2023

Energy-efficient HMAC for wireless communications

Cesar Enrique Castellon Escobar
c.castellon@unf.edu

Energy-Efficient HMAC for Wireless Communications

by
Cesar Enrique Castellon Escobar

A thesis submitted to the School of Computing in partial fulfillment of the requirements for the degree of
Master of Science in Computer Science - Cybersecurity

University of North Florida
College of Computing, Engineering and Construction

April 2023

This thesis titled ***Energy-Efficient HMAC for Wireless Communications***, submitted by ***Cesar Enrique Castellon Escobar*** in partial fulfillment of the requirements for the degree of Master of Science in Computer Science - Cybersecurity , has been:

Approved by the thesis committee: **Date:**

Swapnoneel Roy — 4/18/2023

Swapnoneel Roy, Ph.D.
Associate Professor of Computing
Committee Chair

O. Patrick Kreidl — 04/17/2023

O. Patrick Kreidl, Ph.D.
Associate Professor of Electrical Engineering
Committee Member

Ayan Dutta. — 4/18/2023

Ayan Dutta, Ph.D.
Associate Professor of Computing
Committee Member

ACKNOWLEDGEMENTS

This research study was supported in part by NSF CPS 1932300 and CYBER FLORIDA 220408 grants.

The author would like to express deep gratitude to Dr. Swapnoneel Roy, Dr. Patrick Kreidl, and Dr. Ayan Dutta for their invaluable contributions that played a critical role in the successful completion of the research on energy-efficient computing solutions in energy-constrained environments. Their unwavering support and wise guidance laid the foundation for the investigation and their expertise improved the quality of the work. Moreover, their inputs and insights have inspired the author to pursue further research in the cybersecurity realm.

The author also expresses his gratitude to the University of North Florida's Information Technology Services (ITS) department and Graduate School, particularly Mr. John Sharp and Mrs. Megan Kuehner, for providing financial and professional support throughout his academic journey. He appreciates the opportunities given and looks forward to utilizing the skills and knowledge gained to make a positive impact in his field.

To my beloved and thriving family: Monica, Samuel, and Andres.

Psalms 37:4-5

Contents

List of Figures

ABSTRACT

This thesis introduces the Farming Lightweight Protocol (FLP) optimized for energy-restricted environments that depend upon secure communication, such as multi-robot information gathering systems within the vision of "smart" agriculture. FLP uses a hash-based message authentication code (HMAC) to achieve data integrity. HMAC implementations, resting upon repeated use of the SHA256 hashing operator, impose additional resource requirements and thus also impact system availability. We address this particular integrity/availability trade-off by proposing an energy-saving algorithmic engineering method on the internal SHA256 hashing operator. The energy-efficient hash is designed to maintain the original security benefits yet reduce the negative effects on system availability.

A simulation environment was created to represent several FLPs of practical character, each utilizing HMAC in a consistent manner assuming inputs of configurable size. We then conducted simulation experiments to test our energy-saving algorithmic engineering method for HMAC computations. Using the RAPL API from Intel, we measured computational energy for each input size and FLP protocol variant under study. Our results show that our method reduces energy usage by 11% on average, while maintaining the core capabilities of the FLP protocol without compromising security performance.

Chapter 1

Introduction

1.1 Engineering Motivation

Quality control inspections are crucial in various industrial sectors such as manufacturing, transportation, electricity, infrastructure, and agriculture to prevent the sale of defective products, lower costs, and reduce waste. While automated inspections are necessary due to high production rates, hazardous materials, and small part dimensions, some factories still rely on manual inspections despite the risk of human errors. Additionally, quality control inspections are used to improve operational value and resource efficiency through new automation technologies, but their implementation involves challenges such as new costs, redefined objectives, and redesigned data analytics, known as ”smart inspection” in contemporary applications.

Modern ”Internet-of-Things (IoT)” technologies, such as cutting-edge sensors, cameras, portable drones, and digital communications, are a major component of ”smart inspection” solutions. These technologies do, however, present a number of technical difficulties, including attacks from enemies and failures due to different faults. To fully realize the advantages of the connected ecosystem of smart inspection solutions (i.e.Figure 1.1), fault-tolerance and cybersecurity issues must also be resolved. Based on how they affect the system’s general Confidentiality, Integrity, and Availability, these threats are categorized. (CIA).

Figure 1.1: Diverse communicating robots in smart field operations

1.2 Technical Motivation

In Figure 1.2, a basic Smart Farming system is presented. The devices and main server are connected in a logical star topology to a communication beacon through direct physical or logical connections. When necessary, they can also connect to each other in a peer-to-peer fashion.

Due to the direct connections between the beacon and the ”smart” source devices, critical security issues arise. For instance, an attacker may try to eavesdrop on the communication between actors to obtain information about sensing measurements. Alternatively, an attacker may tamper with control instructions to cause harm to a plantation, such as path disruption or avoidance of specific sectors.

Encryption and digital signatures are well-established mitigations for such security communication vulnerabilities. However, these measures incur additional communication and computation costs, such as memory, processor, and power, particularly in a limited/fixed energy-budget application

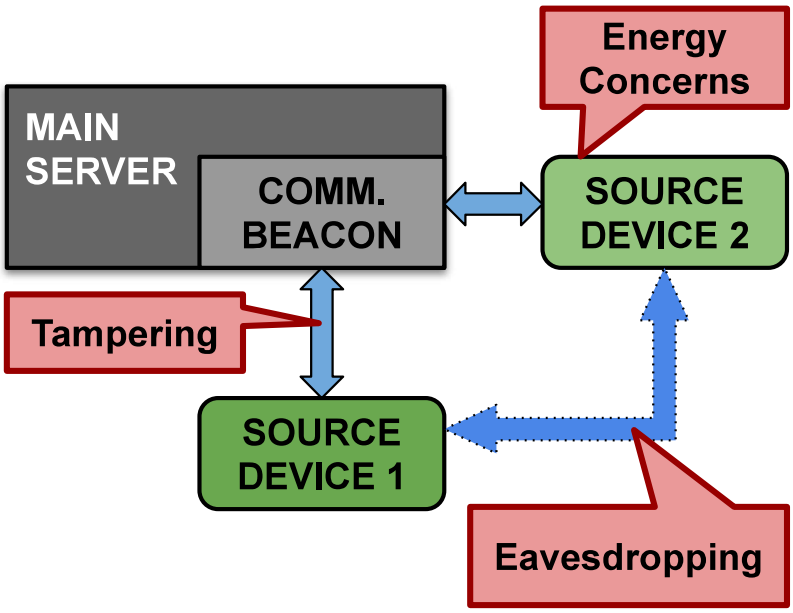


Figure 1.2: SmartFarming : Basic Block Diagram

environment like the IoT system that uses robots to inspect a crop farm's fields. Addressing the upcoming trade-off between performance, energy consumption, and the diverse security measures implemented in the deployed devices becomes challenging in this setting, where energy is the fundamental measure of resource consumption associated with any given implementation.

1.3 Thesis objectives

The key objective of this thesis is to investigate and consider viable strategies for reducing the energy consumption of Hash-based Message Authentication Code (HMAC) powered protocol while maintaining the basic security features associated with message digest generation. The current research will look specifically at these methods for applications with limited or fixed power budgets, such as those related to smart inspection in the Internet of Things (IoT) systems context.

The current study suggests a Farming Lightweight Protocol (FLP) as a first step toward ensuring the integrity and confidentiality of data exchange in bidirectional communication between two entities. In order to further enhance the energy efficiency of the system, an established Energy

Complexity Model [27] will be employed to evaluate a key algorithmic component in the Hash-based Message Authentication Code (HMAC) Message Digest generation, specifically the core cryptographic hashing algorithm. This approach's primary goal is to reduce energy consumption while keeping the required degree of security and reliability in the data exchange process.

Subsequently, in order to measure and quantify the energy reduction achieved by the proposed energy-efficient algorithmic engineering approach, a proprietary software energy measurement tool interface, known as Running Average Power Limit (RAPL) [20], will be utilized to assess the baseline and different implementations included that with the optimized hashing mechanism . The goal of this analysis is to validate our proof-of-concept and the effectiveness of our method in reducing energy consumption without compromising system functionality and security.

1.4 Thesis organization

This thesis consists of three chapters. Chapter 2 is a reproduction of a previously accepted paper at the 2022 International Green and Sustainable Computing Conference (IGSCC) [8], which analyzes the impact of energy savings in HMAC by conducting a synthetic run on a simulation test-bed involving successive calculations. Chapter 3 presents a comparable approach and outcomes for an ECM-optimized-HMAC Farming Lightweight Protocol (E-FLP) run and provides a quantitative and qualitative evaluation. Finally, Chapter 4 concludes the document and provides recommendations for future research in this area.

Chapter 2

Towards an Energy-Efficient Hash-based Message Authentication Code (HMAC)

This chapter is a reproduction of Castellon et al. [8] accepted and presented in the 2023 IEEE 13th International Green and Sustainable Computing Conference. The article's authors are Cesar Castellon, Swapnoneel Roy, O. Patrick Kreidl, Ayan Dutta and Ladislau Boloni.

2.1 Introduction

Hashed-based Message Authentication Code (HMAC) is a well known machine authentication code extensively used in different cybersecurity applications. Advocates for such uses cite the HMAC's ability to provide both integrity and authentication. [4, 30, 47, 24, 39]. One particularly promoted use case is the *Transport Layer Security (TLS)* [4, 47, 28, 1, 34], which is the standard, widely deployed protocol for securing client-server communications over the Internet. The Transport Layer Security (TLS) protocol, sometimes referred to, the Secure Sockets Layer (SSL) protocol, is a stateful, connection-oriented, client-server protocol. TLS is the most widely deployed communications security protocol on the Internet, providing confidentiality, integrity and authentication for parties in communication.

Other usages of HMAC include error correction codes [23], remote attestation [46], VANETs [39, 11, 29, 31], security and privacy in systems [19], and Internet of Things (IoT) [43].

A balance exists in the amount of security, performance, and energy consumption one wants to achieve. In general, increasing security generally causes additional energy consumption while decreasing performance. Finding perfect balance between energy consumption, performance and security level — becomes a challenge in HMAC based applications, where supply of energy is often very limited [25].

To be viable for an application that values autonomy for greater lengths of time, any system must be configured to make more efficient use of energy. Disabling HMAC integrity and authentication will certainly save energy, but also weaken security: it is in such contexts that the exploration of ways to reduce energy consumption of HMAC alone can be of tremendous practical significance.

2.1.1 Related Work

Energy efficiency in computation is a widely studied topic, with numerous points of view: hardware specific platforms, operating systems, hypervisors and containers [45]; software development and security [14]; and algorithms [36, 37]. Energy measurements are sometimes obtained by uniquely instrumented equipment [35], while other times can leverage hardware providers' Application Programmer Interfaces (APIs) in which firmware counters are recalled to provide near realtime information e.g., Running Average Power Limit (RAPL) technology [40]. Energy-efficient (*greener*) HMAC implementations have been actively researched as a means to improve overall energy efficiency in secured systems [32]. These research have mostly focused to optimize energy efficiency of HMAC in a variety of scenarios (e.g., scheduling [2, 13, 48], system software [3, 18, 44], hardware implementation [17, 12], and hypervisors [16]). However, all of these works treat HMAC as a black box and to the best of our knowledge, no research exist in the literature that deals directlywith the hash algorithms of HMAC and attempts to engineer them to reduce energy consumption in HMAC.

2.1.2 Our Scope and Contributions

We study the extent to which the underlying hash function (SHA256), a principal element of HMAC, can be made more energy efficient. Our approach employs an energy-reducing algorithmic engineering technique, based upon an Energy Complexity Model (ECM) proposed by Roy et al. [36, 37], on the SHA256 encryption algorithm, which is central to HMAC.

Using pyRAPL, a python library to measure an executable's Runtime Average Power Limit, we experiment with both the standard and energy-reduced implementations of HMAC for input sizes (in bytes) that are commonly seen within applications using HMAC. Our results show significant reductions in energy consumption, up to 13.5% but on average around 12.7% across the tested input sizes. At present, it is only a conjecture that reduced energy consumption in the HMAC module itself extrapolates to comparable reduction of an application using HMAC on the whole. In any case, to the best of our knowledge our work is the first to address energy optimization of HMAC by engineering the implementation of one of its component algorithms (SHA256). Moreover, the proposed energy-reducing technique is similarly applicable to other key elements of a secured system, potentially affording even "greener" secured application systems than implied by only the HMAC results obtained thus far.

This paper builds on top of research done in [9]. While [9] experiments on energy efficiency of Merkle Trees in Blockchain, this work experiments on energy efficiency of HMAC, a different algorithm. Also, part of this work was done while the first author pursued his Masters in EE as presented in this M.S thesis [7].

2.2 Methodology

An Energy Complexity Model (ECM) [36, 37] has been applied to the underlying SHA256 function of HMAC. We first describe how the general HMAC function works, followed by a brief discussion of the ECM and its application to the underlying SHA256 of HMAC.

2.2.1 HMAC Message Digest Generation

As mentioned before, HMAC implements both integrity checking and authentication of messages using cryptographic hash functions. Any hash function (e.g. MD5, SHA128, SHA256, etc.) can be used in HMAC combined with a shared secret key. HMAC's strength cryptography-wise is dependent on the strength of its underlying hash function [21, 6].

Fig. 2.1 shows a graphic representation of a simple HMAC message digest generation. The input to HMAC is a message M containing $\ell - 1$ blocks ($Y(1) \cdots Y(\ell - 1)$), each of size b. A signature S_i is concatenated to the left of M before it is input to the underlying hash function (e.g. SHA256) to produce a temporary message digest MD'. MD' is further concatenated with output signature $S_o = K^+ \oplus PAD$, which is then hashed again using the underlying hash (e.g. SHA256) to produce MD, the final message digest.

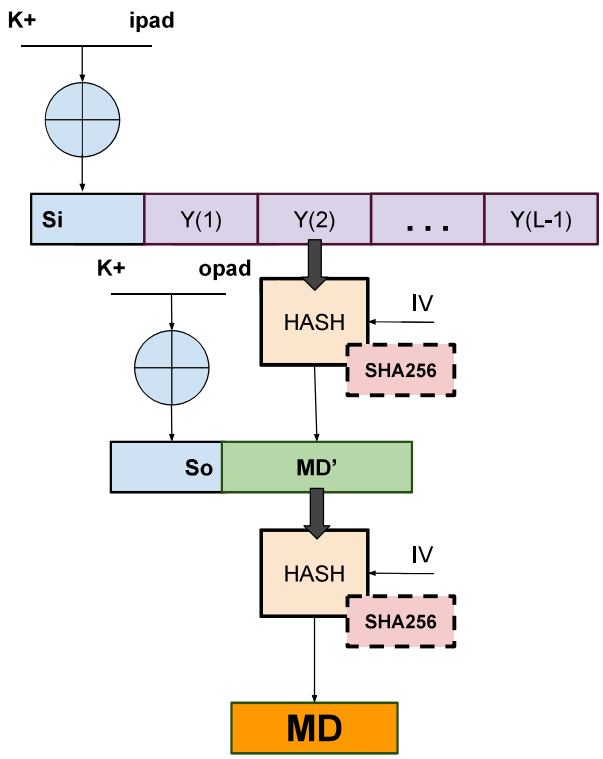


Figure 2.1: Basic HMAC generation

For a recap, in Fig. 2.1, HASH stands for the hash function function (SHA256), M is the input message, S_i and S_o are respectively the input and output signatures, $Y(i)$ is the i^{th} block of M, i ranges from $[1,\ell)$, ℓ is the number of blocks in M. K is the secret key used for the hash. IV is an initial vector (constant values used by SHA256).

2.2.2 The Energy Complexity Model (ECM)

The Double Data Rate Synchronous Dynamic Random Access Memory (DDR SDRAM) is the reference architecture for the energy complexity model (ECM) [36], which has applied to HMAC in this work. As illustrated in Fig. 2.2, the main memory of DDR is divided into *banks*, containing a fixed number of *chunks*[1].

Allocation of data happens in each bank over chunks. Additionally, every bank contains a special chunk called the *sense amplifier*. For any data access, the chunk containing the data to be accessed has to be brought inside the corresponding bank's sense amplifier. Each sense amplifier can house one chunk at a given time, so the present chunk has to be returned to its bank before a new one can be brought in for the next access. At a given time, therefore, only one chunk of a given bank can be accessed; however, chunks of different banks can be accessed in parallel (within each bank's own sense amplifier). Hence, for a P bank DDR memory (e.g., $P = 4$ in Fig. 2.2), at any point of time we can access P chunks. The sense amplifier is called per-bank cache in DDR3 version of the DDR architecture.

The P banks of a given DDR3 SDRAM resource is denoted by $M_1, M_2, \ldots, M_P$ by the ECM. There are multiple chunks of size-B (in bytes) and a cache C_i respectively in each bank M_i. Fig. 2.3 illustrates an example with $P = 4$ banks similar to the case in Fig. 2.2 with each bank having only four chunks. Labels in numbers $1, 2, \ldots, 16$ were assigned to the chunks. Given the constraint in DDR that a single chunk may be put inside a given cache C_i at any time, examples of completely

[1]The term "block" is used in DDR specifications, but we use the term "chunk" to avoid confusion within our HMAC context.

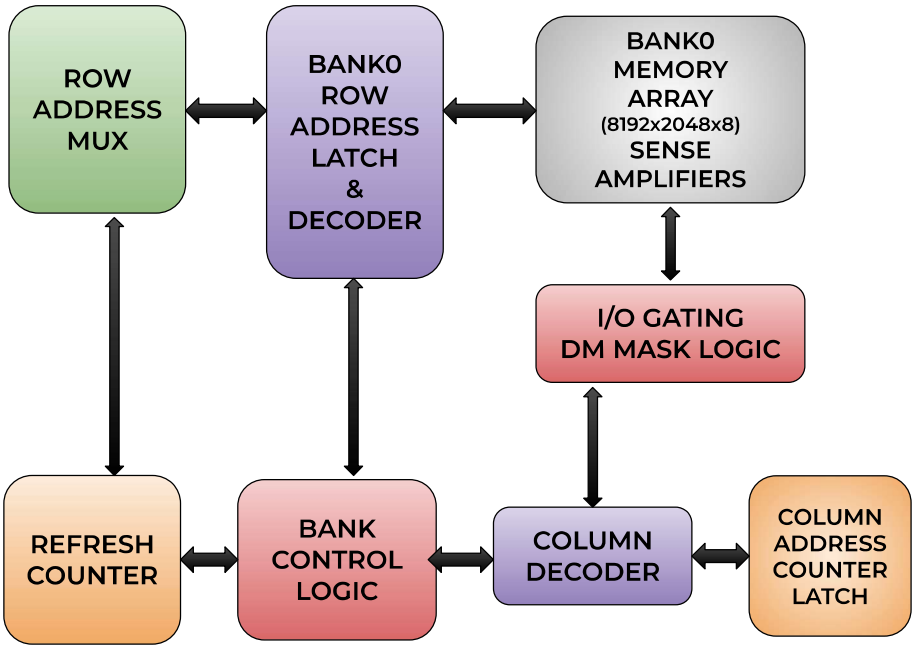


Figure 2.2: Internal DDR SDRAM memory chip block diagram.

serial execution are the access patterns $(1,2,3,4)$ or $(5,6,7,8)$, while $(1,5,9,13)$ or $(3,8,10,13)$ are examples of completely parallel execution. The authors of [36] discovered (1) accessing same number of chunks (sequentially or in parallel) account for very similar amount of power consumption and (2) execution time of an algorithm is reduced significantly when chunks are accessed in parallel than when chunks are accessed sequentially. Since energy consumption of an algorithm is dependent on both time and power, it was implied that energy consumption in any algorithm is potentially reduced by parallelizing chunk accesses during the execution of that algorithm. Formally, as derived by Roy *et al.* [36], the energy consumption (in Joules) of an algorithm $\mathscr{A}$ with execution time τ, assuming a P-bank DDR3 architecture with B bytes per chunk, is given by

$$E(\mathscr{A}) = \tau + (P \times B)/I \tag{2.1}$$

where the so-called *parallelization index* is denoted by I, which is essentially the number of parallel

block accesses across memory banks per P block accesses made by $\mathscr{A}$ on the whole. In other words, an algorithm's potential for energy reduction is inversely proportional to the degree it can be engineered for parallelization of its memory accesses, according to ECM.

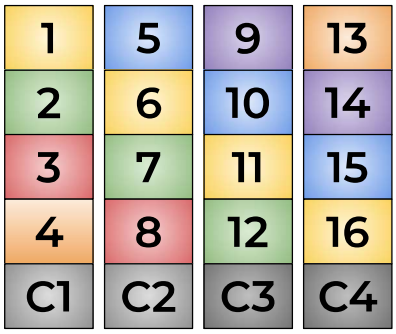


Figure 2.3: ECM for DDR3 Resource with $P = 4$ Banks

2.2.3 Engineering Hash Calculations Using ECM

The energy consumption of the underlying hash algorithm (SHA256) of HMAC has been reduced by engineering it based on ECM in this work. First, how any algorithm $\mathscr{A}$ can be parallelized based on ECM is described. Then we illustrate how SHA256, the underlying hash algorithm for HMAC is engineered for parallelization based on ECM.

Parallelizing any algorithm

For algorithm $\mathscr{A}$, the most common access sequence of $\mathscr{A}$ on execution for a given input is first identified. The vector formed by this access sequence is then engineered to achieve the desired level of parallelism by framing a logical mapping over chunks of memory that store data accessed by $\mathscr{A}$. Physical location of the input (chunks) is static in the memory and is controlled by the memory controller of DDR. But order of access over chunks is different for different levels of parallelization. Different page table vectors $\mathbf{V}$ is framed each time for implementing different levels of parallelization of access over physical chunks. $\mathbf{V}$ defines the ordering of access among chunks (Fig. 2.4).

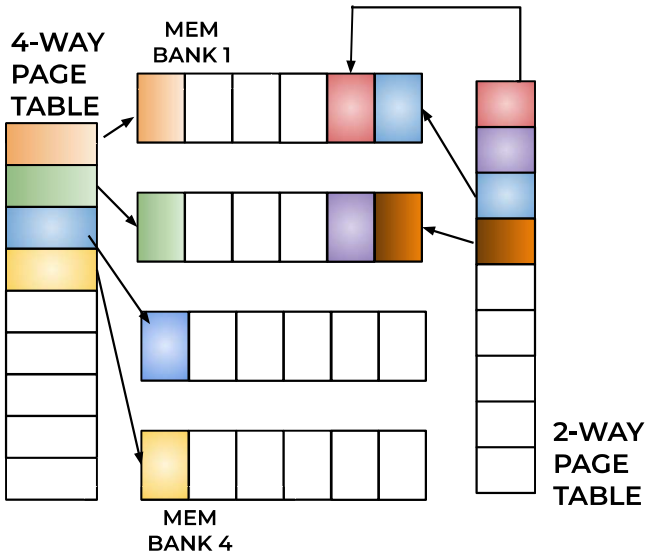


Figure 2.4: Memory Layout ($P = 4$) and Role of Page Tables

For 1-way access, the page table vector $\mathbf{V}$ has the pattern $(1,2,3,4,\ldots)$ and for 4-way access it has the pattern $(1,5,9,13,\ldots)$. A function is then created to map the pattern of the page table vector $\mathbf{V}$ to the original physical locations of the input. Algorithm 1 shows the function to create an ordering among the chunks. The ordering is based on the way we want to access the chunks (P-way would mean full parallel access). The page table is populated by picking chunks with *jumps*. For P-way access, jumps of P are selected that ensure the consecutive chunk accesses lie in P different banks. Going by the above example, for $P = 1$, jumps of 1 ensure that 4 consecutive chunk accesses lie in the same bank (bank 1 of Fig. 2.3). On the other hand, for $P = 4$, jumps of 4 ensures that 4 consecutive chunk access lie in 4 different banks (banks 1 through 4 of Fig. 2.3).

Algorithm 1: Create a Page Table for N Chunks

Input: Page table vector $\mathbf{V}$, jump amount *jump*.

$factor = 0$;

for $i = 0$ *to* $\frac{N}{B} - 1$ **do**

  if $i > 1$ *and* $(i \times jump) \mod \frac{N}{B} = 0$ **then**

    $factor = factor + 1$;

  end

  $\mathbf{V}_i = (i \times jump + factor) \mod \frac{N}{B}$;

end

Parallelizing SHA256

As illustrated in Fig. 2.1, HMAC generates the final message digest (MD) by applying SHA256 twice. The SHA256 algorithm partitions its input into fixed size message blocks, presented in sequence to separate compression functions, as shown in Fig. 2.5. This block sequence is identified in correspondence with the access pattern of the SHA256 algorithm, which we subject to engineering based on the ECM. The SHA256 input vector (see Fig. 2.5), is pre-processed into another vector by applying Algorithm 1. The mapping is then stored in a page table to be used in subsequent hash calculations. An example of this operation for 16 blocks and a parallelization index (jump) of 4 is shown in Fig. 2.3.

Fig. 2.6 shows the outcome of engineering the SHA256 algorithm based on ECM. In our experimentation, an 8-bank DDR3 SDRAM is used and the parallelization index is set to $I = 8$. This essentially means that for any set of eight consecutive block access in SHA256, we created a virtual mapping using techniques described in [37] to ensure that each size-8 access occurs across all eight banks.

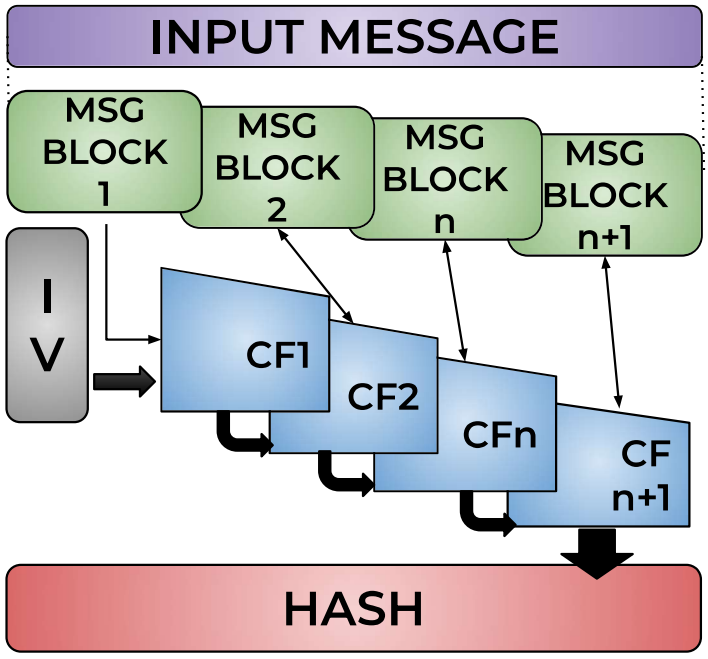


Figure 2.5: The SHA256 Algorithm

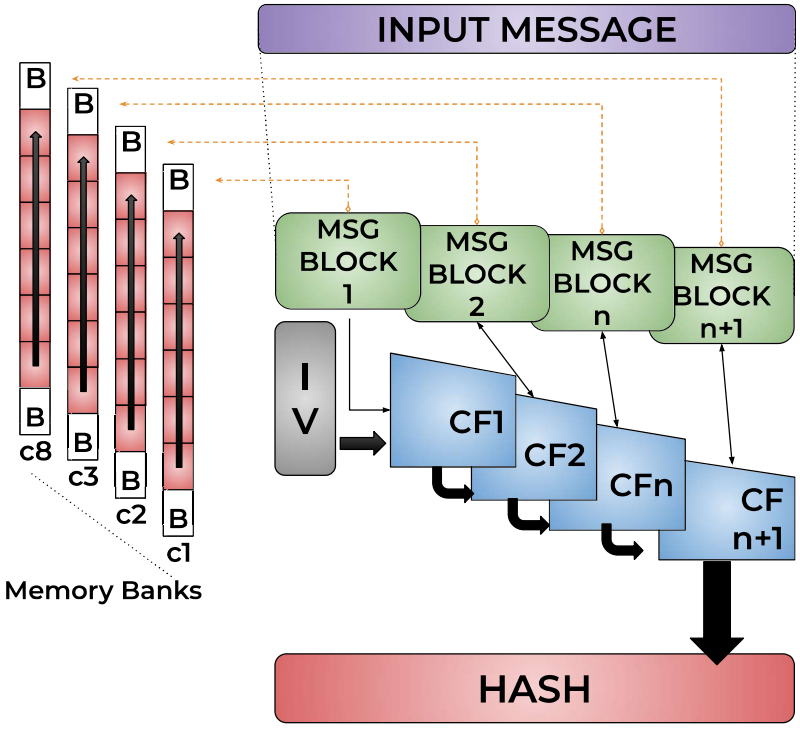


Figure 2.6: ECM-Enhanced SHA256

Theorem 1. *The engineered SHA*256 *algorithm has the same computational complexity as the original SHA*256 *algorithm.*

Proof. SHA256 has a computational complexity of $\Theta(N)$, where N is the number of blocks in Fig. 2.5 [33]. Algorithm 1 has a computational complexity of $\Theta(N)$ since the for loop of line 2 executes exactly $\frac{N}{B}$ times. Therefore applying Algorithm 1 to SHA256 as illustrated in Fig. 2.4 does not change the overall computational complexity of SHA256. □

2.3 Experiments

This section describes experiments performed to measure energy efficiency of HMAC out of the engineering illustrated in the previous section. The ECM required a hardware with a DDR RAM architecture. According to the ECM, maximum energy efficiency is attained by the parallelization

index set to the number of memory banks, which depends upon the DDR version: 4 for DDR2, 8 for DDR3 and 16 for DDR4 and higher. We used a machine with a 64-bit dual-core processor (Intel i5-2410M @ 2900MHz with cache size L2 256KB and L3 3072KB), running Linux Mint version 19.3 with a 8GB DD3 RAM and 500GB SSD storage. Also, pyRAPL, a software toolkit, was used to measure the host machine's energy footprint along the execution of Python code for comparing energy consumption between HMAC with standard and ECM-enhanced of the underlying SHA256. pyRAPL is built upon Intel's Running Average Power Limit (RAPL) technology that estimates a CPU's power consumption; depending on the hardware and operating system configurations, pyRAPL can measure energy consumption of the following CPU domains: CPU socket, GPU, and DRAM [40].

2.3.1 Implementation Details and Setup

Standard and ECM-enhanced versions of the SHA256 algorithm have been implemented in two different C language programs, these are called from a master Python program via the `ctypes` module) as an external command. This permits the use of Python pyRAPL to measure energy events having at the same time the low-level memory control to implement the ECM-enhanced SHA256 functionality.

Our experiments simulated the HMAC calculation with Python code that runs one complete round of Message Digest generation having pyRAPL methods invoked yielding in a single energy measurement per event. Since measurement implementation is subject to noise we have invoked 1000 repetitions for the process and report the average energy (mean and deviation). Our experiments also vary the input size (i.e., the message size) to the HMAC calculations, choosing 64, 128, 256, 384, 512, 768, and 1024 bytes motivated by having standard message's not to exceed traditional MTU limit of 1500 bytes and selecting standard steps of size increase.

2.3.2 Results and Discussion

Our experimental setup features two implementations of HMAC calculations, the standard one (which we label by "O" as it uses the standard SHA256) and the engineered one using ECM (which we label by "E" as it uses the energy efficient SHA256), as well as seven different input sizes.

Per implementation and per input size, our experimental Python script leverages the pyRAPL toolkit to measure the average energy (mean and deviation over 1000 trials) of simulated HMAC calculations. Fig. 2.7 summarizes the seven average energy measurements in a bar chart, per input size comparing the Standard HMAC (O) and the Enhanced HMAC (E) average energy (in μJoules). We observe that the ECM-enhanced implementation consistently consumes less energy that the standard implementation.

The first set (Fig. 2.7) compares average energy measurements between the energy efficient ('E') and the original ('O') HMAC implementations, starting from input size 64 byte to 1024 bytes. To summarize, the ECM-engineered HMAC shows an average energy consumption increment of around 100% with increase in input size from 64 byte to 1024 bytes.

The standard ('O', non-enhanced) model in comparison showed an average energy consumption increment of around 75% over the same input sizes. For example, with input sizes of 64,128, and 256 bytes, the average energy consumption are 8380 and 9600 $\mu Joules$ for 'E' and 'O' respectively, while for input size 1024 bytes, they are 14866 and 17213 $\mu Joules$ respectively. It can be concluded that memory parallelism implementation in SHA256 based on ECM has an overhead impact on energy consumption of HMAC. This is in line with the ECM model proposed in [36].

Fig. 2.8 presents average energy savings on more relative terms, namely as a percent reduction achieved by the ECM-enhanced implementation over the standard implementation of HMAC over all seven input sizes. The energy savings for the smaller input sizes range between $12-13\%$, while

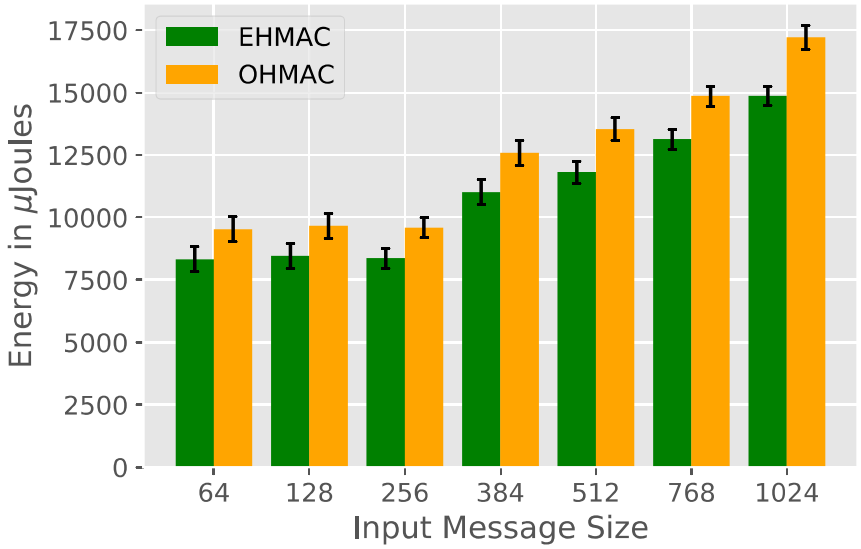


Figure 2.7: Comparison of Average Energy Consumption in HMAC per Message Size (with 1-sigma standard deviation over 1000 trials)

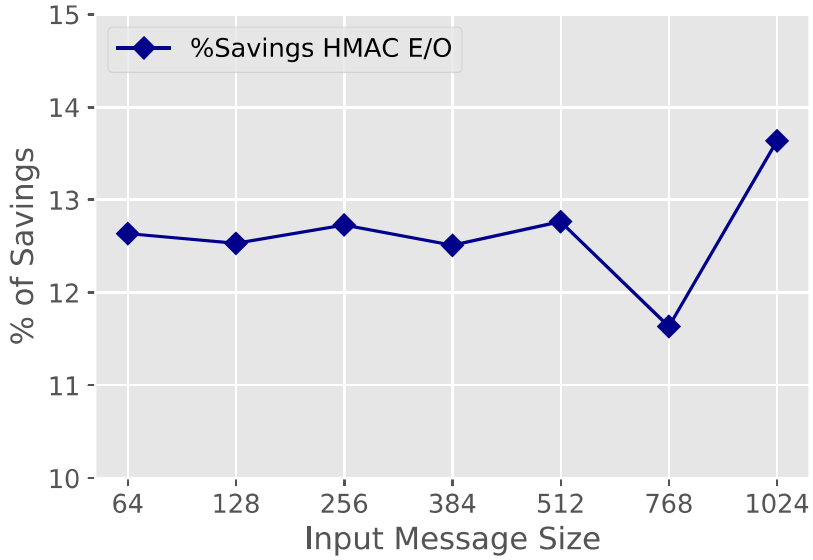


Figure 2.8: Percentage of Energy Saving per process in HMAC per Message Size

the energy savings for the larger input sizes range between 13% and 14%,. As observed, the 768B input renders a savings lower than the average, yet it is still around the margins over 10% for single operation.

Chapter 3

Energy considerations in HMAC secured communications

3.1 Peer to Peer communications

Wireless peer-to-peer (P2P) communication is a type of wireless communication that allows devices to share data without the use of an intermediary network infrastructure [42]. Unlike traditional wireless communication, which relies on a central hub or base station to relay data between devices, peer-to-peer communication enables devices to connect directly to one another and share data in a decentralized way.

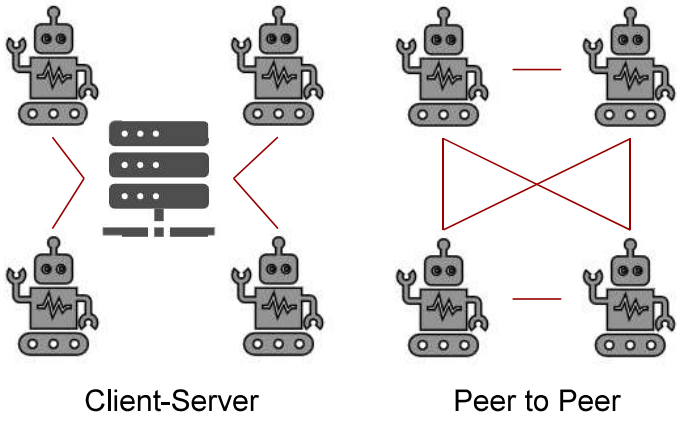


Figure 3.1: Client-Server vs. Peer to Peer

The proliferation of mobile devices with wireless networking capabilities, such as cellphones and

tablets, has fueled the rise of wireless P2P communication. In today's society, these devices have become ubiquitous, and users increasingly expect them to be able to interact with one another seamlessly and without interruption. Wireless P2P communication helps to meet these expectations by allowing devices to share information, cooperate on tasks, and communicate in real time.

There are numerous advantages to wireless P2P transmission. For one thing, it eliminates the need for a centralized network infrastructure, which can be costly and challenging to implement in some environments. Furthermore, P2P communication is more resilient and reliable than traditional wireless communication because devices can interact with one another even in the absence of a central hub or base station. Furthermore, because data is not routed through a central intermediary who could possibly intercept or eavesdrop on the communication, P2P communication can be more secure and private than conventional wireless communication.

Despite these advantages, wireless P2P communication poses a number of obstacles. One major issue is device compatibility, which arises when devices use different wireless communication protocols or are unable to establish a direct link with one another. Another issue is scalability, as P2P communication becomes more complex and challenging to manage as the number of devices increases.

This section offers an overview of important topics related to the implementation of peer-to-peer information exchange. The discussion starts with an examination of raw socket communications as a fundamental component of peer-to-peer communications schema. Following that, the section goes into a discussion of the widely used cryptographic mechanism, HMAC, which is used to verify message authenticity and integrity. In addition, a breakdown of the widely used secure communication scheme, TLS/SSL, is provided. Finally, the section ends with an explanation of the proposed Farming Lightweight Protocol's fundamentals.

3.1.1 Raw socket communications

Raw socket communication is a type of network communication that provides direct access to the network stack, allowing applications to send and receive packets at the raw protocol level. Raw socket communication is often used for network monitoring and debugging, as well as for implementing custom network protocols. This paper provides an overview of raw socket communication, its basic principles, and its practical applications. [15]

Raw socket communication allows applications to bypass the transport layer protocols (See Fig 3.2 below), such as TCP and UDP, and interact directly with the network layer protocols, such as IP and ICMP. This provides applications with fine-grained control over network traffic and enables the implementation of custom network protocols.

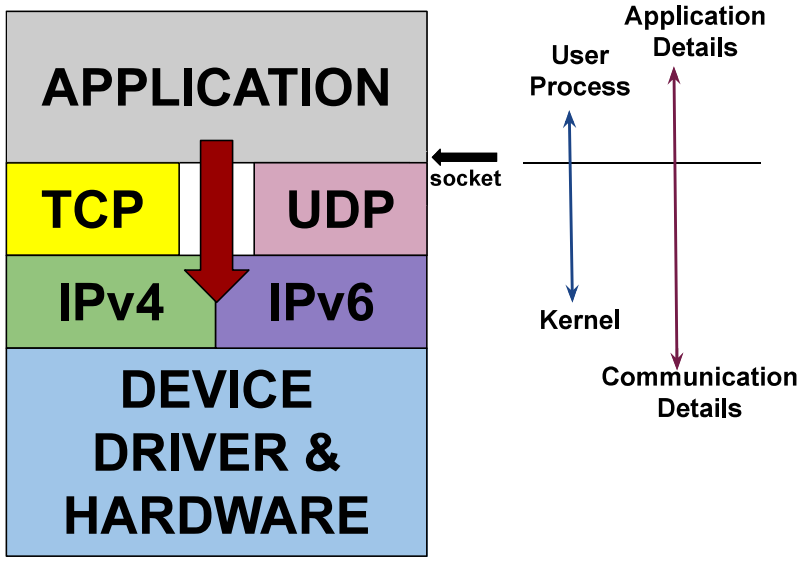


Figure 3.2: RAW socket communications

Raw socket communication is implemented using the socket API, which provides a set of system calls for creating, binding, and communicating with sockets. To create a raw socket, an application specifies the protocol family as AFPACKET and the protocol as ETHPALL, which indicates that the socket should capture all packets, regardless of their protocol type.

Once a raw socket is created, an application can send and receive packets using the sendto() and recvfrom() functions, respectively. When sending a packet, the application specifies the destination address and protocol type. When receiving a packet, the application receives the raw packet data, including the protocol header and payload.

Raw socket communication has a wide range of practical applications, including network monitoring, packet capture, and custom protocol implementation. Raw socket communication is often used by network administrators to monitor network traffic and detect anomalies, such as packet loss or excessive latency.

Raw socket communication is also used for packet capture, which is the process of intercepting and analyzing network traffic. Packet capture is often used for debugging network issues, analyzing network performance, and detecting security threats.

In addition, raw socket communication is used for implementing custom network protocols. For example, an application might use raw sockets to implement a custom protocol for transferring data between two systems, bypassing the standard transport layer protocols and providing greater control over the transmission process. Yet, it may result in additional complexity when resolving logical multiplexing with concurrent applications and sequencing operations for flow and error control.

3.1.2 TCP/UDP Communications

Solving some of the complexity that a raw socket communication may involve, there exists two protocols used in Layer4(Transport) fur such purpose , those are User Datagram Protocol (UDP) and Transmission Control Protocol (TCP).

The User Datagram Protocol (UDP) is a connectionless protocol for transmitting data across a network. Unlike TCP, UDP does not guarantee error checking or data integrity, making it a less reliable but faster choice for data transmission. Instead, UDP prioritizes speed and efficiency,

making it perfect for real-time communication in applications such as video conferencing and online gaming.

UDP transmits data in the form of packets to a particular IP address and port number. In contrast to TCP, there is no connection establishment procedure, so data can be sent immediately without any initial delay. However, there is no way to guarantee that the data was received correctly or in the right order.

While UDP is faster than TCP, it has some drawbacks. Because it lacks error checking and reliability assurances, it is vulnerable to packet loss and data corruption, which can degrade the user experience. As a consequence, to ensure that data is transmitted quickly and reliably, UDP is usually used in conjunction with other protocols that provide these guarantees, such as TCP or the Real-time Transport Protocol (RTP).

TCP is a basic protocol of the Internet Protocol Suite that serves as a dependable, connection-oriented mechanism for transmitting data over the internet. Its main function is to divide data into packets and ensure that these packets are transmitted and received correctly, ensuring end-to-end data integrity [42].

TCP uses a three-way handshake procedure to establish a connection between two endpoints, sending and receiving SYN, SYN-ACK, and ACK packets to ensure that both endpoints are ready to communicate. TCP uses a sliding window method to transmit data once the connection is established. This method includes sending a specified amount of data, or a ”window,” and then waiting for the receiver to acknowledge before sending the next window, preventing data overload.

TCP’s congestion control mechanism, which monitors the network for signs of congestion and dynamically adjusts the transmission rate to avoid network congestion, is one of its most important characteristics. TCP helps to keep the stability and reliability of the communication’s flow, by utilizing algorithms such as slow start and congestion avoidance; along with the congestion control mechanism of the protocol , the system ensures that packet loss and degraded performance are

minimized.

3.1.3 Hash-based Message Authentication Code (HMAC)

Hash-based Message Authentication Code (HMAC) [22] is a widely used cryptographic mechanism for verifying the integrity and authenticity of messages. HMAC provides a way for two parties to ensure that a message has not been tampered with or altered during transmission. This paper provides an overview of HMAC, its basic principles, and its practical applications.

HMAC is a mechanism for generating a message authentication code using a cryptographic hash function and a secret key. HMAC works by combining the message with a secret key, and then hashing the result using a cryptographic hash function, such as SHA-256 or SHA-512. The resulting hash value is the HMAC, which can be used to verify the integrity and authenticity of the message. HMAC is designed to be a computationally secure mechanism for verifying message integrity and authenticity. The security of HMAC is based on the assumption that the cryptographic hash function is secure and that the secret key is kept secret. HMAC is widely used for message authentication in a variety of applications, including secure communication protocols, such as SSL/TLS, and digital signature schemes, such as RSA. HMAC is also used for data integrity checks in file transfer protocols, such as FTP and SCP.

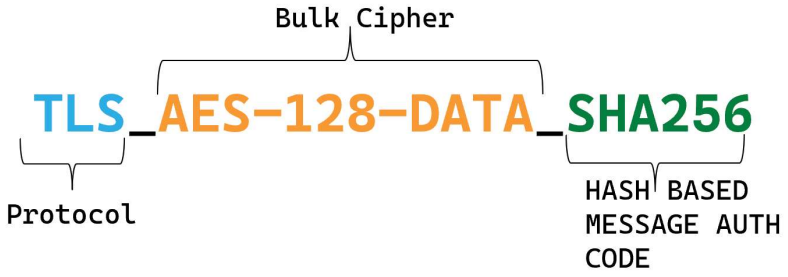


Figure 3.3: TLS/SSL DATA with HMAC [41]

In secure communication protocols, such as SSL/TLS (Figure 3.3), HMAC is used to ensure the integrity and authenticity of the data transmitted between the client and server. When a message is sent, the sender generates an HMAC using a secret key and a hash function.

The receiver then generates an HMAC using the same key and hash function, and compares it with the HMAC received from the sender. If the HMACs match, the receiver can be sure that the message has not been tampered with during transmission.

In digital signature schemes, such as RSA, HMAC is used to ensure the authenticity of the signature. When a message is signed, the sender generates an HMAC using a secret key and a hash function, and then encrypts the HMAC using the sender's private key. The receiver can then verify the signature by decrypting the HMAC using the sender's public key, and comparing it with the HMAC generated from the message using the same key and hash function.

3.1.4 Secure communications over IP - SSL/TLS

The Secure Socket Layer (SSL) and its successor, Transport Layer Security (TLS), are cryptographic protocols used for securing internet communication. SSL and TLS are used to encrypt data between two entities, such as a web server and a client, to prevent third-party eavesdropping or tampering. [10]

SSL was first introduced by Netscape in the 1990s as a way to secure internet communication. In 1999, SSL was updated to version 3.0, which was later standardized as TLS. TLS has since been updated several times, with the latest version being TLS 1.3, which was released in 2018.

SSL/TLS works by encrypting data that is transmitted over the internet. The encryption process is initiated when a client requests a secure connection to a server. The server responds by sending a digital certificate that contains a public key. The client then uses the public key to encrypt a session key, which is used to encrypt all subsequent communication between the client and server.

SSL/TLS uses a combination of symmetric and asymmetric encryption. Symmetric encryption is used to encrypt data using the session key, while asymmetric encryption is used to encrypt the session key itself.

This two-step process provides an additional layer of security, as it ensures that even if the session key is intercepted, the encrypted data cannot be decrypted without the private key associated with the public key used to encrypt the session key.

SSL/TLS is widely used to secure online transactions, such as those performed on e-commerce websites. SSL/TLS is also used to secure other types of communication, such as email and instant messaging. In addition, SSL/TLS is used to secure connections between web servers and web browsers, as well as connections between servers in a distributed system.

SSL/TLS has become the most deployed protocols for secure communications over data networks; the evolution of such protocols now comprehensively addresses several facets of communication that need to be authenticated, encrypted, and authenticated encryption. One of the underlying records of TLS is the MAC record, generated by a cryptographic hash function (nominally SHA256, although SHA-1 and MD5 can also be used) and is used to create an SSL payload, showing a promise of energy savings if modifying the underlying SHA library for the E-SHA [8] version would permit us to have robust and secure communications for our devices.

Nevertheless, after examining the SSL/TLS handshaking operation portrayed in Figure 3.4, it became evident that the phased handshake consisting of three distinct stages would add complexity to the working code of the robots. The first stage involves Establishing Security Capabilities, with a decision process implemented at each node to determine which parameters to accept.

The next stage is Server Authentication and Key Exchange, which requires validation by a centralized Certification Authority. Finally, there is Client Authentication, which necessitates a unique process of certificate exchange and verification.

As a result, this elaborate protocol would require more communication resources, resulting in higher energy consumption. Therefore there is an evident need of a simpler yet secure approach detailed in the following section.

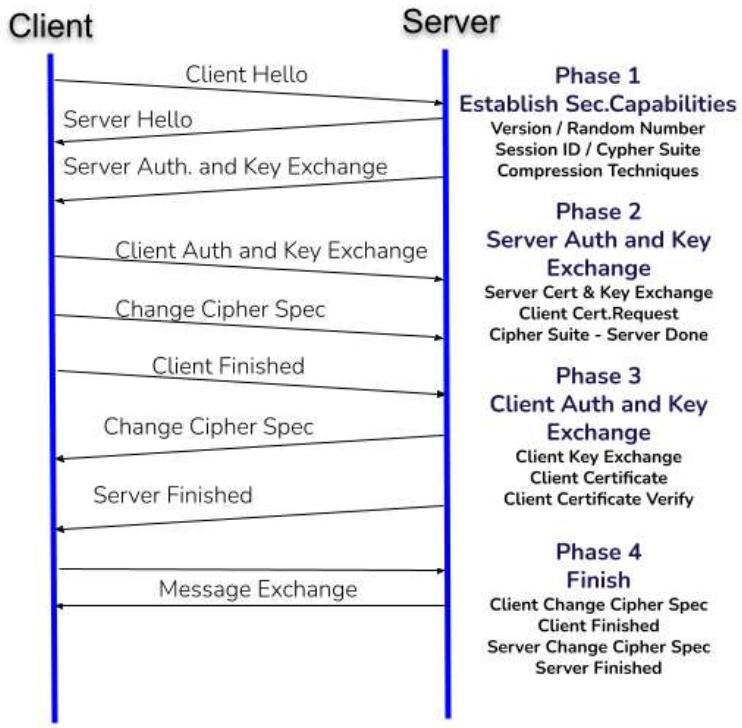


Figure 3.4: SSL/TLS handshaking protocol operation [41]

3.1.5 Farming Lightweight Protocol (FLP)

The current project introduces the Farming Lightweight Protocol (FLP) as a potential solution to address the need for message integrity validation in a multi-robot collaboration environment as presented in the work of Samman et al. in [38]. The scenario involves multiple surveying robots collecting information and sharing it to improve the prediction accuracy. The proposed FLP aims to secure the exchanged messages between the robots and enhance the authenticity of the data.

For this implementation we assume the following conditions :

- Authentication will be leveraged to other mechanisms already present in the solution (either passcode, timestamp, temporary one-time password (OTP), or even a message from an authentication blockchain),
- For communication's security, we let Layer1 and/or Layer2 address it with any appropriate mechanism (e.g., scrambling, modulation) while transmitting.

- Authorization will depend on the robot's resident program , the data reading and validation; within a successful authentication and a correctly resolved challenge, incoming data is processed, and only then data is transmitted outbound.

The designed FLP is depicted in Figure 3.5 below, this is a data-sharing scenario where at first Robot1 senses the vicinity of Robot2 (condition to transmit):

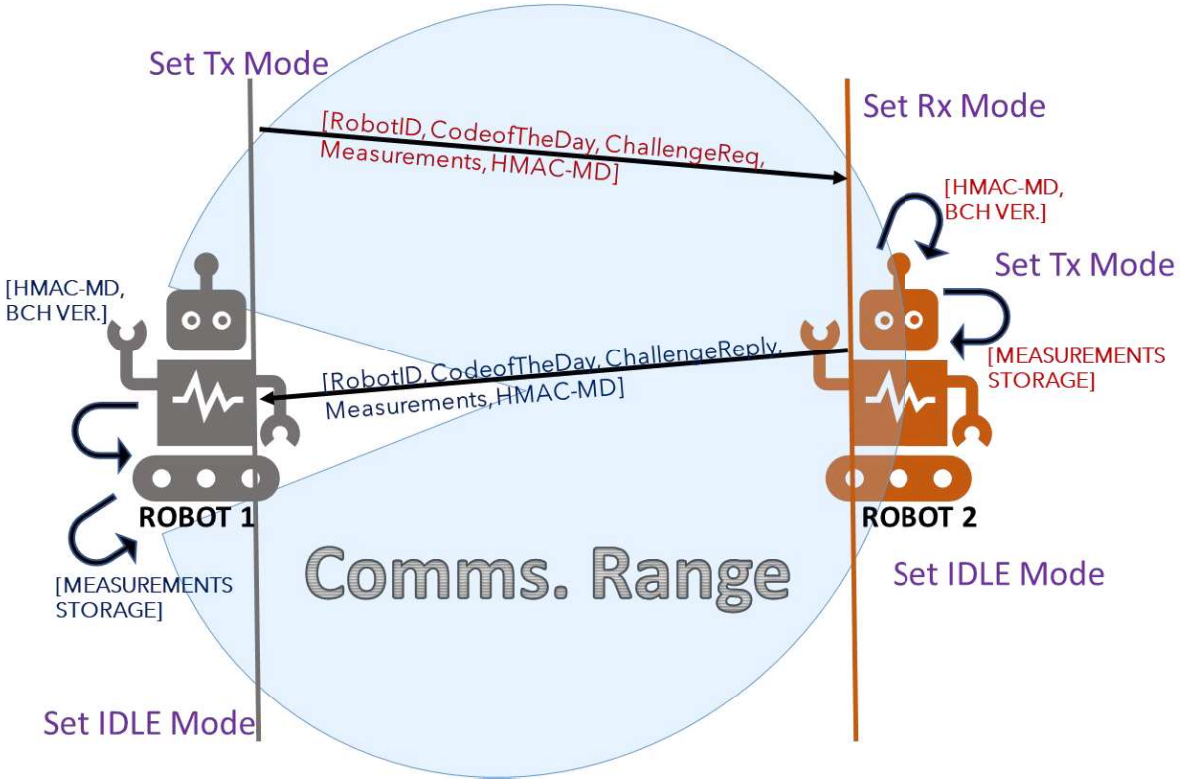


Figure 3.5: Basic FLP message flow

The process after the initial sensing is as follows:

1. Robot1 sends a datagram containing RobotID, CodeofTheDay, ChallengeReq, ChallengeReply,Measurements, HMAC-MD.

2. Robot2 receives the message, calculates the HMAC-MD of the message received, and immediately verifies the information against its own BlockChainLedger.

   If successful, it processes the Measurements and sends a reply datagram containing the

following information: RobotID, CodeofTheDay, ChallengeReply, Measurements, HMAC-MD.

If it is not verified, Robot2 drops the message and sets itself in IDLE mode.

3. Robot 1 receives Robot 2's message, performs the same verification and storage tasks, and sets itself in IDLE mode.

   If there is no answer within a timer, having that Robot2 is still in the vicinity, Robot1 will send a new message with updated information after a contention timer ends.

Message Fields

The information exchanged between the two devices are comprised of the following fields.

1. RobotID : The Device's Network identification in Hexadecimal Format (16 Bytes)
2. Code of the Day : Random Code Generation for Daily operation in Hexadecimal Format (16 Bytes)
3. Challenge Request : Input Data1 + OperatorCode + InputData2 (16Bytes + 16 Bytes + 16 Bytes) . OperatorCode specifies type of operation to be performed (loaded at start of daily operations)
4. Challenge Reply : Result of operation (16 Bytes long)
5. Measurements : All the measurements information available (padded to 128Bytes)
6. HMAC-MD : Overall Message Digest (32 Bytes)

Architecture

Figure 3.6, shows the communication layers for a data transmission between two peers (in the figure Robots 1 and 2):

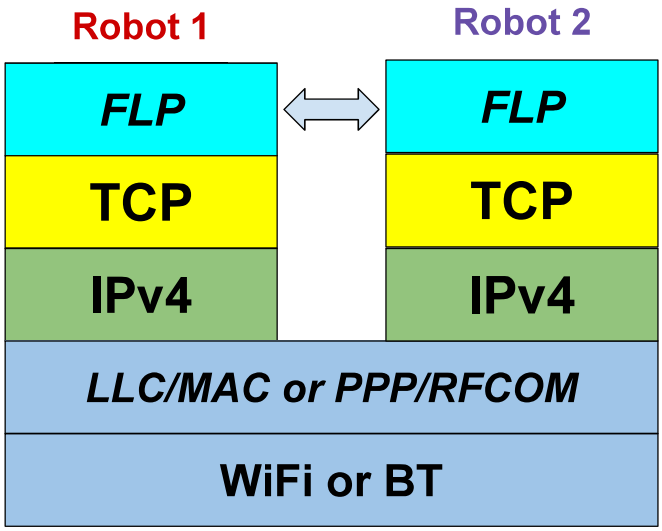


Figure 3.6: FLP Protocol Architecture

We see that at Physical and Link layer we could implement various technologies such as WiFi or Bluetooth, although other technologies like RFID could also be implemented we are sticking to the mostly deployed ones.

The decision to keep IPv4 in the network layer provides both with flexibility on addressing the nodes and in addition means of securing the access to other network instances by means of a stateful firewall.

While transport would support s both the option of a RAW socket or a TCP managed communication, it would depend on the type of connection we require to implement any of those, for our study we will keep TCP. Finally we implemented the FLP on top of this layer , the FLP protocol can be hardenized using HMAC or left as is for the insecure version.

Work development phases

FLP simulation implementation went through the following phases :

- Implemented a SSL/TLS version (Secure FLP / S-FLP).

- Replaced SSL/TLS for a TCP-socket version with HMAC running a standard SHA256 C library (Original FLP / O-FLP).Compared to Figure 3.3, modification occur in the data transmitted format, as shown in Figure 3.7 below

Figure 3.7: FLP DATA with HMAC

- Replaced the standard SHA256 C library with the ECM enhanced SHA256 (Enhanced FLP / E-FLP).

3.2 Experiment Setup

This subsection describes experiments where we measured the energy required for each one of the implementations proposed in the previous subsection.

The application of ECM, simmilarly to the setup mentioned *section* 2.2, uses as well a DDR RAM hardware. The simulations were run on a Linux Mint 20.1 machine with Python 3.8 and the PyRAPL module installed, and C code compiled with GCC 8.3.1 20190507. The simulation hardware included a 64-bit dual-core CPU (Intel i5-2410M @ 2900MHz) with L2 256KB and L3 3072KB caches, 8GB of DD3 RAM, and a 500GB SSD for storage.

The basic operation of the wrapper program is depicted in Figure 3.8. The program controlling the experiments is shown as the Python block on the left, it then calls the C program block on the right for running the hashing mechanism(e.g., either O or E - SHA256), and then records the measurements.

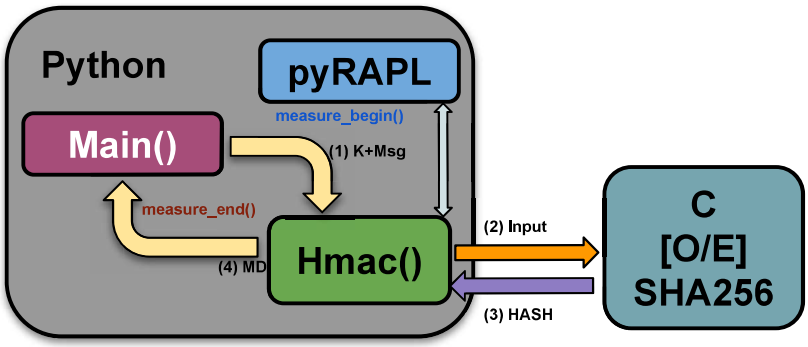


Figure 3.8: Python wrapper for PyRAPL energy measurements [8]

Since energy measurement implementation is subject to noise, we have invoked 100 repetitions for each process and reported the average energy (mean and deviation).

3.3 Evaluation and Results

We have defined possible metrics we will gather for evaluating measurement results; those are the average and standard deviation of :

- Process duration in $\mu seconds$. Obtained from PyRAPL as an internal calculation of timestamps (time elapsed since the epoch).
- Energy consumed in process in $\mu Joules$. List of the CPU energy consumption (per socket).
- Computed Power consumed in $Watts$. Calculated value Energy per Unit of time.

Given the close relationship between the metrics, we focused our data processing and interpretation efforts on the energy use dataset.

Phases of experiments

Recalling Section 3.2 we prepared following measurements scenarios

- Basic HMAC with [Standard hash function] (O-HMAC). vs. item Enhanced HMAC with

[ECM optimized hash function](E-HMAC). [8]

- Secured FLP with [TLS/SSL] Robot1-Robot2 Data Exchange (S-FLP).
- Original FLP with standard [HMAC256] Robot1-Robot2 Data Exchange (0-FLP).
- Enhanced FLP with ECM optimized [HMAC256] Robot1-Robot2 Data Exchange(E-FLP).

Results

- **O-HMAC and E-HMAC**

  Recalling the results by Castellon et al. [8] with input sizes of 64,128, and 256 bytes, the average energy consumption were 8380 and 9600 $\mu Joules$ for 'E' and 'O' respectively, while for input size 1024 bytes, they were 14866 and 17213 $\mu Joules$ respectively. In these tests the energy savings for the smaller input sizes ranged between $12-13\%$, while the energy savings for the larger input sizes ranged between 13% and 14%,

- **S-FLP , O-FLP and E-FLP**

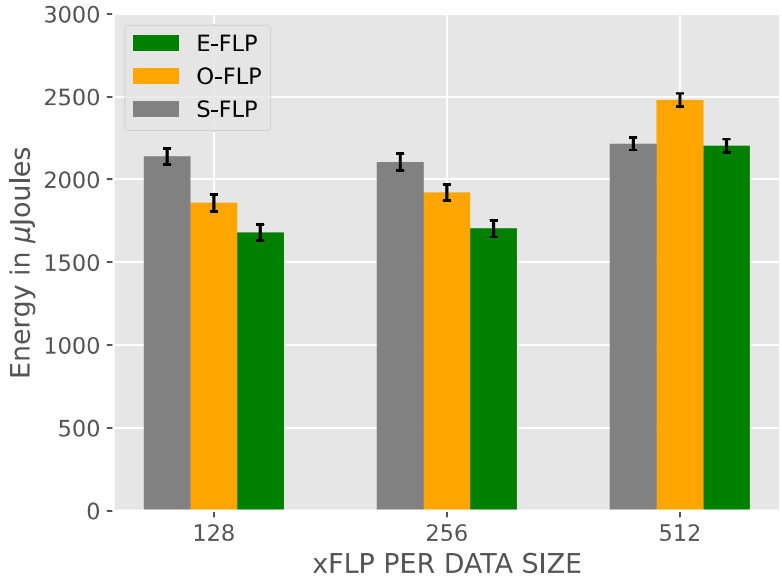


Figure 3.9: Average Energy Consumption in S-FLP , O-FLP and E-FLP per Message Size (with 1-sigma standard deviation over 100 trials)

The Secured FLP ('S,' secured) protocol, in comparison showed an average energy consumption increment of around 20% over the exact input sizes as it can be seen in Figure 3.9. The implementation of TLS/SSL for inter-robot communications has an overhead impact on energy consumption per communication cycle. Figure 3.9 shows the measurements for the SHA256 HMAC-powered FLP. It can be seen that the O-FLP ('O', Original) protocol, in comparison, showed an average energy consumption increment of around 26% over the exact input sizes, whereas the E-FLP showed a constant 23% increment.

Concerning one another, E-FLP has proven to be more energy efficient in a 10% average than O-FLP ; therefore, we can conclude that implementation of E-SHA256 for secure inter-robot communications has an overhead impact on energy optimization per communication cycle as shown in Figure 3.10 below.

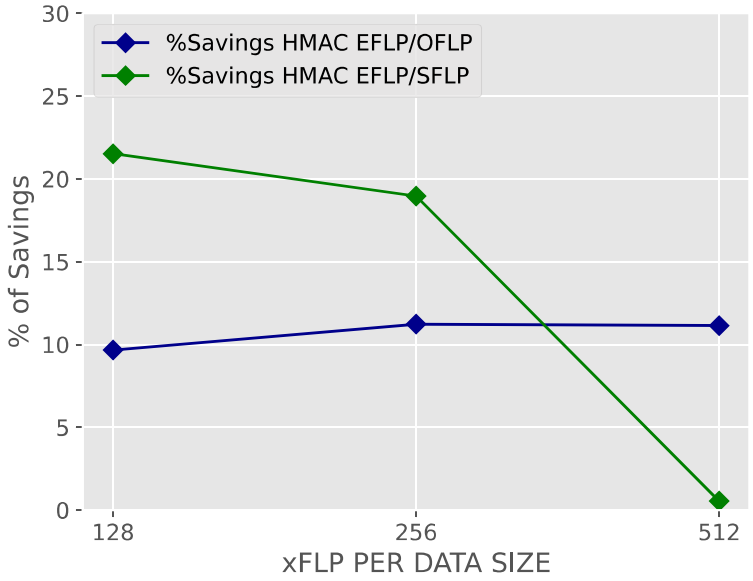


Figure 3.10: Average Savings E-FLP vs. O-FLP and S-FLP per Message Size

- **Statistical Analysis of measurements**

E-FLP vs. O-FLP dataset were selected for further statistical analysis. For the statistical analysis, energy in μJoules was measured for 100 runs of a communication algorithm with 8-way parallelism. Each test runs the algorithm with different inputs of the same size to

avoid caching of results in order to make results independent for statistical analysis.

Since it is not feasible to determine if the measurements are uniformly distributed, we found it suitable to use the Mann-Whitney U-Test for the statistical analysis. The Mann-Whitney U Test is a non-parametric test of the null hypothesis that " for randomly selected values X and Y from two populations, the probability of X being more significant than Y is equal to the probability of Y being more significant than X." [36]

OFLP/EFLP Descriptive Statistics			
Energy Consumed	Runs	Mean	STD.DEV
	100	181.32	2.12
	MIN	MAX	VARIANCE
	175.38	186.36	4.51

Table 3.1: OFLP and EFLP Measurement's Statistical Indicators

In Table 3.1, it can be observed that the variance of range (maximum–minimum) is lower than the mean value. Thus, the standard deviation shows a higher spread as it is nearly half the variance value. This shows that the data is not uniformly distributed and confirms that we could not conduct a t-test. Minitab Statistics software was used to generate the rank table shown in the following Table 3.2. A higher mean value in this Table indicates higher energy consumption.

The results show that O-FLP consumes higher energy than E-FLP and the difference is statistically significant.

OFLP/EFLP RANKS				
Energy Consumed	GROUP	N	Mean Rank	Sum of Ranks
	EFLP	100	8.5	136
	OFLP	100	24.5	392

Table 3.2: OFLP and EFLP Ranks

3.3.1 Energy Savings Extrapolation over Real-World Systems

McNulty et. al in [26] establishes the energy consumption for unmanned vehicle devices (UVD) to be divided into three subsystems: (1) Navigation, (2) Sensing, and (3) Locomotion as shown in Fig. 3.11.

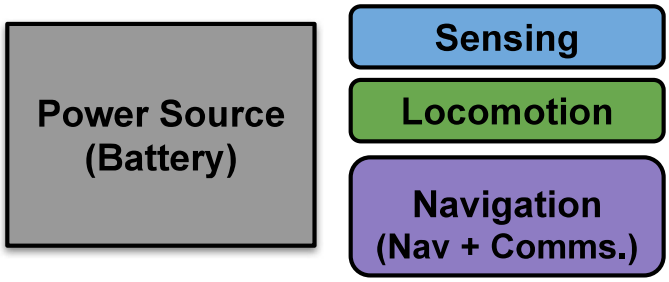


Figure 3.11: Block diagram of energy consumption sources for Unmanned devices.

In our work, the communications module is part of the navigation subsystem. The total energy consumption of an unmanned device can be therefore expressed by the following equation.

$$ET = EL + EP + EN \tag{3.1}$$

where ET stands for the total energy consumption, EL, EP, and EN stands for the energy consumption respectively by the Locomotion, Sensing, and Navigation. Furthermore, EN can be expressed as follows.

$$EN = Egps + Ecomm \tag{3.2}$$

where $Egps$ stands for energy consumed by the global positioning system, and $Ecomm$ for the energy consumed by the communications . Finally Ecomm can be expressed as

$$Ecomm = Etrx + Esec \tag{3.3}$$

where $Etrx$ stands for energy consumed by the Transmission/Reception process, and $Esec$ stands for energy consumed by the security tools implemented. $Esec$ in Equation 3.3, will be $0Joules$

for a system with no security. On the other hand, $Esec$ adds to the total energy consumption for security applications implemented in the system. For our estimation process we will use as input some important values contained in the specifications sheet of any UVD. Those values are:

- Battery Power Capacity (BWh) expresed in [wh]
- Current Output (I) expressed in [mA]
- Operating Voltage (V) expressed in [V]
- Maximum Distance Autonomy (Dmax) expressed in [m]

We can now calculate the Total Energy (TE) Stored at the battery as follows [5]

$$TE_{[Joules]} = I_{[mA]} \times V_{(V)} \times 3600 \tag{3.4}$$

Then the Energy needed by the robot to cover one meter (REL) is :

$$REL_{[Joules/m]} = TE_{[Joules]} / Dmax_{[m]} \tag{3.5}$$

$Dmax$ for any UVD has a direct relation to TE, if it is affected by any source of energy drain, then the maximun distance the device will cover will be lesser than the specified $Dmax$ value, we will refer to this affected distance as $Dmin$.

$$Dmin < Dmax \tag{3.6}$$

We can then calculate the effect any power drainage has in Distance ($Dcost$) by dividing the total amount of energy drained by other sources(Ecost) over the REL coefficient

$$Dcost_{[m]} = Ecost_{[Joules]} / REL_{[Joules/m]} \tag{3.7}$$

Given that the current scenario involves the use of a blockchain as a security mechanism, $Ecost$ will have a component of energy used in the generation of every blockchain block, that is a complete operation of mining (Merkle Tree generation and Proof of Work calculation). We assume that every time a UVD robot communicates with other robot a block is generated consuming a certain amount of energy ($Eblock$).

In [8] the authors measured the energy needed for a HMAC Message Digest generation for both scenarios ,that is with the inner SHA256 ECM optimized (EFLP) or with the original SHA256 library (OFLP). Additionally ,the energy consumption for a SSLTLS operation (SFLP) is presented. All of this values are for a 256 Bytes input size.

$$OFLP = 1,916$$

$$EFLP = 1,672$$

$$SFLP = 2,128$$

Resulting $Ecost$ as a function of number of blocks ($Bnum$) generated can be calculated as follows:

$$Ecost_{[Joules]} = Eblock_{[Joules]} \times Bnum_{[units]} \tag{3.8}$$

$Bnum$ will depend in how many times throughout $Dmax$ displacement the UVD exchanges information with other device in the vicinity, this value depends inversely to the average distance ($step$) displaced between one and the subsequent information exchange.

$$Bnum_{[units]} = Dmax_{[m]} \times step_{[m]} \tag{3.9}$$

We will then calculate $Ecost$ using $O-SHA$ first, and later $E-SHA$ to generate values using

previous equations and then calculate *Dmin* by using

$$Dmin_{[m]} = Dmax_{[m]} - Dcost_{[m]} \tag{3.10}$$

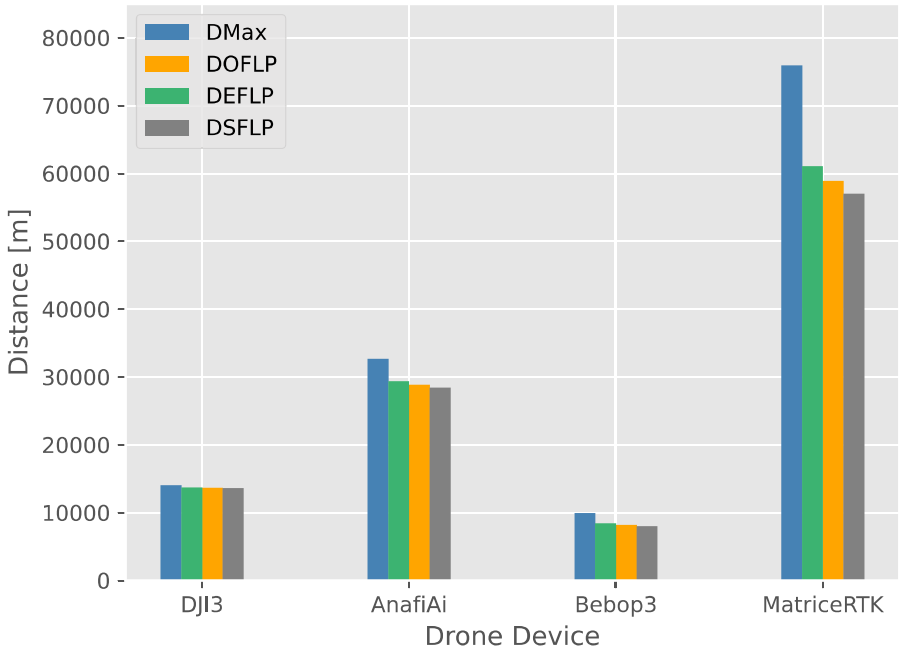
Following Table 3.3.1 shows the calculations of *Dmax* and *Dmin* labeled as "D-OFLP" when using the O-FLP for $Ecost$, "D-EFLP" when using the E-FLP for $Ecost$, and "D-SFLP" when using the TLS for $Ecost$ for a single round of a Field Survey; we also assumed that one communication takes place every 5m ($step = 5$) the robot traveled.

Type	Name	Dmax	D-OFLP	D-EFLP	D-SFLP
Drone	DJI3	14,000	13,652	13,697	13,614
Drone	Anafi Ai	32,640	28,850	29,332	28,430
Drone	Bebop 3	9,900	8,161	8,383	7,969
Drone	Matrice RTK	75,900	58,866	61,036	56,982
Robot	TurtleBot4	7,200	5,667	5,862	5,498
Robot	Jackal	28,800	28,064	28,158	27,983
Robot	Husky	10,800	10,593	10,619	10,570
Robot	TurtleBot Waffle Pi	1,872	1,804	1,813	1,797

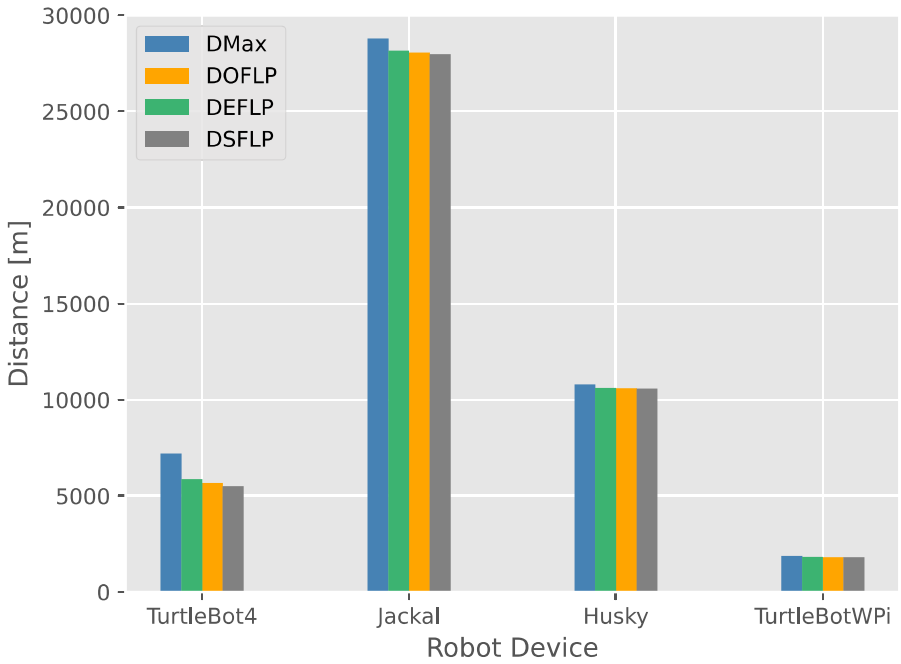
Table 3.3: UVD Distance autonomy

The "Unsecured" *Dmax* version may have a longer distance autonomy than any of the "Secured" ($D-SFLP$,$D-OFLP$,$D-EFLP$) implementations. Nevertheless, it is still the $E-FLP$ secured version that has a better distance autonomy than the other two secured implementations as it can be seen in Figure 3.12 .

The variation in distance autonomy, i.e., $Dmax - Dmin$, indicates the compromise between resources and capabilities involved in securing a system. To determine the cost-benefit ratio of preventing a security breach that could potentially affect a node (e.g., a robot), revenue assurance calculations are necessary. However, this preliminary analysis does not consider such an assessment, as other factors may also be relevant for a comprehensive security evaluation.



(a)



(b)

Figure 3.12: Estimated distance autonomy per implementation

Chapter 4
Recommendations and Conclusion

This study considers implementing an energy-optimized lightweight communication protocol for energy-constrained environments, such as smart farming robot applications. We label this concept the Farming Lightweight Protocol, or FLP for short.

For this purpose, we proposed a low-load protocol for information sharing among robots, leveraging tasks such as authentication and authorization to other components of the overall solution. The first step implemented a standard SSL/TLS protocol for communication security having FLP embedded into the ciphered payload. Furtherly, we took out FLP and combined it with HMAC for message authentication moving away from SSL/TLS. Lastly we implemented an energy-optimized HMAC for FLP integration.

This optimized HMAC reduces the energy consumption of the HMAC by engineering the underlying hashing algorithm (SHA256) based on the Energy Complexity Model (ECM) [36]. We measured and evaluated the HMAC ECM-enhanced implementation compared with the standard implementation via experimental energy measurements with various input sizes of practical significance. These results show approximately 12% energy savings.

In the next step, we measured and evaluated the different simulation implementations, demonstrating that a) there is a statistical correlation between measurements, and we could rule out noise that may have induced some bias in measurement, b) the ECM applied to the SHA256 integrated on the

HMAC proved to be more energy efficient than the standard implementation, and c) the operation of E-FLP represents less energy consumption compared to the other options.

For future work, there are two primary paths to follow. One is related to the swarms of robots and the energy simulation per process. This simulation can be run in a different hardware scenario (e.g., High-Performance Computers), where each processor can be set as one robot and emulate an entire survey cycle. The second path is to test E-SHA256 in IoT devices, keeping in mind that the memory architecture is DIMM4 compatible, but also considering that RAPL as a means for energy measurement is not available for processors other than Intel. However, these measurements could be addressed using a different kind of tool, such as a hardware-based or by means of establishing a correlation with elapsed time. Moreover, for any of the previous scenarios it would also be interesting to test and implement a RAW communication version of the FLP.

Lastly, other important challenges are to assess possible energy savings in other applications of HMAC (e.g., Key Derivation Functions, Secure Data Transfer Protocols, One Time Password generation, Secure Routing Protocols) and evaluate their respective trade-offs between security and resource usage.

REFERENCES

[1] Nadhem AlFardan, Daniel J Bernstein, Kenneth G Paterson, Bertram Poettering, and Jacob CN Schuldt. On the security of {RC4} in {TLS}. In *22nd USENIX Security Symposium (USENIX Security 13)*, pages 305–320, 2013.

[2] L Sherly Puspha Annabel and K Murugan. An energy efficient wakeup schedule and power management algorithm for wireless sensor networks. In *2012 International Conference on Recent Trends in Information Technology*, pages 314–319. IEEE, 2012.

[3] Anya Apavatjrut, Wassim Znaidi, Antoine Fraboulet, Claire Goursaud, Katia Jaffrès-Runser, Cédric Lauradoux, and Marine Minier. Energy efficient authentication strategies for network coding. *Concurrency and Computation: Practice and Experience*, 24(10):1086–1107, 2012.

[4] Lennart Beringer, Adam Petcher, Q Ye Katherine, and Andrew W Appel. Verified correctness and security of openssl. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 207–221, 2015.

[5] Liesle Caballero, Álvaro Perafan, Martha Rinaldy, and Winston Percybrooks. Predicting the energy consumption of a robot in an exploration task using optimized neural networks. *Electronics*, 10(8), 2021.

[6] R Canetti, M Bellare, and H Krawczyk. Keyed hash functions and message authentication. In *Proceedings of Crypto*, volume 96, 2001.

[7] Cesar Castellon. Energy considerations in blockchain-enabled applications. Master's thesis, University of North Florida, 2021.

[8] Cesar Castellon, Swapnoneel Roy, O. Kreidl, Ayan Dutta, and Ladislau Bölöni. Towards an energy-efficient hash-based message authentication code (hmac). In *13th International green and sustainable computing conference .*, 10 2022.

[9] Cesar Castellon, Swapnoneel Roy, Patrick Kreidl, Ayan Dutta, and Ladislau Bölöni. Energy efficient merkle trees for blockchains. In *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 1093–1099. IEEE, 2021.

[10] Tim Dierks and Eric Rescorla. Rfc 5246: The transport layer security (tls) protocol version 1.2, 2008.

[11] Mohamed Amine Ferrag and Lei Shu. The performance evaluation of blockchain-based security and privacy systems for the internet of things: A tutorial. *IEEE Internet of Things Journal*, 8(24):17236–17260, 2021.

[12] Christian Haas, Stephan Munz, Joachim Wilke, and Anton Hergenröder. Evaluating energy-efficiency of hardware-based security mechanisms. In *2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pages 560–565. IEEE, 2013.

[13] Jawad Ahmad Haqbeen, Takayuki Ito, Mohammad Arifuzzaman, and Takanobu Otsuka. Traffic adaptive hybrid mac with qos driven energy efficiency for wsns through joint dynamic scheduling mode. In *2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS)*, pages 3–9. IEEE, 2018.

[14] Priyanka Dhoopa Harish. *Towards Designing Energy-Efficient Secure Hashes*. PhD thesis, UNF Digital Commons, 2015.

[15] Jens Heuschkel, Tobias Hofmann, Thorsten Hollstein, and Joel Kuepper. Introduction to raw-sockets. In *Introduction to Raw-sockets*, 2017.

[16] Wassim Itani, Ali Chehab, and Ayman Kayssi. Energy-efficient platform-as-a-service security provisioning in the cloud. In *2011 International Conference on Energy Aware Computing*, pages 1–6. IEEE, 2011.

[17] Marcio Juliato and Catherine Gebotys. Fpga implementation of an hmac processor based on the sha-2 family of hash functions. *University of Waterloo, Tech. Rep*, 2011.

[18] PT Kalaivaani and A Rajeswari. An energy efficient analysis of s-mac and h-mac protocols for wireless sensor networks. *International Journal of Computer Networks & Communications*, 5(2):83, 2013.

[19] Awais Abdul Khaliq, Adeel Anjum, Abdul Basit Ajmal, Julian L Webber, Abolfazl Mehbodniya, and Shawal Khan. A secure and privacy preserved parking recommender system using elliptic curve c cryptography and local differential privacy. *IEEE Access*, 2022.

[20] K.Nizam, M.Hirki, T.Niemi, J.Nurminem, and Z.Ou. Rapl in action: Experiences in using rapl for power measurements. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, 0, 2018.

[21] H. Krawczyk, M. Bellare, and R. Canetti. Rfc2104: Hmac: Keyed-hashing for message authentication. *IETF.*, 1997.

[22] Hugo Krawczyk, Mihir Bellare, and Ran Canetti. Rfc2104: Hmac: Keyed-hashing for message authentication, 1997.

[23] Tandoh Lawrence, Fagen Li, Ikram Ali, Michael Y Kpiebaareh, Charles R Haruna, and Tandoh Christopher. An hmac-based authentication scheme for network coding with support for

error correction and rogue node identification. *Journal of Systems Architecture*, 116:102051, 2021.

[24] Julian Lettner, Benjamin Kollenda, Andrei Homescu, Per Larsen, Felix Schuster, Lucas Davi, Ahmad-Reza Sadeghi, Thorsten Holz, and Michael Franz. {Subversive-C}: Abusing and protecting dynamic message dispatch. In *2016 USENIX Annual Technical Conference (USENIX ATC 16)*, pages 209–221, 2016.

[25] Sebastian Litzinger, Oliver Körber, and Jörg Keller. Reducing energy consumption of hmac applications on heterogeneous platforms. In *2019 International Conference on High Performance Computing & Simulation (HPCS)*, pages 152–158, 2019.

[26] David McNulty, Aaron Hennessy, Mei Li, Eddie Armstrong, and Kevin M. Ryan. A review of li-ion batteries for autonomous mobile robots: Perspectives and outlook for the future. *Journal of Power Sources*, 545:231943, 2022.

[27] Esther Mengelkamp, Benedikt Notheisen, Carolin Beer, David Dauer, and Christof Weinhardt. A blockchain-based smart grid: towards sustainable local energy markets. *Computer Science-Research and Development*, 33(1-2):207–214, 2018.

[28] Christopher Meyer, Juraj Somorovsky, Eugen Weiss, Jörg Schwenk, Sebastian Schinzel, and Erik Tews. Revisiting {SSL/TLS} implementations: New bleichenbacher side channels and attacks. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 733–748, 2014.

[29] Shafika Showkat Moni and D Manivannan. Crease: Certificateless and reused-pseudonym based authentication scheme for enabling security and privacy in vanets. *Internet of Things*, 20:100605, 2022.

[30] Job Noorman, Pieter Agten, Wilfried Daniels, Raoul Strackx, Anthony Van Herrewege, Christophe Huygens, Bart Preneel, Ingrid Verbauwhede, and Frank Piessens. Sancus: Low-cost trustworthy extensible networked devices with a zero-software trusted computing base. In *22nd USENIX Security Symposium (USENIX Security 13)*, pages 479–498, 2013.

[31] Vincent Omollo Nyangaresi, Anthony J Rodrigues, and Nidhal Kamel Taha. Mutual authentication protocol for secure vanet data exchanges. In *International Conference on Future Access Enablers of Ubiquitous and Intelligent Infrastructures*, pages 58–76. Springer, 2021.

[32] Nachiketh R Potlapally, Srivaths Ravi, Anand Raghunathan, and Niraj K Jha. Analyzing the energy consumption of security protocols. In *Proceedings of the 2003 international symposium on Low power electronics and design*, pages 30–35, 2003.

[33] Dian Rachmawati, JT Tarigan, and ABC Ginting. A comparative study of message digest 5 (md5) and sha256 algorithm. In *Journal of Physics: Conference Series*, page 012116. IOP Publishing, 2018.

[34] Ivan Ristic. *Bulletproof SSL and TLS: Understanding and Deploying SSL/TLS and PKI to Secure Servers and Web Applications*. Feisty Duck, 2013.

[35] Crystal Andrea Roma and M Anwar Hasan. Energy consumption analysis of xrp validator. In *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–3. IEEE, 2020.

[36] Swapnoneel Roy, Atri Rudra, and Akshat Verma. An energy complexity model for algorithms. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 283–304, 2013.

[37] Swapnoneel Roy, Atri Rudra, and Akshat Verma. Energy aware algorithmic engineering. In *2014 IEEE 22nd International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems*, pages 321–330. IEEE, 2014.

[38] Tamim Samman, James Spearman, Ayan Dutta, O. Patrick Kreidl, Swapnoneel Roy, and Ladislau Bölöni. Secure multi-robot adaptive information sampling. In *2021 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 125–131, 2021.

[39] Guohuai Sang, Jingwei Chen, Yiliang Liu, Haiqing Wu, Yong Zhou, and Shunrong Jiang. Pacm: Privacy-preserving authentication scheme with on-chain certificate management for vanets. *IEEE Transactions on Network and Service Management*, 2022.

[40] Mário Santos, João Saraiva, Zoltán Porkoláb, and Dániel Krupp. Energy consumption measurement of c/c++ programs using clang tooling. In *SQAMIA*, 2017.

[41] Ashutosh Satapathy and Jenila Livingston. A comprehensive survey on ssl/ tls and their vulnerabilities. *International Journal of Computer Applications*, 153:31–38, 11 2016.

[42] A.S. Tanenbaum and N. Feamster. *Computer Networks*. Pearson Education, 2019.

[43] Leki Chom Thungon, Md Hussain, et al. Performance evaluation of zero knowledge and hmac-based authentication in fog-based internet of things. In *Proceedings of the International Conference on Computing and Communication Systems*, pages 633–641. Springer, 2021.

[44] Heping Wang, Xiaobo Zhang, and Ashfaq Khokhar. Wsn05-6: An energy-efficient low-latency mac protocol for wireless sensor networks. In *IEEE Globecom 2006*, pages 1–5. IEEE, 2006.

[45] Jonathan Westin. Evaluation of energy consumption in virtualization environments: proof of concept using containers, 2017.

[46] Luca Wilke, Jan Wichelmann, Florian Sieck, and Thomas Eisenbarth. undeserved trust: Exploiting permutation-agnostic remote attestation. In *2021 IEEE Security and Privacy Workshops (SPW)*, pages 456–466. IEEE, 2021.

[47] Tatu Ylonen. Ssh–secure login connections over the internet. In *Proceedings of the 6th USENIX Security Symposium*, volume 37, pages 40–52, 1996.

[48] SU Yu, QU Yugui, and LIN Zhiting. Hmac: An energy efficient mac protocol for wireless sensor networks. *Journal of University of Science and Technology of China*, 40(10):1054, 2010.

VITA

Cesar Castellon earned a Bachelor of Science in Electronics from the Military School of Engineering (EMI) – Bolivia in 1997. After graduating, he earned an internship at Philips CFT - The Netherlands. Upon returning to Bolivia, between 2000 and 2019, Cesar worked a Senior Engineer for ENTEL. During that time, he also earned a Master of Science in Telecommunications at the Universidad Mayor de San Andres – 2004. Later on, he worked as a Value Added Services Product Manager for VIVA, and then as the Technical Operations Manager for CyT Bolivia. Lastly, he worked as a Technical Consultant in Health Informatics for the Interamerican Development Bank (IDB).

Early in 2020, Cesar was accepted as a graduate student at the University of North Florida and began working as a graduate research assistant in the Electrical Engineering department on the NSF-funded project ”Towards Efficient and Secure Agricultural Information Collection Using a Multi-Robot System” under the supervision of Drs. Patrick Kreidl and Swapnoneel Roy. Satisfying the course prerequisites in Fall 2021 he received a Master’s degree in Electrical Engineering.

Following term Cesar was accepted into the Cybersecurity graduate program at the University of North Florida in 2022, where he was appointed as an ITS Graduate Assistant and became involved in HPC computing. In addition to coursework, he continued to volunteer with the aforementioned Research group, extending his previous work. In Spring 2023, he completed all of the course requirements for a Master’s degree in Computer Science.