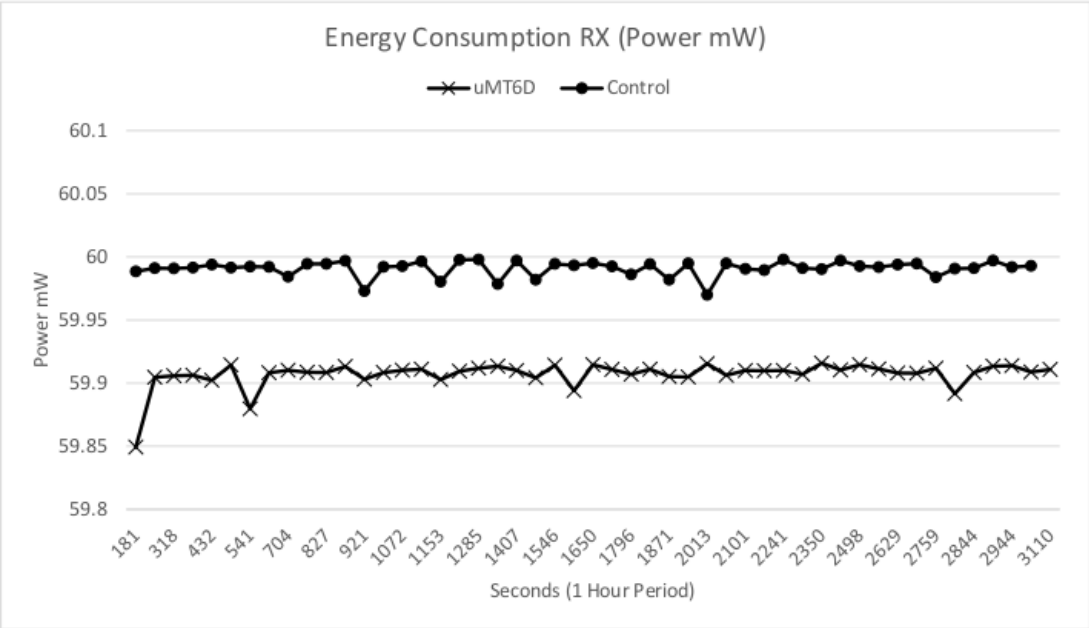

Figure 11.4: RX Power Consumption One Hour Sample (Note: The Y-axis begins at 59.8 mW)

smaller internal state sizes and be less intense on power consumption [79], this is expected to decrease power consumption further. Other additions include an assessment of the border-based implementation. Finally, an initial large scale simulation will be used to explore the scalability of μMT6D.

This chapter has given the initial power consumption results of the use of μMT6D with SHA256. This security mechanism was based on the established algorithms and fundamentals of the full scale MT6D, but has been designed and implemented to suit the power and resource constraints of IoT devices and applications that are becoming prevalent in our society. Due to the need for privacy and resilience to targeted attacks, μMT6D is a viable security mechanism for future use. The following chapters expand upon this knowledge highlighting other simulations and experiments furthering this work.

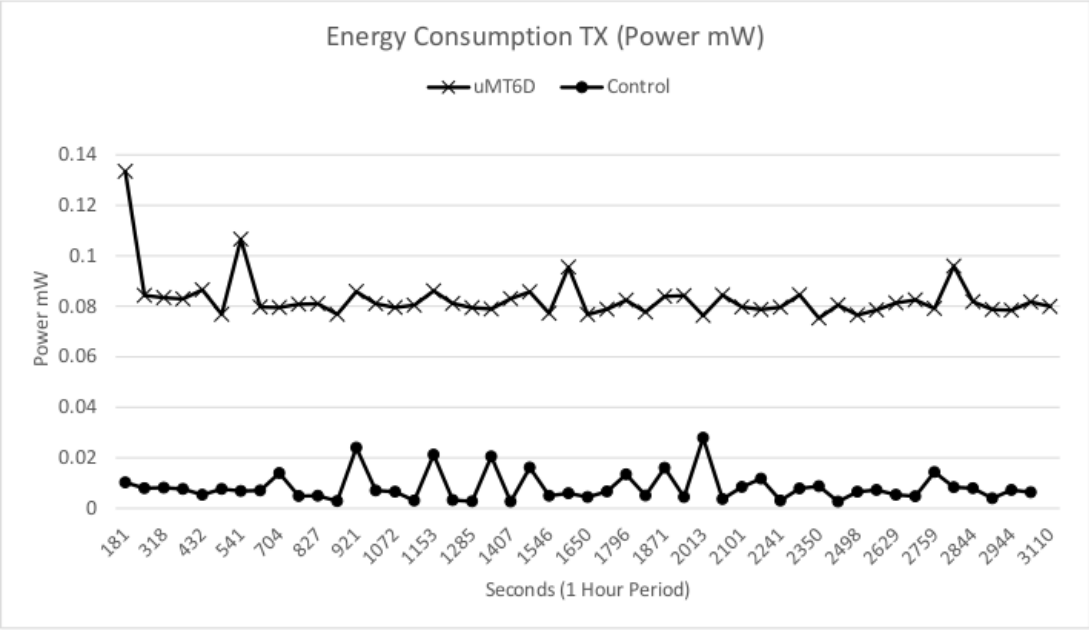

Figure 11.5: TX Power Consumption One Hour Sample (Note: The Y-axis begins at 0 mW)

Chapter 12

Host Based Lightweight Hash Algorithm Simulation: Footprint and Power Analysis

12.1 Experiment Introduction

As small embedded systems and new IoT applications are becoming even more commonplace, security techniques are needed which are specifically geared for securing these systems. The low power and low resource constraints of small embedded systems and wireless sensor networks are challenging for traditional security techniques applied to larger scale systems. μMT6D is a Micro Moving Target IPv6 Defense which has been designed for the purpose of preventing targeted attacks on small embedded systems. This security technique makes use of an algorithm for changing the IPv6 address of small embedded devices and we detail the results of simulations of this technique utilizing a hash library with different lightweight hashing algorithms for comparison and show the viability and flexibility of this security so-

lution. This chapter presents power metrics and the footprint for each lightweight hashing algorithm and compares them to each other and show the difference from a control with no security technique implemented. Future work also presents the next phase for μMT6D which includes a border based solution for wireless sensor networks and future experiments.

The setup for this experiment consists of the architecture design given in [113], with the addition of the connection to the full scale MT6D server left to upcoming future work. The following is background on the lightweight hash functions considered and also those selected for inclusion in this study.

12.2 Lightweight Hash Function Library

This hash library allows for μMT6D to be updated based on new lightweight hashing algorithms as they are released. Lightweight hash functions such as QUARK are designed to have smaller internal state sizes and be less intense on power consumption [79]. This is expected to decrease power consumption further from the use of SHA256. So far, SHA, BLAKE, SPONGENT, and QUARK are all included in the hash library and implemented for use in μMT6D [12, 13, 22, 26]. This particular experiment specifically included SHA256, BLAKE2s, SPONGENT256/256/16, and S-QUARK.

12.3 Implementation

There are many challenges in designing a system to test the use of a moving target defense with lightweight hashing algorithms. With resource constrained devices the methods used must be compiled with the testing application which may for example send packets or readings from the WSN to a border router. We used the different lightweight hashing algorithms discussed in earlier sections to begin experimentation and gather data on feasibility. All were compiled for use on the WiSMote, a platform by Arago Systems and described as an IPv6 WSN solution. Essentially, a WiSMote is a wireless sensor/actuator module with a TI MSP430 series 5 (16 bits), a TI CC2520 2.4GHz IEEE 802.15.4 compatible with 6LoWPAN and embedded sensors for luminosity and temperature, and a 3 axes accelerometer.

This experiment consisted of an implementation of an application with no security mechanisms as our reference base, and the same application with the added security of μMT6D added to each WSN device and enabling them to change their own address. The code base was compiled for the WiSMote in versions utilizing each lightweight hash function in the hash library. The application is the same for each in that that wireless sensor node connects to a border router and periodically sends a UDP packet. The control has no address rotation and the μMT6D versions rotate their own address at a given interval. The WSN mote is connected via the gateway device and a Serial Line Internet Protocol, SLIP, bridge provided from the Contiki Tunslip utility to an Ubuntu Virtual Machine on an external IPv6 network with an IPv6 router connected via the Hurricane Electric IPv6 Tunnel Broker.

The selected lightweight operating system was Contiki 3.0 which is an open source operating system for the IoT. It provides the IPv6 network support including methods and evaluation techniques utilized in this research for our implementation and power testing [38]. The Contiki virtual machine was used for the simulation with Cooja, the Contiki network simulation tool. The WSN motes are composed of WisMotes which are considered medium low power wireless motes complete with light and temperature sensors and are IEEE 802.15.4 compliant. The gateway in the implementation is also a WisMote and was run as an RPL, Routing Protocol for LLNs, Border Router. The RPL provided through the Contiki implementation was used for this research for routing the packets within the WSN. The IPv6 router was configured on a Raspberry Pi 3, Model B. The implementation has been successfully loaded on physical hardware, but these initial tests were completed through the Cooja Simulator.

12.4 Optimizations

The test application was compiled with the already mentioned Contiki OS and several techniques were used for size optimizations with the MSP430 GCC compiler. The -0s flag was used for size optimization along with -ffunction-sections and -gc-sections. These ensure that the compiler is optimizing functions and that unused functions are dropped. Another Makefile flag, #define PROCESS_CONF_NO_PROCESS_NAMES 1, can save flash and RAM by not including the ASCII strings for the naming of headers. The #define NETSTACK_CONF_RDC nullrdc_driver flag sets the radio duty cycle to the smallest size setting.

The ContikiMAC driver is looked at in Chapter 13. These optimizations as well as good code practices of eliminating unnecessary debug and logging code allowed the application, the μMT6D implementation, and the experiment data collection tools to all be included on a single WisMote with the Contiki OS target memory model set to medium.

12.5 Experiment

This experiment looked at three setups. For the first, the control WSN motes sent a UDP packet every minute and the μMT6D WSN motes sent a packet every minute while also rotating their own IPv6 address every five seconds with the μMT6D technique. For the second, the control WSN motes sent a UDP packet every hour and the μMT6D WSN motes additionally rotated their own IPv6 address every minute with the μMT6D technique. For the third and final setup, the control WSN motes sent a UDP packet every hour and the μMT6D WSN motes sent a packet every hour while also rotating their own IPv6 address every hour with the μMT6D technique. Contiki ENERGEST and Collect View tools were used to collect the measurements every minute. Data was collected over an eight hour period. The data collected included measures of power consumption of radio transmission time (TX), radio receiving/listening time (RX), full power CPU time (CPU), and reduced power CPU time (LPM) total power consumption in milliwatts.

12.6 Size Comparisons

An overview of the footprint for each variation can be seen in Figure 12.1. The sizes of the implementations are detailed in Tables A.1, A.2 and A.3 located in Appendix Section A.1. The (C) attribute marks that this was the size with the included overhead of the code for collecting the power consumption data metrics. Overall, SPONGENT has the smallest size footprint across all of the experimental versions. Utilizing Collect Tools added about 1.87 kB of overhead. The WiSMote has 16kB RAM and more than 128kB Flash.

12.7 Power Consumption

The overall average power consumption data results from all experiment types can be seen in Figures 12.2, 12.3, and 12.4. Results are given for the overhead over the control for each of the μMT6D lightweight hash function implementations. Detailed data can be seen in Tables C.1, C.2, and C.3 located in Appendix Section C.1. The confidence intervals are given in Table 12.1.

Table 12.1: 95 % Confidence Intervals in mW

	CASE 1	CASE2	CASE 3
CONTROL	±0.029197	±0.013499	±0.013563
SHA	±0.029650	±0.013417	±0.013456
BLAKE	±0.029702	±0.013806	±0.012646
QUARK	±0.031282	±0.013901	±0.013457
SPONGENT	±0.259656	±0.059911	±0.018543

The results from this experiment show that the average amount of power consumption over-

head when μMT6D is added to an IoT application utilizing varying implementations of lightweight hash functions is between 0.0125 and 2.6606 mW for a rotation interval of five seconds. This decreases to between 0.0028 and 1.0588 mW for a rotation interval of one minute, and between 0.0023 to 0.0229 for a rotation interval of every one hour.

The hashing algorithm with the highest power consumption was SPONGENT 256/256/016 for all experiment setups. There are many other versions of SPONGENT with varying levels of preimage, 2nd preimage, and collision security. The selected 256/256/16 has a higher number of rounds and results in more power consumption. Further studies could compare results from more than just this one version. For the experiment that sent a packet every minute and had the address rotating every five seconds, SHA marginally had a lower power consumption than BLAKE. For the experiment that sent a packet every hour and rotated address every minute, this was reversed with BLAKE having the least power consumption average followed by SHA. For the case where the address was only rotated every hour, the larger amount of time that the mote was not computing the address allowed for the extra power consumption by SPONGENT to remain closer to the other hash function versions, but still remained the top spot for most consumed. The hash function with the lowest power consumption was BLAKE followed closely by QUARK.

The main differences in power consumption from the setup where the mote would rotate the address every 5 seconds versus the experiments where the mote would only rotate the address every minute and hour can be seen in a much higher power consumption overhead. This is expected because the WSN node is doing computation for the address rotations

more often in the first case and then communicating this address. The μMT6D mote has the additional overhead of when it would compute a new address which keeps the average power consumption consistently higher than the control, but not by a large percentage. An additional experiment with a radio duty cycle allowing the mote to enter sleep can be seen in Chapter 13. However, the data from these experiments show that there is a future for the use of a moving target defense to protect resource constrained embedded devices.

For the protection against targeted attacks provided by μMT6D, the power overhead, although a challenge for some IoT applications, can be worth accepting for use with critical applications and communications in which security is required to prevent target attacks.

The following chapters explain the next steps in this research including a refined simulation experiment.

12.8 Experiment Conclusion & Future Work

On-going and future work for this research includes further development and testing to gather more data on the feasibility and strength of this technique. To change the power consumption overhead, the time interval for each address change can be altered to be more or less frequent. This trade-off can be explored further since it affects the window of time that a device address remains static and potentially vulnerable. Further, the duty cycle was optimal for footprint and not power consumption, therefore the motes duty cycle was not being regulated to allow the motes to go into the power saving modes. Even with low

power modes, a mote just listening for radio traffic can still draw around 60 milliwatt of power which is why this experiment data hovered around 58 to 60. Other additions and experiments may also include the connection with a server running the full scale MT6D and a border-based implementation. A border-based implementation could allow a device with more power or resources to do the calculations and address change algorithm for the whole WSN. A method in which the WSN nodes share the computational load to compute addresses could also be explored. The addition of a lightweight encryption scheme for packet encryption would further the strength of this technique and is used by the MT6D version. Finally, large scale simulation can be used to explore the scalability of μMT6D.

This chapter has presented power consumption and footprint results for different hash functions used in a hash library and implemented for use in μMT6D. This security mechanism was based on the established algorithms and fundamentals of the full scale MT6D, but has been designed and implemented to suit the power and resource constraints of IoT devices and applications that are becoming prevalent in our society. Due to the need for privacy and for IoT device resilience to targeted attacks, μMT6D is a viable security mechanism for future use.

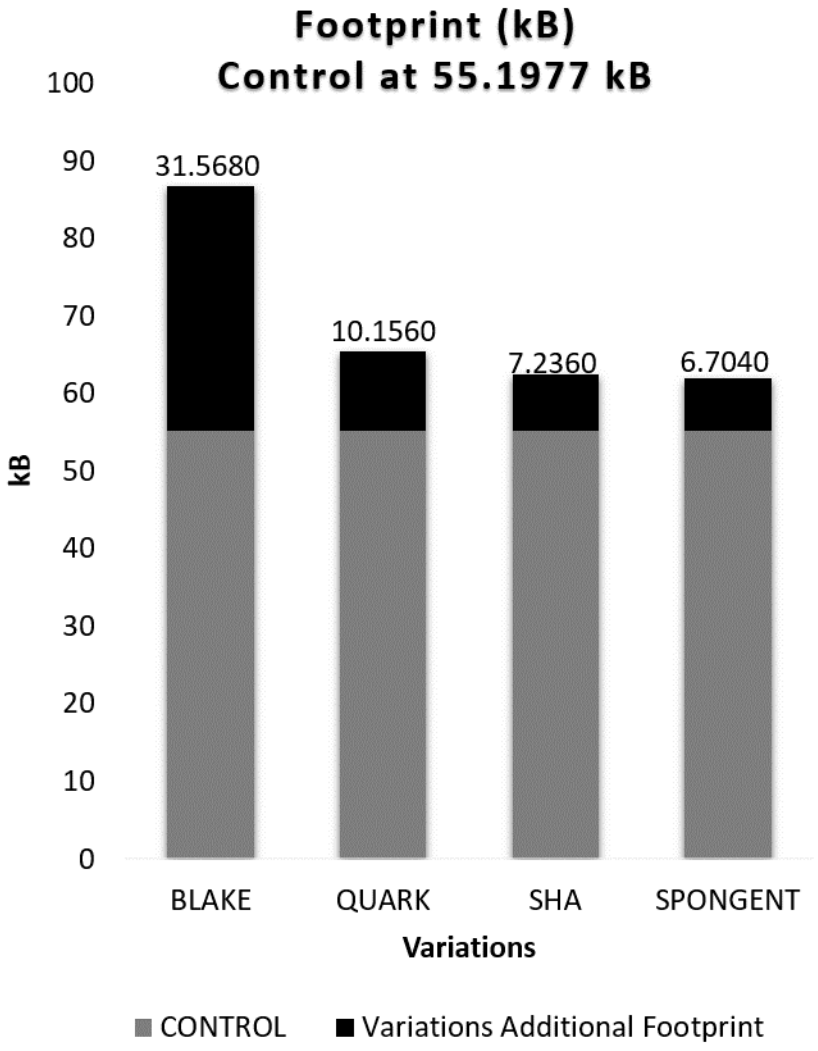

Figure 12.1: μMT6D Footprint Overhead

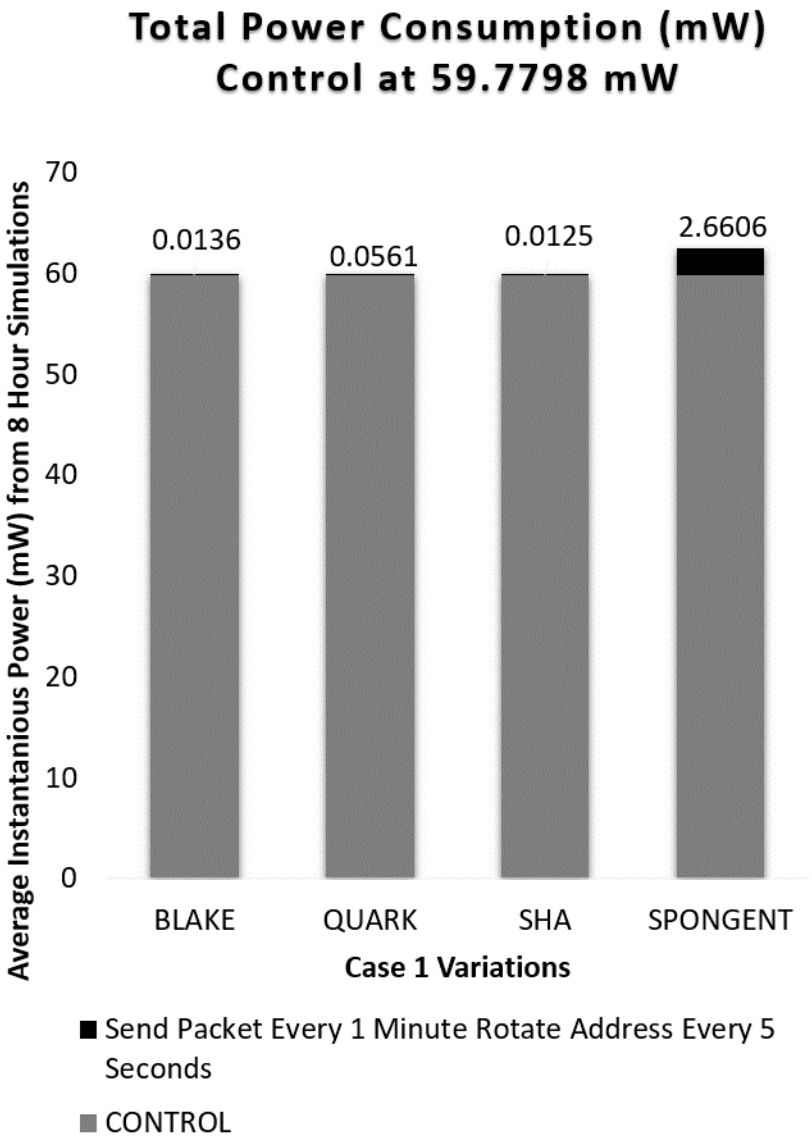

Figure 12.2: Average Total Power Case 1

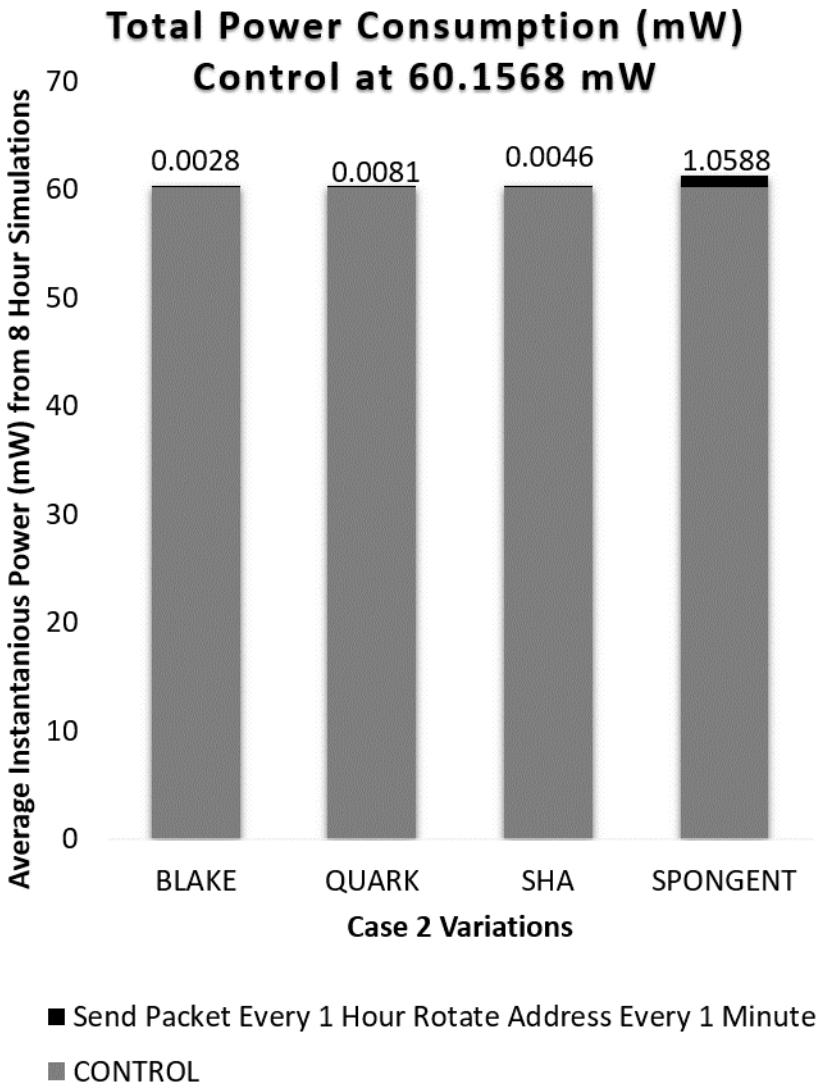

Figure 12.3: Average Total Power Case 2

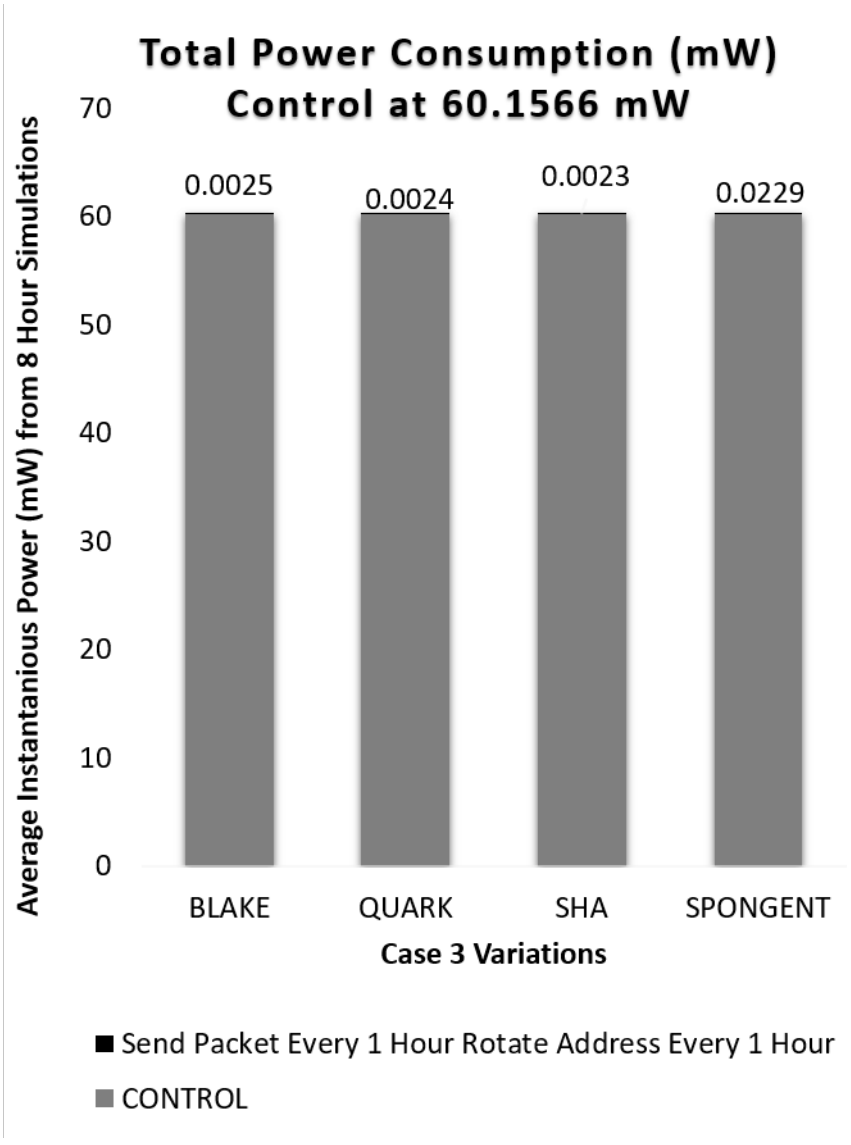

Figure 12.4: Average Total Power Case 3

Chapter 13

Refined Host Simulation of μMT6D

13.1 Simulation Refinements

The refinements which led to this new simulation experiment are listed below:

1. The inclusion of energy metrics in addition to footprint and power metrics

2. The inclusion of simulation cases for isolating metrics for the logical pieces making up the security technique:

   (a) Still included "Everything" cases: Each hash function included in full μMT6D logic

   (b) Still included a control

   (c) Added cases for each hash function computation only: SHA, Blake, Quark, and Spongent

   (d) Added cases for the address setup and the ending global address

3. Use of the power and energy efficient Contiki Radio Duty Cycle driver: ContikiMAC

4. Simulation data gathering start time and length adjusted based on calculated steady state time with Welch's test results [106]

5. Simulation replications based on desired confidence intervals

6. Use of Ten-step systematic approach to performance evaluation [61]

7. Used direct measurements from Contiki Powertrace [39] instead of from Collect View Tools.

13.2 Ten-step Systematic Approach to Performance Evaluation

The ten-step systematic approach to performance evaluation [61] was used to clearly describe the goals and setup of this refined simulation experiment and is given in the sections below.

13.3 State Goals and Define the System

The general goal of this research is to assess the viability of the use of μMT6D to protect IoT low-powered embedded devices by limiting the time an attacker may conduct reconnaissance and therefore preventing them from being able to target a device. The viability was explored with the metrics described in the Select Metrics section below.

The simulation setup described previously in Chapters 7 and 12 defines the system utilized

for this experiment. It is a WSN simulated in Cooja with nodes sending UDP packets at set intervals for the experiment. Simulations were run with nodes without μMT6D and also with the technique in order to compare the overhead. The WSN mote is connected via the gateway device and a Serial Line Internet Protocol, SLIP, bridge provided from the Contiki Tunslip utility to an Ubuntu Virtual Machine on an external IPv6 network with an IPv6 router connected via the Hurricane Electric IPv6 Tunnel Broker. The gateway in the implementation is also a WisMote and was run as an RPL, Routing Protocol for LLNs, Border Router. The RPL provided through the Contiki implementation was utilized for this research for routing the packets within the WSN. The IPv6 router was configured on a Raspberry Pi 3, Model B.

13.4 List Services and Outcomes

The services of this system include the routing of the UDP packets sent at different intervals within the WSN. The gateway utilized was the Contiki RPL, Routing Protocol for LLNs, Border Router already described in a previous session. The individual WiSMotes for the μMT6D cases did all of the computation independently and changed their own addresses.

13.5 Select Performance Metrics

Two main performance metrics were selected for this research and assigned utility classes. The utility classes for performance metrics include Lower is Better (LB), Higher is Better (HB), and Nominal is Best (NB) [61].

1. Viability was measured with footprint or size metrics (LB)
2. Low or acceptable overhead was measured in terms of energy consumption (LB)

13.6 List Parameters

1. System
   (a) Radio Duty Cycle
   (b) Lightweight hash functions
2. Workload
   (a) UDP packet send interval
   (b) For μMT6D simulations, the address rotation interval

13.7 Select Factors to Study

1. Radio Duty Cycle

(a) ContikiMAC driver

(b) NullRDC driver (Chapter 12)

2. Lightweight Hash Functions

(a) None (Control)

(b) Blake

(c) Quark

(d) SHA 256

(e) Spongent

3. UDP Packet Send Interval

(a) Every minute

(b) Every hour

4. Address Rotation Interval

(a) None (Control)

(b) Every five seconds

(c) Every minute

(d) Every hour

13.8 Select Evaluation Technique

The chosen evaluation technique for this experiment is simulation.

13.9 Select Workload

UDP packets sent all had the same data with string "Hello" and "Sent by Mote #." The UDP packets sent varied from one every minute to one every hour. The workload placed on the system from the motes changing addresses included every five seconds, every minute, and every hour, or none for the control.

13.10 Design Experiment

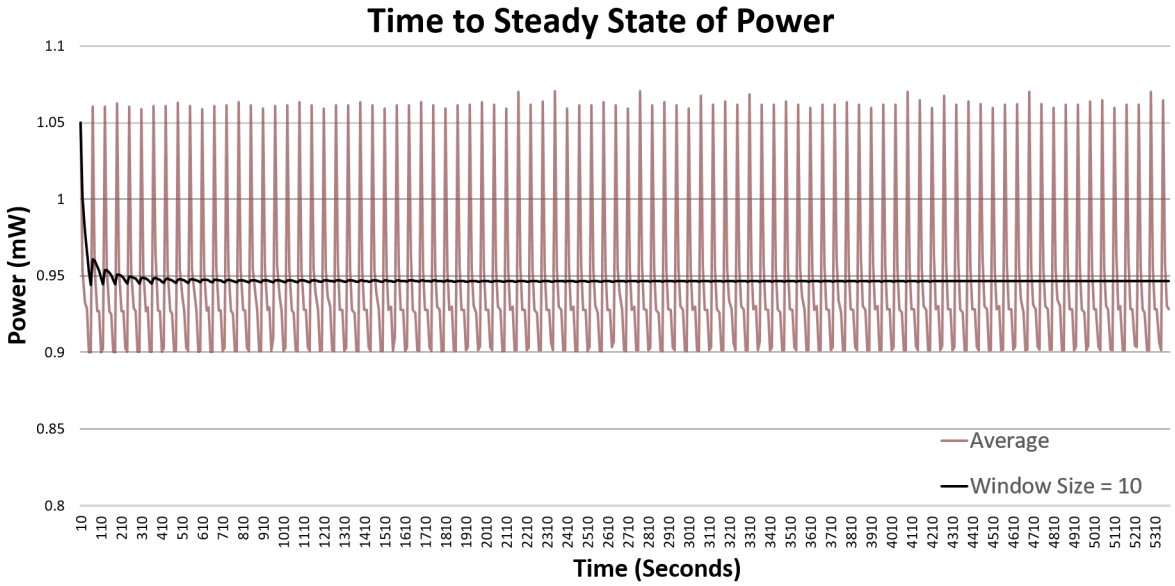

Figure 13.1: Graph to Find Warm-up Period with Welch's Test [106]

For this refined simulation, Welch's Test [106] as described in [74] and [33] was used to find the warm-up period or initial transient which is the time required by networks to reach steady-state in simulation. A graph was created with the windowed moving average set at ten and this can be seen in Figure 13.1. A graph was created with the windowed moving average set at ten and one example can be seen in Figure 13.1. These graphs were done for every variation being tested across all the cases for a total of 32 graphs. All of these can be seen in Appendix E.1. Based on these results, steady state is reached after ten minutes and data collection was selected to start at the ten minute mark which is 600 seconds. The simulation time frame should be at least ten times the initial transient, so the simulation length was determined to be 1 hour and 40 minutes or 6000 seconds.

The number of replications selected were calculated based on desired statistically significant confidence intervals as was done in [91]. The formula for computing the replications based on a 95 % certainty that the power and energy measurements will fall within ±0.01 of the mean power and energy can be seen below:

$$\epsilon \geq \left[\frac{t_{\frac{\alpha}{2}}, (R_0 - 1) \times s}{\sqrt{R}} \right] \tag{13.1}$$

or

$$R \geq \left(\frac{t_{\frac{\alpha}{2}}, (R_0 - 1) \times s}{\epsilon} \right)^2 \tag{13.2}$$

An initial set of ten simulation replications were run. For the confidence interval to be within $\pm\epsilon$ of true value Ω with a probability of $1 - \Omega_s$ the above equations must apply, where ϵ is

0.01 times the mean value and s is the standard deviation of the ten replications for that scenario. The initial number of replications, R_0 is 10. $t_{\frac{\alpha}{2}}, (R_0 - 1)$ is the quantile value from the t distribution table for $\alpha = 0.05$, our selection of 95% confidence. Our $t = 2.776$. The number of simulation repetitions to meet this 95% confidence that the power and energy metrics will be within 1% of the mean was 10.

13.11 Analyze and Interpret Data

13.11.1 Size Comparisons

An overview of the footprint for each variation can be seen in Figure 13.2. The sizes of the implementations are detailed in Tables B.1, B.2 and B.3 located in Appendix Section B.1. The (C) attribute marks that this was the size with the included overhead of the code for collecting the power and energy consumption data metrics. Overall, SPONGENT still has the smallest size footprint across all of the experimental versions. Utilizing the ContikiMAC Radio Duty Cycle had a small increase on the footprint. Utilizing Powertrace added about 1.98 kB of overhead. The WiSMote has 16kB RAM and more than 128kB Flash.

13.11.2 Power Consumption

The overall average power consumption data results from all experiment types can be seen in Figures 13.3, 13.4, and 13.5. Results are given for the overhead over the control for each of

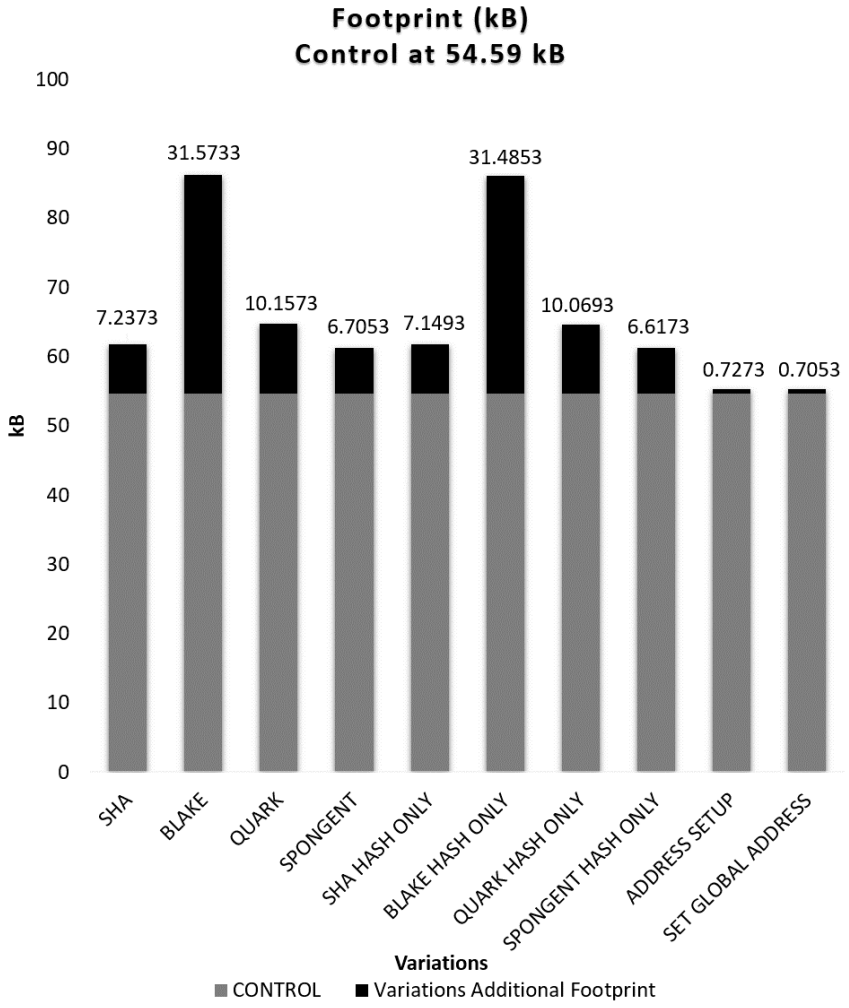

Figure 13.2: μMT6D Footprint Overhead

the μMT6D lightweight hash function implementations, hash only cases, and the remaining address setup cases. A detailed look at the power consumption can be seen in Tables D.1, D.2, and D.3 located in Appendix Section D.1. The confidence intervals are provided in Table 13.1.

Table 13.1: 95 % Confidence Intervals in mW

	CASE 1	CASE2	CASE 3
CONTROL	±0.000884	±0.000882	±0.000882
SHA	±0.000881	±0.000809	±0.000885
BLAKE	±0.000882	±0.000881	±0.000882
QUARK	±0.000896	±0.000891	±0.000856
SPONGENT	±0.016421	±0.016782	±0.001675
SHA HASH	±0.000884	±0.000884	±0.000810
BLAKE HASH	±0.000867	±0.000850	±0.000811
QUARK HASH	±0.000869	±0.000892	±0.000883
SPONGENT HASH	±0.016588	±0.016783	±0.001675
ADDRESS SETUP	±0.000824	±0.000883	±0.000882
SET GLOBAL ADDRESS	±0.000883	±0.000881	±0.000882

13.11.3 Energy Consumption

In addition to power, this experiment included metrics on energy. The average energy consumption data results from all of the types can be seen in Figures 13.6, 13.7, and 13.8. Again, these results are given for the overhead over the control for each of the μMT6D lightweight hash function implementations, hash only cases, and the remaining address setup cases. A detailed look at the energy consumption can be seen in Tables D.4, D.5, and D.6 located in Appendix Section D.2.

13.12 Present Results

Much like the previous experiment, the results of this experiment showed that μMT6D is viable in regards to footprint size and the amount of overhead in power and energy consumption. The choice of lightweight hash from the hash library impacts these often at odds.

Spongent had the smallest footprint, but the highest energy consumption. The inclusion of more test cases showed that the majority of the overhead associated with μMT6D is related to the hash function computations. When comparing to the models calculated in Chapter 10, for Case 1, the Blake, SHA, and Quark hash functions fell in the 10% CPU range and Spongent in the 60% to 70% range. For Case 2, all of the hash functions were in the 10% to 20% range except Spongent in the 30% range. Last, for Case 3 all of the hash functions were in the 10% to 20% range. As more hash functions algorithms are introduced or improved this security technique can be catered to use the best suited in terms of security levels, footprint, and power and energy consumption. Related, the selection of the address rotation interval is a balance between security and resource overhead. The more frequent the address rotation the more secure and the higher the power and energy overhead. Likewise, the less frequent the address rotation the less secure, but the lower the power and energy overhead. The customization available with the design of μMT6D and the hash library shows the suitability of the technique for resource constrained embedded devices.

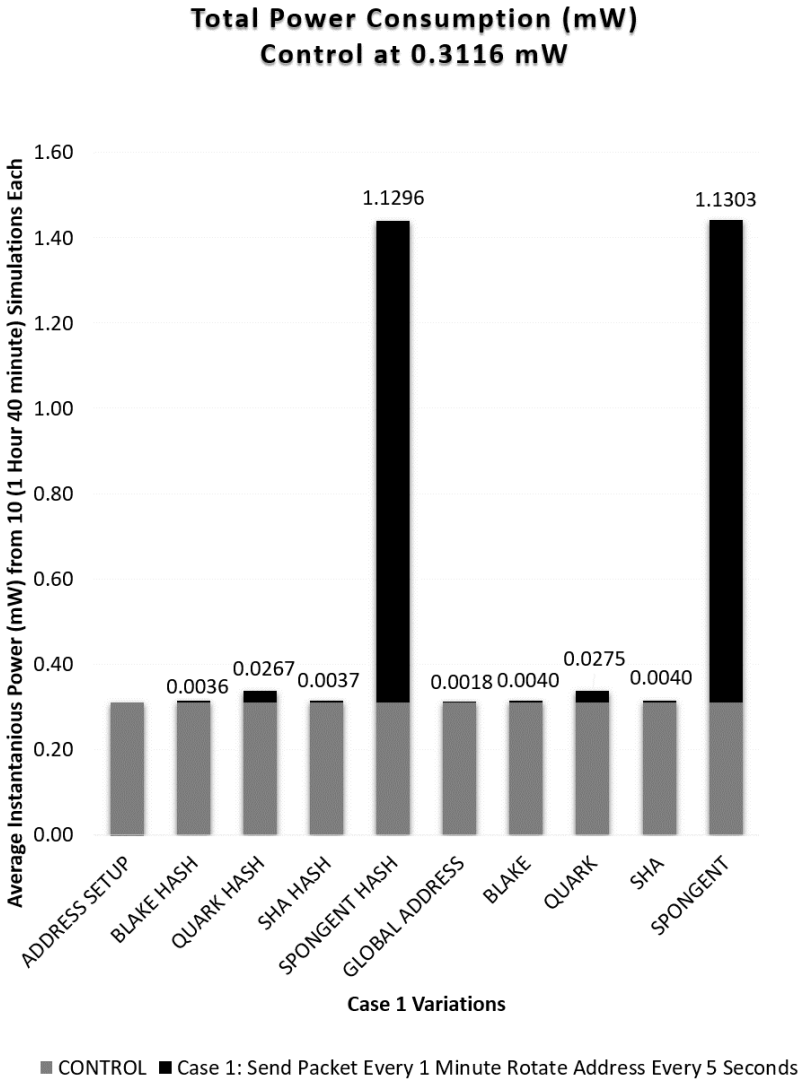

Figure 13.3: Average Total Power Case 1

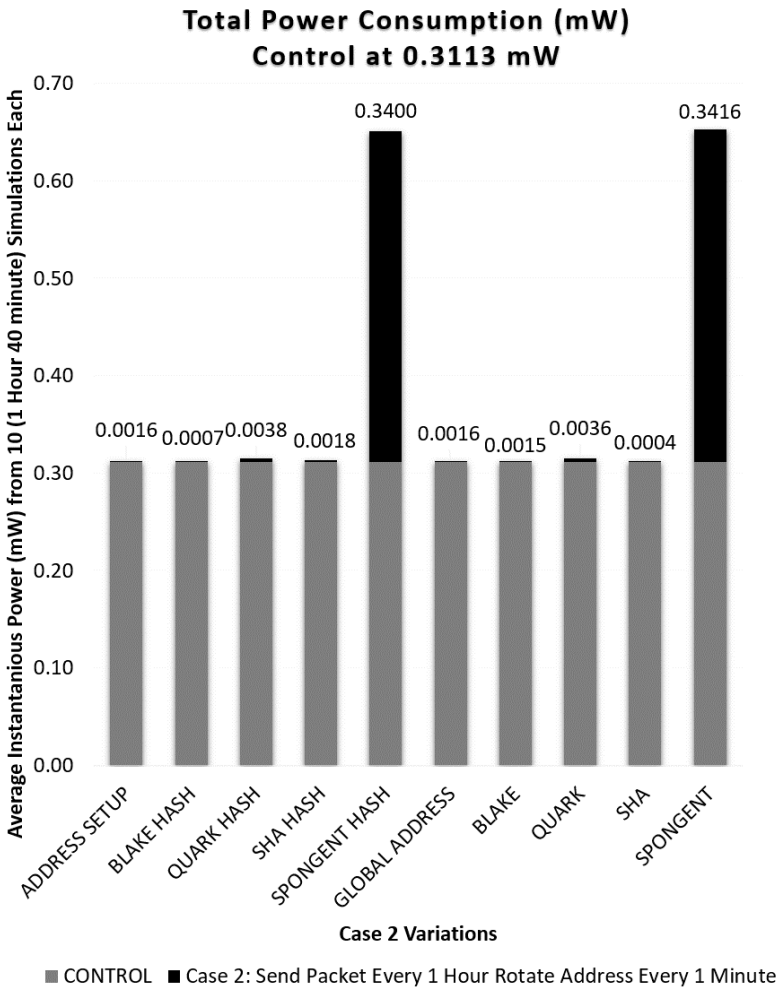

Figure 13.4: Average Total Power Case 2

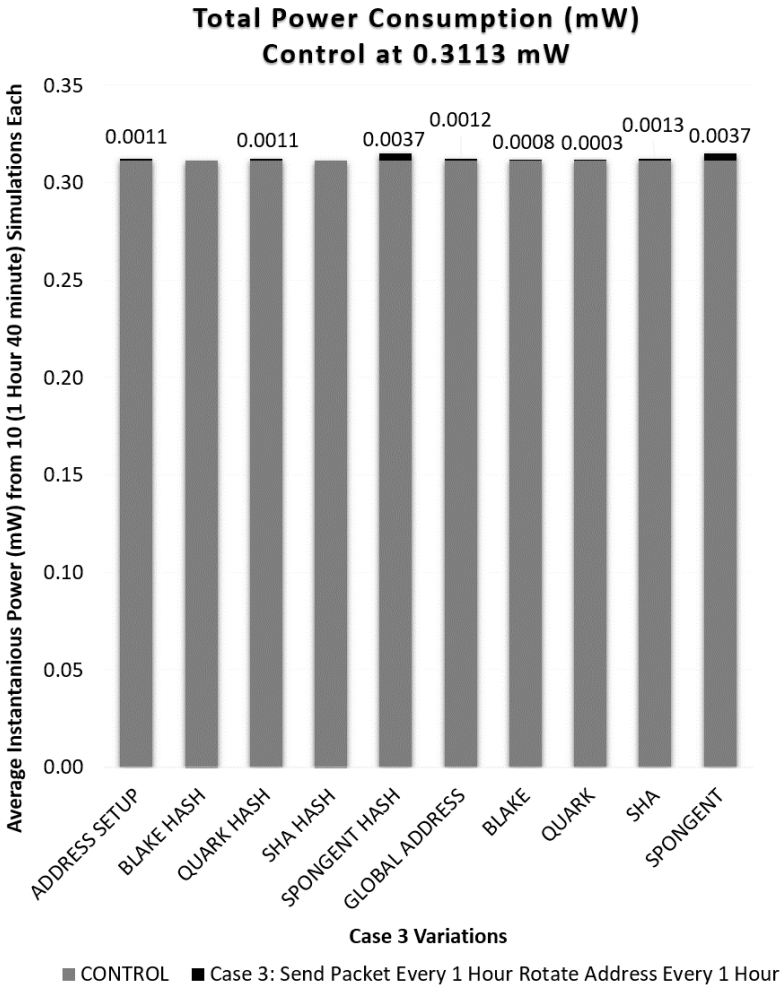

Figure 13.5: Average Total Power Case 3

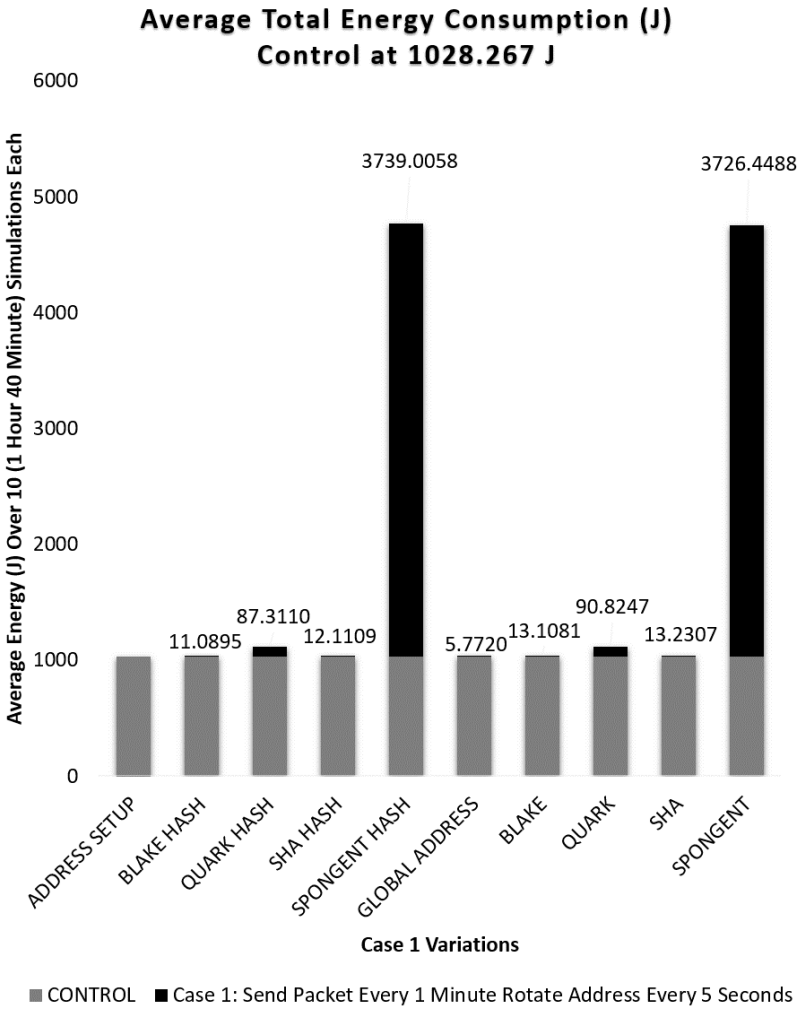

Figure 13.6: Average Total Energy Case 1

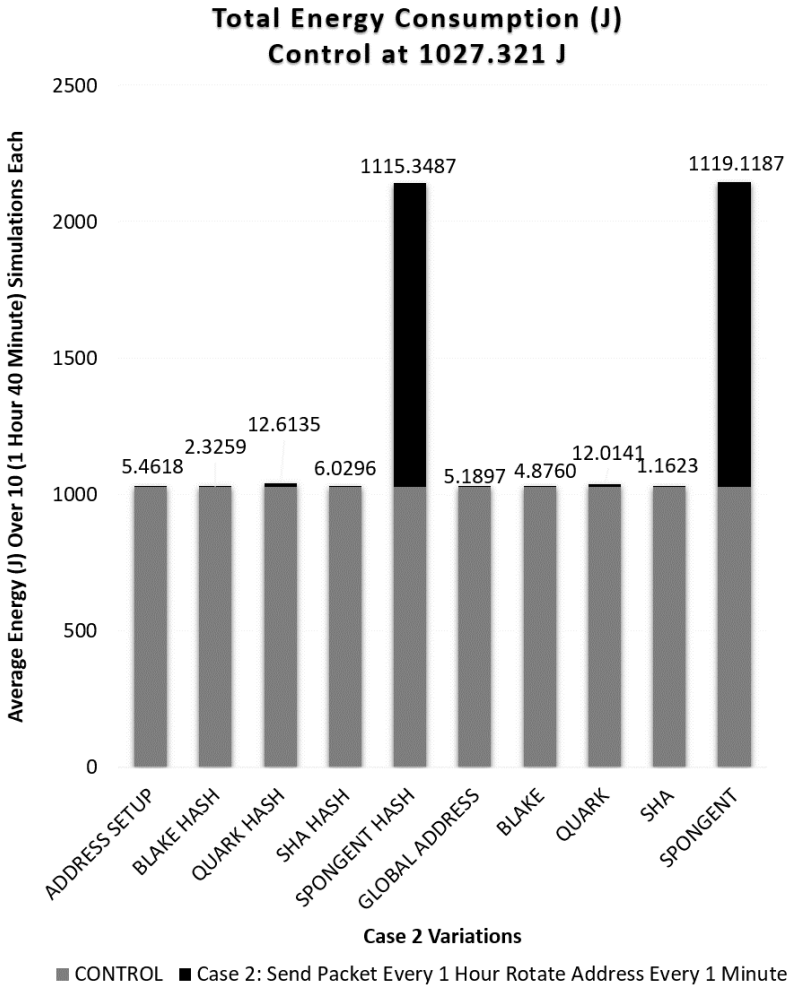

Figure 13.7: Average Total Energy Case 2

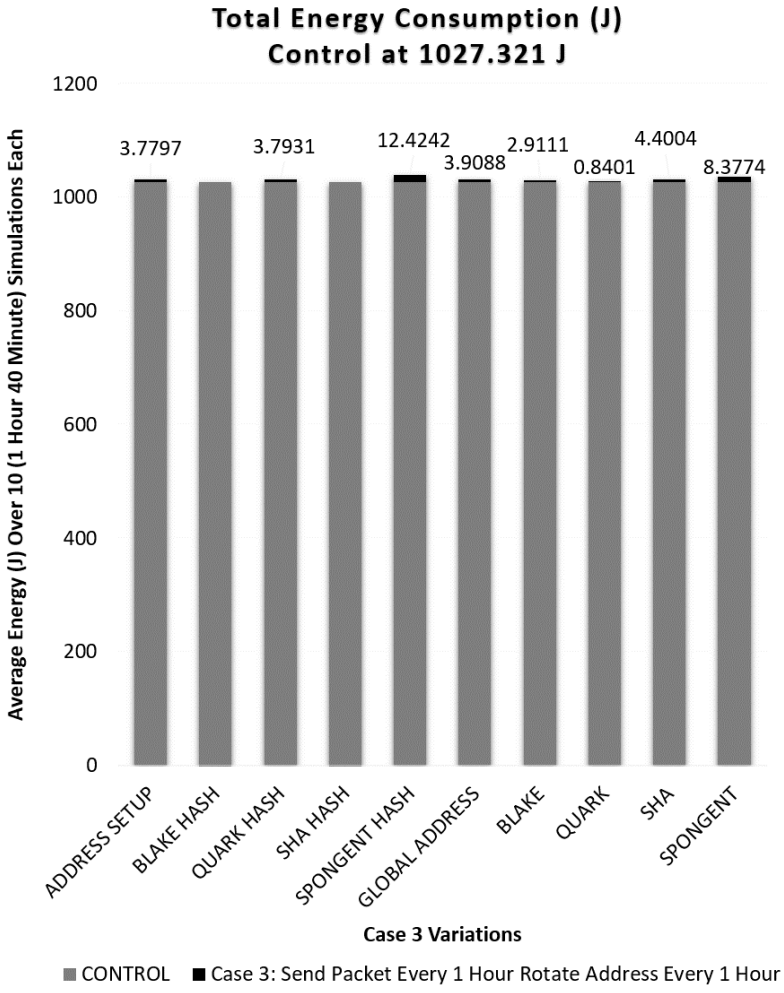

Figure 13.8: Average Total Energy Case 3

Chapter 14

Border Based Simulation Experiment

Initial metrics were gathered to compare the power consumption of the host based and border based designs of μMT6D. The simulations were run over a period of one hour and forty minutes. The total average power consumption for both the border router and host based simulations can be seen in Table 14.1. Overall the host based power overhead was the smallest, but the border based client overhead was also manageable. Having a border router constantly transmitting data after computing the addresses has a high power consumption. The footprint overhead for the border based client is where the greatest benefit of the border based design of μMT6D can be seen.

Figure 14.1 shows the case 1, case 2, and case 3 average power consumption overhead in milliwatts of μMT6D with the lightweight hash function Blake utilized for the hash computations compared to the control of the border router that would respond to requests for an address rotation. Case 1 was every five seconds, case 2 every one minute, and case 3 every hour. The packet send intervals were every minute for case 1 and every hour for cases 2 and 3.

Figure 14.2 shows the average power consumption overhead in milliwatts for a client request-

Table 14.1: Power Consumption Total (mW)

	Case 1	Case 2	Case 3
HOST BASED CONTROL	0.3116	0.3113	0.3113
HOST BASED BLAKE	0.3156	0.3128	0.3121
BORDER BASED CLIENT CONTROL	0.5303	0.1901	0.1901
BORDER BASED CLIENT BLAKE	0.6903	0.2005	0.1925
BORDER BASED ROUTER BLAKE	20.0725	20.0640	20.0628
BORDER BASED ROUTER CONTROL	20.0892	20.0629	20.0629

ing and address of a border router. This version was implemented with μMT6D with Blake utilized for the hash computations compared to the control. All of the cases included in in border router simulations were included. For case 1, this client would request an address every five seconds, for case 2 this client would request an address every minute, and for case three this client would request and address every hour. These were paired with sending packets every minute, and every hour for the last two cases. These simulations also included WSNs with 10, 25, and 50 nodes. The large WSNs added around 4 to 5 times the amount of overhead.

Figure 14.3 shows the average power consumption overhead in milliwatts of μMT6D with Blake utilized for the hash computations in comparison to the control for the host based version. This case would also send a packet every hour for the control and send a packet every hour and rotate the address every one hour for μMT6D. Again, these simulations had address rotation intervals of five seconds, one minute, and one hour. The confidence intervals are given in Table 14.2.

The footprint of these cases allowed for the clients to remain small in comparison to the border router for the border based version. Table 14.3 and Figure 14.4 and 14.5 show the

Table 14.2: 95 % Confidence Intervals in mW

	CASE 1	CASE 2	CASE 3
CONTROL CLIENT	±0.002068	±0.001445	±0.001445
CONTROL SERVER	±0.000171	±0.000069	±0.000069
BLAKE CLIENT	±0.002219	±0.001832	±0.003837
BLAKE SERVER	±0.000132	±0.000097	±0.000225

Table 14.3: Size Data (Bytes)

	text	data	bss	Dec Total	Hex Total
HOST BASED CONTROL	45639	290	8660	54589	d53d
BORDER BASED ROUTER CONTROL	48066	306	7734	56106	db2a
BORDER BASED CLIENT CONTROL	48038	306	7790	56134	db46
BORDER BASED ROUTER BLAKE	48880	328	8600	57808	e1d0
BORDER BASED CLIENT	48954	314	8680	57948	e25c
HOST BASED BLAKE	72987	334	12840	86161	15091

comparisons for these and for the Blake host based case.

Having the client mote communicate with the border router so frequently, especially in the five second rotation interval case led to a high power consumption from the radio transmission state.

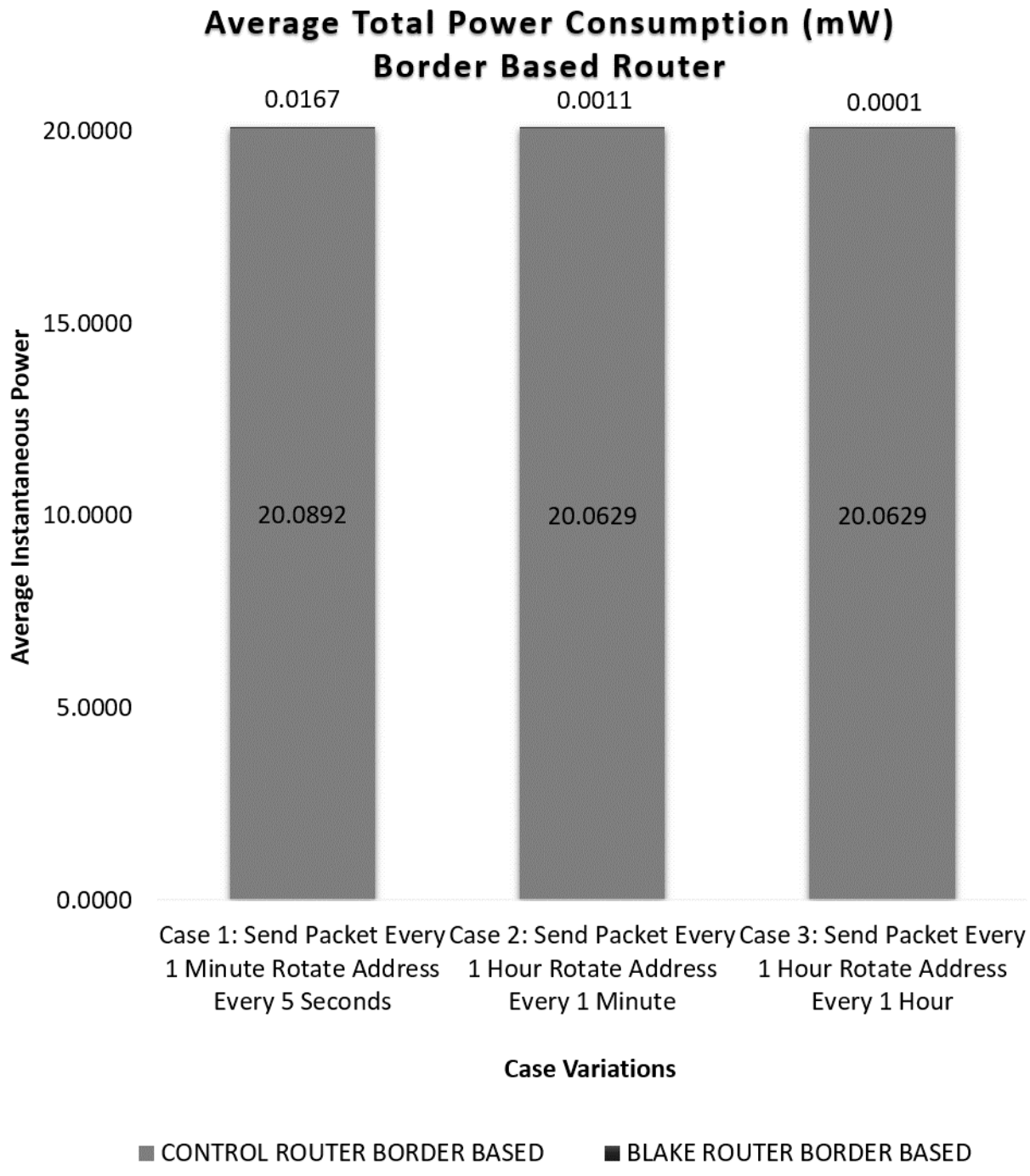

Figure 14.1: Average Total Power Border Based Router

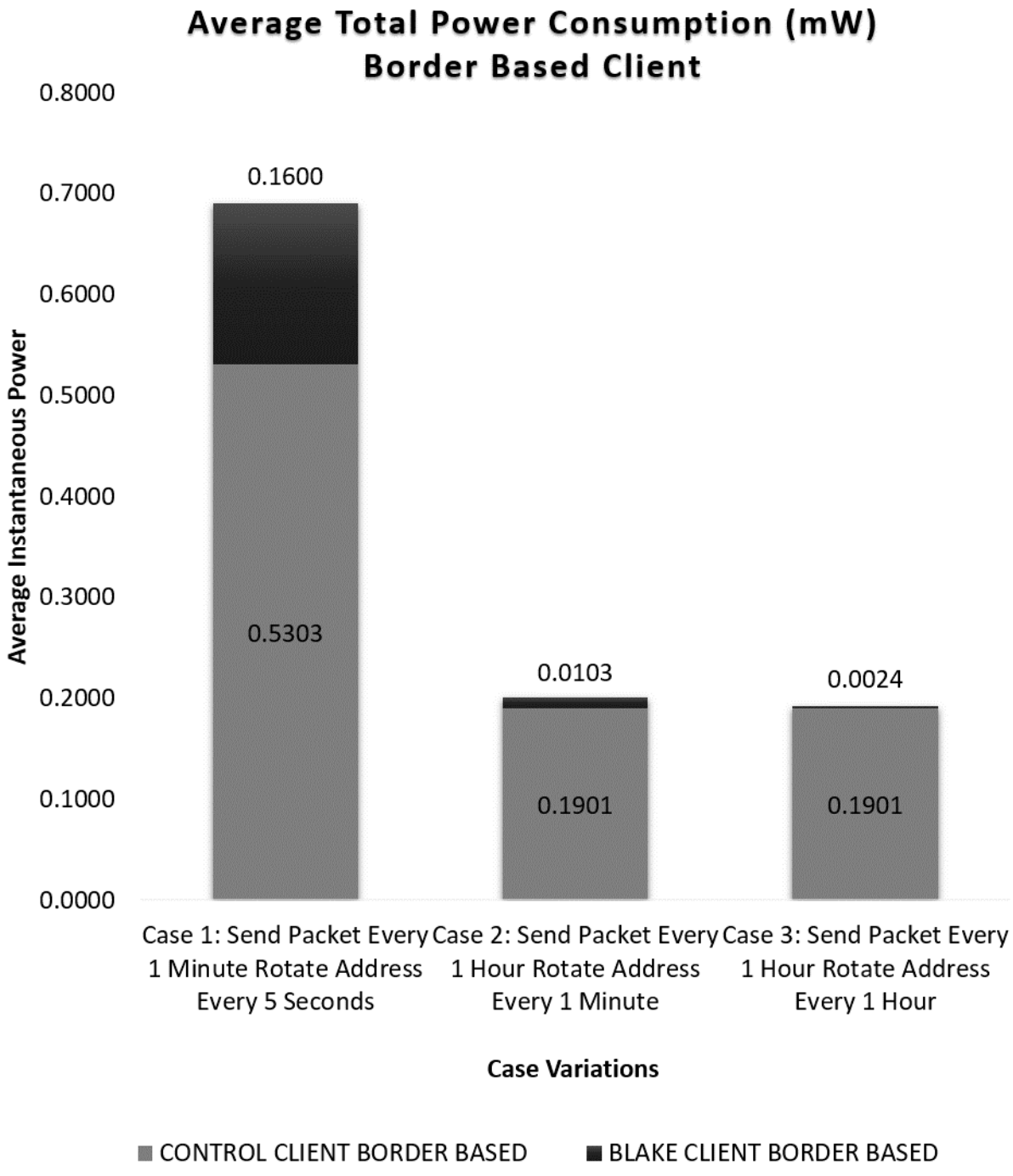

Figure 14.2: Average Total Power Border Based Client

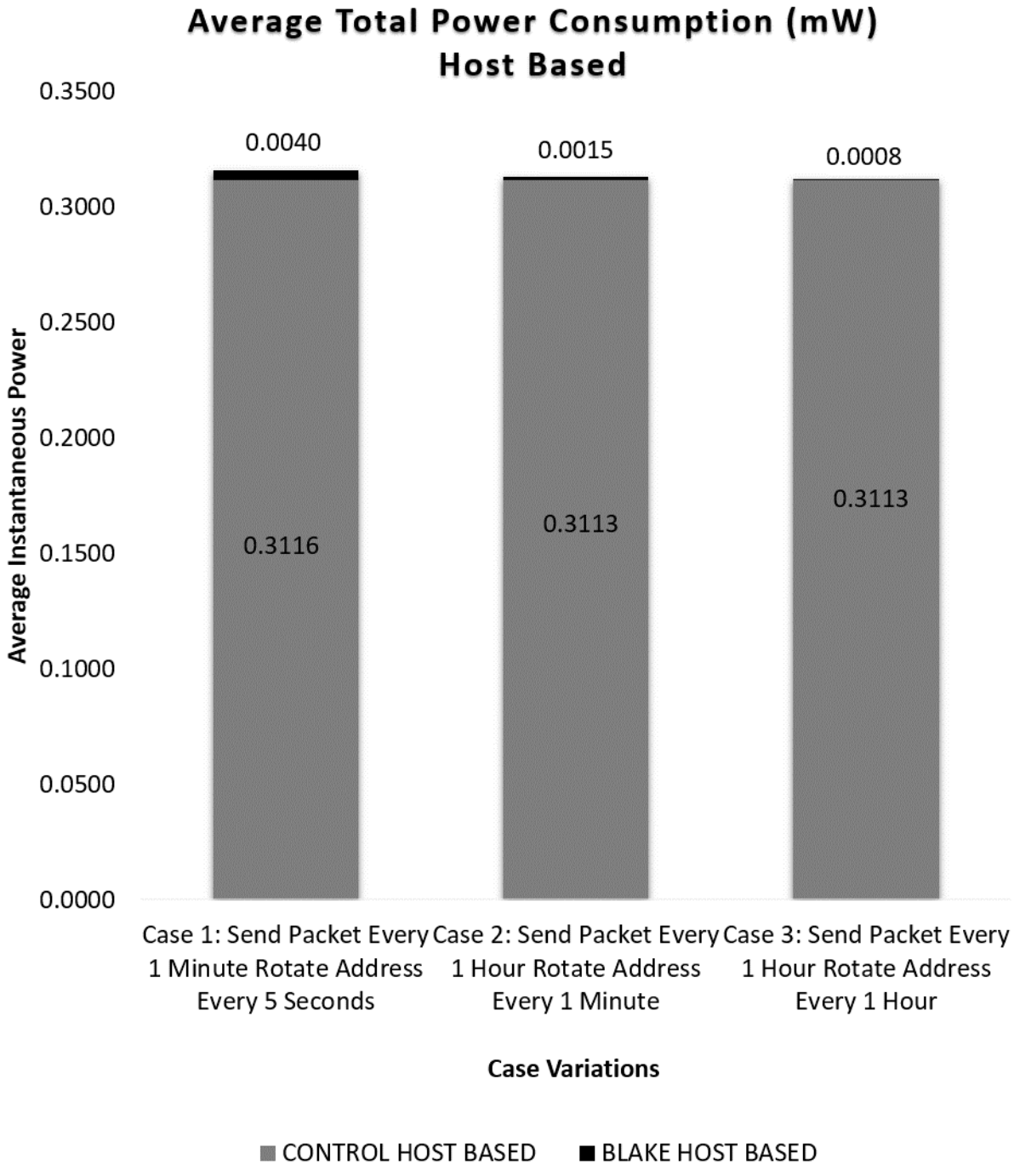

Figure 14.3: Average Total Power Host Based Host

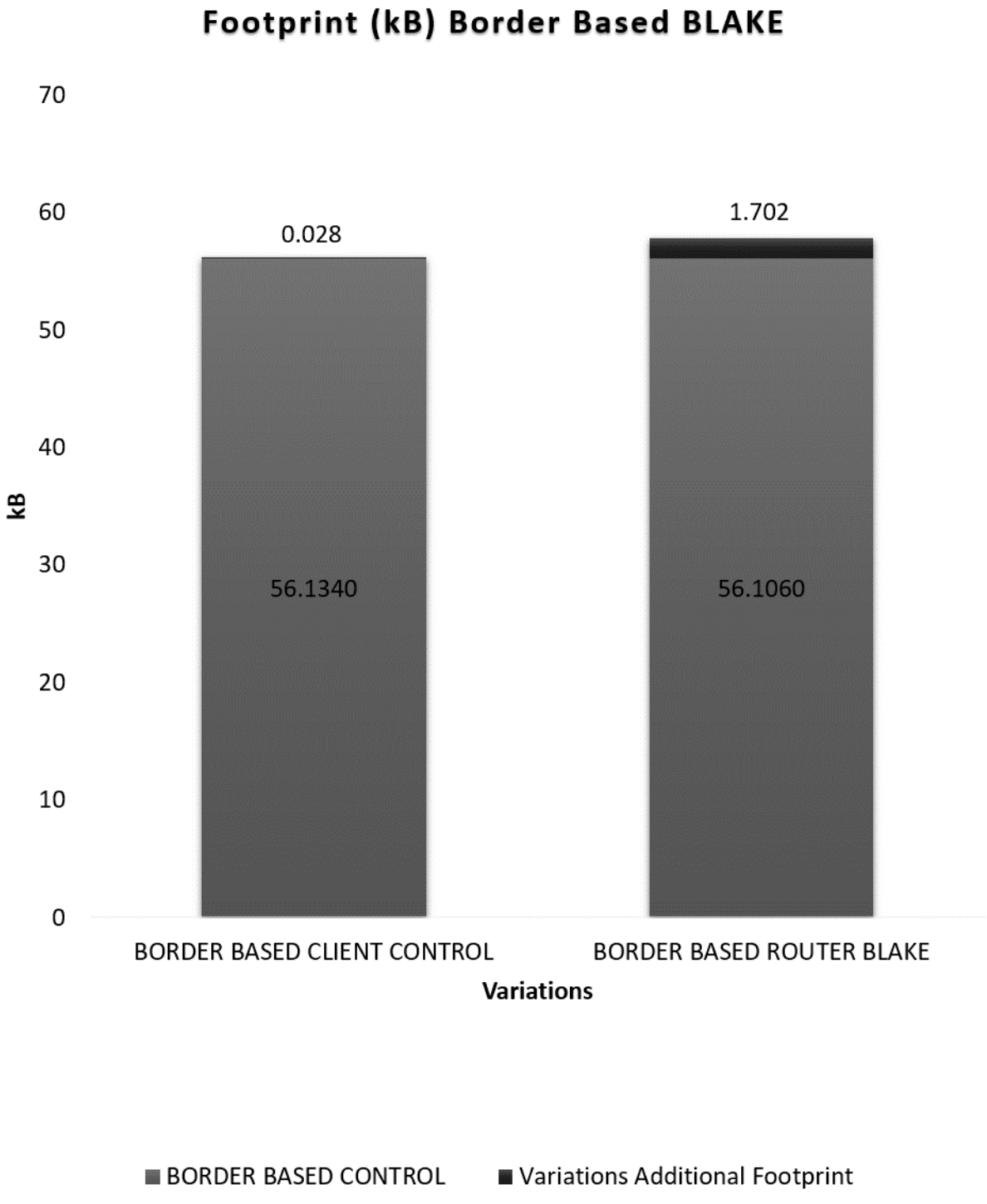

Figure 14.4: Footprint Border Based

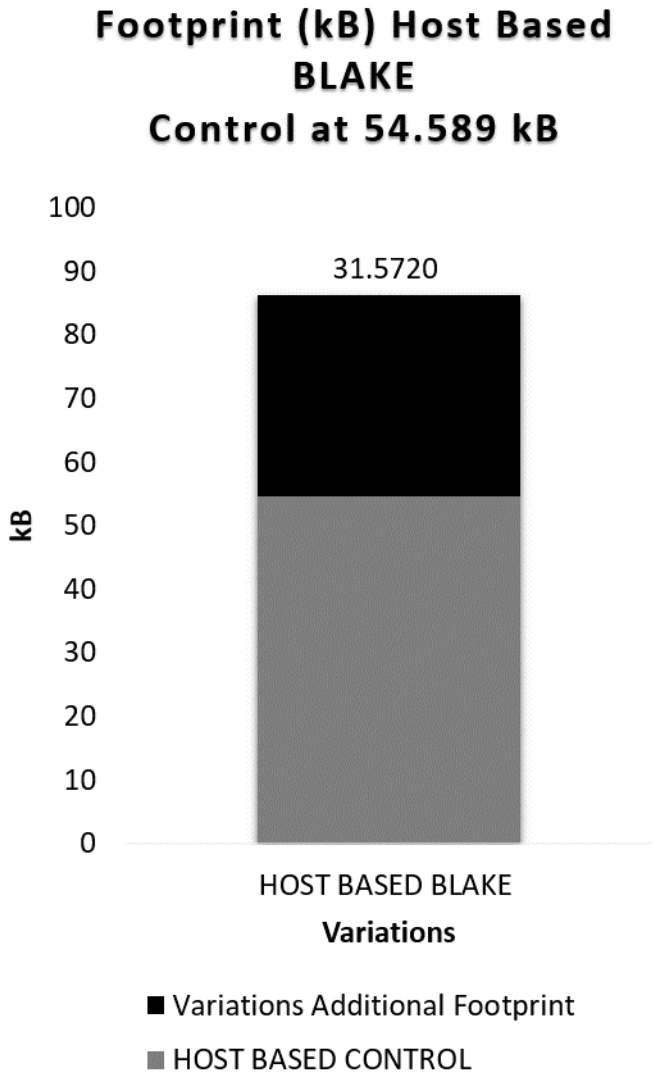

Figure 14.5: Footprint Host Based

Chapter 15

Large Scale Simulation

Initial simulations were run in order to begin to understand what concerns would need to be addressed in regards to scaling of μMT6D. Figure 15.1 shows the case 1 average power consumption overhead in milliwatts of μMT6D with the lightweight hash function Blake utilized for the hash computations compared to the control. This case would send a packet every minute for the control and send a packet every minute and rotate the address every five seconds for Blake across simulations including WSNs with 10, 25, and 50 nodes. The large WSNs added 4.5 to 10 times the amount of overhead.

Figure 15.2 shows the case 2 average power consumption overhead in milliwatts of μMT6D with Blake utilized for the hash computations compared to the control. This case was set to send a packet every hour for the control and send a packet every hour and rotate the address every one minute for μMT6D. These simulations also included WSNs with 10, 25, and 50 nodes. The large WSNs added around 4 to 5 times the amount of overhead.

Figure 15.3 shows the case 3 average power consumption overhead in milliwatts of μMT6D with Blake utilized for the hash computations in comparison to the control. This case would also send a packet every hour for the control and send a packet every hour and rotate the

Table 15.1: 95 % Confidence Intervals in mW

	CASE 1	CASE2	CASE 3
CONTROL 10 Nodes	±0.002353	±0.002202	±0.002202
CONTROL 25 Nodes	±0.003175	±0.003287	±0.003287
CONTROL 50 Nodes	±0.024285	±0.031615	±0.031615
BLAKE 10 Nodes	±0.002303	±0.002157	±0.002147
BLAKE 25 Nodes	±0.126221	±0.060139	±0.003371
BLAKE 50 Nodes	±0.090308	±0.060084	±0.026947

address every one hour for μMT6D. Again, these simulations included WSNs with 10, 25, and 50 nodes. The large WSNs added very little overhead for 25 nodes compared to 10 and about 2 times the amount of overhead with 50 nodes. The confidence intervals can be seen in Table 15.1.

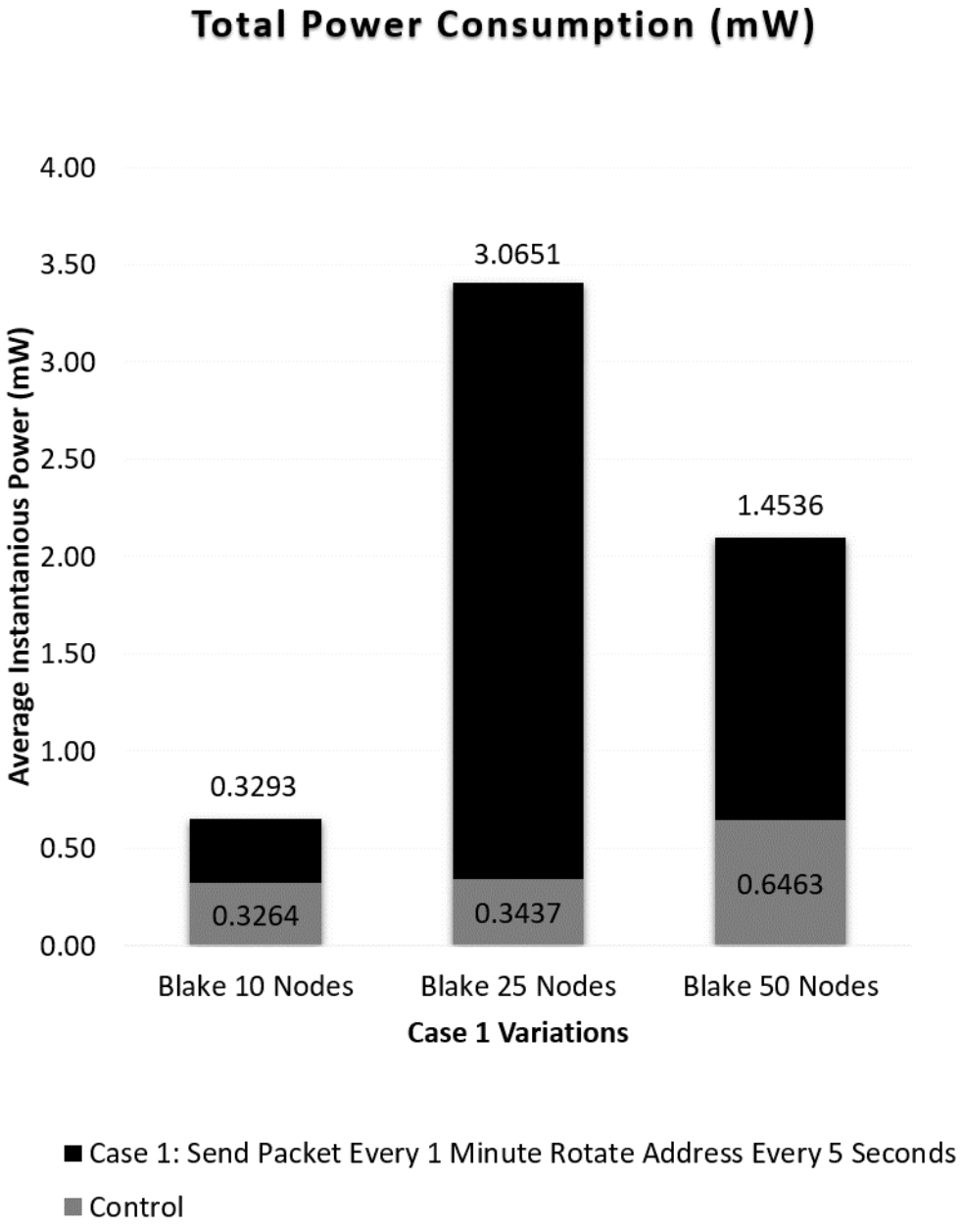

Figure 15.1: Average Total Power Case 1

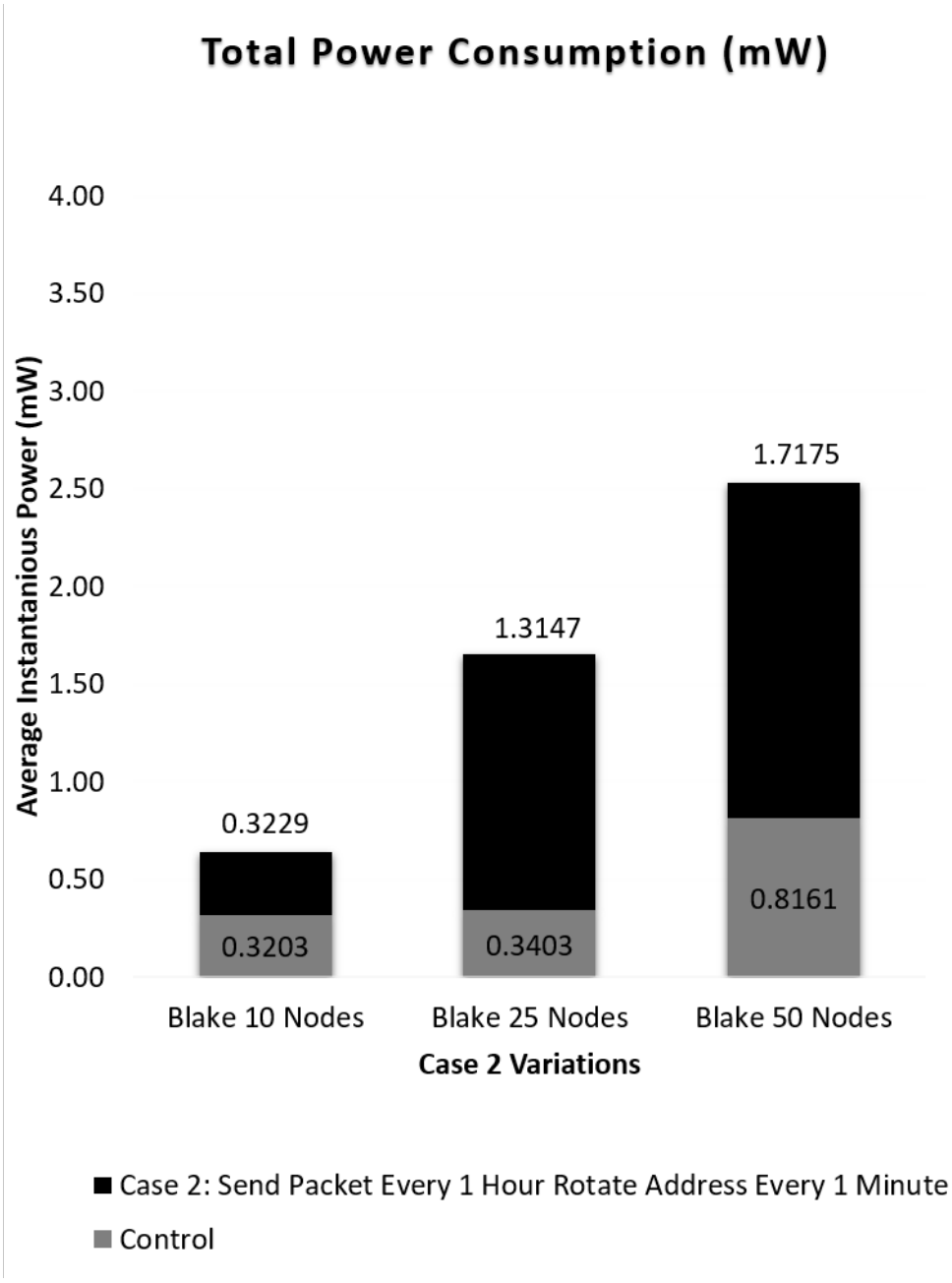

Figure 15.2: Average Total Power Case 2

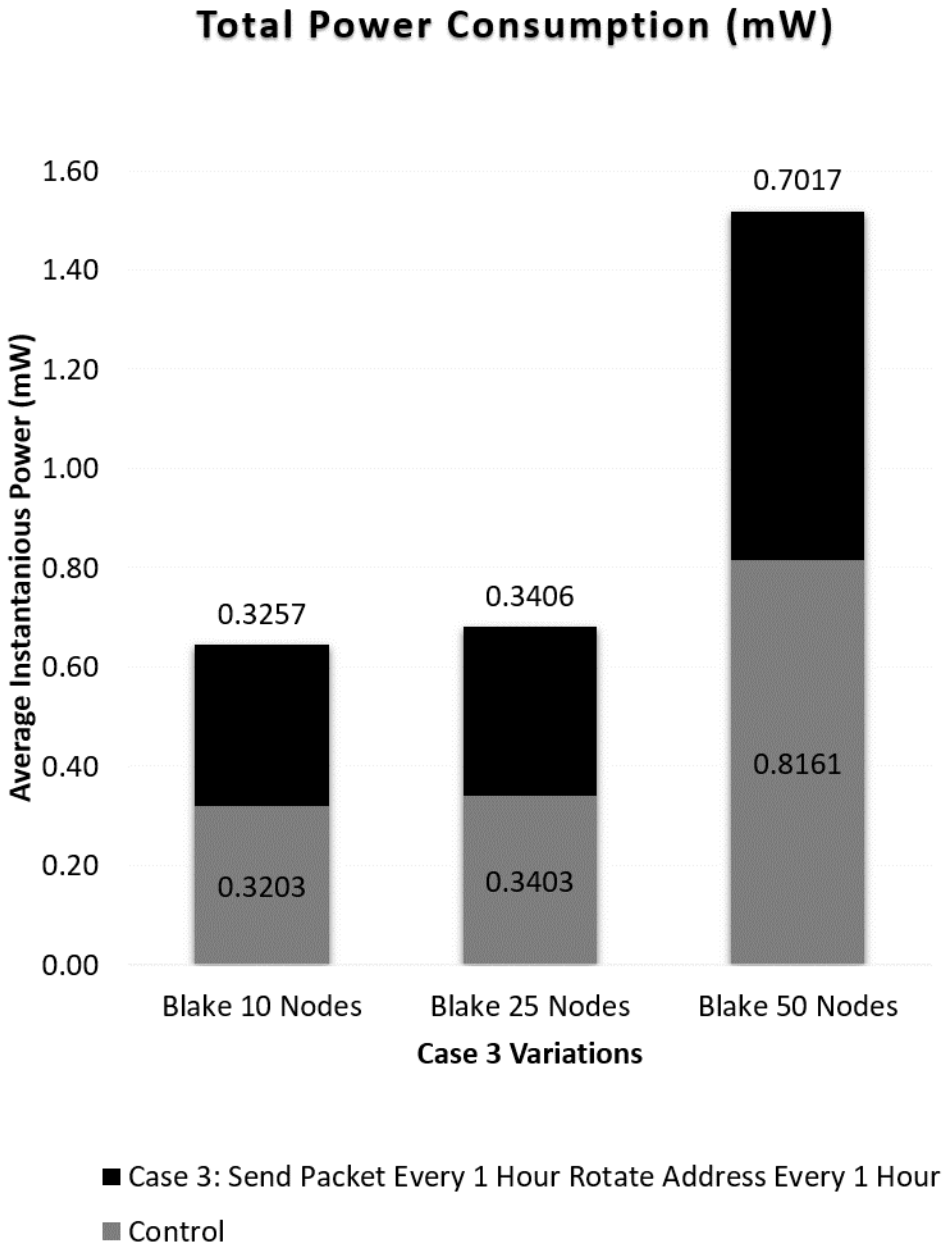

Figure 15.3: Average Total Power Case 3

Chapter 16

Future Work

There is more future work that could further this research and aid in answering the question of "Is a moving target defense methodology a viable defense for limiting reconnaissance time and thwarting attacks for IoT devices?" As touched on in Chapter 9, scaling is one factor which could be considered even further. Simulations and physical hardware metrics were utilized for this research, but actual utilization of an IoT application that is employed on a live network could be gathered and analyzed. Another area to explore includes the security implications of utilizing the different lightweight hashing algorithms. The ability to switch and select the algorithm needs to be paired with research into any resulting factors and/or security issues which may be linked to this change. This future research would be more dependent on the lightweight hash algorithms themselves. In addition, research can be done to test both modes of operation of μMT6D and compare their use in communication with an MT6D server. This will be important as new technologies and techniques are developed and different lightweight hash algorithms are added to the hash library. Research on traffic analysis and the structural characteristics of IPv6 addresses could be considered as another project to ensure μMT6D is not vulnerable to an attacker gleaning any information which

could compromise the motes or information [45]. This work focused on footprint and power and energy consumption and these could be furthered with more cases from other hash functions or address rotation intervals, or the addition of data about packet throughput or session establishment with a server. Randomizing the rotation intervals could also be explored. In regards to computing the hash function, one method to be explored is the sharing of the computational load between different nodes to compute the addresses. Finally, this work could be combined with the client/server MT6D and a method implemented to expire session keys to keep the WSN connected to a large scale system securely. There are many avenues for furthering research in this area and adding knowledge to the MT6D and Micro MT6D timeline which can provide security against targeted attacks.

Chapter 17

Conclusion

This research has outlined the design, optimizations, and analysis of simulations and experiments for a Micro-Moving Target IPv6 defense, μMT6D. This included initial metric gathering of footprint and power over simulations of twenty four and eight hours considering many WSN devices are left unattended for prolonged periods of time. The first experiment utilized μMT6D with one rotation interval set and with SHA256 compared to a control to see initial feasibility moving forward. Next, an experiment was conducted to test three address rotation interval times, two packet send interval times, and four different lightweight hash functions. A total of 14 simulations were run in this second experiment. The proof of concept application for all cases fit on the WiSMote and footprints were reported along with power consumption data. These, however, used the smallest Contiki Radio Duty Cycle in terms of footprint, but the radio remained on which increases the power consumption. Then, a detailed experiment was done of the different components of μMT6D. This refined simulation included all the four hash functions alone and also as part of μMT6D. It looked at three address rotation intervals and two packet send intervals. In addition, the Radio Duty Cycle chosen was the ContikiMAC driver that includes logic for turning the radio on and

off to allow for power saving. All of these cases and variations were included in simulations lasting an hour and forty minutes and replicated ten times based on a computed steady state for data gathering start time and length. This was a total of 32 simulations run to ensure a 95 % certainty that the power and energy measurements would fall within ± 0.01 of the mean power and energy. Finally, two more initial experiments were done. The first began to explore the power consumption of the host based μMT6D when the WSN is scaled from ten to fifty motes with a control compared to the Blake hash function μMT6D for the different address rotation intervals and packet send times. The simulations were also one hour and forty minutes. The second, followed the same set up for cases and variations except that the proof of concept implementation was for the border router based μMT6D in which a an RPL Border Router did the address hash computation and would send the data to the host nodes for the address change. All of these experiments showed that μMT6D had a footprint that met the size requirements for the WiSMote and power and energy overheads that would be acceptable considering the often required benefits of the security techniques.

Further, the flexibility to utilize different lightweight hash algorithms will greatly enhance the potential for the use of μMT6D with different applications based on the varying performance and size constraints, but still within the realm of resource constrained embedded systems as seen by the footprint and power and energy analysis simulations just discussed. Also, it allows for this security technique to be customized based on the selected hash algorithm as future ones are released. The ability to control the address rotation interval is also important as seen with the savings in power and energy consumption for applications that can accept

the risk of a large window in which an attacker can conduct reconnaissance. The described host-based and also border-based modes of operation showed the suitability of μMT6D for different applications of resource constrained devices with smaller footprint and power and energy consumption overheads for the host devices and allowing for the router which may have less constraints to hold the largest overhead. Ultimately, this research has proven that the use of a Micro Moving Target IPv6 Defense to limit the time an attacker has to conduct reconnaissance and therefore prevent targeted attacks is viable for IoT applications designed for low-power and low-resource embedded devices.

Bibliography

[1] M. A. Abdelraheem, C. Blondeau, M. Naya-Plasencia, M. Videau, and E. Zenner. Cryptanalysis of armadillo2. In *Advances in Cryptology–ASIACRYPT 2011*, pages 308–326. Springer, 2011.

[2] H. Abie and I. Balasingham. Risk-based adaptive security for smart iot in ehealth. In *Proceedings of the 7th International Conference on Body Area Networks*, pages 269–275. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2012.

[3] S. Agrawal and M. L. Das. Internet of things a paradigm shift of future internet applications. In *Engineering (NUiCONE), 2011 Nirma University International Conference on*, pages 1–7. IEEE, 2011.

[4] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials*, 17(4):2347–2376, 2015.

[5] E. Al-Shaer. Toward network configuration randomization for moving target defense. In *Moving Target Defense*, pages 153–159. Springer, 2011.

[6] M. Albanese, A. De Benedictis, S. Jajodia, and K. Sun. A moving target defense mech-

anism for manets based on identity virtualization. In *Communications and Network Security (CNS), 2013 IEEE Conference on*, pages 278–286. IEEE, 2013.

[7] I. Alqassem. Privacy and security requirements framework for the internet of things (iot). In *Companion Proceedings of the 36th International Conference on Software Engineering*, pages 739–741. ACM, 2014.

[8] S. Antonatos, P. Akritidis, E. P. Markatos, and K. G. Anagnostakis. Defending against hitlist worms using network address space randomization. *Computer Networks*, 51(12):3471–3490, 2007.

[9] K. Ashton. That internet of things thing. *RFiD Journal*, 22(7):97–114, 2009.

[10] L. Atzori, A. Iera, and G. Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[11] J.-P. Aumasson, L. Henzen, W. Meier, and M. Naya-Plasencia. Quark: A lightweight hash. *Journal of cryptology*, 26(2):313–339, 2013.

[12] J.-P. Aumasson, L. Henzen, W. Meier, and R. C.-W. Phan. Sha-3 proposal blake. *Submission to NIST*, 2008.

[13] J.-P. Aumasson, S. Neves, Z. Wilcox-O'Hearn, and C. Winnerlein. Blake2: simpler, smaller, fast as md5. In *Applied Cryptography and Network Security*, pages 119–135. Springer, 2013.

[14] S. Badel, N. Dağtekin, J. Nakahara Jr, K. Ouafi, N. Reffé, P. Sepehrdad, P. Sušil, and S. Vaudenay. Armadillo: a multi-purpose cryptographic primitive dedicated to hardware. In *Cryptographic Hardware and Embedded Systems, CHES 2010*, pages 398–412. Springer, 2010.

[15] I. E. Bagci, S. Raza, T. Chung, U. Roedig, and T. Voigt. Combined secure storage and communication for the internet of things. In *Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2013 10th Annual IEEE Communications Society Conference on*, pages 523–531. IEEE, 2013.

[16] J. Balasch, B. Ege, T. Eisenbarth, B. Gérard, Z. Gong, T. Güneysu, S. Heyse, S. Kerckhof, F. Koeune, T. Plos, et al. Compact implementation and performance evaluation of hash functions in attiny devices. In *International Conference on Smart Card Research and Advanced Applications*, pages 158–172. Springer, 2012.

[17] D. Basam, R. Marchany, and J. G. Tront. Attention: moving target defense networks, how well are you moving? In *Proceedings of the 12th ACM International Conference on Computing Frontiers*, page 54. ACM, 2015.

[18] D. K. Basam. *Strengthening MT6D Defenses with Darknet and Honeypot capabilities*. PhD thesis, Virginia Tech, 2015.

[19] N. Benamar, A. Jara, L. Ladid, and D. El Ouadghiri. Challenges of the internet of things: Ipv6 and network management. In *2014 Eighth International Conference*

on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), pages 328–333. IEEE, 2014.

[20] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. Sponge functions. In *ECRYPT hash workshop*, volume 2007. Citeseer, 2007.

[21] A. Bogdanov, M. Knežević, G. Leander, D. Toz, K. Varıcı, and I. Verbauwhede. Spongent: A lightweight hash function. In *Cryptographic Hardware and Embedded Systems–CHES 2011*, pages 312–325. Springer, 2011.

[22] A. Bogdanov, M. Knežević, G. Leander, D. Toz, K. Varici, and I. Verbauwhede. Spongent: The design space of lightweight cryptographic hashing. *Computers, IEEE Transactions on*, 62(10):2041–2053, 2013.

[23] R. Bonetto, N. Bui, V. Lakkundi, A. Olivereau, A. Serbanati, and M. Rossi. Secure communication for smart iot objects: Protocol stacks, use cases and practical examples. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2012 IEEE International Symposium on a*, pages 1–7. IEEE, 2012.

[24] G. Brambilla, M. Picone, S. Cirani, M. Amoretti, and F. Zanichelli. A simulation platform for large-scale internet of things scenarios in urban environments. In *Proceedings of the First International Conference on IoT in Urban Space*, pages 50–55. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2014.

[25] T. K. Buennemeyer, R. C. Marchany, and J. G. Tront. Ubiquitous security: Privacy versus protection. 2006.

[26] R. Cabral and J. López. Fast software implementation of quark on a 32-bit architecture. In *International Workshop on Lightweight Cryptography for Security and Privacy*, pages 115–130. Springer, 2015.

[27] A. Chavez, J. Hamlet, E. Lee, M. Martin, and W. Stout. Network randomization and dynamic defense for critical infrastructure systems. *Sandia National Laboratories ReportSAND2015-3324 (April 2015)*, 277:13, 2015.

[28] C. Chen and S. Helal. A device-centric approach to a safer internet of things. In *Proceedings of the 2011 international workshop on Networking and object memories for the internet of things*, pages 1–6. ACM, 2011.

[29] P. L. R. Chze and K. S. Leong. A secure multi-hop routing for iot communication. In *Internet of Things (WF-IoT), 2014 IEEE World Forum on*, pages 428–432. IEEE, 2014.

[30] A. Clark, K. Sun, L. Bushnell, and R. Poovendran. A game-theoretic approach to ip address randomization in decoy-based cyber defense. In *International Conference on Decision and Game Theory for Security*, pages 3–21. Springer, 2015.

[31] A. Clark, K. Sun, and R. Poovendran. Effectiveness of ip address randomization in decoy-based moving target defense. In *52nd IEEE Conference on Decision and Control*, pages 678–685. IEEE, 2013.

[32] B. Clore, M. Dunlop, R. Marchany, and J. Tront. Validating a custom ipv6 security application using opnet modeler. In *MILCOM 2012-2012 IEEE Military Communications Conference*, pages 1–6. IEEE, 2012.

[33] C. S. Currie and R. C. Cheng. A practical introduction to analysis of simulation output data. In *2016 Winter Simulation Conference (WSC)*, pages 118–132. IEEE, 2016.

[34] L. Deru, S. Dawans, M. Ocaña, B. Quoitin, and O. Bonaventure. Redundant border routers for mission-critical 6lowpan networks. In *Real-world wireless sensor networks*, pages 195–203. Springer, 2014.

[35] P. L. DiMarco. *Evaluation of Moving Target IPv6 Defense and Distributed Denial of Service Defenses*. PhD thesis, Virginia Tech, 2013.

[36] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram. Blockchain for iot security and privacy: The case study of a smart home. In *Pervasive Computing and Communications Workshops (PerCom Workshops), 2017 IEEE International Conference on*, pages 618–623. IEEE, 2017.

[37] A. Dunkels. Contiki: The open source os for the internet of things, 2012.

[38] A. Dunkels, J. Eriksson, N. Finne, F. Österlind, N. Tsiftes, J. Abeillé, and M. Durvy. Low-power ipv6 for the internet of things. In *Networked Sensing Systems (INSS), 2012 Ninth International Conference on*, pages 1–6. IEEE, 2012.

[39] A. Dunkels, J. Eriksson, N. Finne, and N. Tsiftes. Powertrace: Network-level power profiling for low-power wireless networks, 2011.

[40] M. Dunlop. *Achieving Security and Privacy in the Internet Protocol Version 6 Through the Use of Dynamically Obscured Addresses*. PhD thesis, Virginia Tech, 2012.

[41] M. Dunlop, S. Groat, R. Marchany, and J. Tront. The good, the bad, the ipv6. In *Communication Networks and Services Research Conference (CNSR), 2011 Ninth Annual*, pages 77–84. IEEE, 2011.

[42] M. Dunlop, S. Groat, R. Marchany, and J. Tront. Ipv6: now you see me, now you don't. In *International Conference on Networks (ICN)*, pages 18–23, 2011.

[43] M. Dunlop, S. Groat, W. Urbanski, R. Marchany, and J. Tront. Mt6d: A moving target ipv6 defense. In *Military Communications Conference, 2011-Milcom 2011*, pages 1321–1326. IEEE, 2011.

[44] M. Dunlop, S. Groat, W. Urbanski, R. Marchany, and J. Tront. The blind man's bluff approach to security using ipv6. *Security & Privacy, IEEE*, 10(4):35–43, 2012.

[45] P. Foremski, D. Plonka, and A. Berger. Entropy/ip: Uncovering structure in ipv6 addresses. *arXiv preprint arXiv:1606.04327*, 2016.

[46] O. Garcia-Morchon, S. L. Keoh, S. Kumar, P. Moreno-Sanchez, F. Vidal-Meca, and J. H. Ziegeldorf. Securing the ip-based internet of things with hip and dtls. In *Pro-*

ceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks, pages 119–124. ACM, 2013.

[47] A. Gluhak, S. Krco, M. Nati, D. Pfisterer, N. Mitton, and T. Razafindralambo. A survey on facilities for experimental internet of things research. *IEEE Communications Magazine*, 49(11):58–67, 2011.

[48] F. Gont. A method for generating semantically opaque interface identifiers with ipv6 stateless address autoconfiguration (slaac). Technical report, 2014.

[49] F. Gosset, F.-X. Standaert, J.-J. Quisquater, et al. Fpga implementation of squash. In *Proceedings of the 29th Symposium on Information Theory in the Benelux*, 2008.

[50] J. Granjal, E. Monteiro, and J. Sa Silva. End-to-end transport-layer security for internet-integrated sensing applications with mutual and delegated ecc public-key authentication. In *IFIP Networking Conference, 2013*, pages 1–9. IEEE, 2013.

[51] D. Green, R. Mayo, and R. Reddy. Ipv6 application performance characterization using a virtual/live testbed. In *MILCOM 2006-2006 IEEE Military Communications conference*, pages 1–4. IEEE, 2006.

[52] M. Green, D. C. MacFarland, D. R. Smestad, and C. A. Shue. Characterizing network-based moving target defenses. In *Proceedings of the Second ACM Workshop on Moving Target Defense*, pages 31–35. ACM, 2015.

[53] S. Groat, M. Dunlop, R. Marchany, and J. Tront. The privacy implications of stateless

ipv6 addressing. In *Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research*, page 52. ACM, 2010.

[54] J. Guo, T. Peyrin, and A. Poschmann. The photon family of lightweight hash functions. In *Advances in Cryptology–CRYPTO 2011*, pages 222–239. Springer, 2011.

[55] O. R. Hardman. *Optimizing a network layer moving target defense by translating software from python to c.* PhD thesis, Virginia Tech, 2016.

[56] V. Heydari and S.-M. Yoo. Moving target defense enhanced by mobile ipv6.

[57] R. Hinden and S. Deering. Ip version 6 addressing architecture. Technical report, 2006.

[58] T. Instrument. Msp430f543x and msp430f541x mixed-signal microcontrollers. *MSP430F5418A datasheet, January*, 2010.

[59] T. Instruments. Cc2520 datasheet: 2.4 ghz ieee 802.15. 4/zigbee rf transceiver. *SWRS068*, 2007.

[60] J. H. Jafarian, E. Al-Shaer, and Q. Duan. Openflow random host mutation: transparent moving target defense using software defined networking. In *Proceedings of the first workshop on Hot topics in software defined networks*, pages 127–132. ACM, 2012.

[61] R. Jain. *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling.* John Wiley & Sons, 1990.

[62] S. Jajodia, A. K. Ghosh, V. Swarup, C. Wang, and X. S. Wang. *Moving target defense:*

creating asymmetric uncertainty for cyber threats, volume 54. Springer Science & Business Media, 2011.

[63] Q. Jia, K. Sun, and A. Stavrou. Motag: Moving target defense against internet denial of service attacks. In *2013 22nd International Conference on Computer Communication and Networks (ICCCN)*, pages 1–9. IEEE, 2013.

[64] Q. Jia, H. Wang, D. Fleck, F. Li, A. Stavrou, and W. Powell. Catch me if you can: A cloud-enabled ddos defense. In *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 264–275. IEEE, 2014.

[65] B. Jungk, L. R. Lima, and M. Hiller. A systematic study of lightweight hash functions on fpgas. In *2014 International Conference on ReConFigurable Computing and FPGAs (ReConFig14)*, pages 1–6. IEEE, 2014.

[66] A. Kanuparthi, R. Karri, and S. Addepalli. Hardware and embedded security in the context of internet of things. In *Proceedings of the 2013 ACM workshop on Security, privacy & dependability for cyber vehicles*, pages 61–64. ACM, 2013.

[67] P. Kasinathan, C. Pastrone, M. A. Spirito, and M. Vinkovits. Denial-of-service detection in 6lowpan based internet of things. In *WiMob*, pages 600–607, 2013.

[68] S. L. Keoh, S. S. Kumar, and H. Tschofenig. Securing the internet of things: A standardization perspective. *Internet of Things Journal, IEEE*, 1(3):265–275, 2014.

[69] D. Kewley, R. Fink, J. Lowry, and M. Dean. Dynamic approaches to thwart adversary

intelligence gathering. In *DARPA Information Survivability Conference & Exposition II, 2001. DISCEX'01. Proceedings*, volume 1, pages 176–185. IEEE, 2001.

[70] N. Khan, N. Sakib, I. Jerin, S. Quader, and A. Chakrabarty. Performance analysis of security algorithms for iot devices. In *Humanitarian Technology Conference (R10-HTC), 2017 IEEE Region 10*, pages 130–133. IEEE, 2017.

[71] J. Ko, A. Terzis, S. Dawson-Haggerty, D. E. Culler, J. W. Hui, and P. Levis. Connecting low-power and lossy networks to the internet. *IEEE Communications Magazine*, 49(4):96–101, 2011.

[72] P. Koshy, J. Valentin, and X. Zhang. Implementation and performance testing of the squash rfid authentication protocol. In *Applications and Technology Conference (LISAT), 2010 Long Island Systems*, pages 1–5. IEEE, 2010.

[73] A. Kumar and A. Aggarwal. Lightweight cryptographic primitives for mobile ad hoc networks. In *International Conference on Security in Computer Networks and Distributed Systems*, pages 240–251. Springer, 2012.

[74] A. M. Law and W. D. Kelton. *Simulation modeling and analysis*, volume 3. McGraw-Hill New York, 2000.

[75] D. C. MacFarland and C. A. Shue. The sdn shuffle: creating a moving-target defense using host-based software-defined networking. In *Proceedings of the Second ACM Workshop on Moving Target Defense*, pages 37–41. ACM, 2015.

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

[77] Y. Maleh, A. Ezzati, and M. Belaissaoui. An enhanced dtls protocol for internet of things applications. In *2016 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, pages 168–173. IEEE, 2016.

[78] G. S. Matharu, P. Upadhyay, and L. Chaudhary. The internet of things: Challenges & security issues. In *Emerging Technologies (ICET), 2014 International Conference on*, pages 54–59. IEEE, 2014.

[79] K. A. McKay, L. Bassham, M. S. Turan, and N. Mouha. Report on lightweight cryptography. *NISTIR*, 8114, 2017.

[80] J. Michalski, C. Price, E. Stanton, E. Lee, K. Chua, Y. Wong, and C. Tan. Network security mechanisms utilizing dynamic network address translation, 2002.

[81] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A.-R. Sadeghi, and S. Tarkoma. Iot sentinel: Automated device-type identification for security enforcement in iot. In *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*, pages 2177–2184. IEEE, 2017.

[82] R. Moore, C. Morrell, R. Marchany, and J. G. Tront. Utilizing the bittorrent dht for

blind rendezvous and information exchange. In *Military Communications Conference, MILCOM 2015-2015 IEEE*, pages 1560–1565. IEEE, 2015.

[83] C. Morrell, R. Moore, R. Marchany, and J. G. Tront. Dht blind rendezvous for session establishment in network layer moving target defenses. In *Proceedings of the Second ACM Workshop on Moving Target Defense*, pages 77–84. ACM, 2015.

[84] C. Morrell, J. S. Ransbottom, R. Marchany, and J. G. Tront. Scaling ipv6 address bindings in support of a moving target defense. In *Internet Technology and Secured Transactions (ICITST), 2014 9th International Conference for*, pages 440–445. IEEE, 2014.

[85] C. F. Morrell. *Improving the Security, Privacy, and Anonymity of a Client-Server Network through the Application of a Moving Target Defense.* PhD thesis, Virginia Tech, 2016.

[86] M. Naya-Plasencia and T. Peyrin. Practical cryptanalysis of armadillo2. In *Fast Software Encryption*, pages 146–162. Springer, 2012.

[87] H. Okhravi, M. Rabe, T. Mayberry, W. Leonard, T. Hobson, D. Bigelow, and W. Streilein. Survey of cyber moving target techniques. Technical report, DTIC Document, 2013.

[88] K. Ouafi and S. Vaudenay. Smashing squash-0. In *Advances in Cryptology-EUROCRYPT 2009*, pages 300–312. Springer, 2009.

[89] S. Poslad, M. Hamdi, and H. Abie. Adaptive security and privacy management for the internet of things (aspi 2013). In *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*, pages 373–378. ACM, 2013.

[90] T. Preiss, M. Sherburne, R. Marchany, and J. Tront. Implementing dynamic address changes in contikios. In *Information Society (i-Society), 2014 International Conference on*, pages 222–227. IEEE, 2014.

[91] D. R. Raymond. *Denial-of-sleep vulnerabilities and defenses in wireless sensor network mac protocols*. PhD thesis, Virginia Tech, 2008.

[92] S. Raza, H. Shafagh, K. Hewage, R. Hummen, and T. Voigt. Lithe: Lightweight secure coap for the internet of things. *Sensors Journal, IEEE*, 13(10):3711–3720, 2013.

[93] M. Saarinen and J. Aumasson. The blake2 cryptographic hash and mac. Technical report, Internet Draft draft-saarinen-blake2-06. IETF, 2015.

[94] P. Sepehrdad, P. Susil, and S. Vaudenay. Fast key recovery attack on armadillo1 and variants. In *CARDIS*, pages 133–150. Springer, 2011.

[95] A. Shamir. Squash–a new mac with provable security properties for highly constrained devices such as rfid tags. In *Fast Software Encryption*, pages 144–157. Springer, 2008.

[96] P. Sharma, D. Girdhar, et al. Security concerns of ipv6-slaac and security policies to diminish the risk associated with them. *arXiv preprint arXiv:1405.4197*, 2014.

[97] M. Sherburne, R. Marchany, and J. Tront. Implementing moving target ipv6 defense to secure 6lowpan in the internet of things and smart grid. In *Proceedings of the 9th Annual Cyber and Information Security Research Conference*, pages 37–40. ACM, 2014.

[98] M. G. Sherburne. *Micro-Moving Target IPv6 Defense for 6LoWPAN and the Internet of Things*. PhD thesis, Virginia Tech, 2015.

[99] N. Sklavos and P. Kitsos. Blake hash function family on fpga: From the fastest to the smallest. In *2010 IEEE Computer Society Annual Symposium on VLSI*, pages 139–142. IEEE, 2010.

[100] J. Sun and K. Sun. Desir: Decoy-enhanced seamless ip randomization. 2016.

[101] P. Sušil and S. Vaudenay. *Multipurpose cryptographic primitive ARMADILLO3*. Springer, 2012.

[102] A. Systems. Wismote schema v1.01, Mar 2011.

[103] A. Systems. Wismote reference document as1006-brf-201, Feb 2012.

[104] P. Tayal. Ipv6 slaac related security issues and removal of those security issues. *International Journal Of Engineering And Computer Science*, 3(09), 2014.

[105] M. Tiloca. Efficient protection of response messages in dtls-based secure multicast communication. In *Proceedings of the 7th International Conference on Security of Information and Networks*, page 466. ACM, 2014.

[106] P. D. Welch. The statistical analysis of simulation results. *The computer performance modeling handbook*, 22:268–328, 1983.

[107] T. Winter. Rpl: Ipv6 routing protocol for low-power and lossy networks. 2012.

[108] W. Wu, S. Wu, L. Zhang, J. Zou, and L. Dong. Lhash: A lightweight hash function (full version). *IACR Cryptology ePrint Archive*, 2013:867, 2013.

[109] T. Xu, J. B. Wendt, and M. Potkonjak. Security of iot systems: Design challenges and opportunities. In *Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design*, pages 417–423. IEEE Press, 2014.

[110] J. Yackoski, J. Li, S. A. DeLoach, and X. Ou. Mission-oriented moving target defense based on cryptographically strong network dynamics. In *Proceedings of the Eighth Annual Cyber Security and Information Intelligence Research Workshop*, page 57. ACM, 2013.

[111] L. Yushi, J. Fei, and Y. Hui. Study on application modes of military internet of things (miot). In *Computer Science and Automation Engineering (CSAE), 2012 IEEE International Conference on*, volume 3, pages 630–634. IEEE, 2012.

[112] K. Zaffarano, J. Taylor, and S. Hamilton. A quantitative framework for moving target defense effectiveness evaluation. In *Proceedings of the Second ACM Workshop on Moving Target Defense*, pages 3–10. ACM, 2015.

[113] K. Zeitz, M. Cantrell, R. Marchany, and J. Tront. Designing a micro-moving target ipv6

defense for the internet of things. In *Internet-of-Things Design and Implementation (IoTDI), 2017 IEEE/ACM Second International Conference on*, pages 179–184. IEEE, 2017.

[114] K. Zeitz, M. Cantrell, R. Marchany, and J. Tront. Changing the game: A micro moving target ipv6 defense for the internet of things. *IEEE Wireless Communications Letters*, 7(4):578–581, 2018.

[115] X. M. Zhang and N. Zhang. An open, secure and flexible platform based on internet of things and cloud computing for ambient aiding living and telemedicine. In *Computer and Management (CAMAN), 2011 International Conference on*, pages 1–4. IEEE, 2011.

[116] Y. Zhang, J. Kim, K. Choi, and T. Shon. High performance and low power hardware implementation for cryptographic hash functions. *International Journal of Distributed Sensor Networks*, 2014, 2014.

[117] Z.-K. Zhang, M. C. Y. Cho, and S. Shieh. Emerging security threats and countermeasures in iot. In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, pages 1–6. ACM, 2015.

[118] W. Zhou, Y. Zhang, and P. Liu. The effect of iot new features on security and privacy: New threats, existing solutions, and challenges yet to be solved. *arXiv preprint arXiv:1802.03110*, 2018.

[119] R. Zhuang, S. A. DeLoach, and X. Ou. A model for analyzing the effect of moving target defenses on enterprise networks. In *Proceedings of the 9th Annual Cyber and Information Security Research Conference*, pages 73–76. ACM, 2014.

[120] R. Zhuang, S. A. DeLoach, and X. Ou. Towards a theory of moving target defense. In *Proceedings of the First ACM Workshop on Moving Target Defense*, pages 31–40. ACM, 2014.

[121] R. Zhuang, S. Zhang, A. Bardas, S. A. DeLoach, X. Ou, and A. Singhal. Investigating the application of moving target defenses to network security. In *Resilient Control Systems (ISRCS), 2013 6th International Symposium on*, pages 162–169. IEEE, 2013.

[122] R. Zhuang, S. Zhang, S. A. DeLoach, X. Ou, and A. Singhal. Simulation-based approaches to studying effectiveness of moving-target network defense. In *National symposium on moving target research*, pages 1–12, 2012.

Appendix A: Simulation Footprint Data

A.1 Simulation Footprint Data

Table A.1: Size Data (Bytes): Send Packet Every 1 Minute Rotate Address Every 5 Seconds

	text	data	bss	Dec Total	Hex Total
CONTROL	44355	290	8610	53255	d007
CONTROL (C)	45961	312	8926	55199	d79f
SPONGENT	46401	842	12794	60037	ea85
SHA	47157	618	12794	60569	ec99
SPONGENT (C)	47929	864	13106	61899	f1cb
SHA (C)	48685	640	13106	62431	f3df
QUARK	50365	330	12794	63489	f801
QUARK (C)	51893	352	13106	65351	ff47
BLAKE	71781	334	12794	84909	14bad
BLAKE (C)	73301	356	13106	86763	152eb

Table A.2: Size Data (Bytes): Send Packet Every 1 Hour Rotate Address Every 1 Minute

	text	data	bss	Dec Total	Hex Total
CONTROL	44435	290	8614	53339	d05b
CONTROL (C)	45963	312	8926	55201	d7a1
SPONGENT	46403	842	12794	60039	ea87
SHA	47159	618	12794	60571	ec9b
SPONGENT (C)	47931	864	13106	61901	f1cd
SHA (C)	48687	640	13106	62433	f3e1
QUARK	50367	330	127944	63491	f803
QUARK (C)	51895	352	13106	65353	ff49
BLAKE	71783	334	12794	84911	14baf
BLAKE (C)	73303	356	13106	86765	152ed

Table A.3: Size Data (Bytes): Send Packet Every 1 Hour Rotate Address Every 1 Hour

	text	data	bss	Dec Total	Hex Total
CONTROL	44435	290	8614	53339	d05b
CONTROL (C)	45955	312	8926	55193	d799
SPONGENT	46407	842	12794	60043	ea8b
SHA	47163	618	12794	60575	ec9f
SPONGENT (C)	47935	864	13106	61905	f1d1
SHA (C)	48691	640	13106	62437	f3e5
QUARK	50371	330	12794	63495	f807
QUARK (C)	51899	352	13106	65357	ff4d
BLAKE	71787	334	12794	84915	14bb3
BLAKE (C)	73307	356	13106	86769	152f1

Appendix B: Refined Simulation Footprint Data

B.1 Refined Simulation Footprint Data

Table B.1: Size Data (Bytes): Send Packet Every 1 Minute Rotate Address Every 5 Seconds

	text	data	bss	Dec Total	Hex Total
Control	45639	290	8660	54589	d53d
Set Global Address	46245	310	8738	55293	d7fd
Address Setup	46267	310	8738	55315	d813
Spongent Hash Only	47531	834	12840	61205	ef15
Spongent	47611	842	12840	61293	ef6d
Sha Hash Only	48287	610	12840	61737	f129
Control (C)	48316	300	7942	56558	dcee
Sha	48367	618	12840	61825	f181
Set Global Address (C)	48922	320	8020	57262	dfae
Address Setup (C)	48944	320	8020	57284	dfc4
Spongent Hash Only (C)	50208	844	12122	63174	f6c6
Spongent (C)	50288	852	12122	63262	f71e
Sha Hash Only (C)	50964	620	12122	63706	f8da
Sha (C)	51044	628	12122	63794	f932
Quark Hash Only	51495	322	12840	64657	fc91
Quark	51575	330	12840	64745	fce9
Quark Hash Only (C)	54172	332	12122	66626	10442

Table B.1: Size Data (Bytes): Send Packet Every 1 Minute Rotate Address Every 5 Seconds

	text	data	bss	Dec Total	Hex Total
Quark (C)	54252	340	12122	66714	1049a
Blake Hash Only	72907	326	12840	86073	15039
Blake	72987	334	12840	86161	15091
Blake (C)	75438	344	12890	88672	15a60
Blake Hash Only (C)	75584	336	12122	88042	157ea

Table B.2: Size Data (Bytes): Send Packet Every 1 Hour Rotate Address Every 1 Minute

	text	data	bss	Dec Total	Hex Total
Control	45641	290	8660	54591	d53f
Set Global Address	46247	310	8738	55295	d7ff
Address Setup	46269	310	8738	55317	d815
Spongent Hash Only	47533	834	12840	61207	ef17
Spongent	47613	842	12840	61295	ef6f
Sha Hash Only	48289	610	12840	61739	f12b
Control (C)	48318	300	7942	56560	dcf0
Sha	48369	618	12840	61827	f183
Set Global Address (C)	48924	320	8020	57264	dfb0
Address Setup (C)	48946	320	8020	57286	dfc6
Spongent Hash Only (C)	50210	844	12122	63176	f6c8

Table B.2: Size Data (Bytes): Send Packet Every 1 Hour Rotate Address Every 1 Minute

	text	data	bss	Dec Total	Hex Total
Spongent (C)	50290	852	12122	63264	f720
Sha Hash Only (C)	50966	620	12122	63708	f8dc
Sha (C)	51046	628	12122	63796	f934
Quark Hash Only	51497	322	12840	64659	fc93
Quark	51577	330	12840	64747	fceb
Quark Hash Only (C)	54174	332	12122	66628	10444
Quark (C)	54254	340	12122	66716	1049c
Blake Hash Only	72909	326	12840	86075	1503b
Blake	72989	334	12840	86163	15093
Blake (C)	75440	344	12890	88674	15a62
Blake Hash Only (C)	75586	336	12122	88044	157ec

Table B.3: Size Data (Bytes): Send Packet Every 1 Hour Rotate Address Every 1 Hour

	text	data	bss	Dec Total	Hex Total
Control	45641	290	8660	54591	d53f
Set Global Address	46251	310	8738	55299	d803
Address Setup	46273	310	8738	55321	d819
Spongent Hash Only	47537	834	12840	61211	ef1b
Spongent	47617	842	12840	61299	ef73

Table B.3: Size Data (Bytes): Send Packet Every 1 Hour Rotate Address Every 1 Hour

	text	data	bss	Dec Total	Hex Total
Sha Hash Only	48293	610	12840	61743	f12f
Control (C)	48318	300	7942	56560	dcf0
Sha	48373	618	12840	61831	f187
Set Global Address (C)	48928	320	8020	57268	dfb4
Address Setup (C)	48950	320	8020	57290	dfca
Spongent Hash Only (C)	50214	844	12122	63180	f6cc
Spongent (C)	50294	852	12122	63268	f724
Sha Hash Only (C)	50970	620	12122	63712	f8e0
Sha (C)	51050	628	12122	63800	f938
Quark Hash Only	51501	322	12840	64663	fc97
Quark	51581	330	12840	64751	fcef
Quark Hash Only (C)	54178	332	12122	66632	10448
Quark (C)	54258	340	12122	66720	104a0
Blake Hash Only	72913	326	12840	86079	1503f
Blake	72993	334	12840	86167	15097
Blake (C)	75444	344	12890	88678	15a66
Blake Hash Only (C)	75590	336	12122	88048	157f0

Appendix C: Simulation Power Data

C.1 Simulation Power Data

Table C.1: Power Consumption (mW): Send Packet Every 1 Minute Rotate Address Every 5 Seconds

	CPU	LPM	TX	RX	Total
CONTROL	0.44968	0.14988	14.3460	44.834	59.7798
SHA	0.46815	0.14933	14.3876	44.787	59.7923
BLAKE	0.46973	0.14928	14.3907	44.784	59.7934
QUARK	0.51188	0.14800	14.3784	44.7976	59.8359
SPONGENT	3.28095	0.0642	15.0762	44.0191	62.4405

Table C.2: Power Consumption (mW): Send Packet Every 1 Hour Rotate Address Every 1 Minute

	CPU	LPM	TX	RX	Total
CONTROL	0.03615	0.1624	0.70005	59.2582	60.1568
BLAKE	0.04032	0.16228	0.7054	59.2516	60.1596
SHA	0.04066	0.16227	0.71184	59.2467	60.1614
QUARK	0.04560	0.16212	0.70427	59.2530	60.1649
SPONGENT	1.12949	0.12930	0.71474	59.2421	61.2156

Table C.3: Power Consumption (mW): Send Packet Every 1 Hour Rotate Address Every 1 Hour

	CPU	LPM	TX	RX	Total
CONTROL	0.0360	0.1624	0.6956	59.2625	60.1566
BLAKE	0.0381	0.1623	0.6991	59.2596	60.1591
QUARK	0.0382	0.1623	0.7003	59.2581	60.1590
SHA	0.0381	0.1623	0.7003	59.2581	60.1589
SPONGENT	0.0543	0.1619	0.6155	59.3478	60.1795

Appendix D: Refined Simulation

Power and Energy Data

D.1 1 Refined Simulation Power Data

Table D.1: Power Consumption (mW) Case 1: Send Packet Every 1 Minute Rotate Address Every 5 Seconds

	CPU	LPM	TX	RX	Total
CONTROL	0.009785	0.054204	0.012205	0.235388	0.311582
ADDRESS SETUP	0.011456	0.054153	0.010488	0.235409	0.311506
BLAKE HASH ONLY	0.014051	0.054075	0.01177	0.235266	0.315161
QUARK HASH ONLY	0.038179	0.053344	0.011413	0.235301	0.338236
SHA HASH ONLY	0.01371	0.054085	0.012218	0.235261	0.315274
SPONGENT HASH ONLY	1.137192	0.020068	0.040467	0.24341	1.441137
SET GLOBAL ADDRESS	0.012007	0.054136	0.012209	0.235028	0.313381
BLAKE	0.014227	0.054069	0.012216	0.235071	0.315583
QUARK	0.038513	0.053334	0.012218	0.235063	0.339127
SHA	0.014151	0.054072	0.012217	0.235154	0.315593
SPONGENT	1.137352	0.020064	0.041925	0.242511	1.441852

Table D.2: Power Consumption (mW) Case 2: Send Packet Every 1 Hour Rotate Address Every 1 Minute

	CPU	LPM	TX	RX	Total
CONTROL	0.009715	0.054206	0.012195	0.235206	0.311322
ADDRESS SETUP	0.011368	0.054156	0.012207	0.235239	0.31297
BLAKE HASH ONLY	0.011412	0.054154	0.011138	0.23532	0.312024
QUARK HASH ONLY	0.01357	0.054089	0.012212	0.235244	0.315115
SHA HASH ONLY	0.011527	0.054151	0.012216	0.23524	0.313133
SPONGENT HASH ONLY	0.338199	0.04426	0.018295	0.250611	0.651365
SET GLOBAL ADDRESS	0.011396	0.054155	0.012207	0.235118	0.312876
BLAKE	0.011334	0.054157	0.012202	0.235128	0.312822
QUARK	0.013658	0.054086	0.012212	0.235009	0.314966
SHA	0.011473	0.054153	0.01038	0.235679	0.311685
SPONGENT	0.33851	0.044251	0.019622	0.250561	0.652943

D.2 2 Refined Simulation Energy Data

Table D.3: Power Consumption (mW) Case 3: Send Packet Every 1 Hour Rotate Address Every 1 Hour

	CPU	LPM	TX	RX	Total
CONTROL	0.009715	0.054206	0.012195	0.235206	0.311322
ADDRESS SETUP	0.010903	0.05417	0.012199	0.235153	0.312425
BLAKE HASH ONLY	0.010692	0.054176	0.010118	0.235112	0.310099
QUARK HASH ONLY	0.01093	0.054169	0.012218	0.235111	0.312428
SHA HASH ONLY	0.010688	0.054176	0.010095	0.235191	0.31015
SPONGENT HASH ONLY	0.013367	0.054095	0.012285	0.235235	0.314982
SET GLOBAL ADDRESS	0.010905	0.05417	0.012205	0.235193	0.312473
BLAKE	0.010629	0.054178	0.012212	0.235124	0.312144
QUARK	0.01086	0.054171	0.011334	0.235287	0.311653
SHA	0.010924	0.054169	0.012216	0.235279	0.312588
SPONGENT	0.013409	0.054094	0.012341	0.235209	0.315053

Table D.4: Energy Consumption (J) Case 1: Send Packet Every 1 Minute Rotate Address Every 5 Seconds

	CPU	LPM	TX	RX	Total
CONTROL	32.38706	178.8724	40.15345	776.8539	1028.267
ADDRESS SETUP	37.84795	178.7071	34.05484	776.8461	1027.456
BLAKE HASH ONLY	46.41025	178.4478	38.03553	776.4627	1039.356
QUARK HASH ONLY	125.9717	176.0388	37.05065	776.5167	1115.578
SHA HASH ONLY	45.35522	178.4797	40.16235	776.3804	1040.378
SPONGENT HASH ONLY	3763.865	66.14029	135.3842	801.8832	4767.273
SET GLOBAL ADDRESS	39.72165	178.6504	40.09579	775.5711	1034.039
BLAKE	47.04256	178.4286	40.1799	775.7239	1041.375
QUARK	127.1461	176.0033	40.17189	775.7703	1119.092
SHA	46.81865	178.4354	40.19311	776.0503	1041.498
SPONGENT	3764.132	66.1322	138.8158	798.8658	4767.946

Table D.5: Energy Consumption (mW) Case 2: Send Packet Every 1 Hour Rotate Address Every 1 Minute

	CPU	LPM	TX	RX	Total
CONTROL	32.1472	178.8797	40.07432	776.2193	1027.321
ADDRESS SETUP	37.61519	178.7141	40.1117	776.3413	1032.782
BLAKE HASH ONLY	37.75457	178.7099	36.58613	776.5959	1029.646
QUARK HASH ONLY	44.91267	178.4932	40.16503	776.3632	1039.934
SHA HASH ONLY	38.14491	178.698	40.18854	776.3187	1033.35
SPONGENT HASH ONLY	1110.684	146.2933	60.69692	824.9949	2142.669
SET GLOBAL ADDRESS	37.71564	178.7111	40.17398	775.9095	1032.51
BLAKE	37.47409	178.7184	40.09812	775.9059	1032.197
QUARK	45.19001	178.4847	40.14079	775.5191	1039.335
SHA	37.95562	178.7038	34.08062	777.7428	1028.483
SPONGENT	1111.655	146.2639	64.78273	824.8998	2147.601

Table D.6: Energy Consumption (mW) Case 3: Send Packet Every 1 Hour Rotate Address Every 1 Hour

	CPU	LPM	TX	RX	Total
CONTROL	32.1472	178.8797	40.07432	776.2193	1027.321
ADDRESS SETUP	36.21077	178.7567	40.10956	776.0231	1031.1
BLAKE HASH ONLY	35.52475	178.7774	33.20849	775.8698	1023.38
QUARK HASH ONLY	36.30107	178.7539	40.15933	775.8993	1031.114
SHA HASH ONLY	35.51201	178.7778	33.11967	776.1804	1023.59
SPONGENT HASH ONLY	44.5796	178.504	40.3642	776.2969	1039.745
SET GLOBAL ADDRESS	36.21063	178.7567	40.09297	776.1691	1031.229
BLAKE	35.32184	178.7835	40.16277	775.9635	1030.232
QUARK	36.04245	178.7617	36.89614	776.4604	1028.161
SHA	36.3085	178.7536	40.16233	776.4965	1031.721
SPONGENT	44.77825	178.498	40.62559	776.1965	1040.098

Appendix E: Steady State Graphs

E.1 1 Steady State Graphs

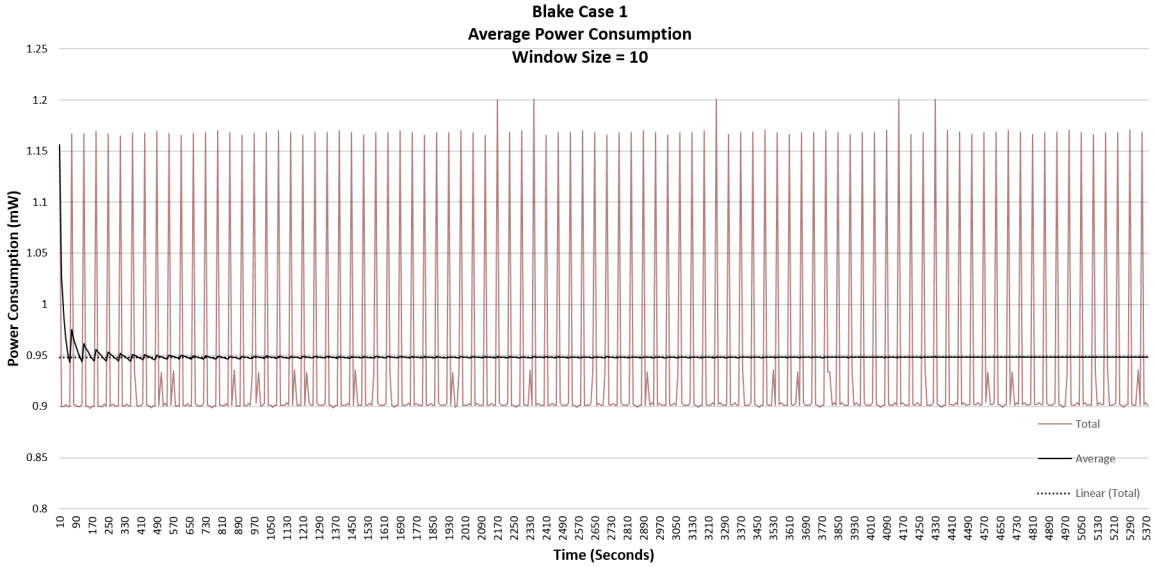

Figure E.1: Steady State Graph Blake Case 1

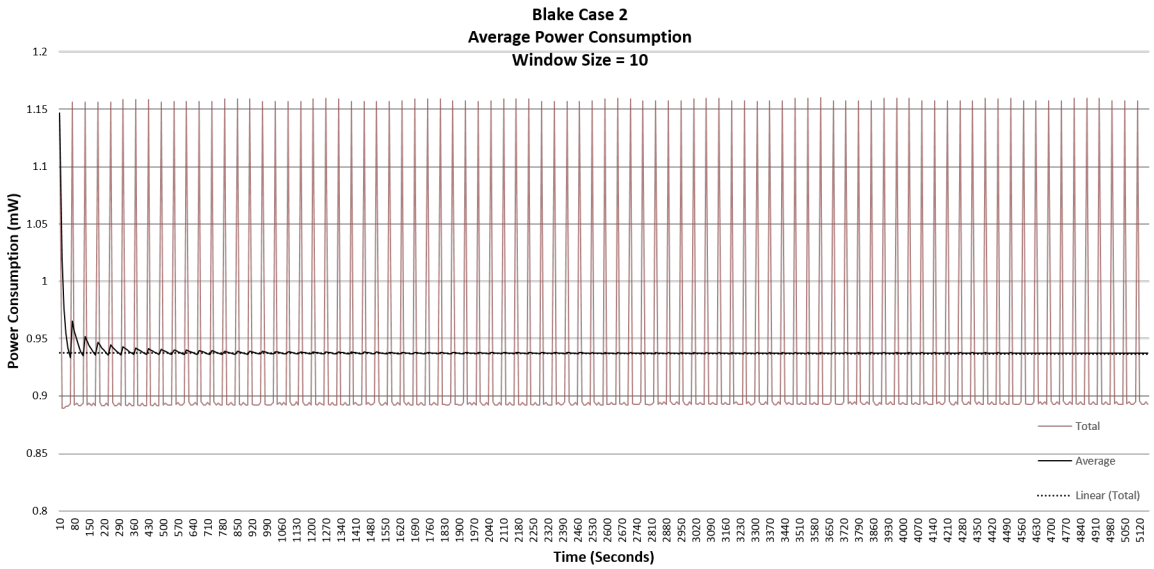

Figure E.2: Steady State Graph Blake Case 2

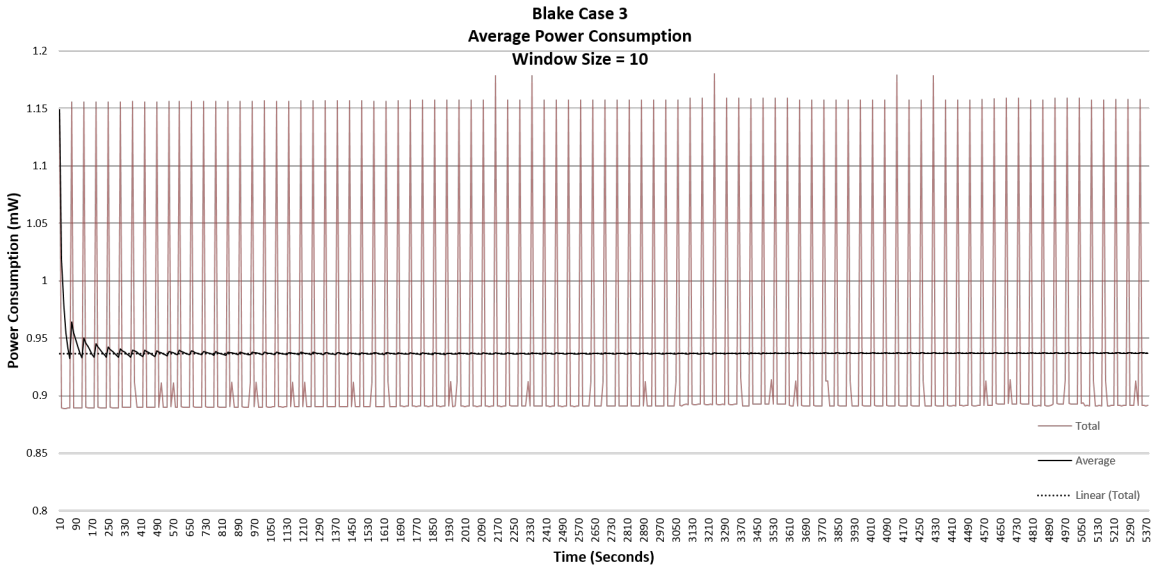

Figure E.3: Steady State Graph Blake Case 3

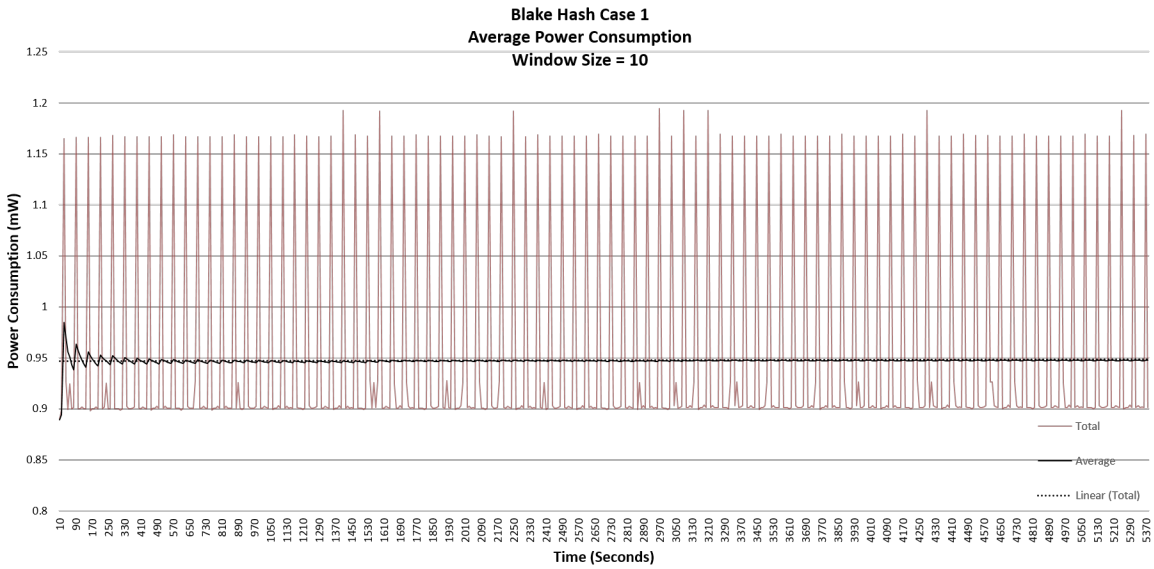

Figure E.4: Steady State Graph Blake Hash Case 1

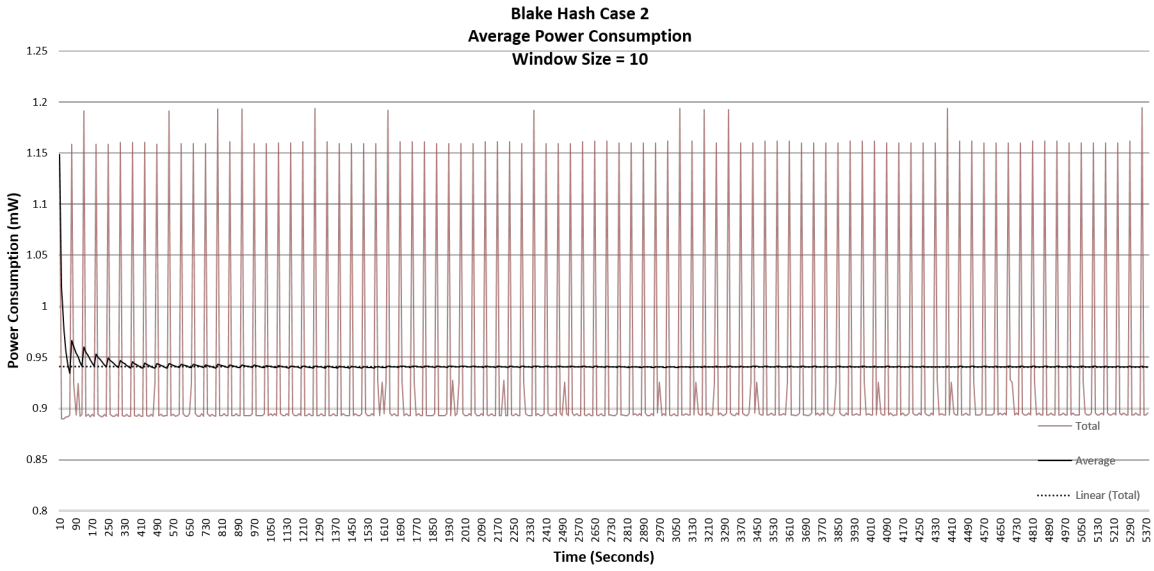

Figure E.5: Steady State Graph Blake Hash Case 2

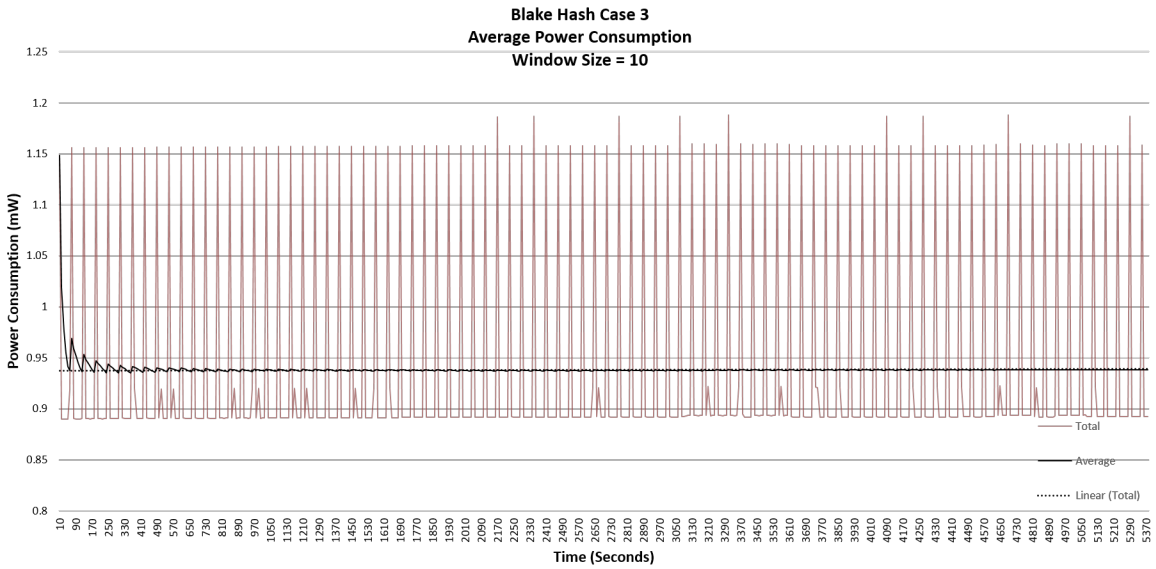

Figure E.6: Steady State Graph Blake Hash Case 3

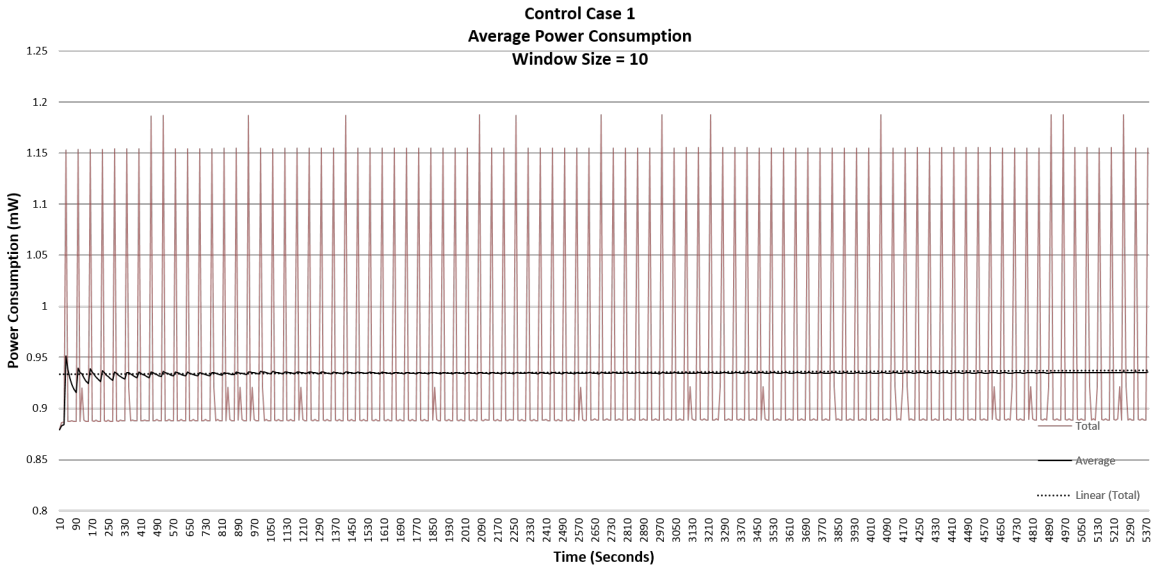

Figure E.7: Steady State Graph Control Case 1

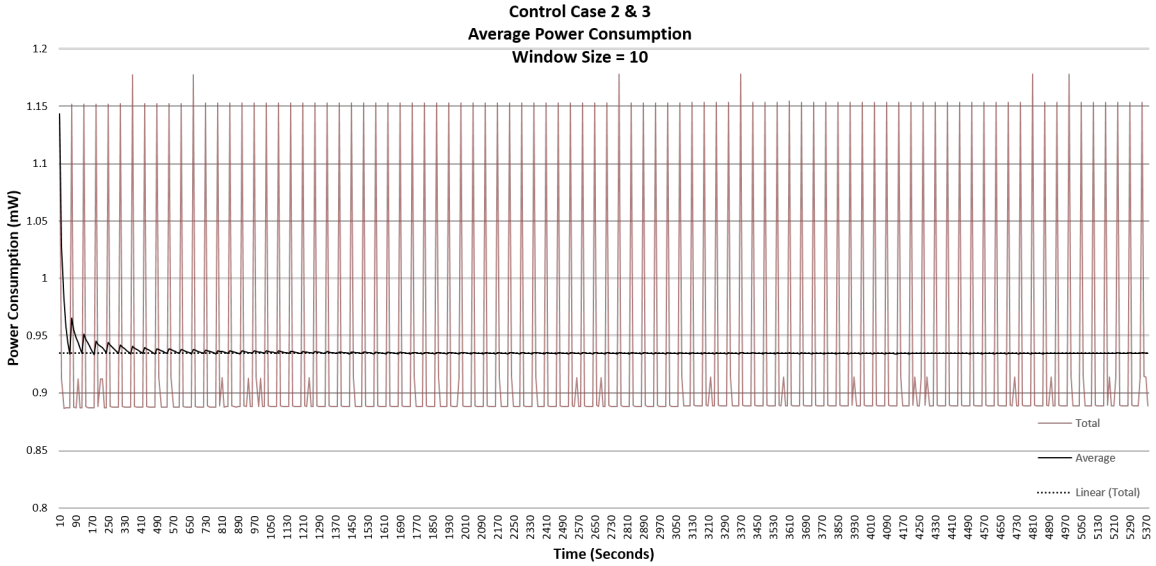

Figure E.8: Steady State Graph Control Case 2

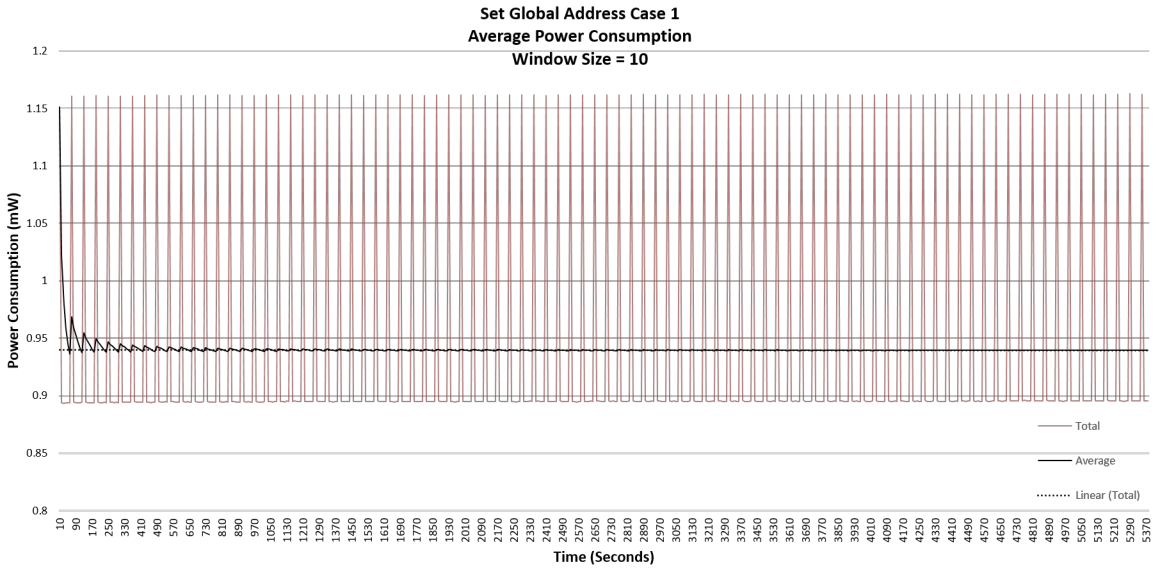

Figure E.9: Steady State Graph Global Case 1

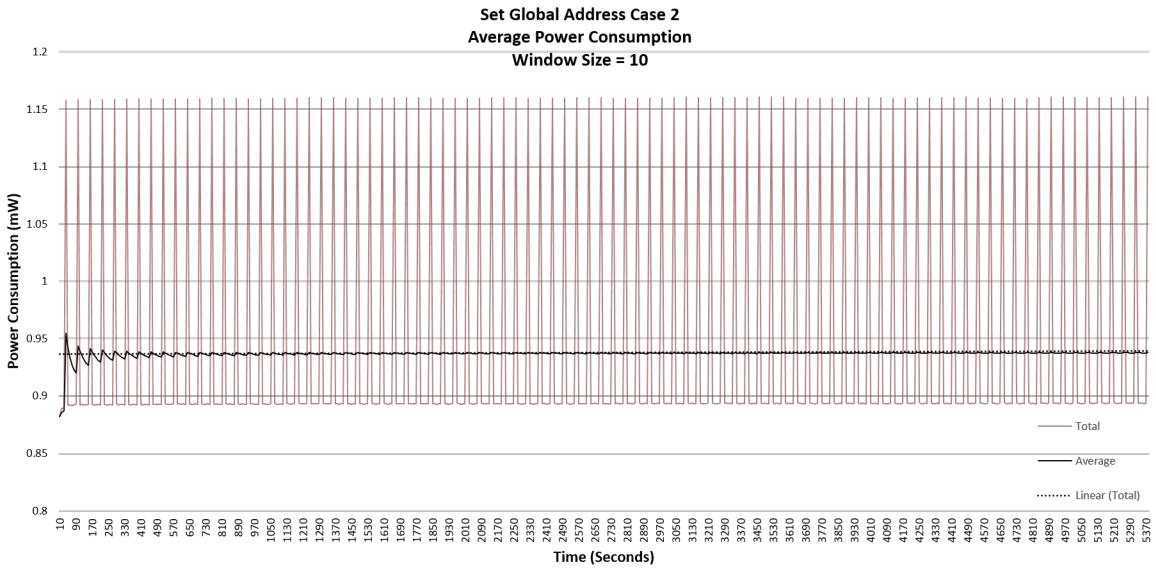

Figure E.10: Steady State Graph Global Case 2

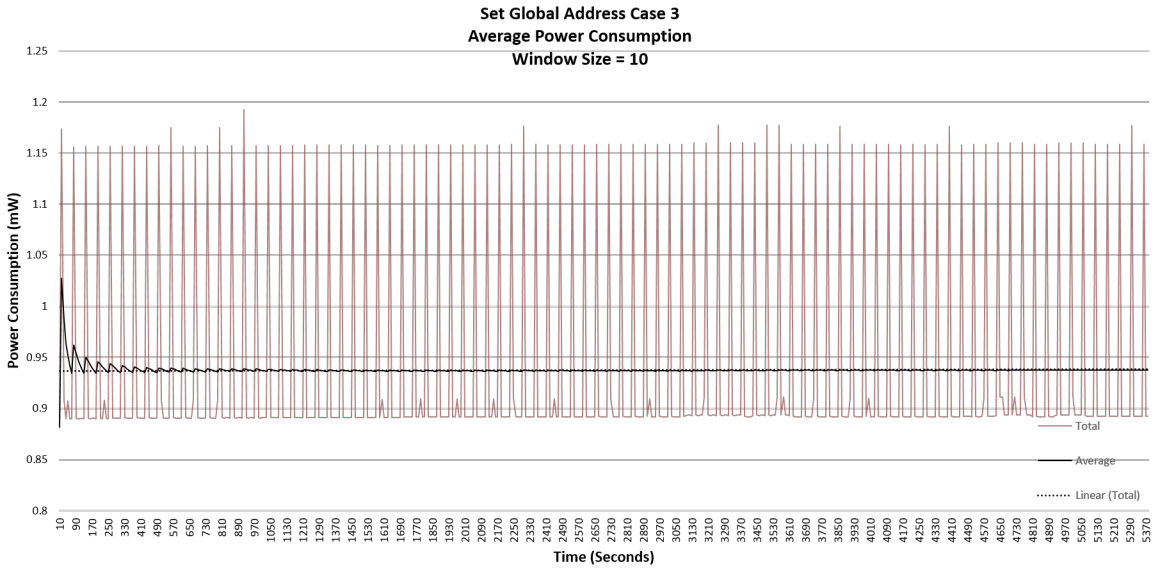

Figure E.11: Steady State Graph Global Case 3

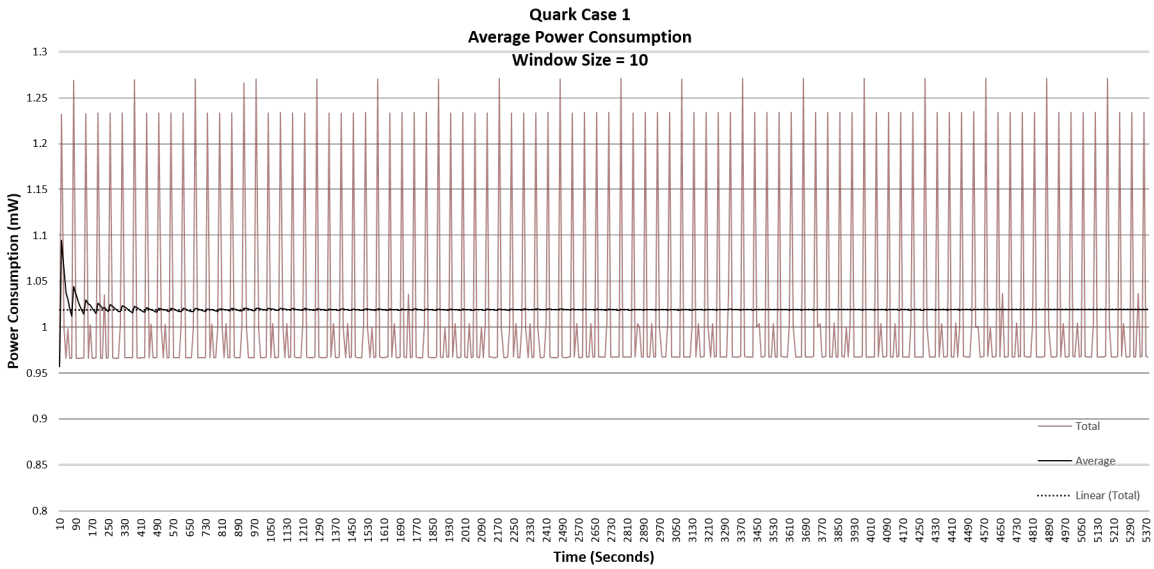

Figure E.12: Steady State Graph Quark Case 1

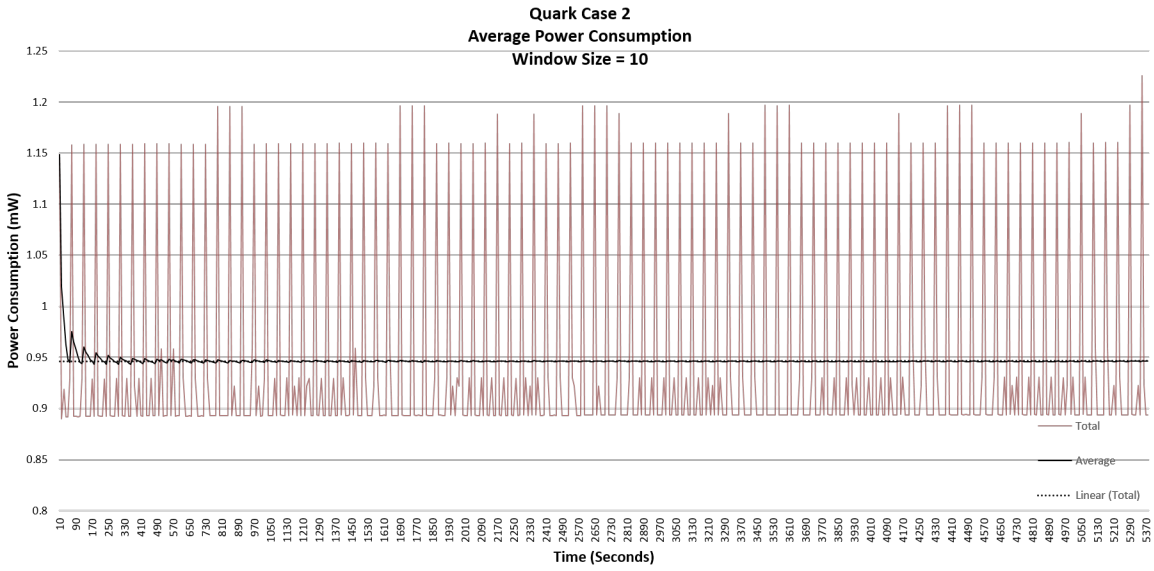

Figure E.13: Steady State Graph Quark Case 2

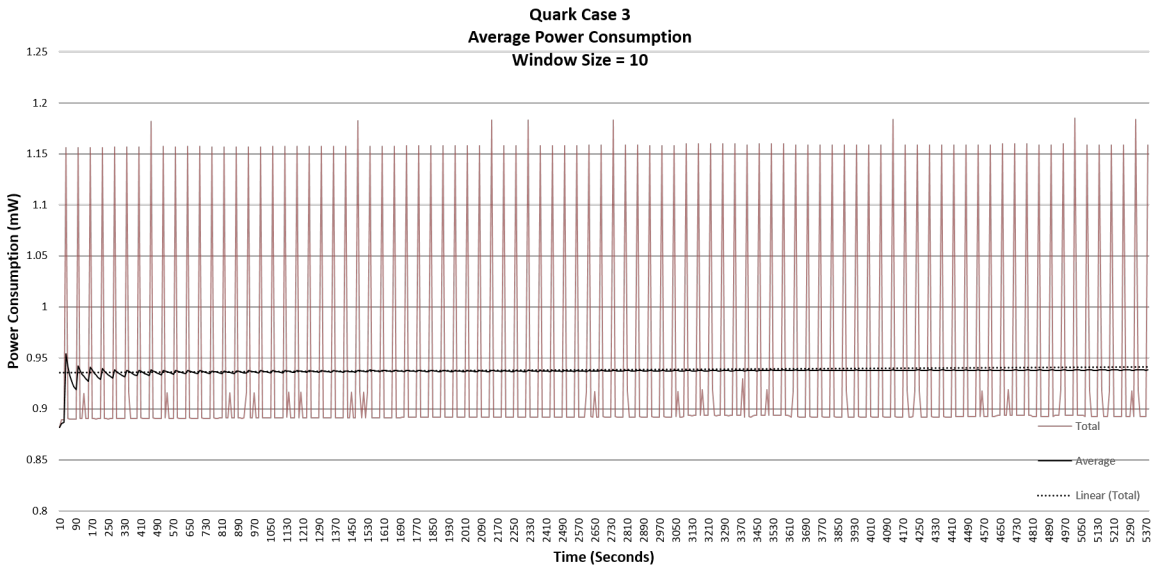

Figure E.14: Steady State Graph Quark Case 3

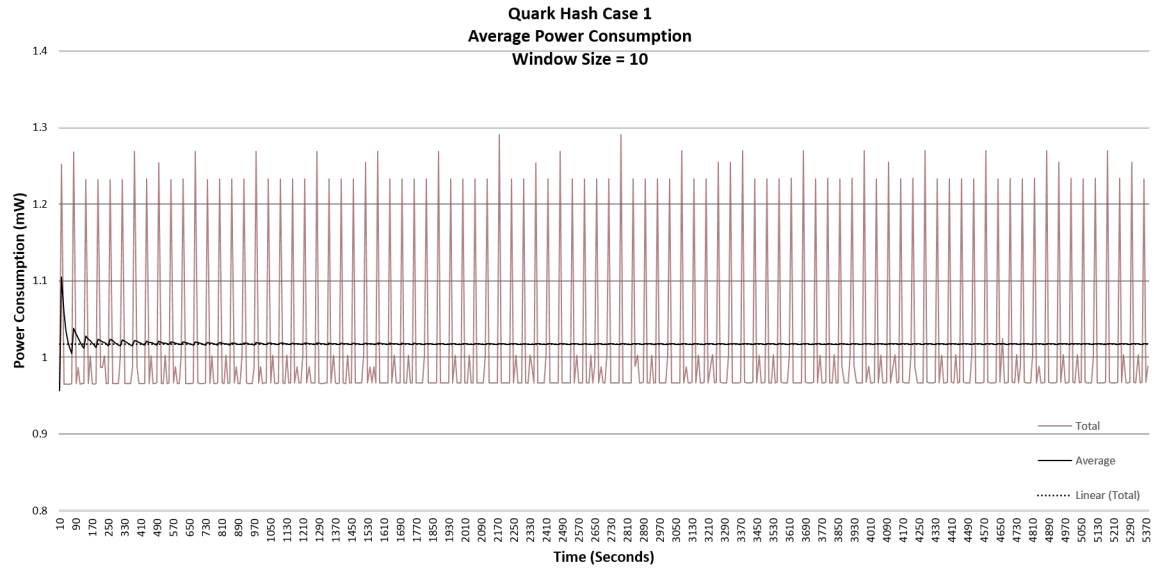

Figure E.15: Steady State Graph Quark Hash Case 1

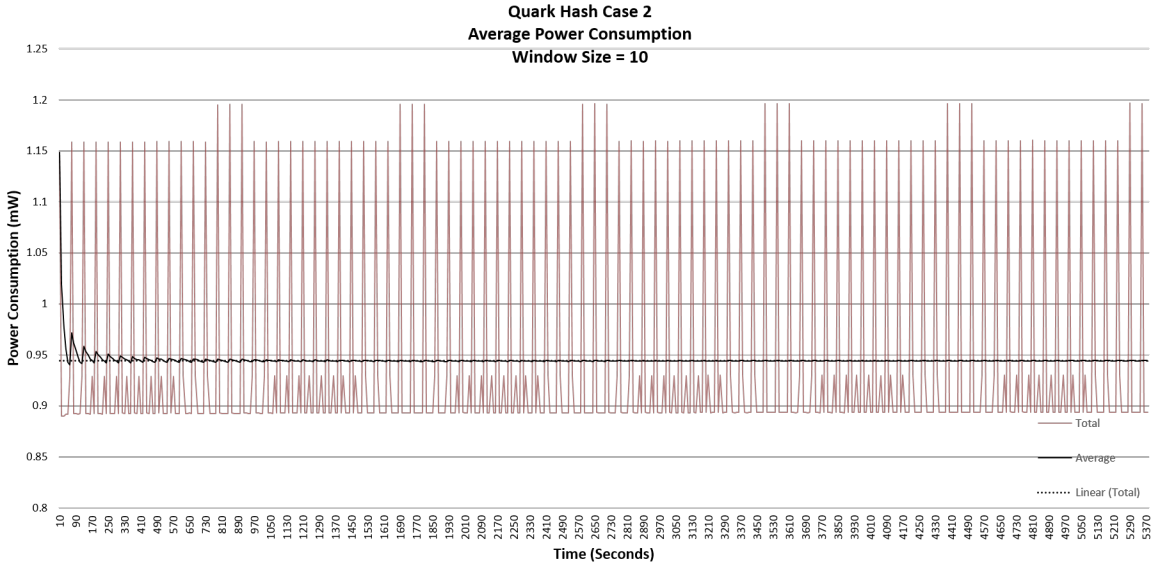

Figure E.16: Steady State Graph Quark Hash Case 2

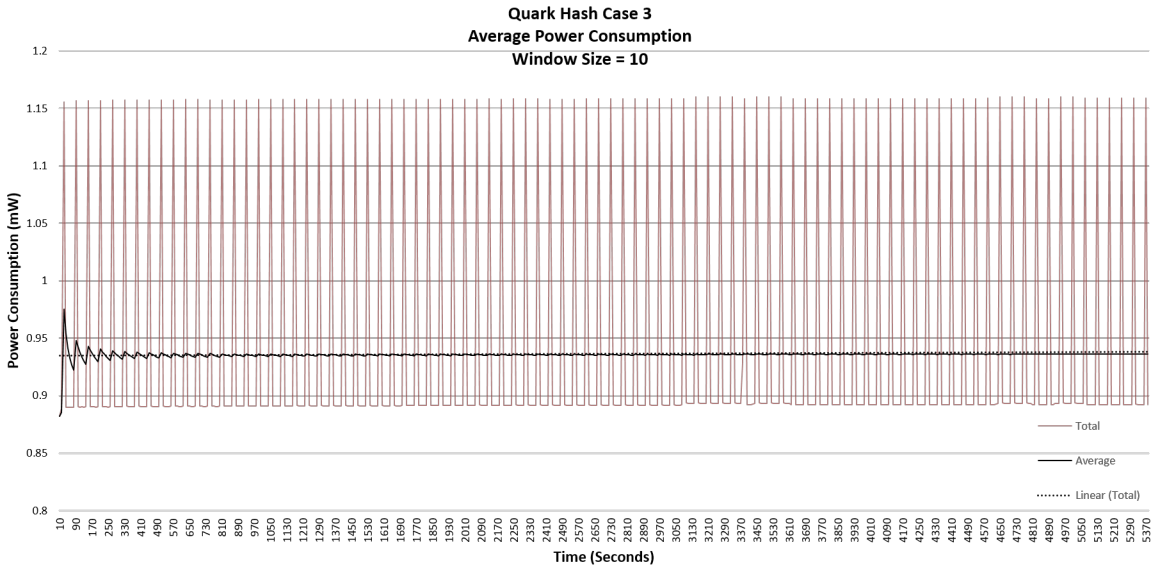

Figure E.17: Steady State Graph Quark Hash Case 3

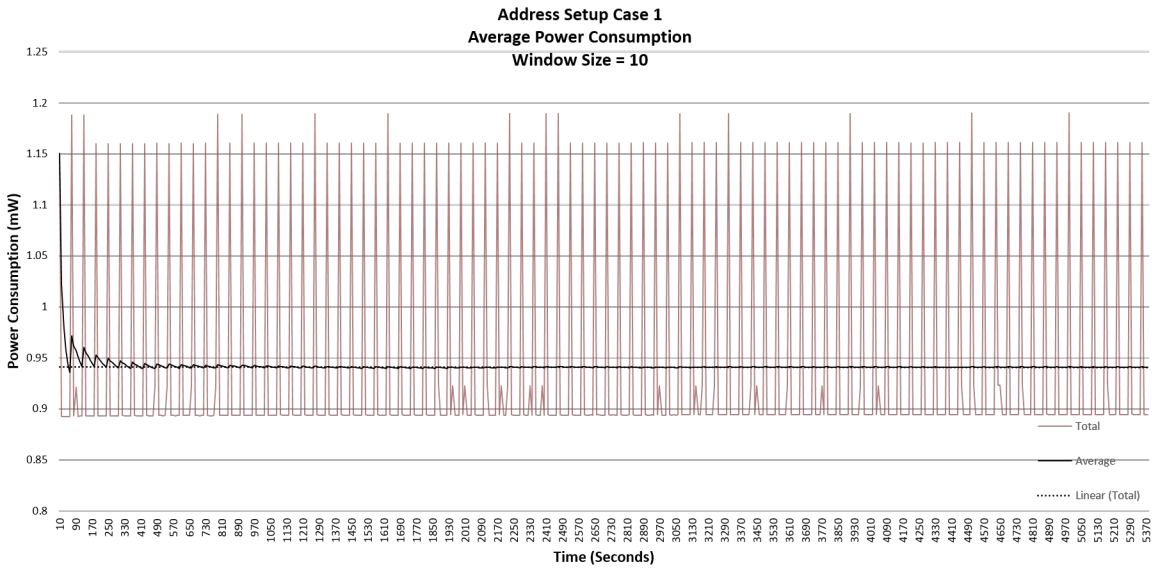

Figure E.18: Steady State Graph Address Setup Case 1

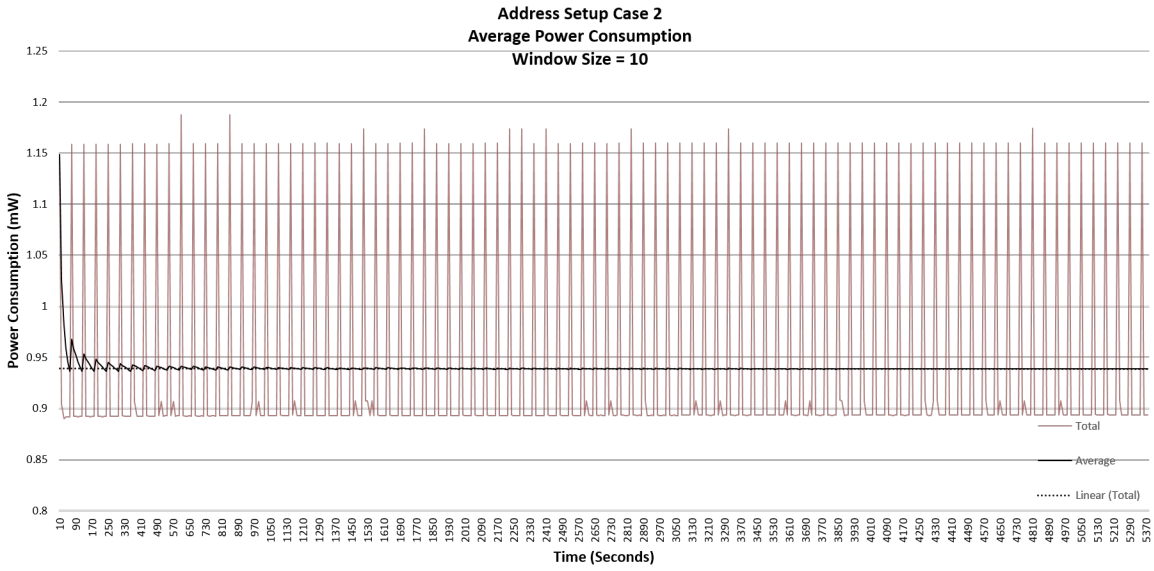

Figure E.19: Steady State Graph Address Setup Case 2

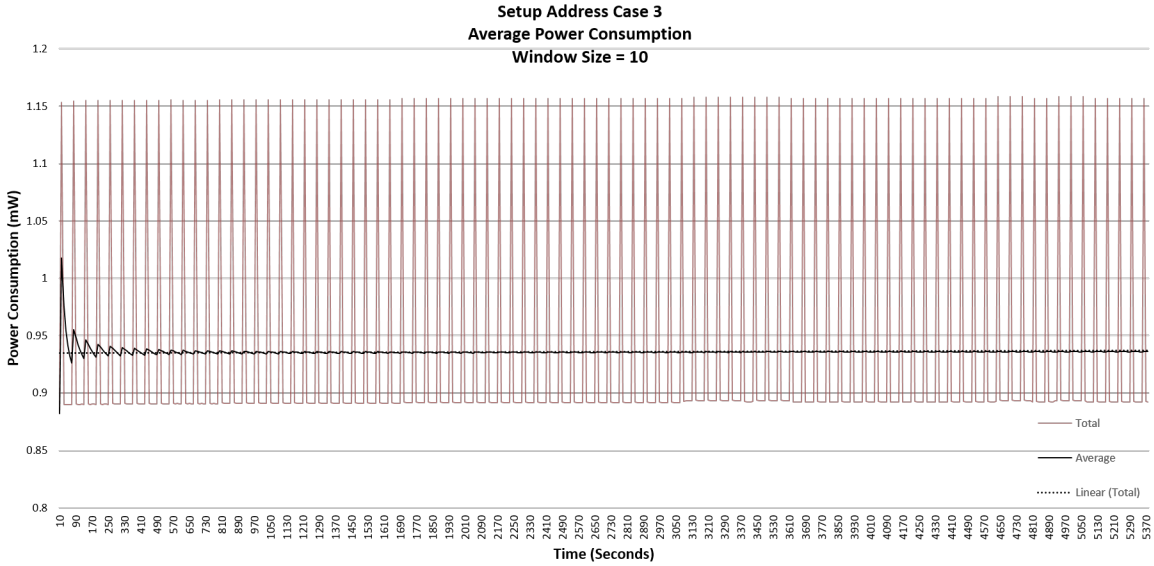

Figure E.20: Steady State Graph Address Setup Case 3

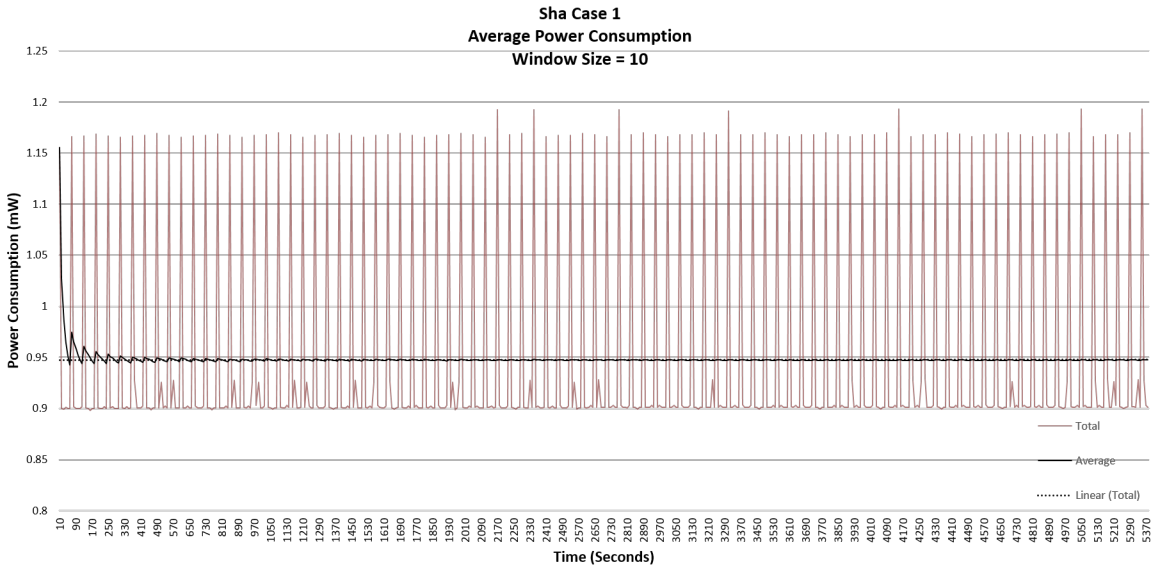

Figure E.21: Steady State Graph Sha Case 1

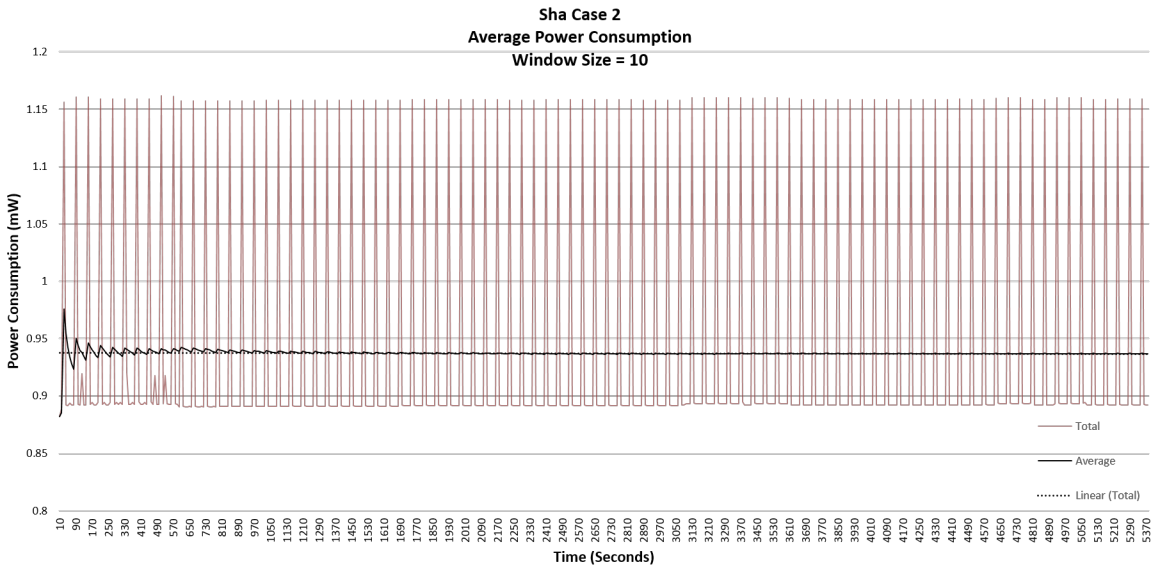

Figure E.22: Steady State Graph Sha Case 2

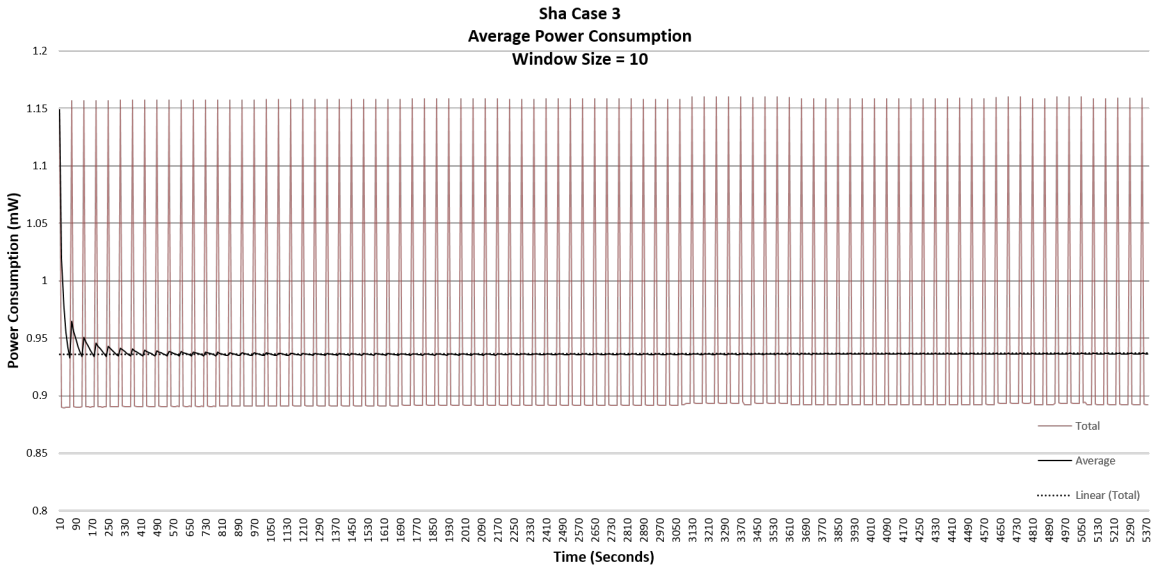

Figure E.23: Steady State Graph Sha Case 3

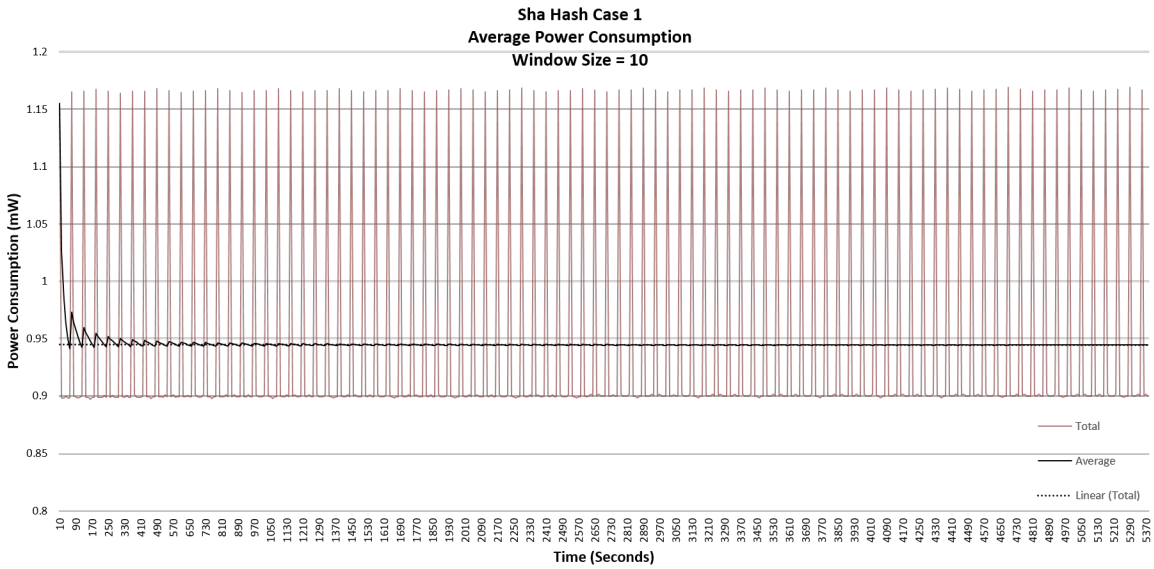

Figure E.24: Steady State Graph Sha Hash Case 1

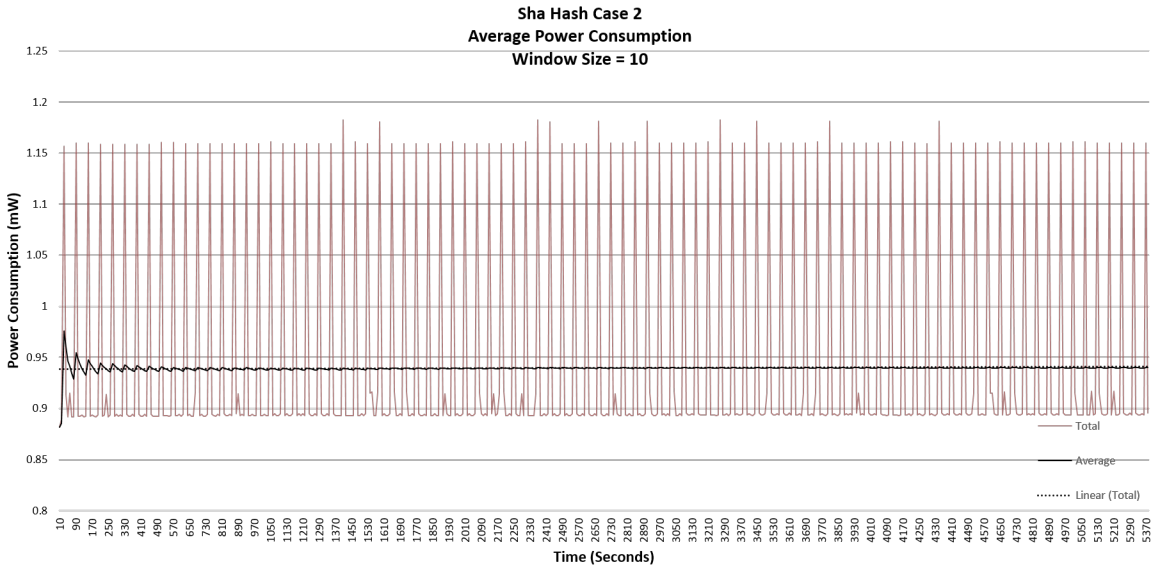

Figure E.25: Steady State Graph Sha Hash Case 2

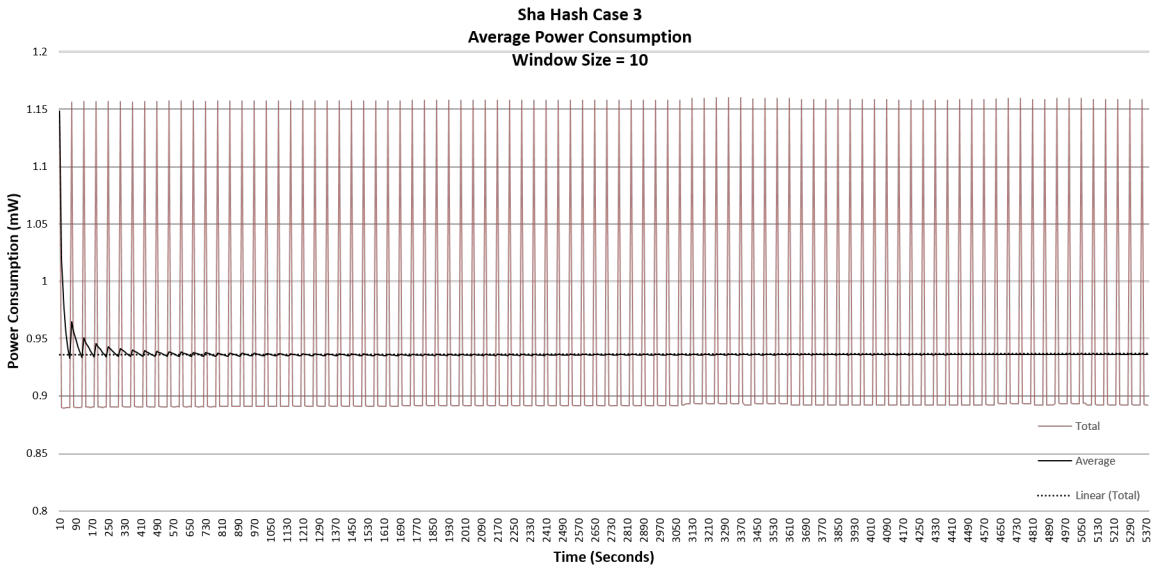

Figure E.26: Steady State Graph Sha Hash Case 3

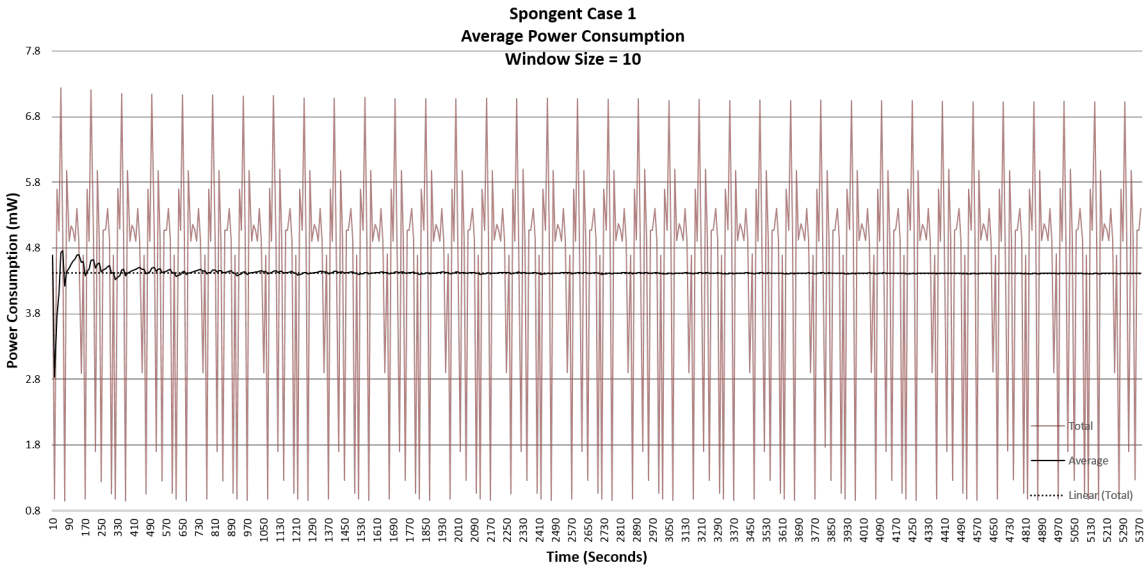

Figure E.27: Steady State Graph Spongent Case 1

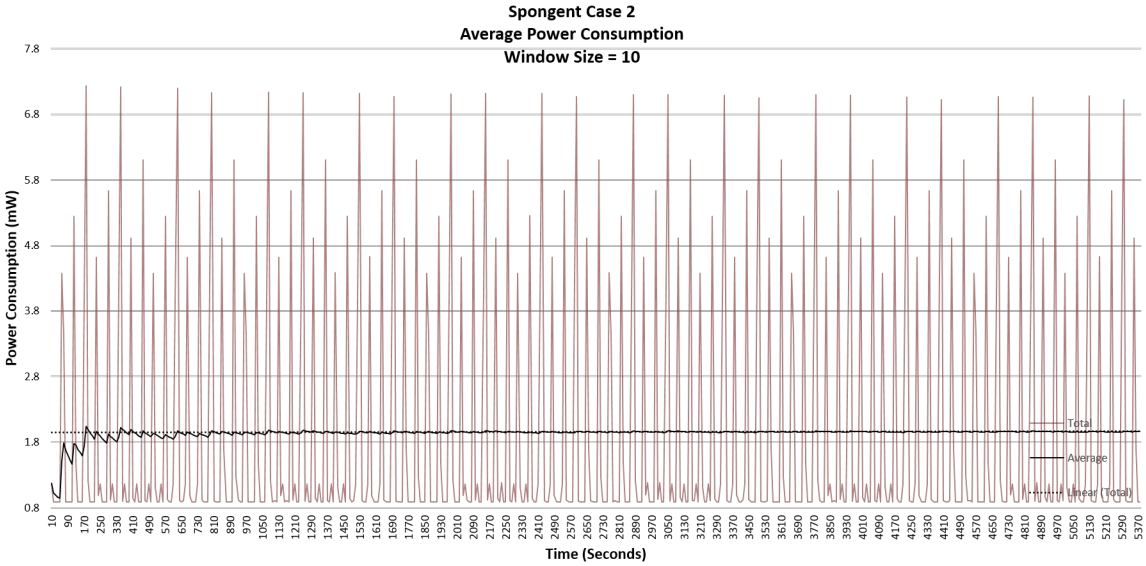

Figure E.28: Steady State Graph Spongent Case 2

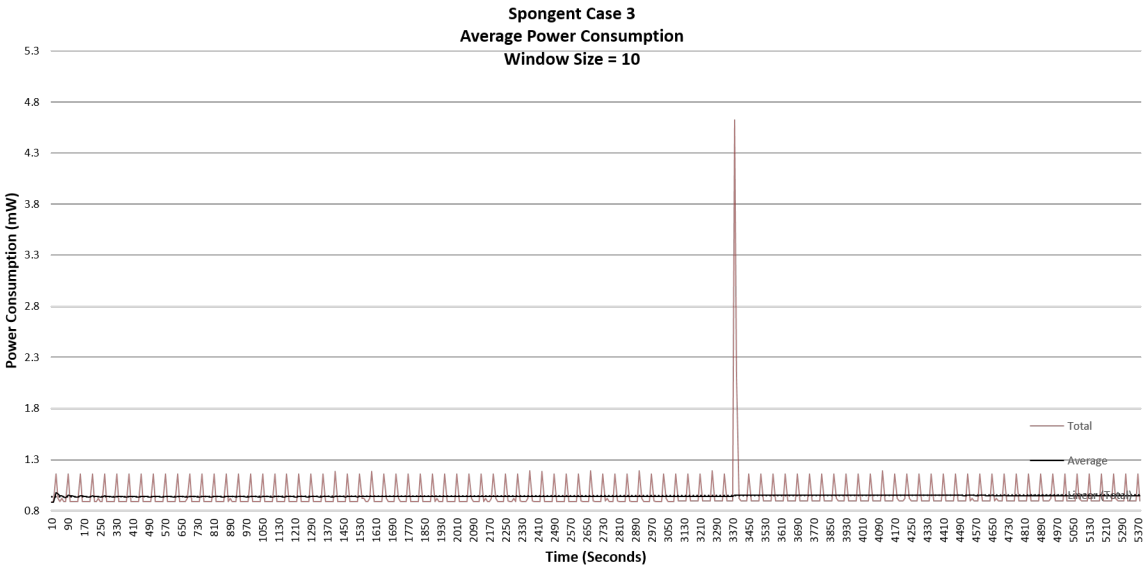

Figure E.29: Steady State Graph Spongent Case 3

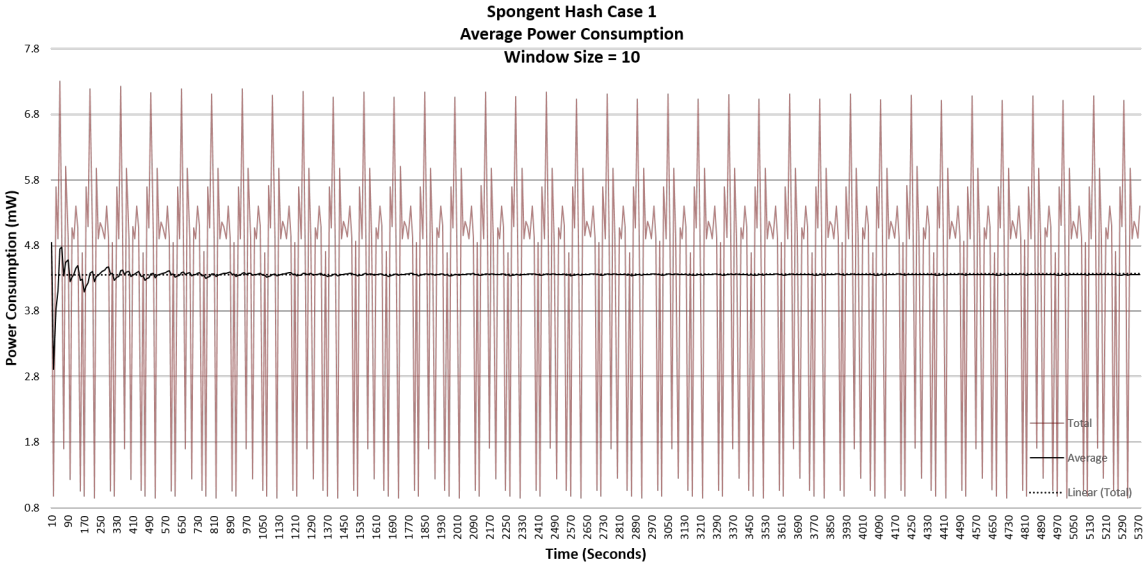

Figure E.30: Steady State Graph Spongent Hash Case 1

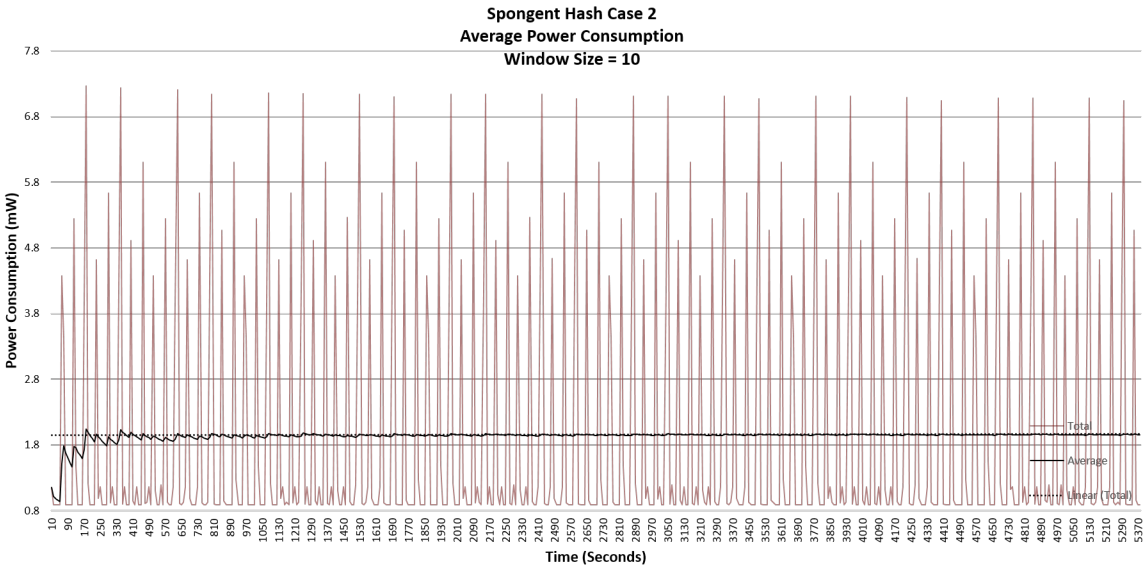

Figure E.31: Steady State Graph Spongent Hash Case 2

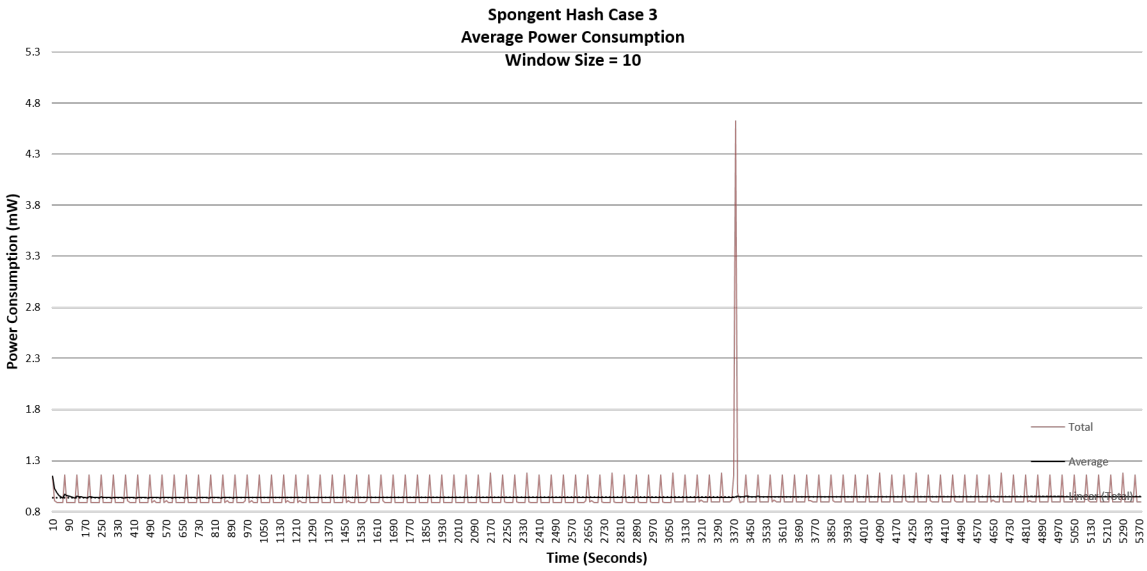

Figure E.32: Steady State Graph Spongent Hash Case 3