Towards a Green Blockchain: Engineering Merkle Tree and Proof of Work for Energy Optimization

Cesar E. Castellon, *Student Member, IEEE,* Swapnoneel Roy, *Member, IEEE,*
O. Patrick Kreidl, *Senior Member, IEEE,* Ayan Dutta, *Member, IEEE,* and Ladislau Bölöni, *Senior Member, IEEE*

***Abstract*—Blockchain-powered smart systems deployed in different industrial applications promise operational efficiencies and improved yields, while significantly mitigating cybersecurity risks. Tradeoffs between availability and security arise at implementation, however, triggered by the additional resources (e.g., memory and computation) required by blockchain-enabled hosts. This paper applies an energy-reducing algorithmic engineering technique for Merkle Tree (MT) root calculations and the Proof of Work (PoW) algorithm, 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 the energy consumption of a computation, we experiment with both the standard and energy-reduced implementations of both algorithms for different input sizes. Our results show that up to 98% reduction in energy consumption is possible within the blockchain's MT construction module, with the benefits typically increasing with larger input sizes. For the PoW algorithm, our results show up to 20% reduction in energy consumption, with the benefits being lower for higher *difficulty levels*. The proposed energy-reducing technique is also applicable to other key elements of blockchain computations, potentially affording even "greener" blockchain-powered systems than implied by only the results obtained thus far on the MT and PoW algorithms.**

***Index Terms*—Blockchain, Merkle Tree, Proof of Work, Energy Optimization.**

I. Introduction

Blockchain technology, popularized by crypto-currency systems, is seeing extensive use in several 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 [1], [2], [3], [4]. One particularly promising use-case is the *Internet of Things (IoT)* [5], [6], [7], which embodies the vision of computing devices communicating with each other to map a physically connected world onto its digital mirror. The IoT vision also motivates prospects of *smart* systems [8] e.g., smart cities, smart homes, smart grid, smart health, smart agriculture. Unfortunately, smart systems also raise critical security and privacy challenges, motivating the vision of blockchain-powered smart systems.

This research study is supported in part by NSF CPS #1932300 and #1931767, Fidelity National Financial Distinguished Professorship in CIS #0583-5504-51, and Cyber Florida #220408 grants.

C. Castellon, S. Roy, and A. Dutta are with the School of Computing, University of North Florida, Jacksonville, FL, USA.

O.P. Kreidl is with the School of Engineering, University of North Florida, Jacksonville, FL, USA.

L. Bölöni is with the Department of Computer Science, University of Central Florida, Orlando, FL, USA.

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 [8] - within a world that demands more environment-friendly ("green") solutions, IoT realizations also face the challenge of energy efficiency i.e., minimizing their energy footprint. Thakore et al. [9] 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 [10]. 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 the energy consumption of blockchain functionality can be of tremendous practical significance.

A. Related Work

Energy efficiency in computation is a widely studied topic, with numerous points-of-view: hardware-specific platforms, operating systems, hypervisors and containers [11]; software development and security [12]; and algorithms [13], [14]. Energy measurements are sometimes obtained by specifically instrumented equipment [15], while other times can leverage hardware providers' Application Programmer Interfaces (APIs) in which firmware counters are queried to provide near real-time information e.g., Running Average Power Limit (RAPL) technology [16]. 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 [17], [18], micro-grid networks [19], [20], [21] and smart contracts [22]). 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.

Some past studies do recognize that the blockchain itself will draw energy away from any symbiotic system it is integrated with. Examples include Sankaran et al. [10] and Sanju et al. [23], 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 [15] 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. [24], 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 by Fu et al. [1], 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 [25], within which the Merkle Tree construction module is essential. Also essential to Nakimoto's blockchain concept are consensus protocols, for which Proof of Work (PoW) is arguably the most popular as the backbone of popular crypto-currencies such as Bitcoin ($\sim$\$700B industry) and Ethereum[1] [26]. However, PoW is notorious for its resource-intensive nature [27], [28] and, therefore, is a concern for resource-limited (mobile) sensors e.g., robots [29]. A quantification metric to measure the effect on the carbon cost of such crypto-currencies is studied in [30]. In recent literature, roboticists have used Blockchain-based consensus protocols, such as PoW, for preventing malicious data integrity attacks on multi-robot systems [31], [32], [33]. As standard robots in today's market have limited on-board battery resources, using another resource-intensive application onboard might be prohibitive. Therefore, novel engineering techniques to reduce such energy consumption in Blockchains is needed, especially for the most popular underlying protocols–this paper takes a significant step in that direction.

Also worth mention is the recent flurry of research to reduce energy consumption in network services and management, in general. Examples include techniques to prevent Distributed Denial-of-Service (DDoS) attacks [34], improve connectivity of autonomous vehicles [35], or implement delay tolerant networks (DTN), all addresssing different root causes of network disconnection problems [36]. We believe this paper, though focused on algorithmic techniques to reduce energy consumption of blockchain-enabled applications, similarly contributes to the broad field of network services and management.

B. Our Scope and Contributions

We study the extent to which two principal elements of blockchain computations, Merkle Tree (MT) construction and the Proof of Work (PoW) algorithm, 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. [13], [14], on the SHA256 encryption algorithm, which is central to both MT and PoW algorithms. Using pyRAPL, a python library that measures an executable's Runtime Average Power Limit (RAPL), we experiment with both the standard and energy-reduced implementations of both algorithms, varying for MT the input size and for PoW the difficulty level, in both algorithms choosing configurations commonly seen within contemporary blockchain implementations. The main contributions of this work are:

- To the best of our knowledge, the first to address the energy optimization of blockchains by re-engineering the implementation of two of its primary component algorithms, Merkle Tree (MT) and Proof-of-Work (PoW).
- Our results show significant reductions in energy consumption: in MT, up to 98% reduction and on-average 50% across the tested input sizes, while in PoW up to 20% reduction and on-average 10% across the tested difficulty levels.

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 MT and PoW results reported here. As a final remark, a preliminary version of this paper appeared in the *2021 IEEE TRUSTCOM* conference [37], treating only the MT construction algorithm. This paper generalizes the experimental approach to examine energy optimization prospects within both MT and PoW and, in turn, presents the PoW results for the first time.

[1]Ethereum will soon adopt the Proof of Stake (PoS) consensus protocol.

II. Overview of Merkle Tree and Proof of Work

A. Merkle Tree (MT) Based Block Generation

A graphic representation of a simple MT based block generation in a blockchain is shown in Fig. 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 involve the recursive hash computation starting from these leaves until a final hash determines the Merkle Tree root (labeled TX_ROOT in Fig. 1).

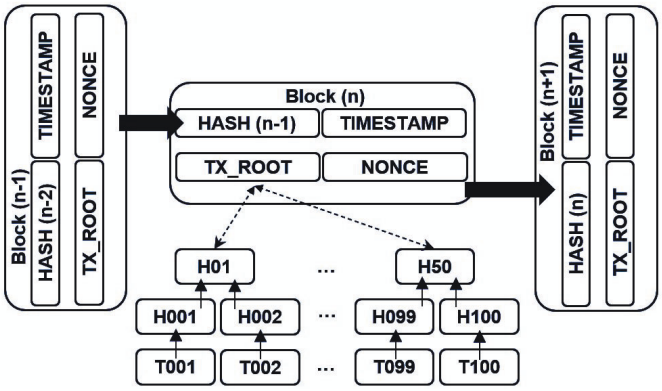


Figure 1: Basic block generation in a blockchain.

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.,

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

$$T = cost[energy, time] \quad (2)$$

That is, the block generation is represented by the state transition in (1), which depends upon a function $f(T)$ with

parameter T denoting the $cost$ as represented by (2). This cost has two main components: one is the energy consumed by the hardware devices to compute the hash of 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 described in Section III-A.

B. Proof of Work (PoW)

In crypto-economic blockchains such as Bitcoin, or Ethereum, node miners run the Proof of Work (PoW) algorithm, competing to create new blocks filled with processed transactions. The PoW algorithm sets the operation rules and mining difficulty setting the pace for adding valid blocks to the chain.

To create a block in PoW, a miner will repeatedly put an actual downloaded dataset through a hashing function, looking for a result below a target nonce as dictated by the block difficulty parameter, 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 *chain selection rule* by which the *correct* chain is decided in the event that multiple paths evolve in parallel. Bitcoin, for instance, uses the Nakamoto rule that selects the "longest chain" as the correct one. Finally, PoW is not only applicable to consensus protocols in blockchains: it also works as block author selector [38] and a Sybil-resistance mechanism [39]. 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.

1) Implementation of PoW: The PoW algorithm involves 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.

Arguably, the most widely adopted algorithm for PoW is the Hashcash scheme [40], [41], [42], [43], [44]. To expedite our experimentation, yet still reliably emulate blockchain hashing operations, we implemented a simplified version of this algorithm (Algorithm 1). In our implementation, a difficulty is defined first – how many trailing or preceding 0's are there in the hash value of a block. The more 0's are there, the more difficult it is to find the hash value. To find this hash value, an integer number, called *nonce*, is used, which is initialized to 0 and increased by 1 within a for loop. If the number of 0's (i.e., the difficulty level) match the number of 0's in the hash value of the nonce variable, the loop breaks since the appropriate nonce is found. The hash value is calculated using the SHA256 algorithm. A maximum loop number is maintained – if the proper nonce is not found by this maximum time, the nonce is reset to 0 and the program returns an exception. This process of finding the proper nonce whose hash matches the the intended difficulty level is called *mining*.

Algorithm 1: BlockHash Mining Algorithm (PoW)

```
Data: nonce, Block(B), difficulty
Result: B.hash, nonce
nonce ← 0;
diff ← fill("0", difficulty);          /* Difficulty = #0s */
for MAX_STEPS do
    B.Hash ← SHA256(B.Params+nonce);
    if substr(B.Hash,difficulty) = diff then
        break ;                        /* Match found */
    else
        if i = 2^256 then
            nonce ← 0 ;                /* Reset nonce */
            break;
        end
    end
    nonce ← nonce + 1 ;                /* Try the next nonce */
end
```

III. Engineering MT and PoW for Energy Conservation

In this section, we describe a technique for reducing the energy consumption of SHA256 by an engineering technique using the Energy Complexity Model (ECM) designed in [13]. We then apply the energy-optimized SHA256 to both MT and PoW to measure the reductions of energy consumption. Before illustrating the engineering techniques, we briefly describe ECM in the next section.

A. The Energy Complexity Model (ECM)

The ECM developed in [13] is built upon an abstraction of the Double Data Rate Synchronous Dynamic Random Access Memory (DDR SDRAM) architecture [45] illustrated in Fig. 2. The main memory in DDR is divided into *banks*, each of which contains a certain number of *chunks*[2]. 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 (each with their

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

own sense amplifier) can be accessed in parallel. Therefore, if the DDR memory is organized into P banks (where $P = 4$ in Fig. 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.

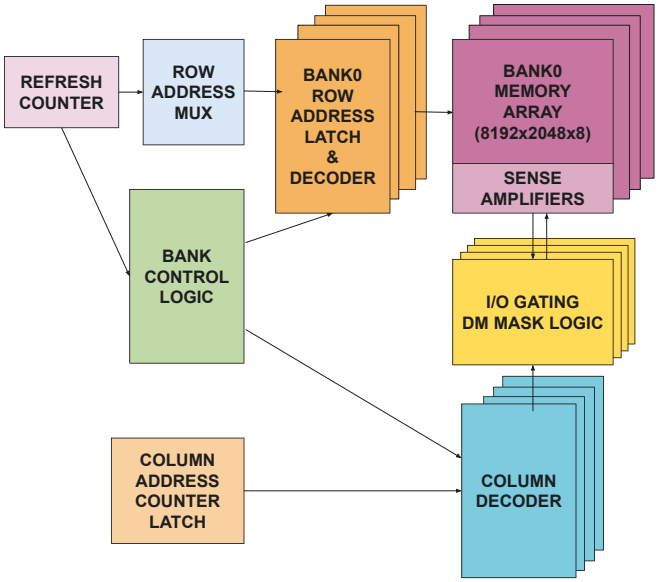


Figure 2: Internal DDR SDRAM memory chip block diagram.

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 bytes and its own cache C_i. The illustrative example of Fig. 3 assumes $P = 4$ banks, as was the case in Fig. 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 [13] 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, parallelizing chunk accesses offers the potential for energy reduction for any algorithm! More formally, the energy consumption (in Joules) of an algorithm $\mathcal{A}$ with computational complexity (execution time) $W(\mathcal{A})$, assuming a P-bank DDR3 architecture with B bytes per chunk, is given by

$$E(\mathcal{A}) = W(\mathcal{A}) + (P \times B)/I \tag{3}$$

where I is the *parallelization index*, essentially the number of parallel block accesses across memory banks per P block accesses made by $\mathcal{A}$ 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.

1	5	9	13
2	6	10	14
3	7	11	15
4	8	12	16
C1	C2	C3	C4

Figure 3: ECM for DDR3 resource with $P = 4$ banks

Equation 3 implies, for $W(\mathcal{A}) \gg (P \times B)/I$, $E(\mathcal{A}) \approx W(\mathcal{A})$. In other words, for algorithms with high computational complexity (e.g. exponential execution time), the energy complexity becomes equivalent to its computational complexity.

B. Re-engineering Hash Calculations Using ECM

In this work, we engineer the hash algorithm of Line 4 of Algorithm 1 based on ECM to reduce energy consumption. First, we briefly describe how any algorithm $\mathcal{A}$ can be parallelized based on ECM. We then illustrate how, specifically the SHA hash algorithm, is re-engineered for parallelization.

1) Parallelizing any algorithm: Given an algorithm $\mathcal{A}$, the input to $\mathcal{A}$ is considered to identify the most common access sequence in $\mathcal{A}$. 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 $\mathcal{A}$. As mentioned above, the order of chunk accesses is different for different levels of parallelization. But the physical location (chunk) 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 $\mathbf{V}$ is generated for each level of parallelization, which defines the ordering among the chunks to be accessed (see Fig. 4).

For 1-way access, the page table vector $\mathbf{V}$ has the pattern $(1, 2, 3, 4, \ldots)$ while 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 2 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. 3). On the other hand, for $P = 4$, jumps of 4 ensure that 4 consecutive chunk accesses lie in 4 different banks (banks 1 through 4 of Fig. 3).

2) Parallelizing SHA for MT and PoW: As described earlier, both MT and PoW (Algorithm 1) perform their respective hash

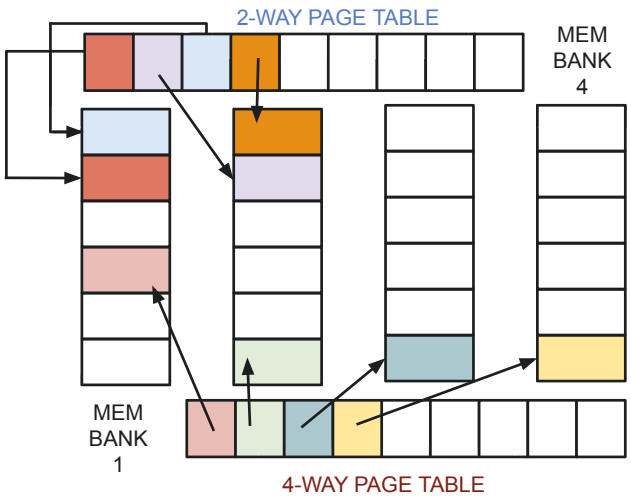


Figure 4: Memory layout ($P = 4$) and role of page tables

Algorithm 2: Create a Page Table for N Chunks

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

1 $factor = 0$;
2 **for** $i = 0$ *to* $\frac{N}{B} - 1$ **do**
3 **if** $i > 1$ *and* $(i \times jump) \mod \frac{N}{B} = 0$ **then**
4 $factor = factor + 1$;
5 **end**
6 $\mathbf{V}_i = (i \times jump + factor) \mod \frac{N}{B}$;
7 **end**

calculations via the repeated use of the SHA256 algorithm. As shown in Fig. 5, the input to SHA256 is partitioned into fixed size message blocks, presented in sequence to separate compression functions.

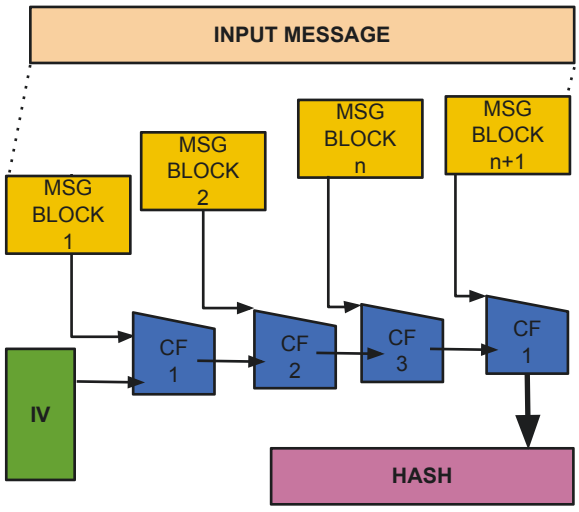


Figure 5: Illustration of the SHA256 algorithm

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 is pre-processed into another vector by applying Algorithm 2. 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. 6. Fig. **??** shows the outcome of re-engineering the SHA256

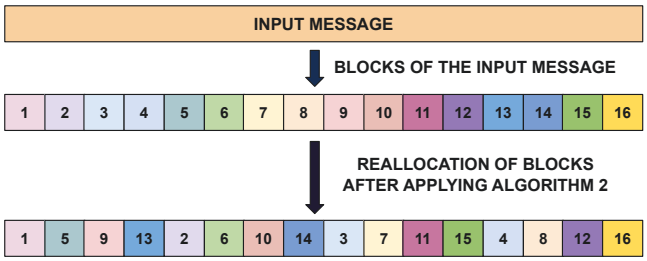


Figure 6: Mapping of SHA input blocks based on ECM

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

Theorem III.1. *The re-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. 5 [46]. Algorithm 2 has a computational complexity of $\Theta(N)$ since the for loop of line 2 executes exactly $\frac{N}{B}$ times. Therefore applying Algorithm 2 to SHA256 as illustrated in Fig. 6 does not change the overall computational complexity of SHA256. □

Theorem III.1 implies that applying the re-engineered SHA256 in place of the original SHA256 in any algorithm *will not modify* the computational complexity of the algorithm.

3) Applying the re-engineered SHA256 to MT: To recap, a graphic representation of a simple MT-based block generation in a blockchain is shown in Fig. 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 involve the recursive hash computation starting from these leaves until a final hash determines the Merkle Tree root (labeled TX_ROOT in Fig. 1). Fig. 7 shows the input to the SHA256 hash for the MT algorithm.

Section IV describes the results of a series of experiments conducted to calculate the energy savings obtained by incorporating the re-engineered SHA256 algorithm into MT.

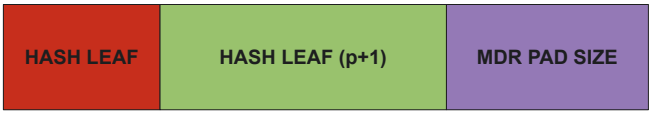


Figure 7: Input to SHA256 for the MT algorithm

4) Engineered SHA256 Applied to PoW: For PoW, Line 4 of Algorithm 1 uses SHA256. The input vector in Line 4 of Algorithm 1 being the concatenation of the string (B.Params) and the nonce (Fig. 8) is pre-processed into another vector by applying Algorithm 2.

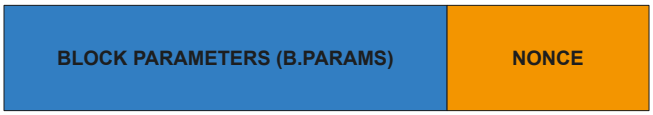


Figure 8: Input to SHA256 for the PoW algorithm

Theorem III.2. *For hash size h, Algorithm 1 has a computational complexity of $O(2^h)$.*

Proof. Algorithm 1 attempts to find a hash value satisfying the difficulty level by brute-forcing over every possible hash values. The computational complexity stated in Theorem III.2 therefore follows. □

Theorem III.2 implies, the energy complexity of Algorithm 1 (or PoW) will converge with its computational complexity based on the size of the hash function used at some point. In this work, we therefore attempted to optimize the *hash function* (SHA256) used in Algorithm 1 to investigate whether that reduces the overall energy consumption of the PoW algorithm. Next, we present the results of our experiments with these proposed energy-optimized MT root calculation and PoW versions.

IV. Experiments

In the following we describe a series of experiments designed to quantify the energy savings of the methodology described 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 the energy consumption 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 [16].

A. Implementation Details and Setup

Our experimental objectives could not be met by using the SHA256 function in the Hash Python library 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 in both MT and PoW algorithms.

1) MT Experimental Details: 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 mean and standard deviation results. 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 [13];
2) the 64B, 96B and 128 inputs are common in blockchain applications [6];
3) the 512B and 1024B inputs are common in file hashing applications [47]; while
4) the 16384B and 262144B inputs are common in the Interplanetary File System (IPFS) [48], [49].

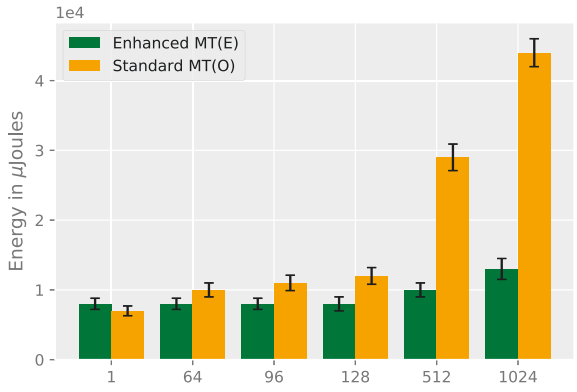
2) PoW Experimental Details: The standard PoW implementation is labeled by "O" that uses the original SHA256 hashing algorithm, while the implementation using the proposed re-engineered SHA256 algorithm is labeled by "E". The input size is fixed to 256 bytes including all the block header parameters and byte padding, while we experimented with six difficulty levels ranged from 1 through 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 standard deviation over repeated trials) of the emulated PoW calculations.

B. Results and Discussion

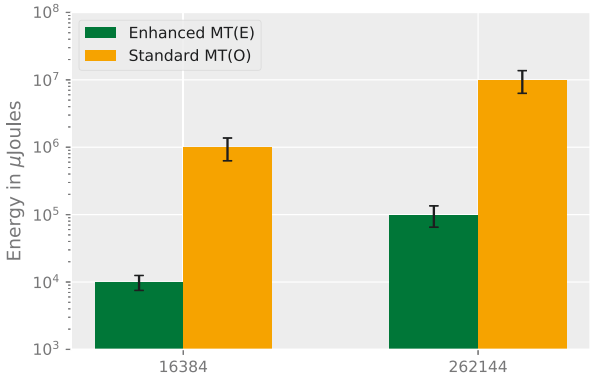
We discuss experimental results from both sets of experiments (MT and PoW) separately in this section.

1) MT Results: 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. 9 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. 9(a) renders the comparison over the six smallest input sizes (using a linearly-scaled vertical axis), while Fig. 9(b) is over the two largest input sizes (using a log-scaled vertical axis). It is seen that the ECM-enhanced implementation consistently requires less energy than the standard implementation, the difference being increasingly significant with the larger input sizes that befit file hashing applications (i.e., 512B and above) while still remaining meaningful for input sizes of 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.



(a) Small input sizes (in Bytes)



(b) Large input sizes (in Bytes)

Figure 9: Comparison of average energy consumption in MT

Fig. 10 presents the average energy comparison on more relative terms, namely as a percent reduction achieved by the

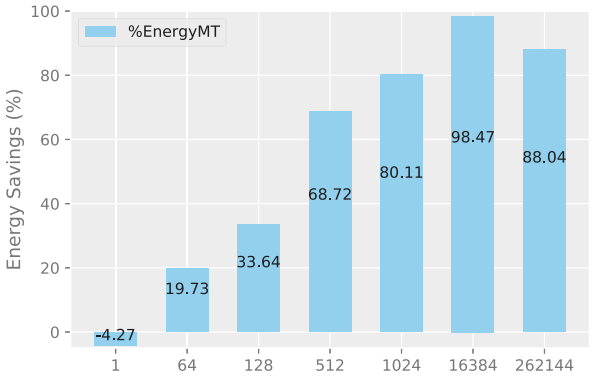


Figure 10: Energy savings based on various input sizes in MT

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 98.47% savings. As noted in Fig. 9, the 1B input renders a savings of -4.27%, 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.

2) PoW Results: Fig. 11 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). It

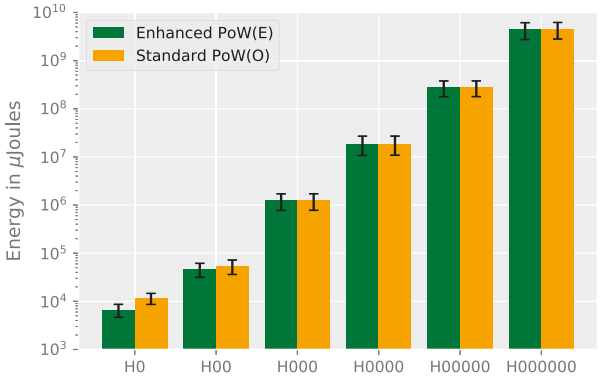


Figure 11: Comparison of average energy consumption in PoW per difficulty level (with 1-sigma standard deviation over 1000 trials)

is observed from Fig. 11 that there exists energy savings when comparing both implementations, although the savings percentage is inversely proportional to the difficulty level. The results in Fig. 11 tally with what we had derived in Section III-B4 in light of Theorem III.2. With higher difficulty levels the energy complexity of PoW based on the ECM converges with its computational complexity based on the

Turing Model. Our experimental results underscore what we had theoretically analyzed about PoW in terms of its energy consumption. To further illustrate this, in Fig. 12, the average

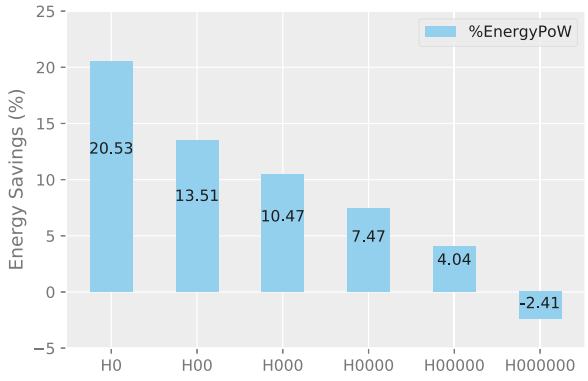


Figure 12: Energy savings in PoW per difficulty

energy comparison in relative terms versus difficulty level, namely as a percent reduction achieved by the enhanced implementation over the standard implementation, is presented. The energy savings in PoW range between 4% and 20%, the case of Difficulty 1 exhibiting a maximum saving of 20%. We observe steady reduction in energy savings as the difficulty level increases (Fig. 12). This again tallies with our theoretical energy complexity analysis of PoW in Section III-B4.

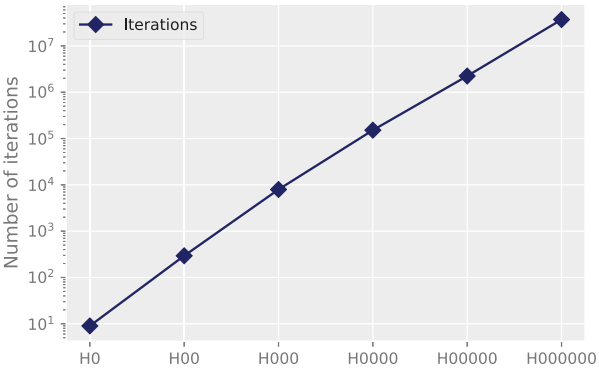


Figure 13: Iterations per difficulty level in PoW (y-axis in log-scale)

As further illustration, Fig. 13 displays the exponential rise in the number of iterations within the PoW operations with increasing difficulty levels, signifying the exponential runtime dependency of PoW (Algorithm 1) based on the hash size and difficulty level as a higher difficulty level forces Algorithm 1 to iterate through the *for* loop (line 3) exponentially more times. This complements Theorem III.2 about the energy complexity of Algorithm 1 converging with its computational complexity based on size of the hash function used at some point.

C. Approximate Cost-Savings Analysis

We present an approximate cost-reduction analysis on implementing the energy optimization techniques on MT and PoW. Evaluating possible power optimization needs total time estimation for every operation. Figs. 14 and 15 respectively illustrate the percentage of savings for time (μsec) and power(μWatt) for MT and PoW.

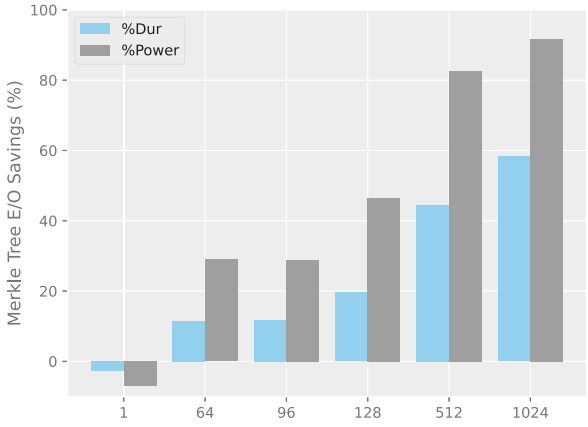


Figure 14: Savings in Merkle Tree duration and power

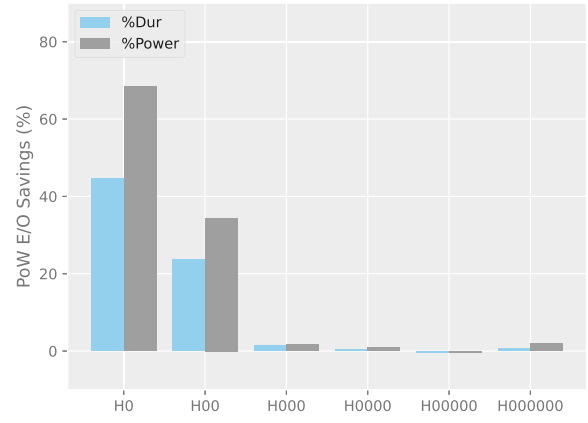


Figure 15: Savings in PoW duration and power

The following formulas have been used to estimate energy costs in the analysis.

Energy consumption calculation. The energy E in kilowatt-hours (kWh) per day is equal to the power P in watts (W) times number of usage hours per day t divided by 1000 watts per kilowatt.

$$E_{(kWh/day)} = P_{(W)} \times t_{(h/day)}/1000_{(W/kW)} \quad (4)$$

Electricity cost calculation. The electricity cost per day in dollars is equal to the energy consumption E in kWh per day times the energy cost of 1 kWh in ¢/kWh divided by 100 (¢/$).

$$Cost_{(\$/day)} = E_{(kWh/day)} \times Cost_{(¢/kWh)}/100 \quad (5)$$

To estimate a yearly energy savings by our optimized PoW we make the following assumptions: The system generates 500 blocks per hour, a total of 16 hours is consumed in daily operation. Additionally, residential electricity rates in Florida average $11.42¢/kWh$. Fig. 16 depicts the calculated yearly energy savings based on the above assumptions.

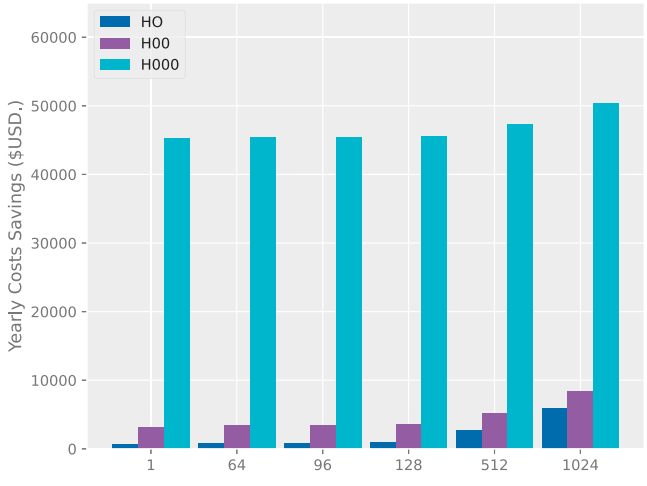


Figure 16: Example of potential yearly savings for a ECM BCH-powered application

D. Discussion on Network Throughput Change

Blockchain nodes are authorized stakeholders to keep track of the distributed ledger, and may also serve as communication proxies (e.g hierarchical distributed cryptocurrency system) for different network tasks, when the underlying topology is complex, or a highly populated ecosystem is used. In simpler topologies (e.g. a fully meshed interconnected robot fleet), all nodes are considered to have access to the ledger, and no proxy functions are implemented on them.

This work has been motivated by the latter topology type, where the ledger life depends only on *duration of the mission*, therefore parameters like node size or network throughput have not been considered for evaluation while conducting experiments (since this is a limited-size, low-volume, small-scale transactions scenario.

Nevertheless, based on Theorem III.1 it can be inferred that since the ECM modification lies in the core programming of the nodes (it is a modification of an algorithm), it does not add any additional data to the transactions. Since the resulting hashes have the same size as the original ones, there will not be any increase in size for any given node. Therefore we conjecture that our energy optimization techniques will not have influence on overall network throughput.

However, it will be interesting to observe any possible changes in Read Latency, Read Throughput, Transaction Latency, Transaction Throughput as Key Performance Indicators depending on the type of Blockchain application to implement while applying our techniques for energy optimization.

V. A Comparison with Existing Techniques

Most of the focus on energy optimization in blockchain has either been in designing new hardware or designing *energy-lighter* versions of existing protocols. A few recent techniques include:

1) **Specializing the Datacenter.** Usage of Clouds in blockchain. Power consumption has been significantly reduced by using GPU and FPGA based clouds especially when dealing with intensive workloads [50].
2) **Resource-efficient mining.** This technique is based on a novel trusted hardware by Intel named as Software Guard Extension (SGX). The idea is to smartly curtail the infinite loop of PoW to design Proof of Useful Work (PoUW), involving miners that provide trustworthy reporting on CPU cycles [51].
3) **Sawtooth blockchain software.** Intel proposed a novel energy-saving blockchain system that incorporates the security features into the chipmaker's CPU. This system software randomly selects users for writing the block. The underlying algorithm is called Proof of Elapsed-Time that makes miners sleep and wake insead of constantly mining as in PoW [52].
4) **Side Chains** This method has evolved over Bitcoin and Ethereum network using again an alternate algorithm called Proof of Authority (PoA) to determine miners. This is mostly used in private blockchains [53].

The notable part is to the best of our knowledge, ours is the first work that attempts to re-engineer PoW based on the ECM to reduce energy consumption in it. Our technique is generic so that it can be also applied to the lighter algorithms as mentioned above to further reduce their energy consumption.

VI. Conclusion and Future Work

This paper described a technique to reduce the energy consumption of the Merkle Tree (MT) and Proof of Work (PoW) computations within blockchanns. At the technique's core is a re-engineered SHA256 hashing algorithm based upon the Energy Complexity Model (ECM) [13]. The ECM-enhanced implementations were compared to the standard implementations via experimental energy measurements, varying input sizes within the MT experiments and varying difficulty level within PoW experiments, both including configurations of practical significance. The results for MT show that up to 34% energy savings is possible for input sizes typically used by blockchains, while up to 98% is possible for input sizes used by other applications (e.g., file systems). The results for PoW show up to 20% energy savings is possible, yielding diminished savings with increasing difficulty levels as the theoretical energy complexity analysis of PoW predicts. Due to limitations of our current experimental infrastructure, we can only conjecture that the reduced energy consumption in these algorithm modules (MT and PoW) extrapolates to a comparable reduction for blockchains on the whole. Should such a conjecture hold, however, numerous applications could render "greener" blockchained-enabled networked systems

e.g., autonomous vehicles [54], crypto-currencies [55], cloud-based software defined networks [56], intrusion detection systems [57] as well as Internet-of-Things (IoT) technology [5], [6], [7] and smart systems [8].

Natural next steps of this work are to assess the energy-saving opportunities in other applications of Merkle Trees e.g., authentication schemes [49], healthcare systems [58], embedded systems [59], network protocols [60], [61]. Also of interest is the exploration to reduce energy consumption of the different variations and implementations of PoW algorithm (e.g. [31], [32], [62], [63], [64]), and the alternative Proof of Stake (PoS) algorithm [31], [65]. Future work could also 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] C. Fan, S. Ghaemi, H. Khazaei, and P. Musilek, "Performance evaluation of blockchain systems: A systematic survey," *IEEE Access*, vol. 8, pp. 126 927–126 950, 2020.

[2] M. Conti, E. S. Kumar, C. Lal, and S. Ruj, "A survey on security and privacy issues of bitcoin," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3416–3452, 2018.

[3] N. Gaur, L. Desrosiers, V. Ramakrishna, P. Novotny, S. A. Baset, and A. O'Dowd, *Hands-on blockchain with hyperledger: building decentralized applications with hyperledger fabric and composer*. Packt Publishing Ltd, 2018.

[4] C. Alcaraz, J. E. Rubio, and J. Lopez, "Blockchain-assisted access for federated Smart Grid domains: Coupling and features," *Journal of Parallel and Distributed Computing*, vol. 144, pp. 124–135, October 2020.

[5] B. Yu, J. Wright, S. Nepal, L. Zhu, J. Liu, and R. Ranjan, "TrustChain: Establishing trust in the IoT-based applications ecosystem using blockchain," *IEEE Cloud Computing*, vol. 5, no. 4, pp. 12–23, 2018.

[6] 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 Things Journal*, vol. 6, no. 3, pp. 5791–5802, 2019.

[7] A. Banafa, "IoT and blockchain convergence: benefits and challenges," *IEEE Internet of Things*, 2017.

[8] A. Reyna, C. Martín, J. Chen, E. Soler, and M. Díaz, "On blockchain and its integration with IoT. Challenges and opportunities," *Future Generation Computer Systems*, vol. 88, pp. 173–190, 2018.

[9] R. Thakore, R. Vaghashiya, C. Patel, and N. Doshi, "Blockchain-based IoT: A survey," *Procedia Computer Science*, vol. 155, pp. 704–709, 2019.

[10] S. Sankaran, S. Sanju, and K. Achuthan, "Towards realistic energy profiling of blockchains for securing internet of things," in *Proc. of 2018 IEEE 38th Int. Conf. on Distributed Computing Systems (ICDCS)*, 2018, pp. 1454–1459.

[11] J. Westin, "Evaluation of energy consumption in virtualization environments: proof of concept using containers," 2017.

[12] P. D. Harish, "Towards designing energy-efficient secure hashes," Master's thesis, University of North Florida, 2015.

[13] S. Roy, A. Rudra, and A. Verma, "An energy complexity model for algorithms," in *Proc. of the 4th Conf. on Innovations in Theoretical Computer Science*, 2013, pp. 283–304.

[14] ——, "Energy aware algorithmic engineering," in *Proc. 22nd IEEE Int. Symp. on Modelling, Analysis & Simulation of Computer and Telecommunication Systems*, 2014, pp. 321–330.

[15] C. A. Roma and M. A. Hasan, "Energy consumption analysis of XRP validator," in *Proc. of 2020 IEEE Int. Conf. on Blockchain and Cryptocurrency (ICBC)*. IEEE, 2020, pp. 1–3.

[16] M. Santos, J. Saraiva, Z. Porkoláb, and D. Krupp, "Energy consumption measurement of C/C++ programs using Clang tooling," in *Proc. of 6th Workshop of Software Quality, Analysis, Monitoring, Improvement, and Applications*, 2017.

[17] F. Imbault, M. Swiatek, R. De Beaufort, and R. Plana, "The green blockchain: Managing decentralized energy production and consumption," in *Proc. of 2017 IEEE Int. Conf. on Environment and Electrical Engineering and 2017 IEEE Industrial and Commercial Power Systems Europe (EEEIC/I&CPS Europe)*, 2017, pp. 1–5.

[18] T. Yang, Q. Guo, X. Tai, H. Sun, B. Zhang, W. Zhao, and C. Lin, "Applying blockchain technology to decentralized operation in future energy internet," in *Proc. of 2017 IEEE Conf. on Energy Internet and Energy System Integration (EI2)*, 2017, pp. 1–5.

[19] E. Mengelkamp, B. Notheisen, C. Beer, D. Dauer, and C. Weinhardt, "A blockchain-based smart grid: towards sustainable local energy markets," *Computer Science-Research and Development*, vol. 33, no. 1-2, pp. 207–214, 2018.

[20] T. Li, W. Zhang, N. Chen, M. Qian, and Y. Xu, "Blockchain technology based decentralized energy trading for multiple-microgrid systems," in *Proc. of IEEE 3rd Conf. on Energy Internet and Energy System Integration (EI2)*, 2019, pp. 631–636.

[21] S. Noor, W. Yang, M. Guo, K. H. van Dam, and X. Wang, "Energy demand side management within micro-grid networks enhanced by blockchain," *Applied Energy*, vol. 228, pp. 1385–1398, 2018.

[22] M. Mylrea and S. N. G. Gourisetti, "Blockchain for smart grid resilience: Exchanging distributed energy at speed, scale and security," in *2017 Resilience Week (RWS)*, 2017, pp. 18–23.

[23] S. Sanju, S. Sankaran, and K. Achuthan, "Energy comparison of blockchain platforms for internet of things," in *Proc. 2018 IEEE Int. Symp. on Smart Electronic Systems (iSES)(Formerly iNiS)*, 2018, pp. 235–238.

[24] S. Fu, L. Zhao, X. Ling, and H. Zhang, "Maximizing the system energy efficiency in the blockchain based Internet of Things," in *Proc. IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6.

[25] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system," 2008.

[26] M. Salimitari, M. Chatterjee, and Y. P. Fallah, "A survey on consensus methods in blockchain for resource-constrained IoT networks," *Internet of Things*, p. 100212, 2020.

[27] A. De Vries, "Bitcoin's growing energy problem," *Joule*, vol. 2, no. 5, pp. 801–805, 2018.

[28] L. Dittmar and A. Praktiknjo, "Could bitcoin emissions push global warming above 2° c?" *Nature Climate Change*, vol. 9, no. 9, pp. 656–657, 2019.

[29] A. Dutta, S. Roy, O. P. Kreidl, and L. Bölöni, "Multi-robot information gathering for precision agriculture: Current state, scope, and challenges," *IEEE Access*, vol. 9, pp. 161 416–161 430, 2021.

[30] M. J. Krause and T. Tolaymat, "Quantification of energy and carbon costs for mining cryptocurrencies," *Nature Sustainability*, vol. 1, no. 11, pp. 711–718, 2018.

[31] T. Samman, J. Spearman, A. Dutta, O. P. Kreidl, S. Roy, and L. Bölöni, "Secure multi-robot adaptive information sampling," in *Proc. 2021 IEEE Int. Symp. on Safety, Security, and Rescue Robotics (SSRR)*, 2021, pp. 125–131.

[32] T. Samman, A. Dutta, O. P. Kreidl, S. Roy, and L. Bölöni, "Secure multi-robot information sampling with periodic and opportunistic connectivity," in *Proc. 2022 IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2022, pp. 1–7.

[33] V. Strobel, E. Castelló Ferrer, and M. Dorigo, "Blockchain technology secures robot swarms: a comparison of consensus protocols and their resilience to byzantine robots," *Frontiers in Robotics and AI*, vol. 7, p. 54, 2020.

[34] R. F. Hayat, S. Aurangzeb, M. Aleem, G. Srivastava, and J. C.-W. Lin, "Ml-ddos: A blockchain-based multilevel ddos mitigation mechanism

for iot environments," *IEEE Transactions on Engineering Management*, 2022.

[35] A. Riyal, G. Kumar, D. K. Sharma, K. D. Gupta, and G. Srivastava, "Blockchain tree powered green communication for efficient and sustainable connected autonomous vehicles," *IEEE Transactions on Green Communications and Networking*, 2022.

[36] A. K. Singh, R. Pamula, and G. Srivastava, "An adaptive energy aware dtn-based communication layer for cyber-physical systems," *Sustainable Computing: Informatics and Systems*, vol. 35, p. 100657, 2022.

[37] C. Castellon, S. Roy, P. Kreidl, A. Dutta, and L. Bölöni, "Energy efficient merkle trees for blockchains," in *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 2021, pp. 1093–1099.

[38] H. Y. Yuen, F. Wu, W. Cai, H. C. Chan, Q. Yan, and V. C. Leung, "Proof-of-play: A novel consensus model for blockchain-based peer-to-peer gaming system," in *Proc. 2019 ACM Int. Symp. on Blockchain and Secure Critical Infrastructure*, 2019, pp. 19–28.

[39] P. Otte, M. de Vos, and J. Pouwelse, "TrustChain: A sybil-resistant scalable blockchain," *Future Generation Computer Systems*, vol. 107, pp. 770–780, 2020.

[40] A. Back *et al.*, "Hashcash-a denial of service counter-measure," 2002.

[41] K. Chatterjee, A. K. Goharshady, and A. Pourdamghani, "Hybrid mining: exploiting blockchain's computational power for distributed problem solving," in *Proc. ACM/SIGAPP Symp. on Applied Computing*, 2019, pp. 374–381.

[42] D. Dasgupta, J. M. Shrein, and K. D. Gupta, "A survey of blockchain from security perspective," *Journal of Banking and Financial Technology*, vol. 3, no. 1, pp. 1–17, 2019.

[43] Z. Feng and Q. Luo, "Evaluating memory-hard proof-of-work algorithms on three processors," *Proc. of the VLDB Endowment*, vol. 13, no. 6, pp. 898–911, 2020.

[44] A. Meneghetti, M. Sala, and D. Taufer, "A survey on PoW-based consensus," *Annals of Emerging Technologies in Computing (AETiC)*, pp. 2516–0281, 2020.

[45] T. Farrell, "Core architecture doubles mem data rate," *Electronic Engineering Times Asia*, vol. 16, 2005.

[46] D. Rachmawati, J. Tarigan, and A. Ginting, "A comparative study of message digest 5 (md5) and sha256 algorithm," in *Journal of Physics: Conference Series*, vol. 978, no. 1. IOP Publishing, 2018, p. 012116.

[47] B. Dowling, F. Günther, U. Herath, and D. Stebila, "Bsecure logging schemes and certificate transparency," in *Proc. European Symp. on Research in Computer Security*, vol. 452, 2016.

[48] R. Kumar and R. Tripathi, "Implementation of distributed file storage and access framework using IPFS and blockchain," *Proc. Fifth International Conference on Image Information Processing*, pp. 246–251, 2019.

[49] Y.-C. Chen, Y.-P. Chou, and Y.-C. Chou, "An image authentication

[58] U. Chelladurai and S. Pandian, "HARE: A new hash-based authenticated reliable and efficient modified Merkle tree data structure to ensure integrity of data in the healthcare systems," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–15, 2021.

scheme using Merkle tree mechanisms," *Future Internet*, vol. 11, no. 7, 2019.

[50] R. Nair, S. Gupta, M. Soni, P. K. Shukla, and G. Dhiman, "An approach to minimize the energy consumption during blockchain transaction," *Materials Today: Proceedings*, 2020.

[51] F. Zhang, I. Eyal, R. Escriva, A. Juels, and R. Van Renesse, "{REM}:{Resource-Efficient} mining for blockchains," in *26th USENIX Security Symposium (USENIX Security 17)*, 2017, pp. 1427–1444.

[52] P. A. H. S. C. PokitDok, "with intel® blockchain technology," *Business Wire (English). 5AD*.

[53] S. Lee, "Explaining side chains, the next breakthrough in blockchain," *Forbes. com*, 2018.

[54] R. Kumar, P. Kumar, R. Tripathi, G. P. Gupta, T. R. Gadekallu, and G. Srivastava, "Sp2f: A secured privacy-preserving framework for smart agricultural unmanned aerial vehicles," *Computer Networks*, vol. 187, p. 107819, 2021.

[55] A. de Vries, "Bitcoin boom: What rising prices mean for the network's energy consumption," *Joule*, vol. 5, no. 3, pp. 509–513, 2021.

[56] P. Velmurugadass, S. Dhanasekaran, S. S. Anand, and V. Vasudevan, "Enhancing blockchain security in cloud computing with iot environment using ecies and cryptography hash algorithm," *Materials Today: Proceedings*, vol. 37, pp. 2653–2659, 2021.

[57] R. Kumar, R. Tripathi, N. Marchang, G. Srivastava, T. R. Gadekallu, and N. N. Xiong, "A secured distributed detection system based on ipfs and blockchain for industrial image and video data security," *Journal of Parallel and Distributed Computing*, vol. 152, pp. 128–143, 2021.

[59] Y. Zou and M. Lin, "Fast: A frequency-aware skewed Merkle tree for FPGA-secured embedded systems," in *Proc. 2019 IEEE Computer Society Annual Symposium on VLSI*, 2019, pp. 326–331.

[60] S. You, F. Xue, Z. Qi, and H. Liu, "Wireless sensor network protocol based on hierarchical Merkle tree," in *Proc. Int. Conf. On Signal And Information Processing, Networking And Computers*, 2018, pp. 9–15.

[61] S. Dhumwad, M. Sukhadeve, C. Naik, K. Manjunath, and S. Prabhu, "A peer to peer money transfer using SHA256 and Merkle tree," in *Proc. 2017 23rd Annual International Conference in Advanced Computing and Communications (ADCOM)*, 2017, pp. 40–43.

[62] L. M. Bach, B. Mihaljevic, and M. Zagar, "Comparative analysis of blockchain consensus algorithms," in *Proc. 2018 41st Int. Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2018, pp. 1545–1550.

[63] V. Gramoli, "From blockchain consensus back to byzantine consensus," *Future Generation Computer Systems*, vol. 107, pp. 760–769, 2020.

[64] A. A. Sabbagh, "Blockchain implementations," Ph.D. dissertation, Lebanese American University, 2019.

[65] C. T. Nguyen, D. T. Hoang, D. N. Nguyen, D. Niyato, H. T. Nguyen, and E. Dutkiewicz, "Proof-of-stake consensus mechanisms for future blockchain networks: fundamentals, applications and opportunities," *IEEE Access*, vol. 7, pp. 85 727–85 745, 2019.