2021

Energy Considerations in Blockchain-Enabled Applications

Cesar Enrique Castellon Escobar
n01453427@unf.edu

Energy Considerations in Blockchain-Enabled Applications

by

Cesar Enrique Castellon Escobar

A thesis submitted to the School of Engineering in partial fulfillment of the requirements for the degree of
Master of Science in Electrical Engineering

University of North Florida
College of Computing, Engineering and Construction

November 2021

This thesis titled ***Energy Considerations in Blockchain-Enabled Applications***, submitted by ***Cesar Enrique Castellon Escobar*** in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering, has been:

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

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

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

Touria El Mezyani, Ph.D.
Assistant Professor of Electrical Engineering
Committee Member

Brian T. Kopp, Ph.D.
Associate Professor of Engineering - Jacksonville University
External Committee Member

ACKNOWLEDGEMENTS

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

The author wishes to thank Dr. Touria El Mezyani and Dr. Brian Kopp, members of the thesis committee, for their valuable comments and suggestions to strengthen this work.

The author wishes to thank the research group's faculty members: Dr. Ayan Dutta for his professional advice, Dr. Swapnoneel Roy for sharing his knowledge and support throughout the research journey, and Dr. O. Patrick Kreidl for his continuous and thoughtful guidance.

To the memory of Prof. Jose Núñez de Arco,dear friend and colleague, who encouraged me to pursue this degree.

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

Isaiah 55:8-9

Contents

List of Figures

ABSTRACT

Blockchain-powered smart systems deployed in different industrial applications promise operational efficiencies and improved yields, while mitigating significant cybersecurity risks pertaining to the main application.

Associated tradeoffs between availability and security arise at implementation, however, triggered by the additional resources (e.g., memory, computation) required by each blockchain-enabled host. This thesis applies an energy-reducing algorithmic engineering technique for Merkle Tree root and Proof of Work calculations, two principal elements of blockchain computations, as a means to preserve the promised security benefits but with less compromise to system availability.

Using pyRAPL, a python library to measure computational energy, we experiment with both the standard and energy-reduced implementations of the Merkle Tree for different input sizes (in bytes) and of the Proof of Work for different difficulty levels. Our results show up to 98% reduction in energy consumption is possible within the blockchain's Merkle Tree construction module, such reductions typically increasing with larger input sizes. For Proof-of-Work calculations, our results show an average energy reduction of 20% across typical difficulty levels.

The proposed energy-reducing technique is potentially applicable to other key elements of blockchain computations, potentially affording even ”greener” blockchain-powered systems than implied by only the Merkle Tree and Proof of Work results obtained thus far.

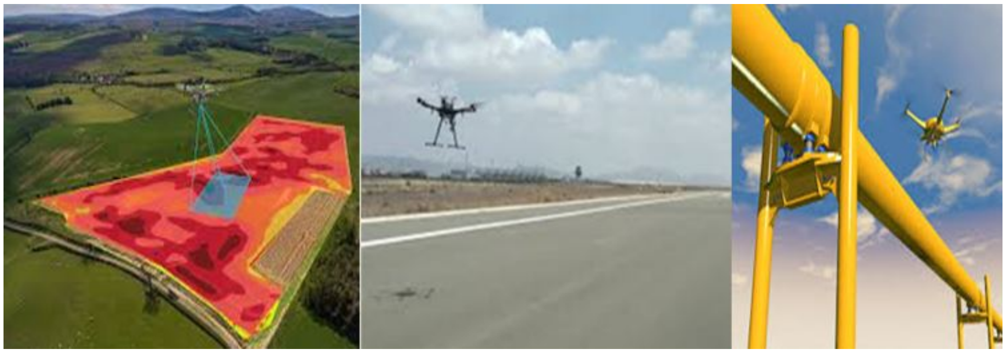
Chapter 1
Introduction

1.1 Engineering Motivation: Smart Inspection

Quality control inspections are common to many different industrial sectors. In factories, for example, inspection systems prevent defective products from reaching a final packaging process and, ultimately, wasting expenditures to market. Most modern-day factories require inspection to be automated (or at least semi-automated) by virtue of small part dimensions, the need for high production rates or possibly hazardous materials or by-products; even in factories that still implement manual inspection, human factors such as boredom and tiredness may render mistakes too frequently. Analogous quality control inspections arise within other societal sectors such as transportation (e.g., highway management, rail safety), power and infrastructure (e.g., delivery lines, energy storage) and agriculture (e.g., crop yields, sustainable practices), all seeking to similarly increase resource efficiency or operational value by leveraging new automation technologies.

Adoption of new automation technology has become an exciting challenge in all the above-mentioned possible inspection scenarios, driven by the need for better and improved sensing (i.e., obtaining measurements with finer resolution and/or broader scope), improved actuation (i.e., invoking responses with greater effectiveness and/or reduced delay), and real-time coordination. However, the subsequent integration of new automation frequently also entails refining objectives or incurring new costs, including time-consuming redesigns of the data analytics in support of the operational decision processes. Such considerations in the context of modern-day quality control applications have been coined ”smart inspection” [28, 1, 25, 20].

As Figure 1.1 illustrates, most contemporary smart inspection solutions are envisioned around so-called “Internet-of-Things (IoT)” technologies (e.g., advanced sensors, cameras, mobile drones, wireless communications) [30, 12, 42, 8]. Indeed, the many technical challenges identified by the IoT community are also challenges for smart inspection. In IoT, any intelligent device may create value from information exchange between connected devices, data repositories and other networks [12]. This feature also makes IoT prone to failure from numerous kinds of faults (e.g., natural disasters, equipment breakdowns, energy exhaustion) as well as attack from numerous kinds of adversaries (e.g., malicious insider, data exfiltration, data tampering). Such threats are often organized by the degree to which they pose risk to the system’s overall Confidentiality, Integrity and Availability (CIA) [36]. Similar threat vectors arise within the connected ecosystem envisioned for smart inspection solutions, implying their potential uses and benefits must also address the associated fault-tolerance and cybersecurity challenges [31, 44].



(a) (b) (c)

Figure 1.1: Outdoor Inspection Solutions: (a) Field Inspection, (b) Road Inspection and (c) Pipeline Inspection. (Sources: *Geoeye Egypt Remote Sensing* [http://geoeyeegypt.com], *Airport Technology* [https://www.airport-technology.com/contractors/airfield-safety/canard-drones/] and *Aerial Inspecting Services* [https://www.indiamart.com/aerialair-goa/])

1.2 Technical Motivation: Blockchain Security

IoT solutions integrate numerous disruptive technologies e.g., big data, cloud computing. The so-called blockchain is another such technology that gets identified for select IoT security issues, the data integrity problem in particular—a blockchain essentially constrains how information may be shared among component devices, establishing trust in a distributed network without the need for oversight by centralized authorities [30]. Blockchain technology originated for crypto-currency systems, but its main properties (i.e., privacy and non-repudiation mechanisms within a decentralized setting [47, 40, 4]) have since been promoted for various other applications.

At its technical core, a blockchain is merely a data structure within which certain cryptographic tools make it possible to evolve a digital ledger of transactions among a network of independent participants. These participants manage the ledger in a peer-to-peer fashion, without a ruling master participant. Change or removal of stored data is rendered extremely difficult by the blockchain's cryptographic tools [27]; in essence, any one participant cannot add or modify a transaction record without other participants collectively validating the proposed transaction, accomplished through some type of ”consensus” protocol.

The implementation details by which participants identify themselves and register to the ledger, as well as the rules that govern information sharing and consensus, are varied. The most prominent implementation is that within the original Bitcoin proposal, the so-called ”Proof-of-Work” (PoW) scheme [41] . It involves number scanning for such a value that, when hashed, produces a result that starts with a determined number of zero bits. The average work required is known to be exponential in the number of zero bits required; on the other hand, any given number is verified by running only a single hash. Furthermore, PoW helps determine representation in majority decision-making because it is essentially ”one-CPU-one-vote;” the majority decision, or the mechanism to force consensus, is related to the longest chain with the greatest PoW effort invested.

To better appreciate both the benefits and pitfalls of blockchain-enabled smart systems, consider the illustration of Figure 1.2, showing a Blockchain ledger and its participants, on both demand-side and supply-side, connected in a logical star topology; conventional source devices (e.g., analog temperature sensors) can connect through the IoT Gateway, while "smart" source devices (e.g., digital valve, camera drone) may connect to the ledger using a more direct physical or logical connection. In addition, the ledger may permit exclusively supply-side actors to connect for special-purpose information retrieval or data monitoring (e.g., by regulatory agencies).

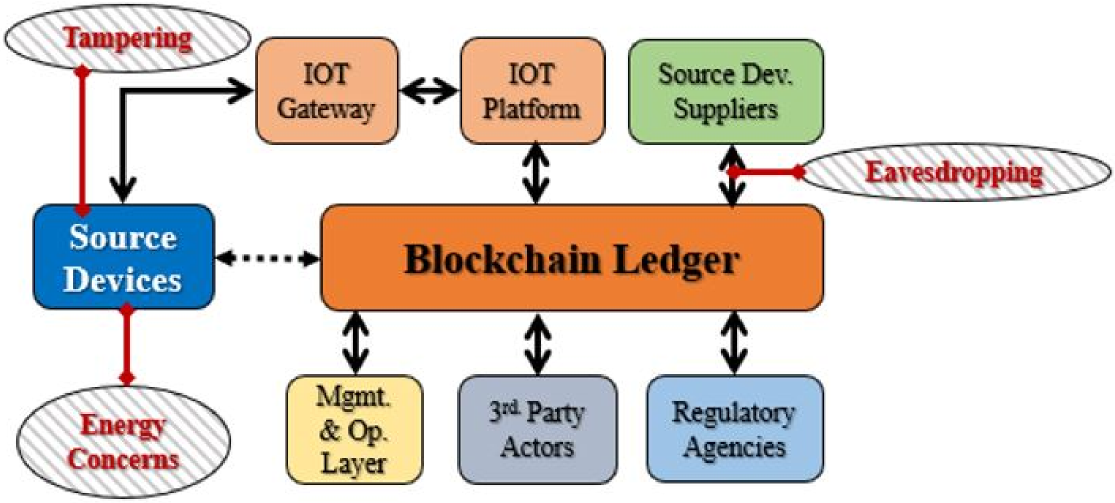


Figure 1.2: A general Blockchain ledger, participants, and threats.

Critical security issues arise from the necessary direct connections between the ledger and the ”smart” source devices or supply-side actors. For example, an attacker interested in getting information about production quantities may try eavesdropping a supply side actor. As another example, an attacker interested in causing harm to a plantation may tamper with humidity measurements by a smart sensor, perhaps to cause irrigation to exceed desired or permitted levels. The risks of such security vulnerabilities have well-established mitigations, such as encryption and digital signatures, but the associated extra communications and computations also cost additional resources (e.g., memory, processor, power) [3]. The quest for efficient and secure blockchain-powered smart systems becomes the core challenge, where energy is emerging as the fundamental measure of resource consumption associated with any given implementation.

1.3 Thesis objectives

This thesis explores possibilities for reducing the energy consumption of blockchain functionalities, preserving the motivating security features even within applications having limited or fixed power budgets (e.g., smart inspection in IoT systems). Our approach first employs an established Energy Complexity Model [26] upon key algorithmic components of any Blockchain Layered Integrated Architecture, specifically a core cryptographic hashing algorithm implemented in the Merkle Tree Root calculation as well as the original Proof-of-Work mining protocol. Then, using a recently-developed energy measurement tool interface called RAPL [19] on both the baseline and optimized algorithms, the achieved energy reduction is quantified. The results provide a proof-of-concept for our energy-reducing algorithmic engineering approach, inviting future work into whether other key components of blockchain computations can similarly be more efficient without sacrifice to functionality.

1.4 Thesis organization

The remainder of this thesis is organized as follows. Chapter 2, which is a reproduction of a paper accepted and presented in the *2021 IEEE Trustcom Conference* [7], examines the effect on energy savings for an ECM-optimized Merkle Tree calculation. Chapter 3 presents the analogous approach and results on an ECM-optimized Proof-of-Work calculation. Finally, Chapter 4 presents the document's conclusion and recommendations for future research.

Chapter 2
Energy Considerations in Merkle Tree Root Operations

This chapter is a reproduction of Castellon et al. [7] accepted and presented in the 2021 IEEE Trustcom Conference. The article's authors are Cesar Castellon, Swapnoneel Roy, O. Patrick Kreidl, Ayan Dutta and Ladislau Boloni.

2.1 Introduction

Blockchain technology, popularized by different crypto-currency systems, is seeing extensive use in different fields. Advocates for such uses cite the blockchain's inherent properties of a decentralized structure alongside enhanced security with mechanisms for privacy and non-repudiation [13, 10, 16, 2]. One particularly promoted use case is the *Internet of Things (IoT)* [48, 41, 5], which embodies the vision by which different computing devices may communicate with each other to map a physically connected world onto its digital mirror. The IoT vision also motivates prospects of so-called *smart* systems [32] e.g., smart cities, smart homes, smart grid, smart health, smart agriculture. The potential uses and benefits of smart systems recognizably also raises critical security and privacy challenges to be addressed, which motivates the vision of blockchain-powered smart systems.

Smart and secure systems implemented upon IoT technology require device inter-connectivity for extended time frames, delivering continuous data. Such operations demand constant power supply [32]—within a world that demands more environmentally-friendly (or "green") solutions, in general, IoT realizations also face the challenge of energy efficiency i.e., minimizing their energy footprint. Thakore et al. [43] acknowledge the additional energy optimization requirements that blockchains require when implemented together with IoT. Depending on the specific type of blockchain-IoT combination, precise analysis of performance and energy requirements becomes critical [37]. As an example of these challenging tradeoffs, consider a particular blockchain-IoT implementation with a fixed power budget. To be viable for an application that values autonomy for greater lengths of time, the system must be configured to make more efficient use of energy. Disabling the blockchain will certainly save energy, but also weaken security: it is in such contexts that the exploration of ways to reduce energy consumption of blockchain functionality 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 [17]; and algorithms [34, 35]. Energy measurements are sometimes obtained by uniquely instrumented equipment [33], while other times can leverage hardware providers' Application Programmer Interfaces (APIs) in which firmware counters are recalled to provide near real-time information e.g., Running Average Power Limit (RAPL) technology [38]. Blockchain implementations are actively under study as providing a decentralized ledger (i.e. record of transactions) by which to optimize energy management in a variety of scenarios (e.g., generation & distribution [18, 46], micro-grid networks [26, 23, 30] and smart contracts [27]). In contrast to our motivation, however, these studies define the optimized management objectives such that the energy footprint of the blockchain itself is out of scope.

There are past studies who also recognize that the blockchain itself will draw energy away from any symbiotic system it is integrated with. Examples include Sankaran et al. [37] and Sanju et al. [36], who perform power measurements and evaluate real experiments on the energy consumption of two different blockchain implementations, namely Ethereum and Hyperledger. A similar analysis of energy consumption is presented in [33] for XRP validation, which is a key element of decentralized consensus processes within many Internet services. A particularly novel theoretical approach is reported by Fu et al. [15], first modeling a blockchain-IoT caching infrastructure and posing its energy optimization within a geometric programming formulation whose solutions allocate resources accordingly. A recent performance evaluation survey, also conducted by Fu et al. [13], illustrates how diverse and sophisticated current implementations of blockchain ledgers are. Despite this diversity, however, all existing implementations at their core remain faithful to Nakamoto's original blockchain concept [29], within which the Merkle Tree construction module is essential.

2.1.2 Our Scope and Contributions

We study the extent to which Merkle Tree construction, a principal element of blockchain computations, 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. [34, 35], on the SHA256 encryption algorithm, which is central to the Merkle Tree. 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 the Merkle Tree for input sizes (in bytes) that are commonly seen within blockchain implementations. Our results show significant reductions in energy consumption, up to 98% but on average 50% across the tested input sizes. At present, it is only a conjecture that reduced energy consumption in the Merkle Tree construction module itself extrapolates to comparable reduction of a blockchain on the whole. In any case, to the best of our knowledge our work is the first to address energy optimization of blockchains by re-engineering the implementation of one of its component algorithms. Moreover, the proposed energy-reducing technique is similarly applicable to other key elements of blockchain computations, potentially affording even ”greener” blockchain-IoT systems than implied by only the Merkle Tree results obtained thus far.

2.2 Methodology

This section describes our application of the Energy Complexity Model (ECM) [34, 35] to the Merle Tree (MT) root construction module of the blockchain. Described first is the process by which a block of the blockchain is computed based on the MT root, in which so-called hash calculations play a central role, followed by a summary of how the ECM works, in general. This section ends with a detailed description of how the central hash calculations of the MT are re-engineered based on the ECM.

2.2.1 Merkle Tree Based Block Generation

A graphic representation of a simple block generation in a blockchain is shown in Fig. 2.1. The bottom layer shows the stored transactions (e.g., $T001$) for the block, which later are converted to their SHA256 Hash signatures (e.g., $H001$) and represent the Merkle Tree leaves. Merkle Tree root calculations involves the recursive hash computation starting from these leaves until a final hash determines the Merkle Tree root (labeled TX_ROOT in Fig. 2.1).

Conceptually, the process of Merkle Tree calculation through hashing can be viewed as a state transition in which an investment of computational resources is required e.g.,

$$\delta_t \xrightarrow{f(T)} \delta_{t+1} \tag{2.1}$$

$$T = cost[energy, time] \tag{2.2}$$

That is, the block generation is represented by the state transition in (2.1), which depends upon a function $f(T)$ with parameter T denoting the *cost* as represented by (2.2). This cost has two main components: one is the energy consumed by the hardware devices to compute the hash of

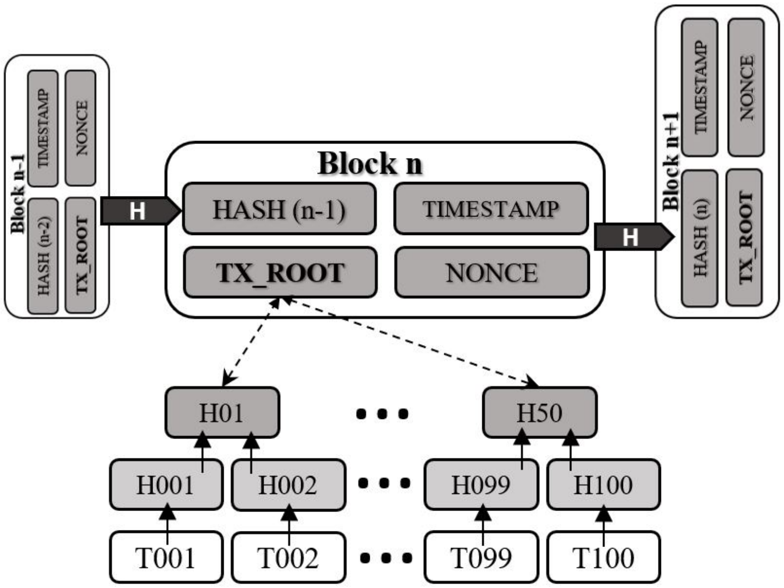


Figure 2.1: Basic Block Generation in Blockchain.

the input vectors, while the other is the execution time of those computations. This paper strives to reduce the overall transition cost T by reducing the energy consumption of the hardware devices, employing a technique based upon the ECM next described.

2.2.2 The Energy Complexity Model (ECM)

The ECM developed in [34] is built upon an abstraction of the Double Data Rate Synchronous Dynamic Random Access Memory (DDR SDRAM) architecture [14], which is illustrated in Fig. 2.2. Main memory in DDR is divided into *banks*, each of which contains a certain number of *chunks*[1]. Data is allocated over chunks in each bank, and each bank also contains a special chunk called the *sense amplifier*. When data needs to be accessed, the chunk containing the data is fetched into the sense amplifier of the corresponding bank. The sense amplifier can only hold one chunk at a time, so the current chunk has to be put back to its bank before the next one can be fetched for access. While only one chunk of a particular bank can be accessed at a time, chunks of different banks

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

(each with their own sense amplifier) can be accessed in parallel. Therefore, if the DDR memory is organized into P banks (where $P = 4$ in Fig. 2.2), then P chunks can be accessed at a given time. In the popular DDR3 architecture, the DDR1 notion of the per-bank sense amplifier is referred to as the per-bank cache, albeit still only capable of accessing one chunk at a given time.

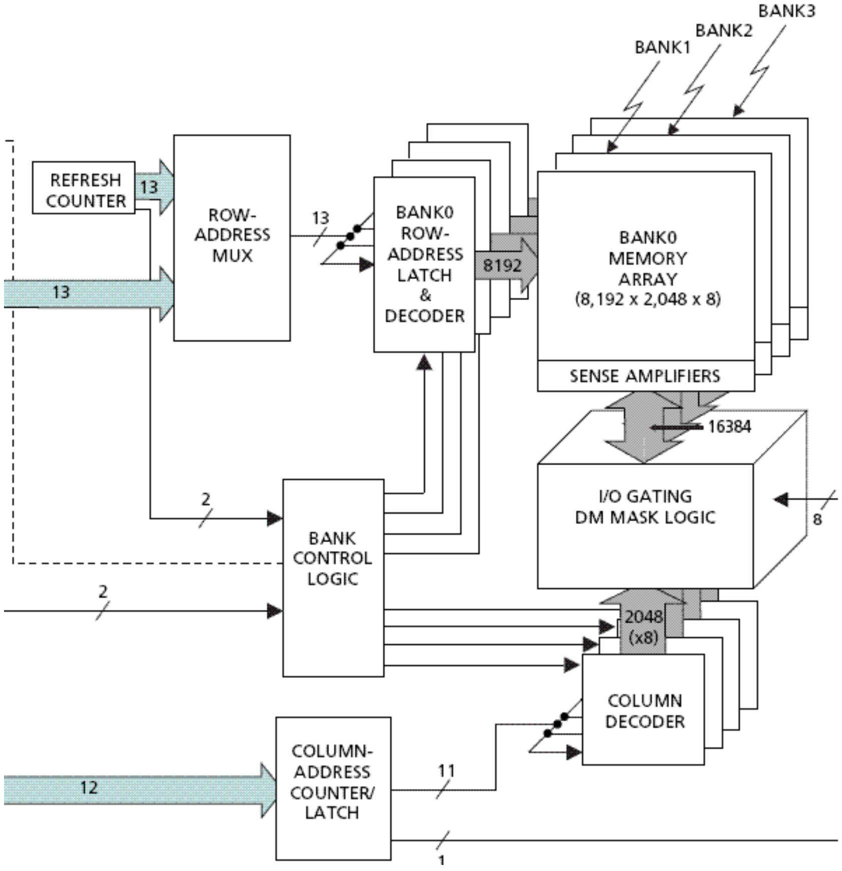


Figure 2.2: Internal DDR SDRAM memory chip block diagram.

The ECM denotes the P banks of a given DDR3 SDRAM resource by $M_1, M_2, \ldots, M_P$, each such bank M_i comprised of multiple chunks of size-B (in bytes) and its own cache C_i. The illustrative example of Fig. 2.3 assumes $P = 4$ banks, as was the case in Fig. 2.2, with just four chunks per bank, assigning numerical labels $1, 2, \ldots, 16$ to the memory's collection of data chunks. Heeding the DDR constraint that each cache C_i may access exactly one chunk at a time, the access patterns $(1,2,3,4)$ or $(5,6,7,8)$ imply a completely serial execution, while the access patterns $(1,5,9,13)$ or $(3,8,10,13)$ are each completely parallel. The authors of [34] discovered two key properties of

DDR memory: firstly, the difference in power consumption between the same number of chunks accessed sequentially or in parallel is not significant; however, the execution time of an algorithm when chunks are accessed in parallel is significantly lower than when chunks are accessed sequentially. Because the associated energy consumption depends upon both power and time, it follows that parallelizing chunk accesses offers the potential for energy reduction of any algorithm! More formally, as derived by Roy *et al.* [34], 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

$$E(\mathscr{A}) = \tau + (P \times B)/I \tag{2.3}$$

where I denotes the so-called *parallelization index*, essentially the number of parallel block accesses across memory banks per P block accesses made by $\mathscr{A}$ on the whole. That is, under the ECM, an algorithm's potential for energy reduction is inversely proportional to the degree to which it can be re-engineered for parallelization of its memory accesses.

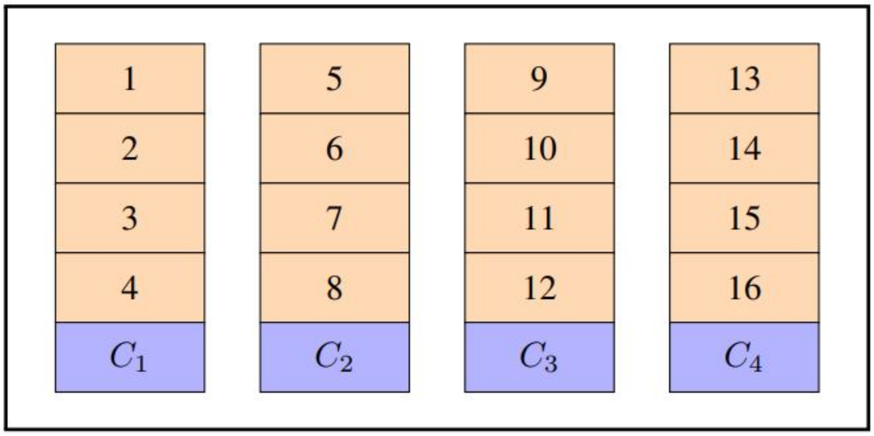


Figure 2.3: ECM for DDR3 Resource with $P = 4$ Banks

2.2.3 Re-engineering Hash Calculations Using ECM

In this work, we engineer the hash algorithm of Merkle Tree (MT) construction based on ECM to reduce energy consumption. First, we briefly describe how any algorithm $\mathscr{A}$ can be parallelized based on ECM. We then illustrate how MT's hash calculation, specifically the SHA encryption algorithm, is re-engineered for parallelization.

Parallelizing any algorithm

Given an algorithm $\mathscr{A}$, the input to $\mathscr{A}$ is considered to identify the most common access sequence in $\mathscr{A}$. The required level of parallelism for the vector formed by the desired access sequence is then engineered using a logical mapping over chunks of memory that store data accessed by $\mathscr{A}$.

As mentioned above, the order of chunk accesses is different for different levels of parallelization. But the physical location (chunks) of the input in the memory is static, and is handled by the memory controller of DDR. Therefore, to implement parallelization of access over physical chunks, a different page table vector **V** is generated for each level of parallelization, which defines the ordering among the chunks to be accessed (see Fig. 2.4).

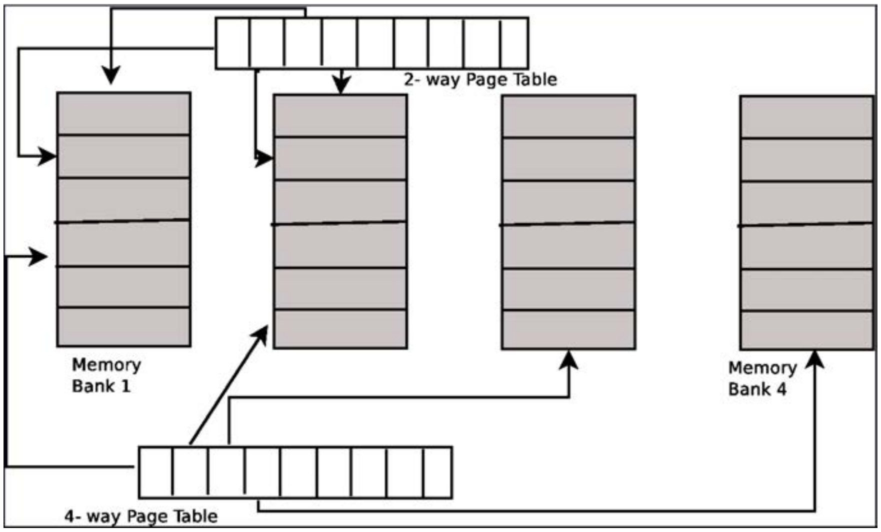


Figure 2.4: Memory Layout ($P = 4$) and Role of Page Tables

For 1-way access, the page table vector **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 **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 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 SHA Encryption

As described earlier, Merkle Tree construction performs its hash calculations via repeated use of the SHA256 encryption algorithm. Specifically, as shown in Fig. 2.5, the input is partitioned into fixed size message blocks, presented in sequence to separate compression functions.

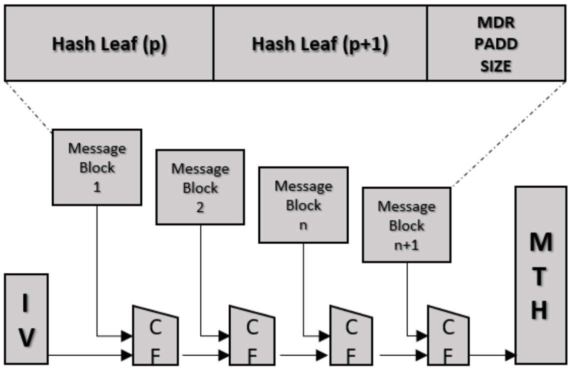


Figure 2.5: SHA256 for Merkle Tree Calculation

This block sequence is identified in correspondence with the access pattern of the SHA256 algorithm, which we subject to re-engineering based on the ECM. The input vector, in a Merkle

Tree being the concatenation of three strings (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. 2.6.

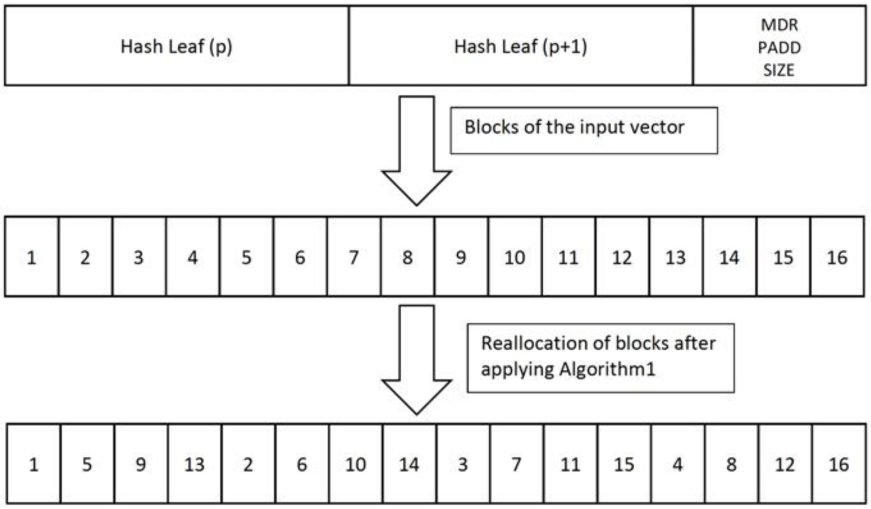


Figure 2.6: Mapping of SHA Input Blocks based on ECM.

Fig. 2.7 shows the outcome of re-engineering the SHA256 algorithm based on ECM.

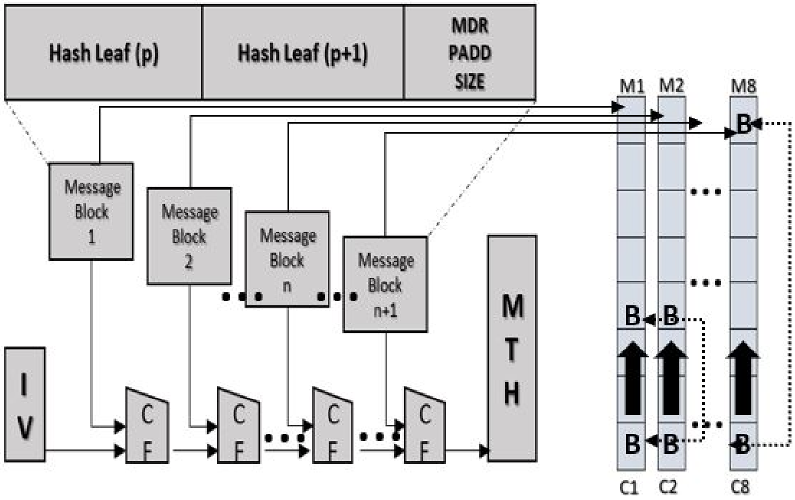


Figure 2.7: ECM-Enhanced Merkle Tree Calculation

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 [35] to ensure that each size-8 access occurs across all eight banks.

2.3 Experiments

We now proceed to describe computer experiments designed to quantify the energy savings of the methodology detailed in the previous section. By virtue of the ECM's formulation, the enhanced implementation requires computer hardware using a DDR RAM architecture. Maximum energy reduction is promised by a parallelization index taken to equal the number of memory banks, which depends upon the DDR version: 4 for DDR2, 8 for DDR3 and 16 for DDR4 and higher. The machine used for our experiments features 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. We use pyRAPL, a software toolkit to measure a host machine's energy footprint along the execution of a piece of Python code, to compare energy consumptions between the standard and ECM-enhanced implementations. 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 [38].

2.3.1 Implementation Details and Setup

Our experimental objectives could not be met by using the SHA256 function in the Hash Python library. This is because memory management in Python involves a private heap, containing all objects and data structures. The control of this private heap is ensured internally by the Python memory manager, with different components dealing with sharing, segmentation, pre-allocation or caching. Our ECM-enhanced implementation of SHA256 requires greater control over memory allocation than Python's memory manager permits. Such low-level control on memory management is possible in the standard C programming language. We thus implement the standard and ECM-enhanced versions of the SHA256 algorithm within separate C programs, which are called from a Python script (upon importing the `ctypes` module) as an external routine. This permits the use of pyRAPL for the needed energy measurements without denying low-level memory control to implement the ECM-enhanced SHA256 functionality during Merkle Tree calculations.

Our experiments simulated the Merkle Tree calculation with Python code that runs 103 consecutive two-leaves-input hashes with pyRAPL invoked. Each execution of the code yields an energy measurement, but because the instrumentation is subject to noise we invoke 5000 repetitions and report the average energy (mean and deviation). Our experiments also vary the input size (i.e., the compounded-leaf size) to the Merkle Tree calculations, choosing 1, 64, 96, 128, 512, 1024, 16384 and 262144 bytes motivated as follows:

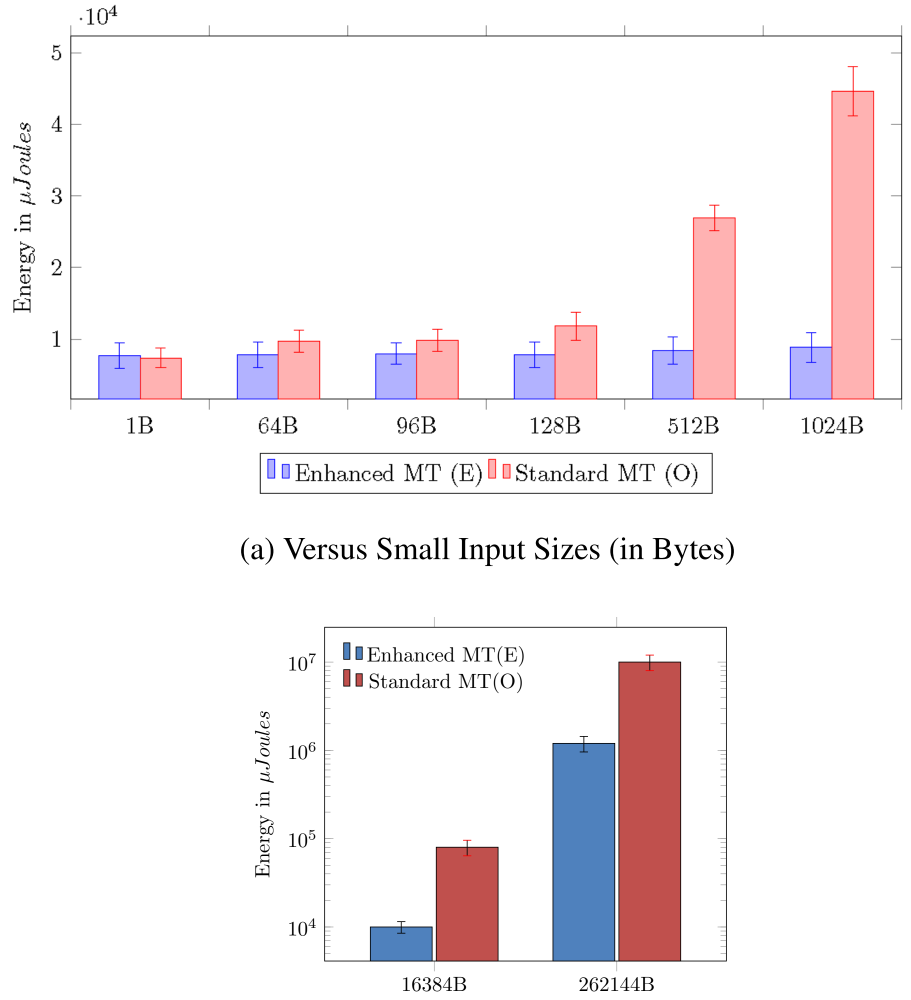
1. the 1B input is the bare minimum that the ECM permits for any algorithm [34];
2. the 64B, 96B and 128 inputs are common in blockchain applications [41];
3. the 512B and 1024B inputs are common in file hashing applications [11]; while
4. the 16384B and 262144B inputs for the Interplanetary File System (IPFS) [21, 9].

2.3.2 Results and Discussion

Recall that our experimental setup features two implementations of Merkle Tree (MT) calculations, the standard one (which we label by “O” as it uses the original SHA256) and the re-engineered one using ECM (which we label by “E” as it uses the enhanced SHA256), as well as eight 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 5000 trials) of simulated Merkle Tree calculations.

Fig. 2.8 summarizes the sixteen average energy measurements in two bar charts, per input size comparing the Standard MT (O) and the Enhanced MT (E) average energy (in μJoules).

Fig. 2.8 (a) renders the comparison over the six smallest input sizes (using a linearly-scaled vertical axis), while Fig. 2.8 (b) is over the two largest input sizes (using a log-scaled vertical axis).



(a) Versus Small Input Sizes (in Bytes)

(b) Versus Large Input Sizes (in Bytes)

Figure 2.8: Comparison of Average Energy Consumption

It is seen that the ECM-enhanced implementation consistently requires less energy that the standard implementation, the difference being increasingly significant with the larger input sizes that befit file hashing applications (i.e., 512B and above) (but still meaningful for input sizes 64B, 96B and 128B that befit blockchain applications).

This observed dependence on input size may be a consequence of CPU memory caching. DRAM memory often allows the memory controller to optimise accesses by L1/L2/L3 caching of data. With smaller inputs, such caching enables parallelization of bank accesses even in the standard implementation.

The comparison for the 1B input size corroborates this point, where we observe the enhanced implementation consume more energy than the standard implementation.

Fig. 2.9 presents the average energy comparison on more relative terms, namely as a percent reduction achieved by the enhanced implementation over the standard implementation versus all eight input sizes. The energy savings for the blockchain-motivated input sizes range between 19% and 34%, while the energy savings for the file-system-motivated input sizes range between 69% and 98%, the case of 16384B exhibiting that maximum 98% savings. As noted in Fig. 2.8, the 1B input renders a savings of -4%, meaning the standard implementation is more energy-efficient by virtue of the parallelism invoked within the CPU's L1/L2/L3 cache in this case.

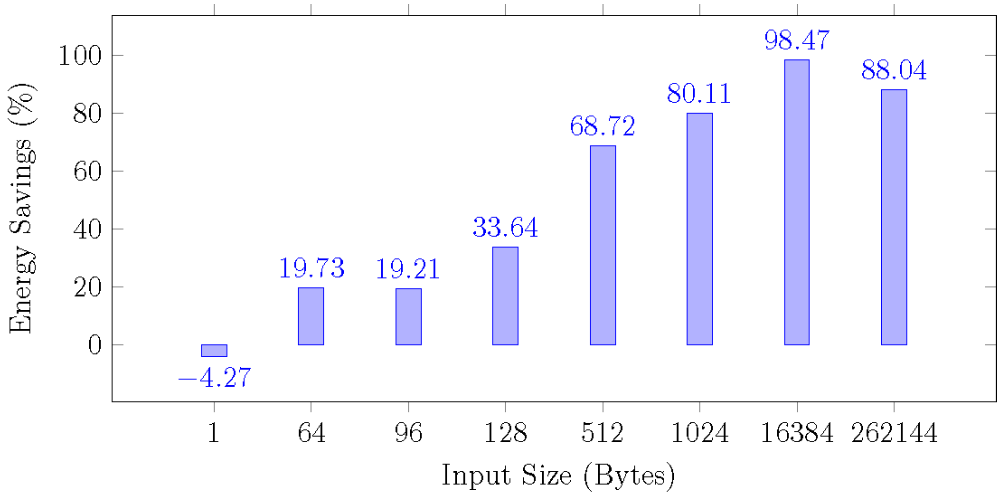


Figure 2.9: Comparison of Energy Savings

Chapter 3

Energy Considerations in Proof-of-Work Operations

The preceding chapter demonstrated significant energy savings are possible within Merkle Tree calculations by simply relying on a re-engineered version of the Sha256 hashing algorithm. This chapter pursues the natural next question, namely whether energy savings remain significant for other blockchain operations. Arguably the most popular blockchain operation is so-called Proof-of-Work (PoW), which essentially achieves distributed consensus by requiring members of the network to expend effort and solve an arbitrary mathematical puzzle in order to be trusted. PoW is used widely in Bitcoin mining to validate transactions and mine new tokens, for example—it is through PoW that Bitcoin transactions can be processed peer-to-peer in a secure manner without the need for a trusted third party.

This chapter begins by framing PoW within the well-studied Byzantine Generals problem, which in the theoretical computer science literature embodies numerous impossibility results and fault-tolerant bounds known for distributed consensus. The chapter continues by describing details of PoW, illustrating how the required work increases as more members join the network and, at scale, requires huge amounts of energy. Preserving the roles of PoW but with smaller energy footprint is, in turn, of great interest. We employ the experimental methodology of Chapter 2 within a blockchain emulation with results pointing to at least modest energy savings being possible.

3.1 Proof-of-Work and The Byzantine Generals Problem

The Byzantine Generals Problem describes the difficulty that decentralized systems have in agreeing on a single truth. The problem is setup as a set of generals having to coordinate an attack on a given city—only if a sufficient number of these generals agree will the city be taken. Bitcoin uses a Proof-of-Work (PoW) mechanism within a blockchain to solve the Byzantine Generals Problem, effectively ensuring that stored information has not been changed or replaced either internally or externally. The allegorical analogy is that any general (i.e., member of the Bitcoin network) can announce an attack (i.e., a next transaction) at any given time with two considerations:

1. The first attack time heard is considered the official one.
2. Different generals may receive other plans at this first time because message transmission times among generals may differ.

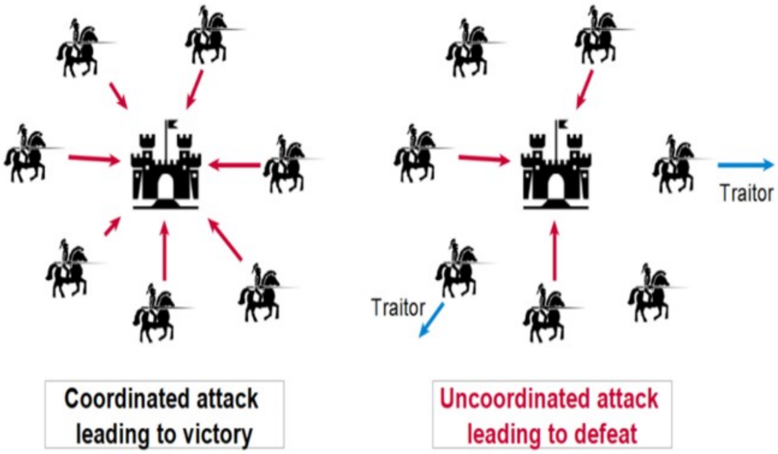

Traitor
Traitor
Coordinated attack leading to victory
Uncoordinated attack leading to defeat

Figure 3.1: Byzantine Generals Problem

All the generals must solve a time-consuming puzzle based on the plan decided. As soon as one general calculates a solution, the solution is added and the resulting plan is sent to the other generals. Every general who receives the new plan starts over again, continually working on the latest updated possible plan.

After numerous rounds, the official determined attack time is in the longest chain of calculations, whose creation will have expended more than half the computational power of the generals; an attacker with less than 50% of the computational power could not have timely created another chain of equal length. In crypto-economic Blockchains, such as Bitcoin or Ethereum, node miners run PoW and essentially compete to create new blocks filled with processed transactions. The node first solving the math puzzle shares the new block with the rest of the network and earns cryptocurrency units, producing a cryptographic link between the current block and the preceding block in the chain.

Proof-of-work (PoW) in blockchains is the mechanism that allows the decentralized blockchain ledger to agree on essential elements such as the order of transactions, the costs of each task, and the actual account balances. The inputs to a PoW algorithm are the operational rules and a mining difficulty, which determines the pace for adding valid blocks to the chain—the more blocks are added, the more difficult successful mining becomes and the more certain the whole network is about the current state of things. To create a block in PoW, a miner will repeatedly put an actual downloaded dataset through a mathematical (hashing) function, looking for a result below a target nonce as dictated by the so-called block difficulty, where a lower target nonce permits a smaller set of valid hashes. Once a successful result is generated, other miners and clients can verify the validity of the block with a single hash operation. If one transaction on the chain were to change, the hash would be completely different, thus signaling possible fraud.

The standard blockchains will accept that an agreement of more than 50% of the total computing capacity suffices to reach a consensus. Thus, although a possible disagreement may appear, the majority vote will solve opposite opinions. That is, another key element of a blockchain's consensus mechanism is the so-called chain selection rule by which the "correct" chain is decided in the event that multiple paths evolve in parallel (e.g., Bitcoin, uses the so-called Nakamoto rule, selecting the "longest chain" as the correct one).

Finally, PoW is not only for consensus protocols in blockchains: it also works as block author selector and a Sybil resistance mechanism. Sybil attacks occur when a group of colluding nodes pretend to be many participant nodes. Resistance to this attack is crucial for decentralized blockchain ledgers. Pow makes users expend a lot of energy or crypto value as protection, as an economic deterrents to Sybil attacks.

3.2 Methodology

To summarize the preceding subsection, Proof-of-Work (PoW) mechanisms involve the following essential steps local to each member of the network sharing a common blockchain:

1. Update the local copy of the chain to the latest version.

2. Solve a mathematical puzzle and create the ”mined” block header hash.

3. Advertise the solution as soon as the math puzzle is solved.

4. Append the block to the chain if authorized.

It is during Step 2, the Puzzle Solving process, that a hashing algorithm such as SHA256 plays its role. In turn, the energy measurement methodology of the preceding chapter should permit a comparable analysis of PoW operations within a blockchain; that is, we consider two blockchain implementations in which all things are equal except whether Step 2 uses the standard SHA256 implementation or the energy-optimized one. However, because a blockchain involves many functions other than hashing, what is unclear is whether the energy savings will be comparable to that demonstrated in Chaper 2 for Merkle Tree calculations in isolation.

One complicating factor to PoW energy analysis is that the operations involve an iterative hashing process. As Figure 3.2 illustrates, the first value of each block’s header is the previous block’s hash (labeled “Hash(n-2)”), also referred to as the `prevhash`. The second value of each block’s header is the associated transaction’s Merkle Tree Root ID (labeled “TXROOT”). The third value is the time stamp (labeled "TIMESTAMP") associated with that block's generation and, finally, the fourth is a variable value that is iteratively adjusted during the computations (labeled as "NONCE").

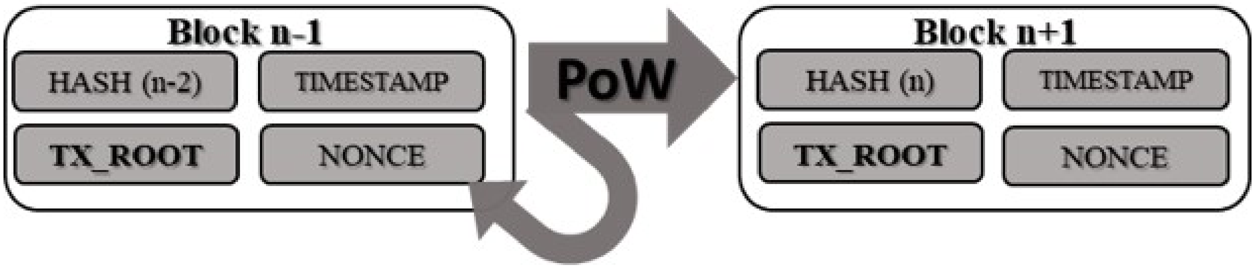


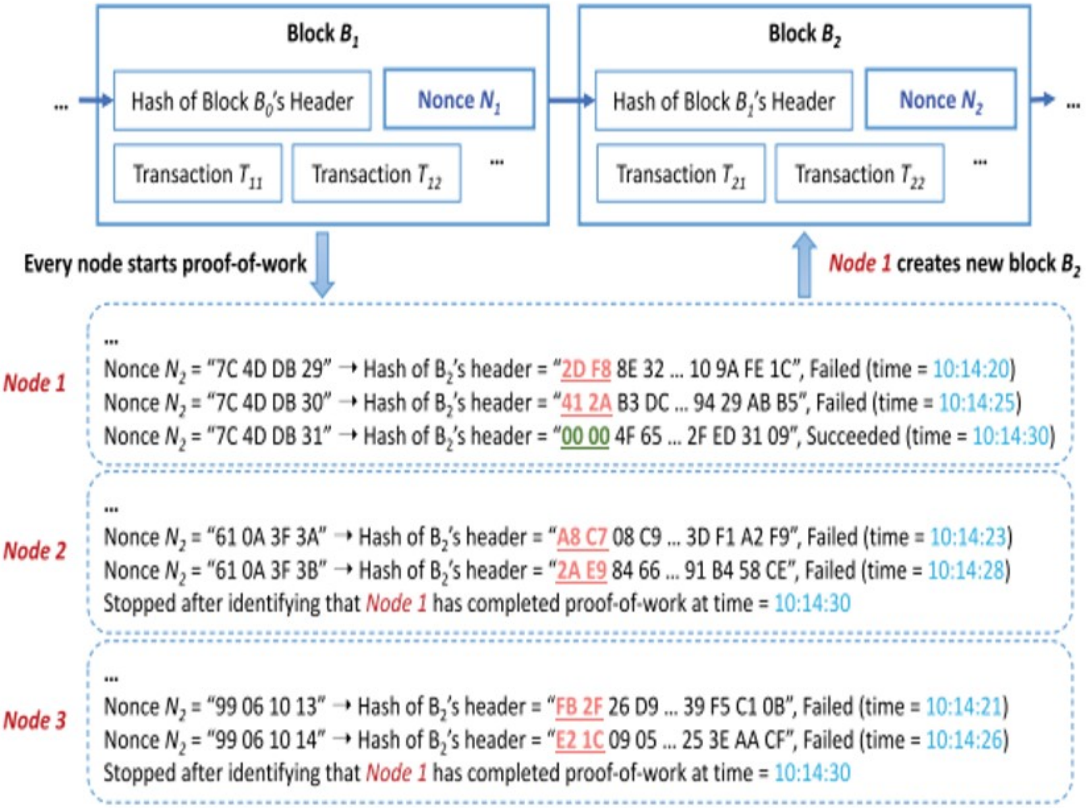
Figure 3.2: PoW Basic Principle

Recall that the nonce determines the degree to which mathematical puzzles become harder to solve as the blockchain grows. More specifically, the puzzle to be solved for block creation requires the "mining" node to find a nonce such that the Block's resulting SHA256 Hash is sufficiently small to fall into a small target space of possible solutions for that header. In other words, the purpose of the "NONCE" value is to render it resource demanding to satisfies specific conditions for adding new blocks [22, 6]. Such behavior is illustrated in Figure 3.3

Arguably the most widely adopted algorithm for Proof-of-Work is the Hashcash scheme. Despite its popularity, two issues are commonly identified in its use. Firstly, the high energy consumption of the scheme is perceived as wasteful because the solutions found provide no useful output [24, 39]. Secondly, the computational complexity class of the scheme is *at least* NP-Complete [24], which means solving PoW is intractable by the Turing Model. To expedite our experimentation, yet still reliably emulate blockchain hashing operations, we bypassed the Hashcash scheme and rather implemented a simplest version of the PoW algorithm—see Algorithm 2.

3.3 Experiments

The PoW experiments use the same hardware setup as discussed in the preceding chapter, remaining compliant with architecture requirements: an Intel i5 64-bit dual-core processor computer with Linux Mint version 19.3 with an 8GB DD3 RAM and 500GB SSD storage. The PoW emulations



```
...
Nonce N2 = "7C 4D DB 29" → Hash of B2's header = "2D F8 8E 32 ... 10 9A FE 1C", Failed (time = 10:14:20)
Nonce N2 = "7C 4D DB 30" → Hash of B2's header = "41 2A B3 DC ... 94 29 AB B5", Failed (time = 10:14:25)
Nonce N2 = "7C 4D DB 31" → Hash of B2's header = "00 00 4F 65 ... 2F ED 31 09", Succeeded (time = 10:14:30)

...
Nonce N2 = "61 0A 3F 3A" → Hash of B2's header = "A8 C7 08 C9 ... 3D F1 A2 F9", Failed (time = 10:14:23)
Nonce N2 = "61 0A 3F 3B" → Hash of B2's header = "2A E9 84 66 ... 91 B4 58 CE", Failed (time = 10:14:28)
Stopped after identifying that Node 1 has completed proof-of-work at time = 10:14:30

...
Nonce N2 = "99 06 10 13" → Hash of B2's header = "FB 2F 26 D9 ... 39 F5 C1 0B", Failed (time = 10:14:21)
Nonce N2 = "99 06 10 14" → Hash of B2's header = "E2 1C 09 05 ... 25 3E AA CF", Failed (time = 10:14:26)
Stopped after identifying that Node 1 has completed proof-of-work at time = 10:14:30
```

Figure 3.3: Example of Nonce Calculation Process

Algorithm 2: BlockHash mining algorithm

```
Data: $nonce = 0, Block(B), parameters, difficulty = 1$
Result: $B.hash, nonce$
$diff = fill("0", difficulty)$;
for $nonce = 0$ do
    $B.Hash = SHA256(B.Params + nonce)$;
    if $substr(B.Hash, difficulty) = diff$ then
        $break$;
    else
        if $i = 2**256$ then
            $nonce = 0$;
            $break$;
```

and energy measurements are also similarly accomplished, namely through a combination of a main Python wrapper program (with the measurement logging routine and API) and a C program (containing all the logic for the hashing and the Proof of Work calculation.)

We first used a single standard program for the PoW for a given input and low difficulty. Then, following Fig 3.4, we performed a baseline measurement. The measurement process was repeated after modifying the previous program so that hashing employed the re-engineered algorithm of Chapter 2, all other things equal, as illustrated in Figure 3.5. In particular, both steps ensured that the exact same input conditions were provided.

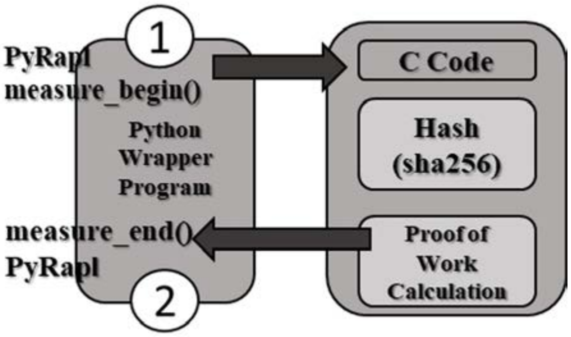


Figure 3.4: Standard Measurement Schema - PoW

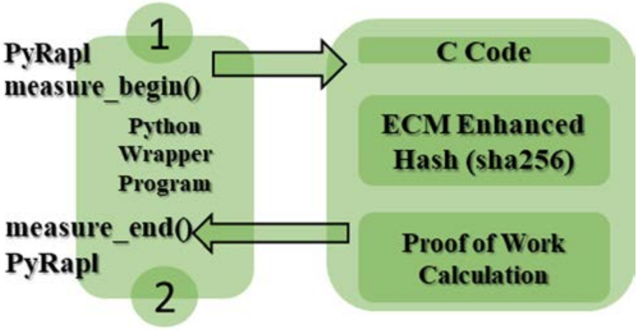


Figure 3.5: ECM Enhanced Code Measurement Schema - PoW

3.3.1 Implementation Details and Setup

The standard PoW implementation is labeled by “O,” using the original SHA256 hashing algorithm, while the implementation using the re-engineered SHA256 algorithm is labeled by “E.” The input size is fixed 256 bytes, including all the Block Header Parameters and byte padding, while the difficulty level ranged from 1 to 6 (encoded by `H0`, `H00`, `H000`, `H0000`, `H00000` and `H000000`).

Per implementation and difficulty level, our experimental Python program leverages the pyRAPL toolkit to measure the average energy (mean and deviation over repeated trials) of the emulated Proof-of-Work calculations.

3.3.2 Results and Discussion

Fig. 3.6 summarizes the average energy measurements (using a log-scaled vertical axis) per difficulty level size, at each level comparing the Standard PoW (O) and the Enhanced PoW (E) average energy (in μJoules).

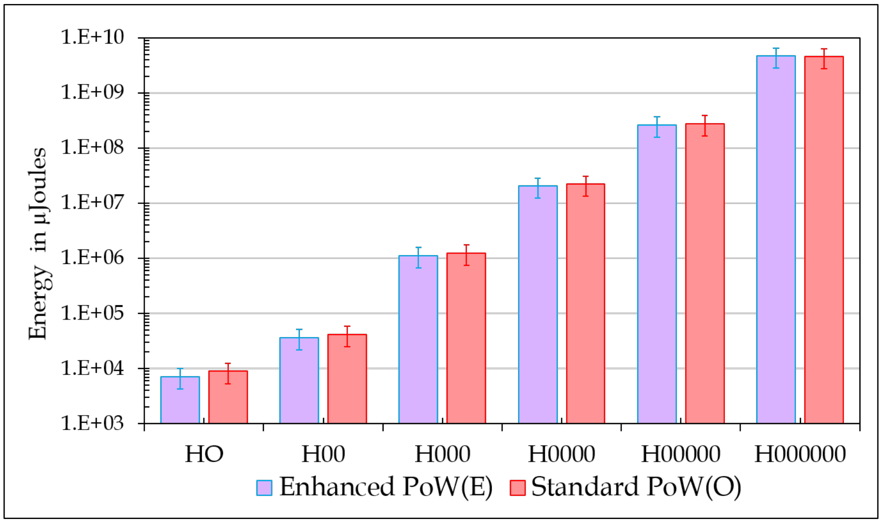


Figure 3.6: PoW Energy Measurements per Difficulty level (with 1-sigma standard deviation over 1000 trials)

It can be seen in the figure that there exists some energy savings when comparing both implementations, although the savings in percentage is inversely related to difficulty level.

Fig 3.7 the average energy comparison in relative terms versus difficulty level, namely as a percent reduction achieved by the enhanced implementation over the standard implementation. The energy savings range between 20% and 4%, the case of Difficulty 1 exhibiting that maximum 20% savings. Observe the steady energy reduction as the difficulty level increases. We conjecture this to be an artifact of the exponential increase in computational resources with increasing difficulty,

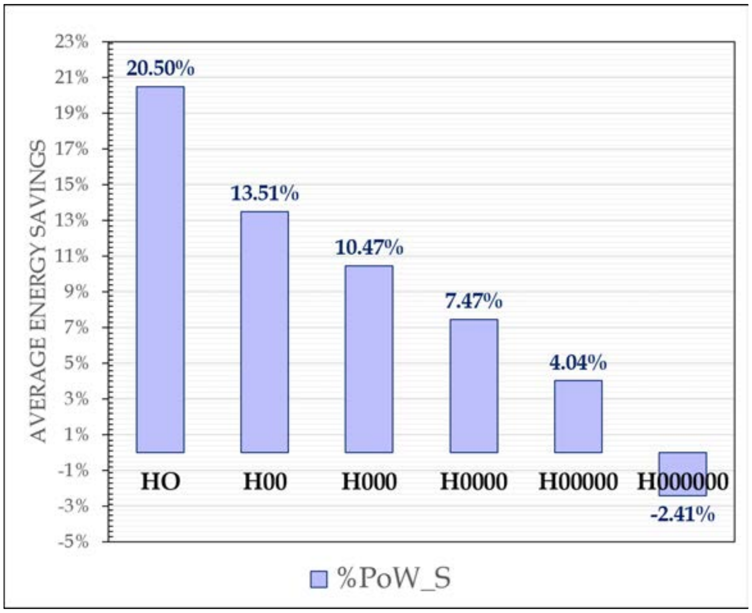


Figure 3.7: Average PoW Savings per Difficulty

which hinders the parallelism upon which energy optimization rests. With the Merkle Tree experiments of Chapter 2, the number of iterations did not exceed hundred per data point, while in PoW measurements these range from 9 in Difficulty 1, 150 thousand in Difficulty 4 and up to more than 37 million in Difficulty 6. Figure 3.8 displays the exponential rise in the number of iterations within the PoW operations.

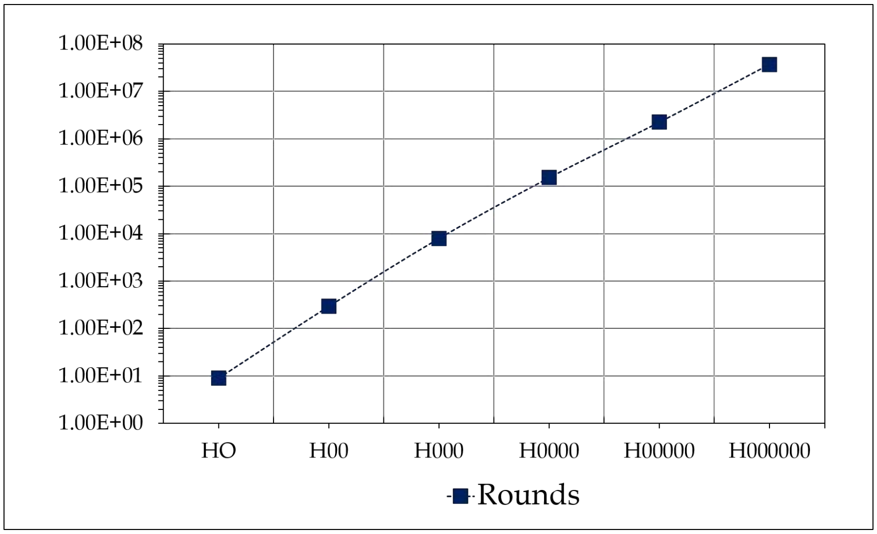


Figure 3.8: Iterations Per Difficulty level

Chapter 4

Conclusion and Recommendations

4.1 Summary of Thesis

This thesis was motivated by technological trends in business and industry (e.g., smart inspection applications), in which sensors are integrated with the ability to retrieve, process and report information through a mesh connected network towards a central information repository. Specifically, this thesis addressed some associated cybersecurity challenges among these trends, such as preserving information integrity, ensuring confidentiality of transactions and validating authenticity of participants. Business and industries are addressing such cybersecurity challenges by adopting Blockchain ledger solutions, employing cryptographic primitives in a manner by which certain cybersecurity risks become mitigated.

However, the benefits of a blockchain-enabled solution also raises some drawbacks in the form of necessitating extra communications and computations that divert resources (e.g., processor, storage, memory, power) from the original application's purpose. If not properly managed, these trade-offs emerge as its own threat to the system's health; for example, while the blockchain may well-maintain integrity, confidentiality, and non-repudiation, there is little value to the increased security if the additional resources it requires compromises the overall availability of the original system or service. This is particularly important in applications (e.g., inspections systems, Internet-of-Things) where the technological devices are already power-constrained (e.g., battery-powered robots). In such power-constrained applications, any additional drain on the power budget in the name of security implies a diminished endurance of the baseline autonomy.

In this context, this thesis explored ways to reduce the energy consumption of specific blockchain operations, namely Merkle Tree (MT) Root calculations and Block Hash generation within Proof-of-Work (PoW) calculations. Our approach firstly employed the Energy Complexity Model to re-engineer the popular SHA256 hash operation, which is a core sub-routine within both MT and PoW operations. The second step in our approach was to emulate a basic version of the blockchain functionalities, using both the Python and C programming languages so that either the baseline or the re-engineered hash algorithm could be employed, all other things equal. The final step of our approach leveraged a third-party energy measurement tool, using the Intel RAPL interface, to experimentally evaluate the energy footprints of both the baseline and optimized implementations. Numerous practical challenges were raised and tackled, including the experimental hardware to guarantee certain boundary conditions as well as repeatability and accuracy of the energy measurements. Also important was the validation that both the baseline and the optimized implementations retained their correctness, involving bit-by-bit debugging rounds between our hash implementations as well as hashing tools out of our experimental scope.

It is worth emphasizing that the energy savings was observed over a variety of input size assumptions. For Merkle-Tree operations, sizes were selected according to what is typical in blockchain applications as well as file-sharing applications. The Proof-of-Work operations were fixed to sizes appropriate for blockchains. It should be noted that the set of sizes are in the realm of what emerging standards in IoT devices are prescribing within both inter-device communications and on-device formats for data collection. Thus, while energy savings are demonstrated in this thesis only within emulations of actual IoT-based systems, that these savings are evident for input sizes and variables that are aligned with IoT technologies remains encouraging.

4.2 Suggestions for Future Work

Arguably the most natural next steps of this work is to assess analogous energy-saving opportunities in other applications of the Sha256 Merkle Tree data structure. These include selected network protocols, authentication schemes and a number of file-sharing services.

While the results in this thesis demonstrate the potential for energy savings using the algorithmic re-engineering techniques, it remains conjecture that the reduced energy consumption in our blockchain emulation would extrapolate to comparable reduction in complete and operational blockchains. Future work could examine directions by which the emulation testbench developed herein can be merged with an actual blockchain implementation, offering all the functionalities that a peer node uses while connected to other peers in true distributed fashion. Real-world usage may also feature different sequencing and/or intermittent reliance on hashing primitives, so achieved energy savings may be only probabilistically related to the results demonstrated under deterministic usage patterns here.

Another avenue for future work is to examine the sensitivity of energy savings to different hardware platforms. The energy measurement tool employed here, namely RAPL, is developed for only Intel processors; meanwhile, the Energy Complexity Model (ECM) by which the hash function was re-engineered, is developed currently only for DDR memory architectures. However, current ”smart” devices technologies are anticipated to use other hardware configurations, such as ARM platforms, for which the ECM is not yet developed to analogously exploit prospects of memory/CPU parallelization.

REFERENCES

[1] *Computer Vision in Transport and Traffic*. Faculty of Transport and Traffic Sciences, (FTTS) Zagreb, Croatia, 2018.

[2] Cristina Alcaraz, Juan E Rubio, and Javier Lopez. Blockchain-assisted access for federated smart grid domains: Coupling and features. *Journal of Parallel and Distributed Computing*, 2020.

[3] M. Anwer, A. Saad, and A. Ashfaque. Security of IoT Using Block chain: A Review. *2020 International Conference on Information Science and Communication Technology (ICISCT)*, page 1–5, February 2020.

[4] A. Banafa. Iot and blockchain convergence: benefits and challenges. *IEEE Internet of Things*, 2017.

[5] Ahmed Banafa. Iot and blockchain convergence: benefits and challenges. *IEEE Internet of Things*, 2017.

[6] B. Cao. Performance analysis and comparison of pow, pos and dag based blockchains. *Digital Communications and Networks*, 6(4), November 2020.

[7] C.Castellon, S.Roy, P.Kreidl, A.Dutta, and L.Boloni. Energy efficient merkle trees for blockchains. *IEEE Trustcom*, 2021.

[8] S. Chaudhary, R. Johari, R. Bhatia, K. Gupta, and A. Bhatnagar. Craiot: Concept, review and application(s) of IoT. *2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU)*, page 1–4, April 2019.

[9] Y.-C. Chen, Y.-P. Chou, and Y.-C. Chou. An image authentication scheme using merkle tree mechanisms. *Future Internet*, 11(7), 2019.

[10] Mauro Conti, E Sandeep Kumar, Chhagan Lal, and Sushmita Ruj. A survey on security and privacy issues of bitcoin. *IEEE Communications Surveys & Tutorials*, 20(4):3416–3452, 2018.

[11] B. Dowling, F. Günther, U. Herath, and D. Stebila. Bsecure logging schemes and certificate transparency. In *European Symposium on Research in Computer Security*, volume 452. IACR, 2016.

[12] S. Duangphasuk, P. Duangphasuk, and C. Thammarat. Review of internet of things (IoT): Security issue and solution. In *2020 17th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, page 559–562. 10, June 2020.

[13] Caixiang Fan, Sara Ghaemi, Hamzeh Khazaei, and Petr Musilek. Performance evaluation of blockchain systems: A systematic survey. *IEEE Access*, 8:126927–126950, 2020.

[14] Todd Farrell. Core architecture doubles mem data rate. *Electronic Engineering Times Asia*, 16, 2005.

[15] Shu Fu, Lian Zhao, Xinhua Ling, and Haijun Zhang. Maximizing the system energy efficiency in the blockchain based internet of things. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2019.

[16] Nitin Gaur, Luc Desrosiers, Venkatraman Ramakrishna, Petr Novotny, Salman A Baset, and Anthony O'Dowd. *Hands-on blockchain with hyperledger: building decentralized applications with hyperledger fabric and composer*. Packt Publishing Ltd, 2018.

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

[18] F Imbault, M Swiatek, R De Beaufort, and R Plana. The green blockchain: Managing decentralized energy production and consumption. In *2017 IEEE International Conference on Environment and Electrical Engineering and 2017 IEEE Industrial and Commercial Power Systems Europe (EEEIC/I&CPS Europe)*, pages 1–5. IEEE, 2017.

[19] 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.

[20] M. Kulbacki, J. Segen, W. Kniec, R. Klempousz, K. Kluwakz, and and A.Serester J.Nikodemz, J.Kulbacka. Survey of drones for agriculture automation from planting to harvest. *22nd IEEE International Conference on Intelligent Engineering Systems*, 2018.

[21] R. Kumar and R. Tripathi. Implementation of distributed file storage and access framework using ipfs and blockchain. *Fifth International Conference on Image Information Processing*, pages 246–251, 2019.

[22] T. Laurence. *Blockchain for dummies*. John Wiley Sons, 2017.

[23] Ting Li, Wei Zhang, Ning Chen, Minhui Qian, and Ying Xu. Blockchain technology based decentralized energy trading for multiple-microgrid systems. In *2019 IEEE 3rd Conference on Energy Internet and Energy System Integration (EI2)*, pages 631–636. IEEE, 2019.

[24] Angelique Faye Loe and Elizabeth A Quaglia. Conquering generals: an np-hard proof of useful work. In *Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems*, pages 54–59, 2018.

[25] N. Mangayarkarasi, G. Raghuraman, and S. Kavitha. Influence of computer vision and IoT for pipeline inspection-a review. *2019 International Conference on Computational Intelligence in Data Science (ICCIDS)*, page 1–6, February 2019.

[26] 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.

[27] Michael Mylrea and Sri Nikhil Gupta Gourisetti. Blockchain for smart grid resilience: Exchanging distributed energy at speed, scale and security. In *2017 Resilience Week (RWS)*, pages 18–23. IEEE, 2017.

[28] M.Yonghzi, X.Benyu, D.Jianwu, and C. Y.Biau. Real time detection system for rail surface defects based on machine vision. *Eurasip Journal on Image and Video Processing*, 2018.

[29] S Nakamoto. Bitcoin: a peer-to-peer electronic cash system. whitepaper bitcoin, 2008.

[30] Sana Noor, Wentao Yang, Miao Guo, Koen H van Dam, and Xiaonan Wang. Energy demand side management within micro-grid networks enhanced by blockchain. *Applied energy*, 228:1385–1398, 2018.

[31] D. Puthal, N. Malik, S. P. Mohanty, E. Kougianos, and C. Yang. The blockchain as a decentralized security framework [future directions]. *IEEE Consumer Electronics Magazine*, 7(2):18–21, March 2018.

[32] Ana Reyna, Cristian Martín, Jaime Chen, Enrique Soler, and Manuel Díaz. On blockchain and its integration with iot. challenges and opportunities. *Future generation computer systems*, 88:173–190, 2018.

[33] 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.

[34] 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.

[35] 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.

[36] Sonam Sanju, Sriram Sankaran, and Krishnashree Achuthan. Energy comparison of blockchain platforms for internet of things. In *2018 IEEE International Symposium on Smart Electronic Systems (iSES)(Formerly iNiS)*, pages 235–238. IEEE, 2018.

[37] Sriram Sankaran, Sonam Sanju, and Krishnashree Achuthan. Towards realistic energy profiling of blockchains for securing internet of things. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 1454–1459. IEEE, 2018.

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

[39] Efe Ulas Akay Seyitoglu, Attila Altay Yavuz, and Thang Hoang. Proof-of-useful-randomness: Mitigating the energy waste in blockchain proof-of-work. *Proceedings of the 18th International Conference on Security and Cryptography - SECRYPT,*, 2021.

[40] Y. Sun, L. Zhang, G. Feng, B. Yang, B. Cao, and M. A. Imran. Blockchain-enabled wireless internet of things: Performance analysis and optimal communication node deployment. *IEEE Internet of ThingsJournal*, 6(3):5791–5802, 2019.

[41] Yao Sun, Lei Zhang, Gang Feng, Bowen Yang, Bin Cao, and Muhammad Ali Imran. Blockchain-enabled wireless internet of things: Performance analysis and optimal communication node deployment. *IEEE Internet of Things Journal*, 6(3):5791–5802, 2019.

[42] S. N. Swamy and S. R. Kota. An empirical study on system level aspects of internet of things (IoT). *IEEE Access*, 8:188082–188134, 2020.

[43] Riya Thakore, Rajkumar Vaghashiya, Chintan Patel, and Nishant Doshi. Blockchain-based iot: A survey. *Procedia Computer Science*, 155:704–709, 2019.

[44] S. Vishal and S. Sekhar. Review-iot security research opportunities. *IEEE International Conference on convergence to Digital World Quo-Wadis*, 2020.

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

[46] Tianyu Yang, Qinglai Guo, Xue Tai, Hongbin Sun, Boming Zhang, Wenlu Zhao, and Chenhui Lin. Applying blockchain technology to decentralized operation in future energy internet. In *2017 IEEE Conference on Energy Internet and Energy System Integration (EI2)*, pages 1–5. IEEE, 2017.

[47] B. Yu, J. Wright, S. Nepal, L. Zhu, J. Liu, and R. Ranjan. Trustchain: Establishing trust in the iot-based applications ecosystem usingblockchain. *IEEE Cloud computing*, 5(4):12–23, 2018.

[48] Bin Yu, Jarod Wright, Surya Nepal, Liming Zhu, Joseph Liu, and Rajiv Ranjan. Trust chain: Establishing trust in the iot-based applications ecosystem using blockchain. *IEEE Cloud computing*, 5(4):12–23, 2018.

VITA

Cesar Castellon earned a Bachelor of Science in Electronics from the Military School of Engineering (EMI) – Bolivia in 1997. After graduation, he earned an internship at Philips CFT - Eindhoven, The Netherlands. Back in Bolivia, between 2000 and 2019, Cesar worked many years as a Mobile Networks 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 has also worked as Value Added Services Product Manager for VIVA, furtherly as the Technical Operations Manager for CyT Bolivia, and lately has worked as a Technical Consultant in Health Informatics for the International Development Bank (iADB).

Cesar was accepted as a graduate student at the University of North Florida in early 2020 and joined in a graduate research assistant position in the Electrical Engineering department under the direction of Dr. Patrick Kreidl, Dr. Swapnoneel Roy in the NSF granted project ”Towards Efficient and Secure Agricultural Information Collection Using a Multi-Robot System.” In addition, he completed the course requirements for the Master’s degree in Electrical Engineering in 2021.