*mathematics*

MDPI

*Article*

# ASIC-Resistant Proof of Work Based on Power Analysis of Low-End Microcontrollers

**Hyunjun Kim, Kyungho Kim, Hyeokdong Kwon and Hwajeong Seo ***

Division of IT Convergence Engineering, Hansung University, Seoul 02876, Korea; amdjd0704@hansung.ac.kr (H.K.); pgmkkh@hansung.ac.kr (K.K.); hyeok@hansung.ac.kr (H.K.)

* Correspondence: hwajeong@hansung.ac.kr; Tel.:+82-2-760-8033

Received: 27 July 2020; Accepted: 8 August 2020; Published: 12 August 2020


**Abstract:** Application-Specific Integrated Circuit (ASIC)-resistant Proof-of-Work (PoW) is widely adopted in modern cryptocurrency. The operation of ASIC-resistant PoW on ASIC is designed to be inefficient due to its special features. In this paper, we firstly introduce a novel ASIC-resistant PoW for low-end microcontrollers. We utilized the measured power trace during the cryptographic function on certain input values. Afterward, the post-processing routine was performed on the power trace to remove the noise. The refined power trace is always constant information depending on input values. By performing the hash function with the power trace, the final output was obtained. This framework only works on microcontrollers and the power trace depends on certain input values, which is not predictable and computed by ASIC.

**Keywords:** ASIC-resistant; proof of work; power analysis; microcontroller; blockchain

## 1. Introduction

Modern cryptocurrency, such as Bitcoin, Ethereum, and Monero, uses a Proof-of-Work (PoW) consensus algorithm in order to ensure the integrity of blocks [1–3]. The work must be hard on the requester side, and should be easy for the service provider. Bitcoin utilizes the hash function to satisfy the special condition. The hash function can be efficiently computed on an Application-Specific Integrated Circuit (ASIC). Since the ASIC is designed for PoW, most applications are dominated by the ASIC machine, which threatens the decentralization of blockchain networks (i.e., Bitcoin). In order to overcome the domination of ASIC machine on the consensus, modern cryptocurrency uses multi-hash PoW, memory hard PoW, and programmatic PoW. Since the ASIC is the targeted chip, the irregular pattern can lead to inefficient computation on ASIC.

In this paper, we present a new technique to prevent a mining monopoly caused by special mining devices, such as ASIC and the field-programmable gate array (FPGA). The proposed methods allow anyone to work on performing a cryptographic module on a microcontroller. In the microcontroller, the power trace generated during the execution of the crypto-module shows different values depending on the input parameters. With this unique power trace, the PoW consensus is performed with ASIC-resistance feature.

*Research Contributions*

- We introduce a novel ASIC-resistant proof of work based on a power analysis: Previous ASIC-resistant PoW methods are based on multi-hash, memory hard, and programmatic approaches. The proposed method presents the first ASIC-resistant PoW based on a power analysis. The power trace during cryptography encryption (e.g., AES) on the microcontroller is utilized for the source of PoW. Since the power trace depends on the target microcontroller, cryptography encryption, and input values,

the proposed PoW cannot be emulated by ASIC and FPGA. For this reason, we achieved the ASIC-resistant feature.

- Post-processing for noise elimination: The raw power trace contains noise information. Small noise can significantly alter the result of cryptography operations. In order to filter out the noise from the raw power trace, a Fast Fourier Transform (FFT) is performed. This method efficiently removes the high frequency. After the post-processing, the refined information is used as a source for PoW.
- In-depth analysis of novel PoW based on various block ciphers: The performance was evaluated on a microcontroller. We performed the experiment with various block ciphers. The result shows that the novel PoW works on all block ciphers without difficulty.

The paper is organized as follows. In Section 2, the background of ASIC-resistance PoW algorithm and power analysis is described. In Section 3, the proposed technique is described. In Section 4 the evaluation of PoW method is given. Section 5 concludes the paper.

## 2. Related Works

### *2.1. Proof-of-Work*

Since 1992, many PoW-based applications have been developed to resolve many real-world problems. In [4], the PoW technique was utilized to block junk emails by requesting that the sender compute some function of the message. Similarly, the PoW technique was applied to an uncheatable benchmark, metering web-sites accesses, lottery schemes, and PoW-puzzles [5–9].

In 1993, the first PoW algorithm was suggested by Cynthia and Moni [4]. Afterward, the PoW method was applied to Bitcoin in 2009 [1]. In order to add a new block in Bitcoin, the value using SHA-256 must be smaller than the target difficulty. Bitcoin is one of the most popular and successful hash cash, which was conceived of in 1997 and detailed in 2002 [10].

The primitive function of hash cash is a hash function. A number of hash functions are employed by modern cryptocurrency to improve both performance and security. SHA-256 is designed in 2001 by the NSA and chosen by NIST as U.S FIPS. SHA-256 generates 256-bit wise output, which is used in Bitcoin and Bitcoin Cash. Ethash is the hash function for the Ethereum consensus protocol. The hash function is combination of Keccak, Hashimoto, and Dagger algorithms to ensure ASIC-resistance [11–13]. Scrypt is a password-based key-derivation function, which uses a large amount of memory than others [14]. The memory-hardness ensures the ASIC-resistant feature. The hash function is used in Litecoin. Random memory access based protocol, namely Cryptonote, is used in the CryptoNight PoW algorithm for Monero [15]. The details of ASIC-resistant PoW are covered in Section 2.2.

The hash cash-based PoW is efficient and easy to implement on any platforms. However, the scheme requires huge computational power to generate the valid block. An unintended consequence of PoW is the centralization of mining power. The large-scale PoW blockchain systems inhibits decentralization [16,17].

According to [18], Bitcoin miners consume 0.33% of the world's electricity. In order to save the wasted electricity, the PoW side of consensus for useful results through PoW was presented (i.e., Bread Pudding Protocols). One example is a solving scientific problems. In 2013, a PoW protocol for Cunningham chains of primes was proposed [19]. Furthermore, a PoW protocol was also designed for other problems such as DNA and RNA sequencing. Alternatively, MicroMint and data handling have been also actively investigated [20,21].

In the PoW protocol, the miner firstly solves the problem and then broadcasts the new block. When the block is valid, the miner receives a certain reward. With above principle, the protocol constructs huge distributed networks under trust. However, unexpected conditions can lead to trouble. In order to resolve this issue, there are certain criteria to select the fork block. First, the oldest fork block is selected. Second, the longest fork block is selected. Third, the chain with the greatest amount of computational power spent is selected [22].

As the PoW is widely used in practice, a number of attacks have been performed on it. The most well-known attacks on blockchain with PoW protocol is 51% attacks. This attack can be triggered when

the malicious user controls more than half of the network's computing power. This attack occurred in 2018 for BTC Gold and ZenCash [23]. Moreover, a number of attacks on the PoW protocol, such as selfish mining attacks, network-level attacks, pool related attacks, and goldfinger attacks have been investigated [24–27].

As we explored the blockchain technology with PoW protocol above, we found that it has many new features together with many open problems. In this paper, we focused on the ASIC-resistant PoW for the low-end Internet of Things.

### *2.2. ASIC-Resistant PoW*

In 1993, the first PoW algorithm was suggested by Cynthia and Moni [4]. Afterward, the PoW method was applied to Bitcoin in 2009 [1]. In order to add a new block in Bitcoin, the value using SHA-256 must be smaller than the target difficulty.

Recently, Application-Specific Integrated Circuits (ASICs) designed for PoW calculation dominate the crypto-mining world, threatening the decentralization of blockchain. In order to prevent the ASIC approach, the ASIC-resistant PoW algorithm was introduced.

#### 2.2.1. Multi-Hash PoW

Multi-hash PoW achieves ASIC immunity by using multiple hashing functions linked together in successive steps to resolve the dependency on a single hashing function, one of the weaknesses of the PoW algorithm. These include Quarkcoin (http://www.quarkcoins.com) and LYRA2RE (https://en.bitcoinwiki.org/wiki/Lyra2RE).However, the multiple hash PoW mechanism is not sufficient to prevent ASIC-based mining. The ASIC resistance is analyzed by implementing a multi-hash PoW in a field-programmable gate array (FPGA). As a result, the multi-hash PoW showed ASIC resistance at a level similar to the PoW mechanism that could not block the ASIC.

#### 2.2.2. Memory-Hard PoW

The memory-hard PoW is a PoW where the proofer uses a lot of memory capacity for the proof and the verifier uses a small amount of memory space and time to verify the work. With this bounding, it makes no sense to use ASICs to speed up computation. Ethereum's Ethash (https://github.com/ethereum/wiki/wiki/Ethash) and CryptNight [15] are memory-hard Pow.However, an ASIC system targeting such a memory-hard PoW algorithm was also introduced.

#### 2.2.3. Memory-Bound PoW

Memory-bound PoW [28–30] traverses a random path on a large table of random numbers, making multiple memory accesses. Memory-bound PoWs have the risk of having a memory access pattern in cryptographic hashing, and have the disadvantage of having to send a large random table between the attestor and the verifier over the network [31].

#### 2.2.4. Programmatic PoW

Programmatic PoW is a new way to increase the versatility of calculations by executing arbitrary code. A randomly generated program [32,33] can be part of the operation. It is impractical to build special hardware modules that target each computational task. Therefore, ASICs perform less than general purpose computers in random programs. RandomX utilized this method approach (https://github.com/tevador/RandomX).

### *2.3. Power Analysis and Its New Applications*

Side-channel analysis is an attack technique that identifies confidential information from side information [34]. The known power consumption and the dependencies of the codes executed can be used as techniques for determining the similarity of the codes [35]. In [36], IP protection through

similar power trace is suggested. According to previous works, the power trace can be utilized for the identification of target program. We were motivated from previous works and present a novel PoW method based on a power trace.

## 3. Proposed Method

Special mining devices, such as ASIC and FPGA, hinder the safety of the blockchain network, which decreases the participation rates and increases entry barriers. The proposed technique attempts to solve the problem by utilizing the power trace of microcontrollers during the cryptography function to ensure the ASIC-resistant PoW. The main idea is based on the PoW of Bitcoin and the source code classification by analyzing power traces [35,36].

The working flow of proposed method is described in Figure 1. First, the block header information is filled with the version, previous blockhash, merklehash, time, bits, and nonce. Afterward the hash function (e.g., SHA-256) is performed on the block header. The output is used for the key value of the encryption module (e.g., AES). For the text part, the nonce is utilized. With these input values, the encryption module is executed on the target microcontroller. By performing the encryption, the microcontroller generates certain power traces. These power traces may contain the noise. In order to remove the noise, the post-processing is performed on it. Then the refined power trace and block header is added together. The output of addition is used for the input value of hash function (e.g., X16R). Finally, the evaluator checks whether the output of hash function meets the target value. If it satisfies the target value, the valid block is generated. Otherwise, the nonce increases and the routine is performed again. Detailed descriptions of the proposed PoW method are given in Algorithm 1. In Step 2, the nonce value is updated. In Step 3 and 4, the output of SHA-256 on block header and nonce are used for key and text of AES-256 encryption, respectively. In Step 5 and 6, the power consumption is measured and post-processed. In Step 7, the block header and the power trace are added together. Finally, the information is used for the generation of target value. If the value is lower than the target, the block is generated, successfully. Otherwise, the computation is performed again after increasing the nonce value.

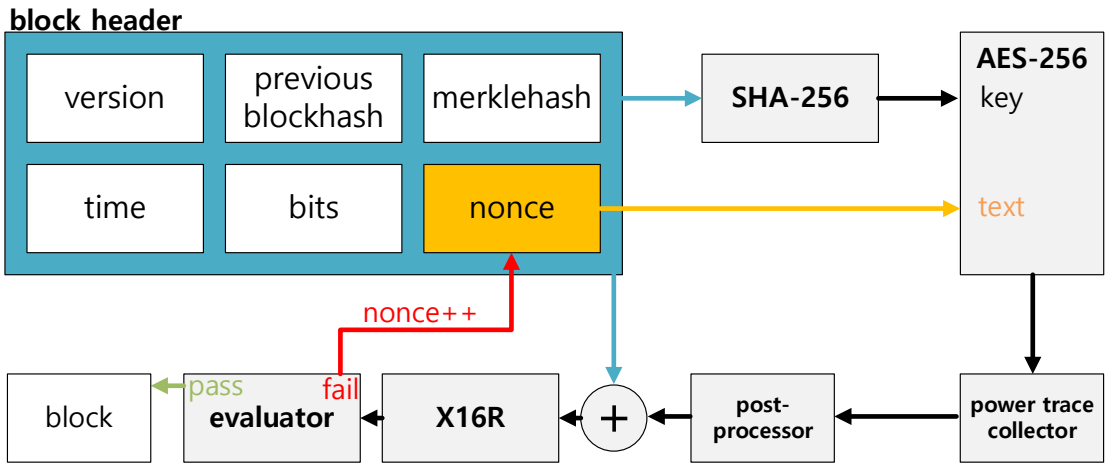


**Figure 1.** Working flow of proposed Proof-of-Work (PoW).

**Algorithm 1** Proposed PoW algorithm

**Input:** *Block*, Power Trace.
**Output:** *Block*.

**while** $H < Target$ **do**
  $Block.header.nonce \leftarrow Block.header.nonce + 1$
  $AES256.key \leftarrow SHA256(Block.header)$
  $AES256.text \leftarrow Block.header.nonce$
  $Block.header.trace \leftarrow PowerCapture(AES256)$
  $Block.header.trace \leftarrow PostProcessor(Block.header.trace)$
  $Block.header \leftarrow Block.header + Block.header.trace$
  $H \leftarrow X16R(Block.header)$
**end while**
**return** *Block*

### *3.1. Power Trace Based Proof-of-Work*

This section describes the power trace based Proof of Work. The benefit of each step is discussed in detail.

#### 3.1.1. Collection of Power Trace from Microcontrollers

In this step, AES-256 encryption is performed with the key and nonce. The output of power trace is stored in the block header. We ensure that the output of power trace contains the key and nonce information, since the power trace is varied depending on the input and algorithm. The power trace is post-processed and then used for the input value of the hash function (i.e., X16R). Since the power trace information is unique, the hash function cannot be performed without previous steps. This is the main purpose of forcing the use of the microcontroller to generate the power trace through above steps.

#### 3.1.2. Features of Power Consumption Trace

The power trace produced by operating the microcontroller exhibits different characteristics depending on the program being operated with certain input values. This feature indicates that the output of each trace is different for each active program, even if the same program is operated. Different output power traces are generated according to the input values of the active program. This unique characteristic of the power consumption is utilized to prove the work. It prevents an attacker from entering an arbitrary trace without rapidly changing or executing code to perform the high-speed mining.

#### 3.1.3. Post-Processing for Noise Filtering

The trace of power consumption from the microcontroller may contain noise. To prevent false positives due to noise, Fast Fourier Transform (FFT) is applied to noise filtering from the power trace. FFT is applied to the acquired power trace, which removes the high frequency, and then converted back to signal at frequency. Power traces with noise and without noise are displayed at the top and bottom of Figure 2, respectively. The power trace with noise shows clearer information than the power trace without.

#### 3.1.4. High Entropy for Input Value

The proposed PoW generates the block depending on valid input value. With the input value, the AES-256 encryption is performed on the microcontroller. The AES-256 cipher uses 32-byte keys and 16-byte plain text, respectively. The key uses the hash (i.e., SHA-256) value of the block header. The plaintext is a 16-byte nonce value. The complexity of key and plain text is $2^{384}$ since key and

plaintext are 256-bit and 128-bit, respectively. For this reason, it is impossible to perform the brute-force attack and make a table through pre-computation in advance. Since the block header contains a nonce, the key changes whenever the nonce value changes. The key has randomness due to hash function. This randomness does not allow the attacker to operate the microcontroller in advance to generate a power consumption trace. Since the power consumption trace is output differently depending on the function and input value operated by the microcontroller, it is difficult to predict the power trace value.

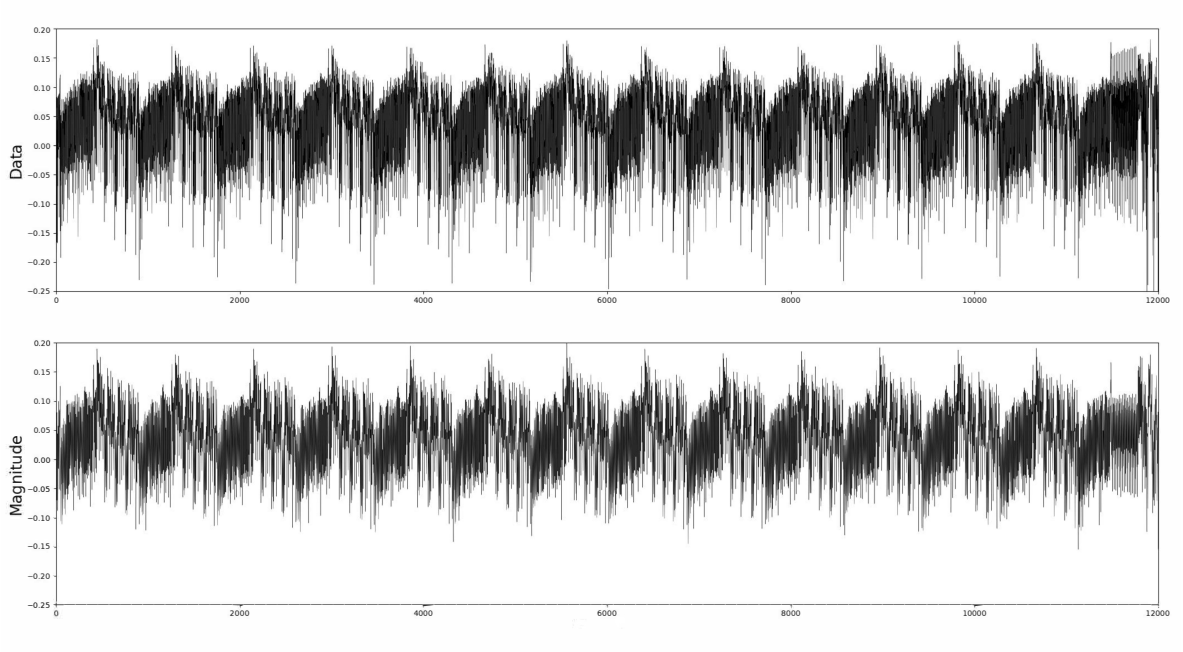


**Figure 2.** Power consumption traces (**top**) with noise and (**bottom**) without noise.

#### 3.1.5. Finding the Target with Hash Function

In PoW of Bitcoin, the hash value of a block is generated by combining the creation time, version, bit, root hash, hash of the previous block, and a temporary value called nonce. If the newly created hash value is smaller than a certain target value, a new block is created and connected to the existing blockchain network. In the proposed method, the block generation time, version, bit, root hash, previous block hash, nonce, and power consumption trace are combined together to generate the block hash value. This step is used to ensure the unique power trace is unpredictable.

#### 3.1.6. AI-Resistant Hash Function

The hash uses X16R. The X16R is a newly designed algorithm to avoid AI-style mining in Ravencoin, consisting of 16 hashing algorithms that work in chain fashion according to the last 8 bytes of the hash of the previous block. ASIC development is difficult due to the random nature of this algorithm. For this reason, the proposed method achieved the highest level of ASIC-resistance.

#### 3.1.7. Verification with Power Trace

Blocks created in the blockchain are propagated to other nodes and verified whether the hash values match to certain level. In the proposed method, the power consumption trace is checked in addition to the existing method to verify that the creator of the block worked on the microcontroller with the correct code and input values.

**Trace verification.** Even if the same program code and input value are used in the microcontroller, the power trace does not appear due to the noise generated during collection. Therefore, simply comparing the values of the power consumption trace cannot confirm whether the power consumption trace uses the same code and input value. For this reason, filtering with FFT is performed on the power trace to remove

the noise. Afterward, each of the same input values are characterized by a high correlation coefficient compared to other input values. In Figure 3, the distribution of the correlation coefficient is narrow for the same input value when FFT is applied to the power trace. This experiment confirmed that filtering with FFT function is working, properly. Differences in correlation coefficients are confirmed during experiments. Detailed descriptions of proposed Trace verification method are given in Algorithm 2. In Step 1, the power trace is captured during the AES256 encryption. In Step 2, the power trace is post-processed. In Step 3, the correlation coefficient between refined power trace and valid power trace is calculated. In Step 4 $\sim$ 8, the correlation is compared with certain threshold. If the value is over the threshold, the verification is confirmed.

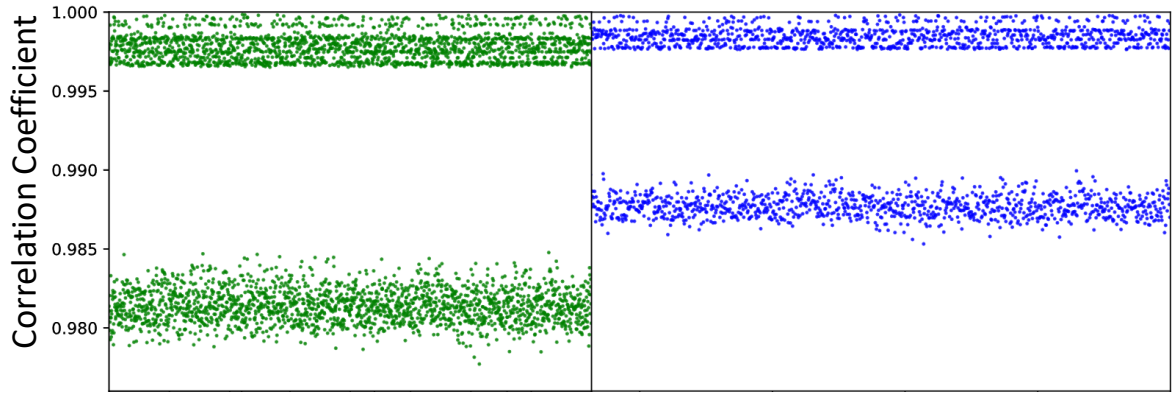


**Figure 3.** Comparison of correlation coefficient for power traces (**left**) without post-processing and (**right**) with post-processing.

**Algorithm 2** Power Consumption Trace Verification algorithm

**Input:** $key$, $nonce$, $validate\_trace$.
**Output:** Verification result.

$trace \leftarrow Power\_capture(AES256(key, nonce))$
$refined\_trace \leftarrow Post\_processing(trace)$
$col \leftarrow Correlation\_coefficient(refined\_trace, valid\_trace)$
**if** $col > Threshold$ **then**
  **return** *True*
**else**
  **return** *False*
**end if**

**Power trace based verification process.** First, the cryptographic program works on the microcontroller using the key and nonce values stored in block values. During the execution, the power consumption trace is generated and captured. The power consumption trace is then filtered using FFT function. Afterward, the correlation coefficient between filtered trace and the valid trace is calculated. If the value exceeds the threshold, the verification is confirmed. This is novel approach to utilize the power trace for verification.

### *3.2. Advantages of the Proposed Method*

This section shows that the above-mentioned proposed technique is faithfully satisfied with the requirements of the PoW technique. In particular, it describes how to solve the problem, unlike the previous known methods.

#### 3.2.1. PoW Requirements

PoW is designed to prevent blockchain network attacks like Sybil Attack. There are several requirements to achieve the safe PoW. The PoW is difficult to prove. However, the verification should be an easy problem for provers. Bitcoin uses a hash to satisfy these requirements. However, Bitcoin is vulnerable to ASIC attack. The proposed method satisfies asymmetry by using the same hash function. PoW should be impossible to optimize depending on certain conditions. If someone finds an optimization method, they will take advantage of it. This creates the centralization problem of the blockchain network. Bitcoin is a cryptocurrency that is difficult for ordinary miners to participate in, as it is a miner that can perform fast calculations such as ASIC using SHA256. In the proposed method, X16R hash function is used to resist the ASIC.

#### 3.2.2. ASIC Resistance

Previous methods can be optimized with ASICs. In the proposed method, the step of using the designated code in the microcontroller must be performed by adding the verification step that the microcontroller performed the same code and input value. Therefore, ASIC is meaningless in our scenario because optimization with ASIC is impossible. Since the output power consumption trace information is used as the input value of the hash calculation (X16R), the hash calculation using parallel processing is impossible. Due to non-optimization and parallelism constraints, the proposed method achieved the ASIC resistance.

#### 3.2.3. Operation Time

The proposed proof-of-work execution time is the microcontroller multiplied by the number of iterations times the sum of the cryptographic module execution time and the hash operation time. Since the number of repetitions cannot be adjusted, the execution time of the cryptographic module in the microcontroller is fixed. Since the parallel operation is impossible, the only way to reduce execution time is to compute the hash algorithm quickly on the microcontroller. However, the computation speed cannot be enhanced, because the target microcontroller has certain resource-constrained features. Therefore, it is expected that miners will use a general-purpose microcontroller capable of hashing at an appropriate speed because there is no reason to purchase special mining equipment at a high cost.

#### 3.2.4. Flexibility of Encryption Module on Microcontrollers

The encryption module operated by the microcontroller is changeable. The reason for using the AES-256 encryption module is that it is international standard. Furthermore, if the input value is the same, the intermediate value and the output value are the same. With this feature the power consumption trace can be verified. The proposed method uses AES-256 in the microcontroller, but other block ciphers can be used for this function if it has good cryptography features and efficiency.

## 4. Evaluation

In this section, we evaluate the proposed method on the microcontroller. We firstly introduce the experiment environment and results in detail.

### *4.1. Experiment Environment*

The proposed method relies on the power trace information. For this reason, a high-quality of power trace information is important. We collected the power consumption of the microcontroller through ChipWhispererLite XMEGA (8-bit processor). The sampling rate was 7.38 million samples per second (MS/s) and this was the working frequency of target processor. The source code of the microcontroller was written in the C language and compiled with the AVR-GCC. For the power trace collection, we used the ChipWhispererLite API version 5.1. The experiment code was written in Python.

### *4.2. Uniqueness of Power Consumption*

We ran the AES-256 encryption module on the microcontroller to collect the power consumption traces generated from the same input value, and collected two 2000 traces of 1000 in order to confirm that the power consumption was unique according to the input variable when the encryption module was executed on the microcontroller. The correlation coefficient between the two groups was obtained. Cases include different plaintext, different key, and different both values. Two-thousand traces of power consumption were collected. All the collected trace were filtered using the FFT function. Afterward, two groups of 1000 were created and correlation coefficients between the two groups were obtained. Table 1 shows the statistical values of the results. Both the key and input showed a relatively higher correlation coefficient than the other case. In particular, the minimum value when the key and input are the same is larger than the maximum value when only the key is different. This is clearly seen in Figure 4. There is no common value for the same input value and different input values. Between the minimum value in the same case and the maximum value in the other case, there is large gap. For this reason, we can classify this, correctly.

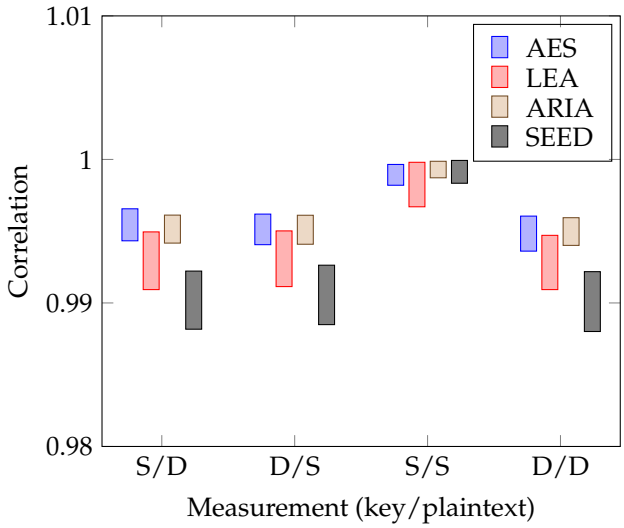


**Figure 4.** Correlation coefficient between power consumption traces, S/D, D/S, S/S, and D/D represent same key/different plaintext, different key/same plaintext, same key/same plaintext, and different key/different plaintext, respectively.

The same experiment was performed for LEA, ARIA, and SEED [37–39] to confirm that other ciphers with the same intermediate value and output value also have these characteristics.

LEA showed a wider range of correlation coefficients than the results of AES. In addition, AES is less than the minimum value of 0.0005 for the same input value. There is no common value for the same input value and different input values. Therefore, it was confirmed that LEA can also be used as a working cryptographic module.

ARIA showed a higher correlation coefficient than AES for other input values. In addition, there was a difference between the minimum value for the same input value and the maximum value for other inputs, such as 0.0017. The ARIA does not have a common value for the same input value and different input values. Therefore, we confirmed that the ARIA can also be used as a working cryptographic module.

SEED is similar to the result of AES. This confirmed that the block cipher can be used as a cryptographic module that works. There is no common value for the same input value and different input values.

**Table 1.** Correlation coefficient between power consumption traces with different block ciphers.

| Measurement (Key/Plaintex) | Maximum | Minimum | Mean | First Quartile | Third Quartile |
|---|---|---|---|---|---|
| AES | | | | | |
| Same key/Different plaintext | 0.99656 | 0.99433 | 0.99549 | 0.99519 | 0.99578 |
| Different key/Same plaintext | 0.99619 | 0.99406 | 0.99516 | 0.99488 | 0.99542 |
| Same key/Same plaintext | 0.99965 | 0.99820 | 0.99884 | 0.99859 | 0.99903 |
| Different key/Different plaintext | 0.99605 | 0.99361 | 0.99493 | 0.99461 | 0.99524 |
| LEA | | | | | |
| Same key/Different plaintext | 0.99495 | 0.99093 | 0.99288 | 0.99236 | 0.99337 |
| Different key/Same plaintext | 0.99502 | 0.99114 | 0.99283 | 0.99231 | 0.99331 |
| Same key/Same plaintext | 0.99980 | 0.99670 | 0.99809 | 0.99756 | 0.99854 |
| Different key/Different plaintext | 0.99471 | 0.99093 | 0.99271 | 0.99221 | 0.99317 |
| ARIA | | | | | |
| Same key/Different plaintext | 0.99612 | 0.99417 | 0.99505 | 0.99483 | 0.99526 |
| Different key/Same plaintext | 0.99611 | 0.99409 | 0.99498 | 0.99476 | 0.99518 |
| Same key/Same plaintext | 0.99987 | 0.99872 | 0.99919 | 0.99900 | 0.99937 |
| Different key/Different plaintext | 0.99594 | 0.99401 | 0.99492 | 0.99469 | 0.99514 |
| SEED | | | | | |
| Same key/Different plaintext | 0.99222 | 0.98817 | 0.99034 | 0.98990 | 0.99080 |
| Different key/Same plaintext | 0.99263 | 0.98849 | 0.99050 | 0.99005 | 0.99097 |
| Same key/Same plaintext | 0.99993 | 0.99834 | 0.99901 | 0.99875 | 0.99927 |
| Different key/Different plaintext | 0.99218 | 0.98801 | 0.99023 | 0.98976 | 0.99068 |

### *4.3. Lightweight PoW for Low-End Microcontrollers*

In Table 2, the comparison result is given. Previous methods suggested the hard problem not to be solved by ASIC. However, the problem is even hard for the users. On the other hand, the proposed method is efficient even for the low-end IoT devices, because the hard problem comes from varied power trace features related with microcontrollers, cryptographic modules, and input values. This is easy for microcontrollers but it is hard to emulate. We ensure that this is the first practical PoW method for low-end IoT devices.

**Table 2.** Comparison of the PoW method.

| Method | Overhead | Low-End IoT |
|---|---|---|
| Multi-hash PoW | HIGH | - |
| Memory-hard PoW [15] | HIGH | - |
| Memory-bound PoW [28–30] | HIGH | - |
| Programmatic PoW [32,33] | HIGH | - |
| This work (Power-trace PoW) | LOW | √ |

## 5. Conclusions

In this paper, we presented the novel ASIC-resistant PoW based on the power analysis of low-end microcontrollers. This approach is based on the unique power consumption pattern performing the cryptography function on the microcontroller. The power trace is post-processed with FFT to meet the certain quality of PoW source. Finally, the refined information is used for the block generation. In order to confirm the practicality of proposed method, we implemented the method on the low-end microcontroller and four different block ciphers were evaluated. The result shows that the proposed method works efficiently on target microcontrollers.

As a future work, we want to apply the proposed method to other platforms ranging from high-end processors to medium-end processors. In terms of attack scenarios, we will investigate the ASIC-based emulation for power trace. Furthermore, we will also explore deep-learning-based emulation to break through the proposed method.

**Author Contributions:** Software, H.K. (Hyunjun Kim) and K.K.; validation, H.K. (Hyeokdong Kwon); investigation, K.K. and H.K. (Hyeokdong Kwon); writing–original draft preparation, H.K. (Hyunjun Kim) and K.K.; writing–review and editing, H.S. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was partly supported as part of the Military Crypto Research Center (UD170109ED) funded by the Defense Acquisition Program Administration (DAPA) and Agency for Defense Development (ADD) and this work was supported by the Institute for Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (< Q | Crypton >, No.2019-0-00033, Study on Quantum Security Evaluation of Cryptography based on Computational Quantum Complexity) and this work was partly supported by Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No.2018-0-00264, Research on Blockchain Security Technology for IoT Services).

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 8 August 2020).
2. Wood, G. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Proj. Yellow Pap.* **2014**, *151*, 1–32.
3. Noether, S.; Mackenzie, A. Ring Confidential Transactions. Available online: http://ledger.pitt.edu/ojs/ledger/article/view/34 (accessed on 8 August 2020).
4. Dwork, C.; Naor, M. Pricing via processing or combatting junk mail. In *Annual International Cryptology Conference*; Springer: Berlin/Heidelberg, Germany, 1992; pp. 139–147.
5. Cai, J.Y.; Lipton, R.J.; Sedgewick, R.; Yao, A.C. Towards uncheatable benchmarks. In Proceedings of the Eigth Annual Structure in Complexity Theory Conference, San Diego, CA, USA, 18–21 May 1993; pp. 2–11.
6. Ar, S.; Cai, J.Y. Reliable Benchmarks Using Numerical Instability. 1994. pp. 34–43. Available online: https://books.google.com.hk/books?hl=zh-TW&lr=&id=SYEy8nTSkMYC&oi=fnd&pg=PA34&dq=Reliable+Benchmarks+Using+Numerical+Instability&ots=5Ujh83NE-k&sig=DRCARrjKZm7LW9KkLtV3PCDRQmY&redir_esc=y&hl=zh-CN&sourceid=cndr#v=onepage&q=ReliableBenchmarksUsingNumericaInstability&f=false (accessed on 8 August 2020).
7. Franklin, M.K.; Malkhi, D. Auditable metering with lightweight security. In *International Conference on Financial Cryptography*; Springer: Berlin/Heidelberg, Germany, 1997; pp. 151–160.
8. Goldschlag, D.M.; Stubblebine, S.G. Publicly verifiable lotteries: Applications of delaying functions. In *International Conference on Financial Cryptography*; Springer: Berlin/Heidelberg, Germany, 1998; pp. 214–226.
9. Rivest, R.L.; Shamir, A.; Wagner, D.A. *Time-Lock Puzzles and Timed-Release Crypto*; Massachusetts Institute of Technology, Laboratory for Computer Science: Cambridge, MA, USA, 1996. Available online: http://bitsavers.trailing-edge.com/pdf/mit/lcs/tr/MIT-LCS-TR-684.pdf (accessed on 8 August 2020).
10. Back, A. Hashcash-A Denial of Service Counter-Measure. Technical Report. 2002. Available online: ftp://sunsite.icm.edu.pl/site/replay.old/programs/hashcash/hashcash.pdf (accessed on 8 August 2020).
11. Bertoni, G.; Daemen, J.; Peeters, M.; Van Assche, G. Keccak. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 313–314.

12. Dryja, T. Hashimoto: I/O Bound Proof of Work. Technical Report. 2009. Available online: https://mirrorx.com/files/hashimoto.pdf (accessed on 8 August 2020).
13. Buterin, V. Dagger: A Memory-Hard to Compute, Memory-Easy to Verify Scrypt Alternative. Technical Report. 2013. Available online: http://www.hashcash.org/papers/dagger.html (accessed on 8 August 2020).
14. Percival, C. Stronger Key Derivation via Sequential Memory-Hard Functions. Technical Report. 2009. Available online: https://pdfs.semanticscholar.org/7c74/956d21f0466c9771bb583e2fdf854c2aedbf.pdf (accessed on 8 August 2020).
15. Van Saberhagen, N. CryptoNote v 2.0. Technical Report. 2013. Available online: https://decred.org/research/saberhagen2013.pdf (accessed on 8 August 2020).
16. Gencer, A.E.; Basu, S.; Eyal, I.; Van Renesse, R.; Sirer, E.G. Decentralization in bitcoin and ethereum networks. In *International Conference on Financial Cryptography and Data Security*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 439–457.
17. Ruoti, S.; Kaiser, B.; Yerukhimovich, A.; Clark, J.; Cunningham, R. SoK: Blockchain technology and its potential use cases. *arXiv* **2019**, arXiv:1909.12454.
18. De Vries, A. Bitcoin's growing energy problem. *Joule* **2018**, *2*, 801–805. [CrossRef]
19. Primecoin, K.S. Cryptocurrency with Prime Number Proof-of-Work. Technical Report. 2013. Available online: https://primecoin.io/bin/primecoin-paper.pdf (accessed on 8 August 2020).
20. Rivest, R.L.; Shamir, A. PayWord and MicroMint: Two simple micropayment schemes. In *International Workshop on Security Protocols*; Springer: Berlin/Heidelberg, Germany, 1996; pp. 69–87.
21. Miller, A.; Juels, A.; Shi, E.; Parno, B.; Katz, J. Permacoin: Repurposing bitcoin work for data preservation. In Proceedings of the 2014 IEEE Symposium on Security and Privacy, Berkeley, CA, USA, 18–21 May 2014; pp. 475–490.
22. Sompolinsky, Y.; Zohar, A. Secure high-rate transaction processing in bitcoin. In *International Conference on Financial Cryptography and Data Security*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 507–527.
23. Jang, J.; Lee, H.N. Profitable double-spending attacks. *arXiv* **2019**, arXiv:1903.01711.
24. Eyal, I.; Sirer, E.G. Majority is not enough: Bitcoin mining is vulnerable. In *International Conference on Financial Cryptography and Data Security*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 436–454.
25. Heilman, E.; Kendler, A.; Zohar, A.; Goldberg, S. Eclipse attacks on bitcoin's peer-to-peer network. In Proceedings of the 24th USENIX Security Symposium ( USENIX Security 15), Washington, DC, USA, 12–14 August 2015, pp. 129–144.
26. Daian, P.; Eyal, I.; Juels, A.; Sirer, E.G. (Short paper) Piecework: Generalized outsourcing control for proofs of work. In *International Conference on Financial Cryptography and Data Security*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 182–190.
27. Kroll, J.A.; Davey, I.C.; Felten, E.W. The economics of Bitcoin mining, or Bitcoin in the presence of adversaries. In Prcoceedings of the WEIS, Washington, DC, USA, 11–12 June 2013; Volume 2013, p. 11.
28. Abadi, M.; Burrows, M.; Manasse, M.; Wobber, T. Moderately hard, memory-bound functions. *Acm Trans. Internet Technol. (Toit)* **2005**, *5*, 299–327. [CrossRef]
29. Dwork, C.; Goldberg, A.; Naor, M. On memory-bound functions for fighting spam. In *Annual International Cryptology Conference*; Springer: Berlin/Heidelberg, Germany, 2003; pp. 426–444.
30. Dwork, C.; Naor, M.; Wee, H. Pebbling and proofs of work. In *Annual International Cryptology Conference*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 37–54.
31. Ren, L.; Devadas, S. Bandwidth hard functions for ASIC resistance. In *Theory of Cryptography Conference*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 466–492.
32. Bradley, W.F. Superconcentration on a Pair of Butterflies. *arXiv* **2014**, arXiv:1401.7263.
33. Cook, S.A. An observation on time-storage trade off. In *Proceedings of the Fifth Annual ACM Symposium on Theory of Computing*; Association for Computing Machinery: New York, NY, USA, 1973; pp. 29–33.[CrossRef]
34. Kocher, P.; Jaffe, J.; Jun, B. Differential power analysis. In *Annual International Cryptology Conference*; Springer: Berlin/Heidelberg, Germany, 1999; pp. 388–397.
35. Durvaux, F.; Gerard, B.; Kerckhof, S. Intellectual Property Protection for Integrated Systems Using Soft Physical Hash Functions. In *International Workshop on Information Security Applications*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 208–225.

36. Samarin, P.; Lemke-Rust, K. Detecting similar code segments through side channel leakage in microcontrollers. In *International Conference on Information Security and Cryptology*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 155–174.
37. Kwon, D.; Kim, J.; Park, S.; Sung, S.H.; Sohn, Y.; Song, J.H.; Yeom, Y.; Yoon, E.J.; Lee, S.; Lee, J.; et al. New block cipher: ARIA. In *International Conference on Information Security and Cryptology*; Springer: Berlin/Heidelberg, Germany, 2003; pp. 432–445.
38. Hong, D.; Lee, J.K.; Kim, D.C.; Kwon, D.; Ryu, K.H.; Lee, D.G. LEA: A 128-bit block cipher for fast encryption on common processors. In *International Workshop on Information Security Applications*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 3–27.
39. Park, J.; Lee, S.; Kim, J.; Lee, J. *The SEED Encryption Algorithm*; KISA: Seoul, Korea, 2005. Available online: http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=53A7C173E093671AB34C78C08C2943DB?doi=10.1.1.374.1600&rep=rep1&type=pdf (accessed on 8 August 2020).