IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 2, 2018, accepted October 28, 2018, date of publication October 31, 2018, date of current version November 30, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2878895

ASIC-Resistance of Multi-Hash Proof-of-Work Mechanisms for Blockchain Consensus Protocols

HYUNGMIN CHO
Department of Computer Engineering, Hongik University, Seoul 04066, South Korea
e-mail: hcho@hongik.ac.kr

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government—Ministry of Science, ICT & Future Planning (MSIP)—under Grant 2017017414.

ABSTRACT Blockchain technology rapidly gained popularity based on its open and decentralized operation. Consensus protocol is the core mechanism of a blockchain network that securely maintains the distributed ledger from possible attacks from adversaries. Proof-of-work (PoW) is a commonly used consensus protocol that requires a significant amount of computation to find a new valid block. As the application-specific integrated circuits (ASICs) that are specially designed for PoW computation begin to dominate blockchain consensus operation, the decentralized nature of blockchain networks is being threatened. Many PoW mechanisms are being proposed to disincentivize the use of ASICs in the consensus operation. Employing multiple hash functions in the PoW computation (i.e., multi-hash PoW) is one of the commonly adopted approaches to achieve such ASIC-resistance. In this paper, we experimentally evaluate the level of ASIC-resistance of the multi-hash PoW mechanisms. We assess the level of ASIC-resistance based on the performance gap between ASICs and general-purpose computing platforms. Contrary to the expectation of the multi-hash PoW mechanisms, our results reveal that ASIC-resistance of these PoW mechanisms is not strong enough to prevent ASIC-based mining. Most of them show similar levels of ASIC-resistance as those of PoW mechanisms that are already defeated by ASIC-based systems.

INDEX TERMS ASIC, blockchain, consensus, FPGA, hash, proof-of-work.

I. INTRODUCTION

Blockchain provides a mechanism to maintain an immutable public record of transactions (*ledger*) without central authorities [1]–[3]. Blockchain consists of a series of records, called blocks, that are connected using cryptography. Each block contains transactions that are based on digital currency circulated in the blockchain network, called *cryptocurrency*. The first incarnation of a blockchain network is Bitcoin [4], which was proposed in 2008 and made public in 2009. Since then, many blockchain networks have been created and draw tremendous attention. As of September 2018, about 2,000 cryptocurrencies are being traded in the market, and the total market size is more than 200 billion US dollars.[1] Beyond the basic blockchain protocol of Bitcoin, new generations of protocols brought new features and improved the performance and practicality of blockchain networks. For example, Ethereum [5] supports the execution of a program recorded in the blockchain. This feature enables an automated execution of transactions, called *smart contracts*. Blockchain technology is expected to bring revolutionary changes in business operations, and many companies began to prepare for the blockchain era [6]–[8].

The key to the decentralized operation between independent nodes in a peer-to-peer network is blockchain's *consensus* mechanism. Only a new block that is accepted by the consensus protocol can be appended to the blockchain. A consensus mechanism should be able to draw a mutual agreement between nodes, even in the presence of selfish or malicious nodes.

Proof-of-Work (PoW) is one of the most widely-adopted consensus mechanisms. PoW requires a significant amount of computation effort to find a block that meets the consensus protocol. Due to its resemblance to digging for gold in a mine, this process is called *mining*. When a new valid block is found and appended to the blockchain, the miner who found the block can collect the predefined rewards or transaction fees.

Any participant can join the mining process using its own computing resource. If an adversary wants to disrupt the consensus operation by breaking the PoW protocol, he needs to possess computing power larger than the aggregated

[1] https://coinmarketcap.com/

computing power of other mining participants (commonly known as the 51% attack). The rationale behind the PoW consensus protocol is that there is no economic incentive to perform such an attack on the blockchain network. However, such a belief is challenged by the introduction of mining systems based on application-specific integrated circuits (ASICs).

ASICs that are designed for the PoW computations deliver much higher performance than general-purpose computing systems, such as CPUs (central processing units) or GPUs (graphics processing units) [9]. The biased computing power threats the decentralized nature of blockchain networks [10]. The aggregated hashrate of the Bitcoin network has dramatically inflated (Fig. 1) by ASIC-based mining, and participating in Bitcoin mining using a general-purpose computing system is not profitable. ASIC-based mining created a high barrier to entry for the general public because they have to invest in special equipment to participate in the mining process. The entities who are capable of investing in and maintaining a large volume of ASIC systems could take control of the blockchain network. Therefore, ASIC-based mining is more vulnerable to the 51% attack. If all PoW computation has to be done using a general-purpose computing platform, it is more expensive to occupy the majority of the computing platforms that can join the mining process. On the other hand, if mining is done using specially-manufactured systems only, an attacker only needs to occupy the majority of the specialized systems, which is a much smaller domain than the group of all general-purpose computing systems. As of September 2018, Bitcoin mining pools BTC.com and Antpool, which are operated by the same company that manufactures ASIC miners, account for more than 30% of the total computing power (hashrate) in the Bitcoin network [11]. ASIC-based mining systems that are monopolized by a specific manufacturer have another vulnerability. Bitmain shipped their ASIC mining hardware with a backdoor program installed, which allows the company to remotely control a large portion of the hashing power in the network [12].

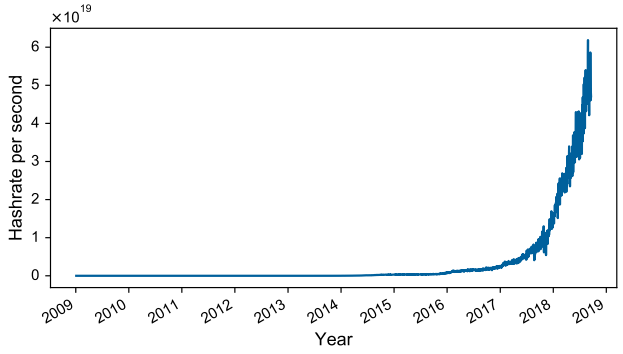

FIGURE 1. Aggregated hashrate of the bitcoin network.

In order to discourage the use of ASIC-based systems for mining, many PoW consensus protocols were introduced to achieve ASIC-resistance. In principle, there is no fundamental difference between a general-purpose computing platform and an ASIC. A part of the general-purpose system might include a dedicated module to accelerate a commonly-used function (e.g., Secure hash algorithms (SHA) modules in modern CPUs), and an ASIC may also use a programmable design (e.g., a microcontroller module that runs firmware). Because it is impossible to completely prevent the use of specially-designed hardware for mining, PoW protocols target ASIC-resistance, not ASIC-proof. The goal of an ASIC-resistant PoW algorithm is reducing the profitability of ASIC-based mining. Therefore, it is crucial to quantitatively study the degree of ASIC-resistance before adopting a PoW mechanism as the consensus protocol of a blockchain network. In fact, many PoW mechanisms that claimed ASIC-resistance turns out to have very little resistance to ASIC-based mining, and some of the blockchain networks that rely on such mechanisms are already dominated by ASIC-based mining.

In this work, we evaluate ASIC-resistance of a class of consensus protocol that use multiple hash functions in their PoW computation (multi-hash PoW). One way to achieve ASIC-resistance is making the initial cost for building an ASIC higher than expected profit. However, this property is difficult to maintain in a long-term especially if the blockchain network expands and the obtainable profit from mining grows. We focus on ASIC-resistance achieved by the performance gap between ASICs and general-purpose computing platforms. A PoW algorithm is resistant to ASICs if it is challenging to design a specialized hardware design that has a significant performance advantage over CPUs and GPUs. This performance gap would determine the ASIC-resistance over an extended period. We implement a multi-hash PoW computing platform on Field-Programmable Gate Array (FPGA) to estimate the expected performance of the PoW mechanisms on ASICs. Unfortunately, the evaluation in this work reveals that multi-hash PoW algorithms have very little ASIC-resistance in terms of the performance gap.

The rest of this paper is organized as follows: Section II introduces blockchain consensus protocols that are widely used in many blockchain networks. Section III explains the multi-hash PoW consensus mechanisms that are evaluated in this work. Section IV describes the setup for the experiment to evaluate ASIC-resistance. Section V shows the ASIC-resistance evaluation results and discusses the results. Finally, Section VI concludes of the paper.

II. BLOCKCHAIN CONSENSUS MECHANISMS

In a public blockchain network, no single entity solely holds the authority to verify the validity of transactions. An adversary may forge a chain of blocks with false transactions and propagate the false blockchain to the network. Blockchain networks employ consensus mechanisms that discourage the attempts to forge a false blockchain.

A. PROOF-OF-WORK CONSENSUS

Although there are differences in PoW mechanisms adopted by blockchain networks, we briefly explain Bitcoin's

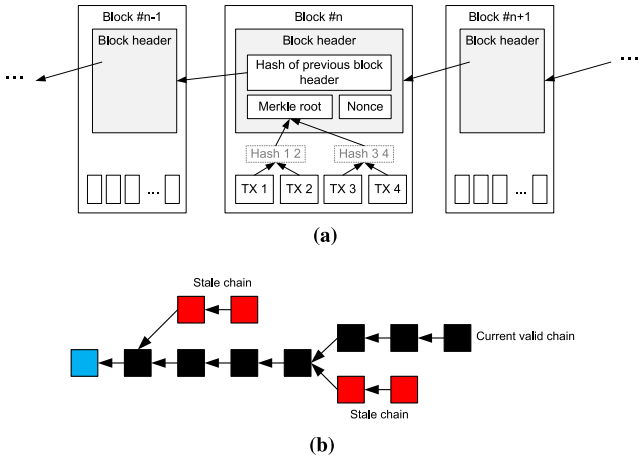

FIGURE 2. Illustration of Bitcoin's blockchain structure. (a) Block header. (b) Longest branch selection.

blockchain structure and PoW mechanism to discuss the basics of the PoW consensus protocol. Each block in Bitcoin's blockchain consists of the *block header* and transactions (TXs) included in the block (Fig. 2). A block header contains the hash value of the previous block header, the Merkle root of the transactions, and a nonce field. The previous hash value in the header makes sure the chain is not mutable after a block is included in the blockchain, and the Merkle root summarizes the transactions in the current block. The nonce field contains a random value that alters the hash value of the block header without the need for changing other fields.

Bitcoin consensus protocol requires the hash value of the block header to be smaller than a certain threshold, determined by the *difficulty* value. The difficulty value is dynamically adjusted by the network depending on the network's aggregated hashrate. Bitcoin uses double SHA-256 (or SHA256d) as its hash function, which applies the SHA-2 hash function twice (i.e., $\text{SHA256d}(x) = \text{SHA256}(\text{SHA256}(x))$) on the block header. The mining process is essentially finding a nonce value that makes the hash value to meet the condition.[2] Because the hash function is not invertible, the only feasible solution to find a valid block is a brute-force search that computes many instances of PoW computations (i.e., hash function computations) with different nonce values. Since the output of the hash function is uniformly distributed, the probability of finding a block with a nonce value that satisfies the condition is proportional to the devoted computing power.

Each node in the blockchain network selects the longest blockchain branch as the valid chain (Fig. 2b). Under this rule, if an attacker wants to forge a false blockchain, the attacker should present a blockchain to the network that is longer than the genuine one. Forging such a false blockchain would require a larger amount of PoW computations than the total sum of the computing power of the miners who are currently participating in the mining process.

The security of the PoW consensus protocol lies in the belief that there is no financial incentive to occupy the majority of the computing hardware and spend electrical energy to attack the blockchain. However, ASIC designs for blockchain mining introduce a great threat to this belief. The PoW computation requires massive computing power, but the computation is limited to specific types of calculations, usually hash functions. When compared to a general-purpose computing system, such as a CPU or a GPU, an ASIC-based system can deliver unparalleled performance and energy efficiency for the designated computation task. For example, an ASIC-based mining platform for Bitcoin [13] can perform 13 trillion PoW computations per second while consuming 1310 W of power, whereas a GPU-based mining platform with NVIDIA GeForce GTX 1080 ti achieves only 1.5 billion PoW somputations per second with 250 W of power consumption. Manufacturing an ASIC requires a considerable investment, but the majority portion of the manufacturing cost is non-recurring engineering (NRE) cost that gets amortized if the production volume increases. As the Bitcoin network grows, more and more ASIC-based mining systems are introduced, and the economic benefit of participating in the mining process is not sustainable unless the miner uses ASIC-based mining systems.

Many ASIC-resistant PoW mechanisms are introduced to avoid the same consequences of the Bitcoin mining process. The common types of ASIC-resistant PoW algorithms can be classified as follows.

1) **Multi-hash PoW:** Unlike the Bitcoin's PoW algorithm that uses only one type of hash function, PoW algorithms in this class employ multiple hash functions to compute the valid condition of the block. Those hash functions are applied to the block's header in a particular sequence. The sequence of hash functions is either fixed or dynamically determined per each block. Many variations of multi-hash PoW algorithms are created, such as X11 [14], X14 [15], X17 [16], X11EVO [17], X16S [18], X16R [19], Quark [20], and TimeTravel [21].
2) **Memory-hard PoW:** Although ASICs provide much higher computation efficiency than general-purpose computing platforms, ASICs are also limited by external data bandwidth similar to general-purpose computing platforms. Memory-hard PoW algorithms, such as Ethereum's Ethash [5], Scrypt [22], and CryptNight [23], focus on this aspect to provide ASIC-resistance. A memory-hard PoW computation requires to fetch random pieces of data out of a large dataset. The size of the entire dataset is large enough to make it difficult to design an ASIC that stores the entire dataset in on-chip memory. For example, Ethash's DAG dataset is larger than several gigabytes and continuously increase its size. The location of the data that has to be fetched per each PoW computation with

[2] A miner may change the hash value of the block header by producing a block with different sequence of transactions, but it is computationally more expensive.

different nonce is determined during the computation, which means it is not feasible to prefetch the required data or group multiple nonces that access the same location of the dataset. Therefore, the effective hashrate would be limited by the available off-chip data bandwidth of the platform.

Currently, memory-hard algorithms have stronger ASIC-resistance than Bitcoin's SHA256d as our analysis in Sec. IV-C shows. However, ASIC systems that target such memory-hard PoW algorithms are also eventually released to the market (e.g., Antminer E3 for Ethash and Antminer L3+ for Scrypt). Moreover, emerging transistor technologies, such as 3D stacking [24] would mitigate the performance bottleneck for memory-hard PoW systems. The ASIC-resistance of memory-hard PoW algorithms has to be tested with such emerging technologies as they get approachable to low-cost ASIC systems.

3) **Programmatic PoW:** One of the new directions for PoW mechanisms to be ASIC-resistant is increasing the diversity of the computations. For example, a large pool of mathematical functions [25] or a randomly-generated program can be a part of the computation [26], [27]. It would be impractical to build specialized hardware modules that target each of the possible computation tasks. This type of PoW mechanisms are not realized in a practical blockchain network yet, but they are actively being developed. However, it is still possible to build specialized hardware with a large array of small, yet efficient programmable cores to increase the computation efficiency over existing general-purpose computing platforms.

In this work, we focus on evaluating the ASIC-resistance of a wide range of multi-hash PoW mechanisms. Although multi-hash PoW algorithms claim ASIC-resistance, some of them are already defeated by ASIC systems. For the remaining multi-hash PoW algorithms, it has not been shown that whether they are truly resistant to ASICs or ASIC systems are not introduced yet merely because the blockchain network is not large enough to invest in the ASIC development. Unfortunately, the evaluation in this work reveals that the ASIC-resistance of the multi-hash algorithms does not have an apparent difference with other PoW mechanisms that are already defeated by ASIC. Our evaluation results mean that a stronger ASIC-resistant mechanism is required to effectively deter ASIC-based mining.

B. PROOF-OF-STAKE CONSENSUS

It is often criticized that PoW mechanisms waste massive electrical energy for wasteful computation. An alternative to PoW consensus protocol is Proof-of-Stake (PoS) consensus mechanism [28]. In PoS, a person who possesses a larger amount of cryptocurrency gets a higher chance of obtaining the right to create the next block. This process is called *minting* compared to mining in PoW. Unlike PoW, where a miner obtains such right through a massive amount of computation, PoS consensus protocol requires little computation effort. However, PoS introduces new possible vulnerabilities to blockchain protocols, such as *Nothing at Stake Problem* and *Bribe Attack* [28], [29]. Nothing at Stake Problem arises because there is no additional cost to participate in the minting of multiple different branches. Bribe Attack might nullify a committed transaction in the main branch by encouraging other minters to mint on a false branch using a relatively small amount of rewards (i.e., bribe). Several mitigation approaches were introduced to prevent such attacks on PoS [30], [31]; however, the security of PoS-based blockchain networks has to be tested in a longer term.

C. OTHER CONSENSUS MECHANISMS

Although not as popular as PoS or PoW, other types of consensus protocols have been proposed for blockchain networks. Practical Byzantine Fault Tolerance (PBFT) is a replication-based consensus protocol between known parties that can tolerate the Byzantine failure of up to 1/3 of the parties [32]. Proof-of-Human-Work requires the involvement of human action, by requiring a solution of a puzzle as the proof of work [33]. The puzzle is designed to be human-efficient, but machine-inefficient similar to CAPTCHA [34]. Some consensus protocols try to connect the PoW computation with meaningful work that benefits the world (i.e., *Proof-of-Useful-Work*). For example, Primecoin's consensus protocol is based on a computation that finding a prime number chain [35].

III. MULTI-HASH PoW MECHANISMS

In this section, we describe the multi-hash PoW consensus mechanisms that are evaluated in this work.

A. HASH FUNCTIONS

To be used as a PoW mechanism, a hash function needs a firm guarantee that it is infeasible to invert and collision-resistant (i.e., difficult to find two inputs a and b such that $H(a) = H(b)$, and $a \neq b$) [36]. It is difficult to create a large pool of distinct hash functions with such properties from scratch. The US National Institute of Standards and Technology (NIST) held an open competition to create a new hash function called SHA-3 [37]. A large number of hash function candidates were submitted to the contest, and the final winner was *Keccak* [38]. Most of the multi-hash PoW algorithms utilize the candidates of the NIST competition as their hash functions. Although other candidates were not selected as the final winner, hash functions proceeded to the second round of the competition satisfied NIST's hash function criteria and received a fair amount of cryptanalyses. Other than Keccak, hash functions accepted to the second round (or even to the final round) are as follows: *Blake* [39], *Blue Midnight With (BMW)* [40], *Grøstl* [41], *JH* [42], *Skein* [43], *Luffa* [44], *CubeHash* [45], *SHAVite* [46], *SIMD* [47], *Echo* [48], *Hamsi* [49], *Fugue* [50], *Shabal* [51].

B. FIXED SEQUENCE HASH CHAINS

In X11, eleven different hash functions form a fixed hash chain. That is, the output of the n^{th} hash function is fed into the $(n+1)^{\text{th}}$ hash function (Fig. 3). A valid block is a block with a nonce value that makes the final hash value (i.e., the output of the eleventh hash function) smaller than the threshold. This fixed hash chain approach was expected to be ASIC-resistant because a mining system must implement a large collection of distinct hash functions. On CPUs and GPUs, it is only a matter of using different hash function codes for each of the hash functions, while an ASIC-based system requires a higher design cost to implement all of the hash functions in hardware. Following X11, other similar multi-hash PoW mechanisms were proposed with more hash functions in the chain (e.g., X14 and X17).

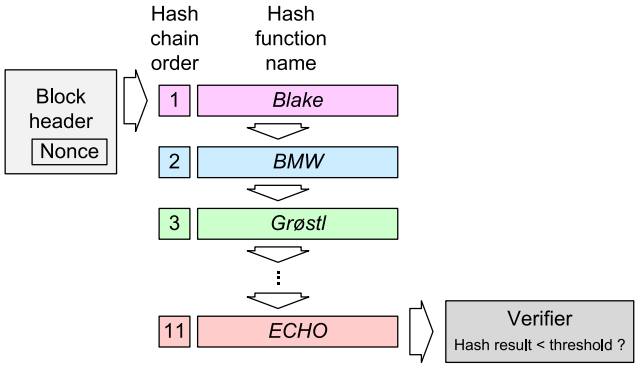

FIGURE 3. Hash functions connected in a fixed chain.

Unfortunately, ASIC-resistance of X11 did not last long. About two years after the launch of the DASH blockchain network, which is the first blockchain network that uses X11 as its consensus mechanism, an ASIC-based mining system that implements the X11 algorithm was launched to the market with superior performance than CPU or GPU performance [52]. Most of the blockchain networks that employed a similar fixed hash chain to provide ASIC-resistance are no longer immune to ASIC-based mining.

There are two reasons why such fixed hash chains quickly lost its ASIC-resistance. First, unlike the expectation of PoW algorithms, design efforts to implement an ASIC that supports a wide variety of hash functions were not so high. The hash functions in X11 are candidates of the NIST competition, where one of the requirements is low hardware overhead. That is, those hash functions are designed to be ASIC-friendly. Secondly, increasing the length of the hash chain not only increased hardware overhead but also increased the computation time on CPU and GPU platforms. Chaining multiple hashes in a row did not decrease the efficiency gap between the general-purpose computing platforms and ASIC-based platforms. We experimentally verify that a fixed hash chain is not an effective way to achieve ASIC-resistance.

C. HASH CHAINS WITH VARIABLE SEQUENCE PER BLOCK

After the introduction of ASIC-based mining systems defeated the ASIC-resistance property of X11, a different class of multi-hash PoW mechanisms was introduced. These PoW mechanisms also use a chain of hash functions, but the sequence of the hash functions gets permuted periodically. The sequence of the hash functions can be determined either by the block number, timestamp, or hash value of the previous block.

The motivation behind this approach is that it is not profitable to create multiple versions of hardware designs for each of the possible hash sequences. However, this expectation is not valid because it assumes that the hash modules in an ASIC have to be connected in a fixed pipeline. It may require additional overhead to provide configurable data path, but a hardware design that supports flexible ordering of the hash functions is entirely possible. Although no commercial ASIC-based mining system is currently available to directly target this class of PoW mechanisms, it is crucial to assess the levels of hardware overhead to support variable-sequence hash chain in advance before the attempts to build such ASIC platforms.

We also implement an FPGA-based multi-hash mechanism that supports the flexible sequencing of hash functions. In Sec. V, we show the implementation and experiment results that show the additional resource and performance impact to support such variable sequence is not high enough to preclude ASIC-based mining.

These variable-sequence hash chains can be further classified into two different types depending on whether they allow repeated use of a hash function in the chain. When the permutation is done without repeats, all hash functions appear only once in the chain, and the computation load per each hash function is identical. However, if the permutation allows repeats, certain hash functions could be utilized more frequently than others. In ASICs, each hash module can only compute a specific type of hash function, and unbalanced use of hash modules would result in performance degradation.

The X16R PoW algorithm composes a new hash chain per each block based on the previous block's hash value (Fig. 4). Each 4-bit in the previous block's hash determines the hash function out of sixteen hash candidates, and total sixteen hash

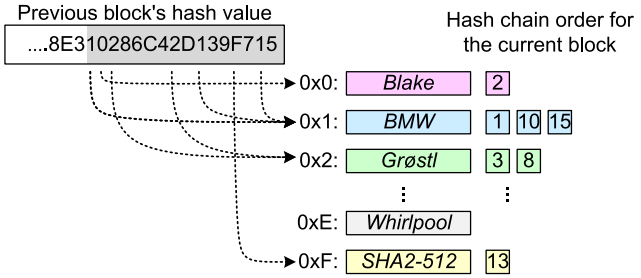

FIGURE 4. Hash function use in the X16R hash mechanism.

functions are chosen to form a hash chain for the following block. As the example in Fig. 4 shows, a hash function can be selected multiple times. Unlike CPU or GPU platforms where repeated use of a hash function can be handled by simply executing the corresponding hash function multiple times, these can result in a load imbalance between the hash modules in an ASIC implementation.

However, the hash value of the previous block would be uniformly distributed, which is an expected property of a cryptographically secure hash function. When the hash functions are selected with uniform probability, the cases with extreme load imbalance are rare. For example, the probability of having more than r repeats of a specific hash function in a chain that selects n hash functions out of n candidates with uniform probability is as the following:

$$Pr(\text{repeats} \geq r) = \sum_{k=r}^{n} \binom{n}{k} \left(\frac{1}{n}\right)^{k} \left(\frac{n-1}{n}\right)^{n-k} \quad (1)$$

The probability quickly diminishes as r grows. When $n = 16$, the probability of having equal or more than three repeats are less than 8%.

D. HASH CHAINS WITH VARIABLE SEQUENCE PER NONCE

We also evaluate a new type of multi-hash PoW mechanism that applies a distinct sequence of hash functions for each nonce value. This PoW mechanism is not realized by an active blockchain network, but the basic idea is an extension of the Quark PoW algorithm. Each PoW computation in Quark goes through nine hash functions. The hash sequence is fixed except for the third, sixth, and ninth hash functions, which are selected out of two candidates based on the second, fifth, and eighth hash result, respectively. We evaluate a PoW mechanism that selects every hash function in the chain dynamically based on the previous hash function's result. We describe the algorithm in detail in Sec. V-C.

IV. PERFORMANCE EVALUATION SETUP FOR MULTI-HASH PoW MECHANISMS

In this section, we describe our evaluation approach to assess the level of ASIC-resistance of multi-hash PoW mechanisms. Several factors would affect the profitability of adopting an ASIC-based system, such as the amount of initial investment for ASIC fabrication and the cost of operation (e.g., electricity and maintenance cost). It is difficult to boil down such factors into a single metric because they have high fluctuations depending on the market condition. To avoid the involvement of unpredictable factors, we investigate ASIC-resistance solely based on the performance aspect. That is, we evaluate how the ASIC-resistance PoW algorithms impose performance disadvantage on ASICs while they retain performance levels on CPUs and GPUs.

We estimate the expected performance of ASIC-based mining by implementing a multi-hash PoW platform on an FPGA. Although the results from FPGA implementations are not actual measurements on an ASIC, FPGA results can be used to assess the performance and the area overhead of ASIC implementations, especially for logic-dominated designs [53]. Hash functions heavily use logic operations, such as XOR, shift, majority, and mux. FPGA implementation results in Sec. V-A also show that the most prominent resource usage is the logic components in look-up tables (LUTs).

A. PERFORMANCE METRIC

The performance of a PoW systems is commonly represented using the *hashrate*. It represents the number of PoW computations per unit of time. A PoW platform with a high hashrate has a higher chance of finding a valid block during a given amount of time. Technically, the hashrate is not the actual number of hash function computations because PoW algorithms may involve multiple hash functions per each instance of PoW operation. A PoW algorithm may result in a lower hashrate if it has more hash functions per each instance of PoW computation. We denote the hashrate of PoW algorithm x on computing platform y as $P_y(x)$. (y is either CPU, GPU, or FPGA in this work).

B. EXPERIMENT PLATFORMS

1) CPU-BASED PLATFORM

We use the *cpuminer* open-source mining software [54] to measure the performance of a CPU-based PoW platform. CPU hashrates are measured on a system with an Intel i7-8700K CPU, which has eight hardware threads.

2) GPU-BASED PLATFORM

To measure the GPU hashrates, we use the *ccminer* open-source mining software [55]. ccminer uses the CUDA programming interface to program NVIDIA GPU devices. Our experiment platform uses an NVIDIA GeForce GTX 1080 ti GPU, which has 3,584 computing cores. The host machine that runs the GPU is the same platform as the CPU-based platform.

3) FPGA-BASED PLATFORM

Our multi-hash PoW platform uses the Xilinx Zynq UltraScale+ ZU9EG FPGA. The hash functions are implemented using the open-source hash modules created by the Cryptographic Engineering Research Group (CERG) at George Mason University [56]. Each hash module is connected to input and output FIFOs, which together forms a hash unit (Fig. 5). These FIFOs provide a temporary storage space while the hash modules are computing the results. Also, these FIFOs support clock domain crossing to enable different clock frequency (Clk_{hash}) in each hash module. Table 1 lists the hash modules used in our evaluation and their FPGA resource utilization. As we mentioned, FPGA resource utilization shown in Table 1 is dominated by the LUT use. More than 77% of the available LUTs are used when the FPGA platform contains all fifteen hash modules, while the

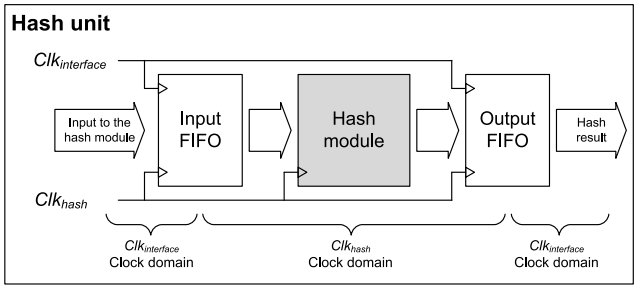

FIGURE 5. Structure of a hashunit.

utilization level of registers and on-chip memory modules are less than 18% and 5%, respectively.

Unlike CPU or GPU platforms where the number of computing cores is fixed, the number of hash modules in an FPGA or ASIC platform can be increased if the resource (i.e., silicon area) is available. Unfortunately, the FPGA module used in this evolution is not large enough to flexibly include multiple instances of multi-hash PoW units. For example, in Table 1, Fixed-12 and Fixed-15 use 72.7% and 77.1% of LUT resources on our FPGA platform, respectively. Only one instance of either case can be instantiated on FPGA even though Fixed-15 requires a higher hardware overhead. Such discrepancy is less prominent on an ASIC platform with a higher logic capacity per chip.

To represent the expected performance without the disproportionate performance evaluation, we scale the measured hashrate ($\hat{P}_{\text{FPGA}}(x)$) based on the ratio of the available LUTs (274,080 on ZU9EG) and the platform's LUT utilization.

$$P_{\text{FPGA}}(x) = \hat{P}_{\text{FPGA}}(x) \times \frac{\text{\# of available LUTs on FPGA}}{\text{\# of LUTs used}} \quad (2)$$

For example, a single instance of SHA256d PoW module on FPGA uses 3,496 LUTs, and it achieves the hashrate of 3.2×10^5 PoW operations per second. On the Xilinx ZU9EG FPGA, we scale the hashrate to match that from $\frac{274{,}080}{3{,}496} = 78.4$ instances of SHA256d PoW modules on FPGA, which is 251 MH/s.

C. EVALUATION METRIC FOR ASIC-RESISTANCE

The goal of the comparison in this work is to investigate whether the multi-hash PoW algorithms make ASIC designs less effective when compared to the general-purpose computing platforms for computing the PoW algorithms. Directly comparing the measured hashrates from the experiment platforms does not give a good representation of ASIC-resistance. Even if a platform has a lower hashrate, the throughput of a whole system can be increased by employing many instances of the same platform. For instance, most of the ASIC-based mining systems include a large number of independent ASIC chips to increase the throughput per system [9].

Instead of comparing the raw measurements from entirely different systems, we define a metric to evaluate ASIC-resistance of PoW algorithms based on the relative performance difference on each platform. We use Bitcoin's SHA256d as the common performance baseline. Due to its relatively simple PoW computation structure, many ASIC-based PoW platforms have been created for Bitcoin, and ASIC-based platforms dominate Bitcoin's PoW consensus operation. Therefore, SHA256d PoW mechanism can be considered not resistant to ASIC. We evaluate the relative performance of multi-hash PoW algorithms compared to that of SHA256d on each of the three computing platforms. On our CPU, GPU, and FPGA platforms, the hashrate of SHA256d is 31.6 megahashes per second (MH/s), 1500 MH/s, and 251 MH/s, receptively.

We measure the level of ASIC-resistance of a PoW mechanism x by calculating the relative performance *disadvantage* of ASIC over the (general-purpose) computing platform y (y = CPU or GPU):

$$AD_y(x) = \frac{P_{\text{FPGA}}(\text{SHA256d})}{P_{\text{FPGA}}(x)} \Big/ \frac{P_y(\text{SHA256d})}{P_y(x)} \quad (3)$$

This ASIC disadvantage (AD) metric is the ratio between the two values, $\frac{P_{\text{FPGA}}(\text{SHA256d})}{P_{\text{FPGA}}(x)}$ and $\frac{P_y(\text{SHA256d})}{P_y(x)}$. The first value ($P_{\text{FPGA}}$) represents the performance impact of using algorithm x compared the baseline SHA256d algorithm on ASIC. Most of ASIC-resistant algorithms seek to increase this value, by making computation harder on ASICs. Being ASIC-resistant also requires algorithm x does not have such degree of performance impact on CPU or GPU platforms; otherwise, ASICs still possess performance advantage. The second value (P_y) of the AD metric addresses the performance impact on CPUs and GPUs. A PoW algorithm x can be considered to have a stronger ASIC-resistance than SHA256d only if its performance degrades in a larger scale on ASICs than it does on CPUs or GPUs (i.e., $\frac{P_{\text{FPGA}}(\text{SHA256d})}{P_{\text{FPGA}}(x)} >> \frac{P_y(\text{SHA256d})}{P_y(x)}$).

For example, consider a case where $P_{\text{FPGA}}(\text{SHA256d}) = 100$ MH/s and $P_{\text{CPU}}(\text{SHA256d}) = 1$ MH/s. Suppose a new PoW algorithm x_1 is proposed with a more complex hash operation and resulted in $P_{\text{FPGA}}(x_1) = 10$ MH/s. However, due to its complex hash operation, $P_{\text{CPU}}(x_1)$ is also reduced to 0.1 MH/s. In this case, $AD_{\text{CPU}}(x_1) = \frac{100\text{MH/s}}{10\text{ MH/s}} / \frac{1\text{ MH/s}}{0.1\text{ MH/s}} = 1\times$, which means x_1 does not impose any more performance disadvantage on ASICs compared to SHA256d. Since SHA256d is not resistant to ASIC, this level of difference would be considered far from a strong ASIC-resistance.

Unfortunately, multi-hash PoW mechanisms discussed in this paper do not show any meaningful ASIC-resistance when evaluated based on the AD metric. Most of them did not show a sufficiently high AD_{CPU} or AD_{GPU} value; even some of them show an AD value lower than $1\times$.

In contrast, memory-hard PoW mechanisms have higher AD levels. Although we did not evaluate any memory-hard PoW mechanism on FPGA in this work, the hashrate of a memory-hard PoW mechanism can be estimated based on the platform's external memory bandwidth. For example, the Ethash PoW algorithm requires 8 KB of external data access per each instance of PoW computation. On the NVIDIA 1080 ti GPU, which has 320 gigabytes per second (GB/s) of external memory bandwidth, the theoretical

TABLE 1. FPGA resource utilization of fixed hash chain implementations on the Xilinx Zynq UltraScale+ ZU9EG FPGA.

Hash Order	Hash unit	Fixed-3		Fixed-6		Fixed-9		Fixed-12		Fixed-15	
		LUT	Register	LUT	Register	LUT	Register	LUT	Register	LUT	Register
1	BLAKE	10,828	6,377	10,831	6,377	10,840	6,377	10,834	6,377	10,835	6,377
2	BMW	28,042	5,776	28,004	5,776	28,000	5,776	28,028	5,776	27,979	5,776
3	Grøstl	22,478	5,834	22,475	5,833	22,479	5,836	22,464	5,833	22,465	5,834
4	Skein			7,567	4,419	7,546	4,419	7,620	4,419	7,632	4,418
5	JH			5,581	4,032	5,587	4,032	5,589	4,030	5,578	4,028
6	Keccak			4,755	3,932	4,754	3,934	4,754	3,937	4,755	3,934
7	Luffa					8,835	3,047	8,834	3,045	8,782	3,046
8	Cubehash					15,028	12,156	15,056	12,164	15,048	12,148
9	SHAvite-3					6,919	5,215	6,913	5,215	6,927	5,215
10	SIMD							52,606	16,656	52,597	16,656
11	ECHO							24,709	7,150	24,688	7,139
12	Hamsi							11,826	6,720	11,768	6,700
13	Fugue									4,497	2,966
14	Shabal									5,296	6,508
15	SHA2-512									2,504	2,734
Total (% of FPGA chip)		61,348 (22.3%)	17,987 (3.28%)	79,213 (28.9%)	30,369 (5.54%)	109,988 (40.1%)	50,792 (9.27%)	199,233 (72.7%)	81,322 (14.8%)	211,351 (77.1%)	93,479 (17.1%)

peak hashrate is 40 MH/s, and the actual hashrate measured on the GPU is about 32.4 MH/s. The Xilinx ZU9EG FPGA has a single DDR4 interface that provides the peak bandwidth of 19.2 GB/s, which translates into 2.34 MH/s of maximum hashrate. The lower bound of $AD_{\mathrm{GPU}}(\mathrm{Ethash})$ is $\frac{P_{\mathrm{FPGA}}(\mathrm{SHA256d})}{P_{\mathrm{FPGA}}(\mathrm{Ethash})} / \frac{P_{\mathrm{GPU}}(\mathrm{SHA256d})}{P_{\mathrm{GPU}}(\mathrm{Ethash})} = \frac{251\mathrm{MH/s}}{2.34\mathrm{MH/s}} / \frac{1500\mathrm{MH/s}}{32.4\mathrm{MH/s}} \approx 2.32\times$, which is much higher than $1\times$.

This ASIC-disadvantage level of Ethash is a conservative estimation. Unlike the evaluation of multi-hash PoW algorithms, where most of the hardware overhead comes from logic elements, an overhead comparison that involves external I/O on FPGA is not analogous to that of an ASIC. On FPGA, logic elements involve high overhead (more than an order of magnitude) due to their programmability, while many external I/O use dedicated hard blocks without such overheads. The performance of Ethash and SHA256d depends on external I/O bandwidth and logic capacity, respectively. The performance gap between those two on ASICs will be smaller than the conservative estimation discussed here (i.e., a higher AD level than $2.32\times$).

Although memory-hard PoW mechanisms prevented many blockchain protocols from being dominated by ASIC-based mining, ASIC mining systems for memory-hard PoW mechanisms are eventually introduced to the market as the blockchain networks grow, which means even the AD level of $2.32\times$ is not strong enough to permanently prevent ASIC-based mining.

V. ASIC RESISTANCE EVALUATION RESULTS

A. FIXED HASH CHAINS

ASIC-based mining systems already defeated Multi-hash PoW mechanisms that use fixed hash chains. We experimentally show that these fixed hash chains indeed have almost no ASIC-resistance based on the AD metric in (3). Starting from X11, which has eleven hash functions, many algorithms continued to increase their hash chain lengths while expecting a longer chain would have a stronger ASIC-resistance. To evaluate the level of ASIC-resistance of this approach, we tested five different hash chain lengths, up to fifteen hashes in line. We set the sequence of those fifteen hash functions the same as the hash sequence of the X15 algorithm,[3] except the last hash function. In X15, the last hash function is *Whirlpool* [57], which is not included in the NIST SHA-3 standard competition. Instead, we use SHA2-512 [58] as the last (fifteenth) hash function of the chain. SHA2-512 is also used as a part of X16R and X17.

We create fixed hash chains on FPGA by directly connecting the hash units in a pipelined architecture. We name those fixed pipeline implementations as Fixed-n, where n is the number of hash functions in the chain. Table 1 shows the FPGA resource utilization of fixed hash chain implementations with five different hash chain lengths.

The pipeline is fed with the block header with a unique nonce value on each PoW operation. The hash modules inside the hash units are also designed using a pipelined architecture, and they start processing of the next computation task as soon as the previous one flows down to the pipeline; that is, multiple nonce trials are computed simultaneously through the pipeline. After the final hash unit completes the hash operation, a verifier module checks if the hash result satisfies the valid condition.

Figs. 6a-6c show the measured hashrates of fixed hash chains on CPU, GPU, and FPGA platforms, respectively. Longer hash chains decrease hashrates on all platforms, not just on FPGA. On CPU and GPU platforms, a longer hash chain results in a lower hashrate because of the elongated computation time per each PoW operation. On FPGA, the throughput of a single pipeline is not directly affected by the length of the hash chain, but the effective hashrate per available hardware resource (i.e., the scaled hashrate) decreases as the number of hash functions in the chain

[3] X12, X13, X14, X15, and X17 follow the same hash sequence as their predecessors. That is, the first eleven hash functions in X12 have the same sequence as X11, and so forth.

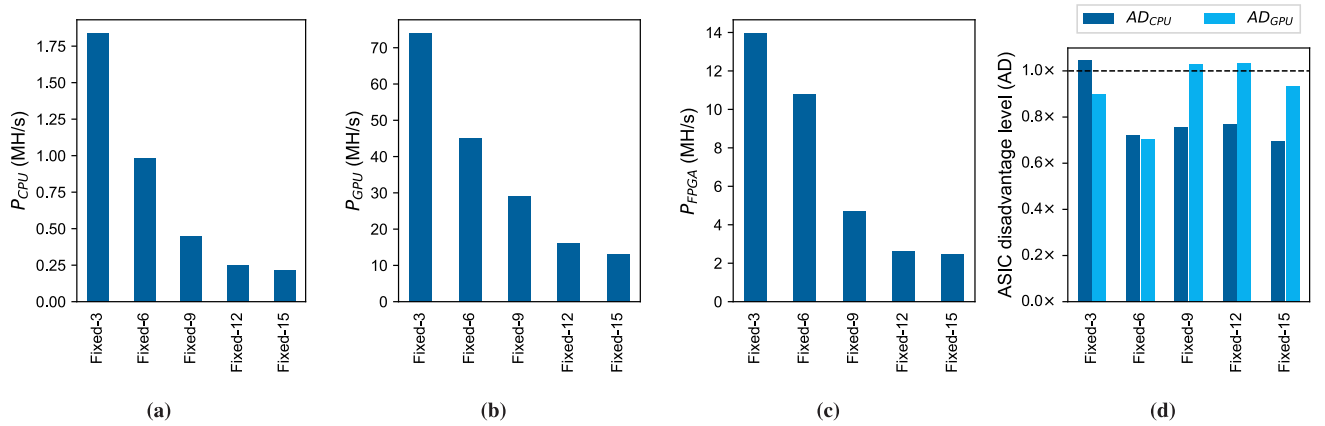

FIGURE 6. Performance results of fixed hash chains on our experiment platforms. (a) CPU platform hashrates. (b) GPU platform hashrates. (c) FPGA platform scaled hashrates. (d) AD levels.

increases. Fig. 6d shows the ASIC-resistance levels represented in the AD values over CPU and GPU. Because the hashrates of all three platforms decrease as the hash chain length increases, the AD levels of fixed hash chains are all close to 1× and do not show any meaningful increase over the baseline. These low levels of ASIC-resistance are well reflected in the fact that multiple ASIC-based mining systems are introduced for the fixed hash chain mechanisms. Merely using a larger number of hash functions in a fixed pipeline does not effectively deter ASIC-based PoW operations.

B. VARIABLE-SEQUENCE HASH CHAINS

If the hash modules in an ASIC are connected in a fixed pipeline as the Fixed-n implementations, such an ASIC design cannot be used for PoW mechanism that dynamically changes the sequence of hash functions in the chain. Many multi-hash PoW mechanisms, such as X11EVO, X16S, and TimeTravel, are based on such assumption and claimed ASIC-resistance by periodically shuffling the sequence of the hash functions in the chain. However, there is no need to create a large number ($n!$) of distinct ASIC implementations to support all possible permutations out of n hash functions. Instead, an on-chip interconnect that provides a configurable data path between the hash modules can accommodate any hash function sequencing. ASIC-resistance through variable-sequence hash chains can be achieved if the additional overhead to provide such a flexible interconnect is big enough to reduce ASIC's performance advantage by a large degree.

We create another multi-hash PoW system that supports variable-sequence hash chains, called *VarChain*, to evaluate the additional overhead for supporting variable-sequence hash chains (Fig. 7). The input and output ports of the fifteen hash units are connected to a 16×16 crossbar interconnect. The initial input (i.e., block header with a nonce value) and the final hash result of the chain are also assigned to separate input and output ports of the interconnect. When a hash unit completes the hash computation, the resulting output is delivered to the next hash unit through the crossbar interconnect,

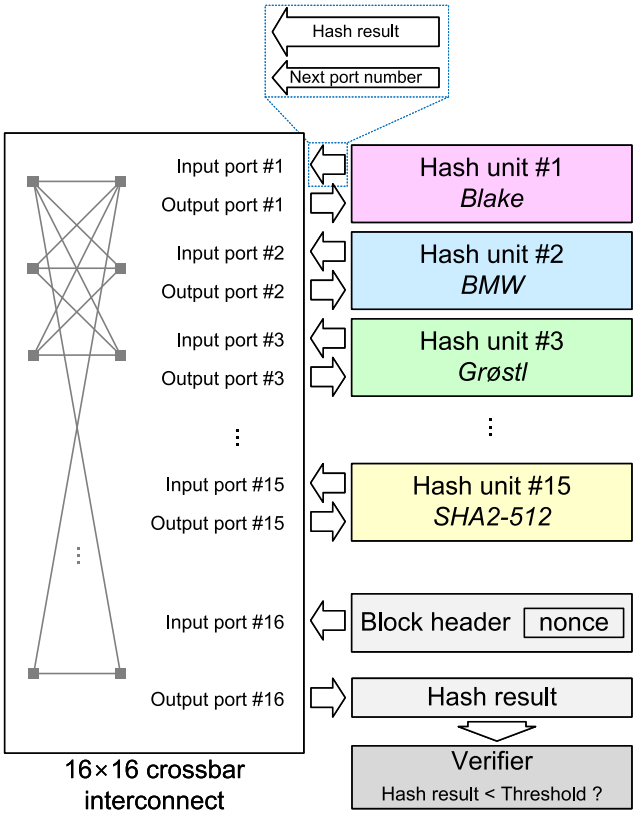

FIGURE 7. Hash modules connected to a crossbar interconnect (VarChain).

and each hash unit informs the crossbar interconnect of the next port number. We choose the crossbar interconnect for VarChain because it provides a dedicated data path between each input and output port pair. Other types of interconnection networks, such as a shared bus or a mesh architecture, can be used for the variable hash sequence with lower overhead but higher latency [59]. Even with a crossbar interconnect, however, the added hardware resource overhead compared to the fixed hash pipeline is not prohibitively large to discourage the use of such a flexible interconnect.

Table 2 compares the hardware overhead of Fixed-15 versus VarChain. In terms of the utilized number of LUTs and registers, VarChain has only 5.8% and 6.2% higher overhead than Fixed-15, respectively. Fig. 8 is the layout of VarChain that displays the placement of the fifteen hash units and the crossbar interconnect module (blue color) on FPGA.

TABLE 2. FPGA resource utilization comparison between Fixed-15 and VarChain.

Hash unit	Fixed-15		VarChain	
	LUT	Register	LUT	Register
BLAKE	10,835	6,377	10,882	6,396
BMW	27,979	5,776	28,017	5,795
Grøstl	22,465	5,834	22,499	5,856
Skein	7,632	4,418	7,684	4,438
JH	5,578	4,028	5,658	4,051
Keccak	4,755	3,934	4,807	3,959
Luffa	8,782	3,046	8,859	3,070
Cubehash	15,048	12,148	15,200	12,224
SHAvite-3	6,927	5,215	6,999	5,234
SIMD	52,597	16,656	54,973	16,675
ECHO	24,688	7,139	24,791	7,152
Hamsi	5,884	3,350	5,956	3,369
Fugue	4,497	2,966	4,561	2,986
Shabal	1,324	1,627	1,364	1,710
SHA2-512	2,504	2,734	2,552	2,753
Crossbar interconnect	-		8,831	5,071
Total (% of FPGA chip)	211,351 (77.1%)	93,479 (17.1%)	223,681 (81.6%)	99,238 (18.1%)

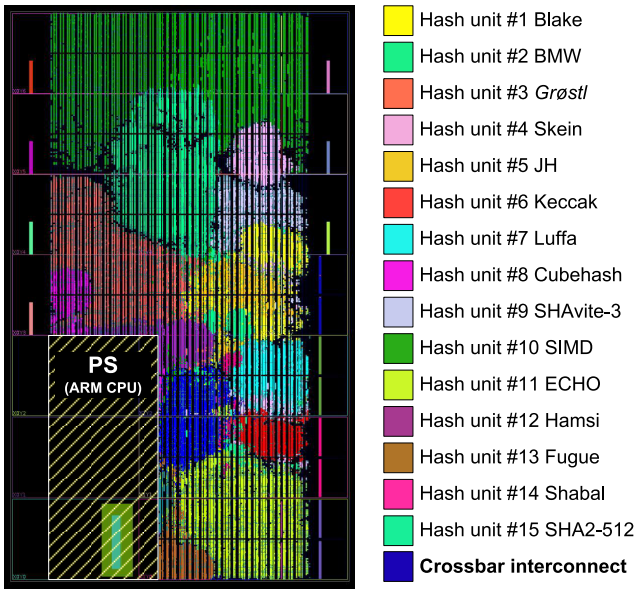

FIGURE 8. Layout of VarChain placed and routed on the Xilinx Zynq UltraScale+ ZU9EG FPGA device.

Using the VarChain platform, we evaluate three different types of the variable-sequence hash chains. The first two of them represent the hash chains that vary the hash sequence per block. *Variable-15P* sets the hash chain sequence by rearranging the order of the hash functions (i.e., permutation without repeats). This scheme is similar to that of TimeTravel, X11EVO, and X16S PoW mechanisms. These have 8, 11, and 16 hash functions, respectively. The hash sequence may get changed after a certain period (e.g., daily) or on every block. We change the hash sequence on every block when evaluating the performance of Variable-15P.

Variable-15R allows repeated use of a hash unit in the chain, similar to X16R (i.e., permutation with repeats). The only difference between Variable-15R and X16R is that Variable-15R lacks the Whirlpool hash in the hash candidates. This approach provides another aspect of ASIC-resistance because even if an ASIC platform can adapt to the variable hash sequences, it cannot dynamically adapt itself to the variable load distribution between the hash functions. In the extreme, it is possible that the previous block's hash happens to select the same hash fifteen times for the following block's hash chain. In that case, the throughput may degrade by $\frac{1}{15}\times$ while the rest unselected hash modules are left idle. However, as we discussed in Sec. III-C, such extreme cases are rare. In most cases, the level of load imbalance is tolerable.

Figs. 9a-9c compares the hashrate of fixed and variable-sequence hash chains on CPU, GPU, and FPGA platforms, respectively. Although the sequence of hash functions changes on each block, Variable-15P has almost no performance variation because all hash functions are used exactly once in the hash chain. On the other hand, the resulting hashrates of Variable-15R have high variations depending on the previous block's hash value because different combinations of hash functions are included in the chain. The hashrates of Variable-15R in Figs. 9a-9c are shown in box plots that displays five values: 1 percentile, the first quartile, mean (red bar), the third quartile, and 99 percentile. Fig. 9d shows the corresponding AD levels. The AD values for Variable-15R are derived using its mean hashrate.

The hashrate of Variable-15P does not have a significant difference from the hashrate of Fixed-15 on CPU and GPU platforms. On FPGA, the scaled hashrate gets reduced by 6% from P_{FPGA}(Fixed-15), which reflects the small added overhead from the crossbar interconnect. The AD levels of Variable-15P have no significant improvement from those of Fixed-15.

Although the hashrates of Variable-15R have a high variance per block, the mean hashrate converges very closely to the hashrate of Fixed-15 and Variable-15P on CPU and GPU platforms. If a miner participates in the PoW operation over multiple blocks, the expected hashrate would not have a noticeable difference even if the hashrate fluctuates on each block. On FPGA, on the other hand, the mean hashrate of Variable-15R is worse than the hashrates of other multi-hash PoW mechanisms due to the load imbalance. The mean hashrate of Variable-15R on FPGA is 32.6% lower than P_{FPGA}(Variable-15P). The performance degradation due to the load imbalance is reflected in AD_{GPU}(Variable-15R) value of 1.33×, which tells that Variable-15R has a better ASIC-resistance than other multi-hash PoW mechanisms.

C. VARIABLE-SEQUENCE HASH CHAINS PER NONCE

Using the VarChain FPGA platform, we evaluate another case of possible multi-hash PoW algorithm that changes the hash

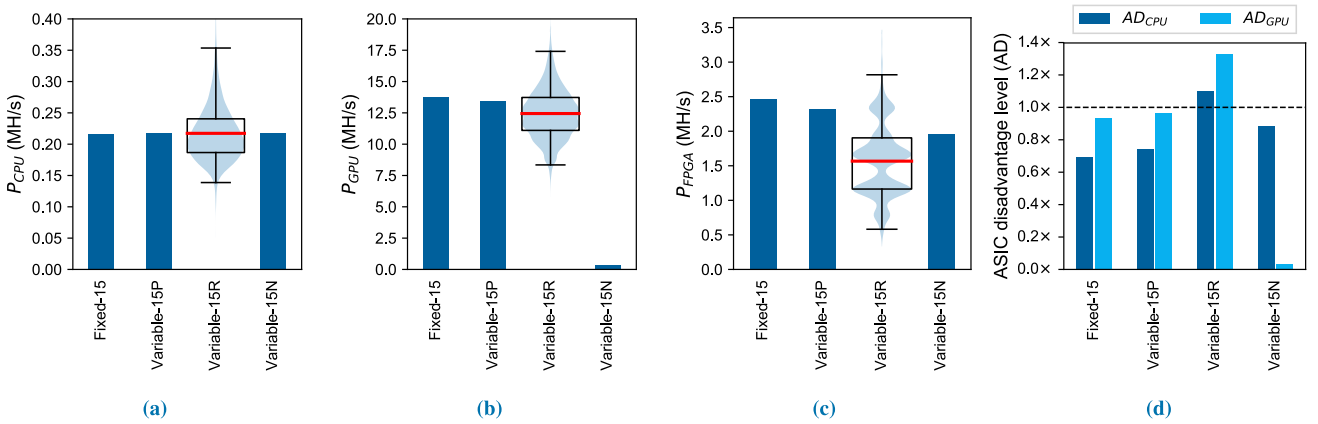

FIGURE 9. Performance results of variable-sequence hash chains on our experiment platforms. (a) CPU platform hashrates. (b) GPU platform hashrates. (c) FPGA platform scaled hashrates. (d) AD levels.

sequence on every nonce value (*Variable-15N*). Variable-15N selects the i^{th} hash function in the chain based on the $(i-1)^{\text{th}}$ hash result during the PoW computation. That is, the output of a hash function is not only sent to the next hash function, but a part of the computed hash value determines the next hash function. Because the hash value would be different if the nonce value changes, the sequence of hash functions applied to each PoW computation instance is not identical even in the same block. Algorithm 1 describes the computation flow of Variable-15N.

Algorithm 1 Pseudocode for Varaible-15N

Input: B := Block header
n := Nonce
th := Hash value threshold for a valid block
Output: v := Validity of the block header

$Hashes[] \leftarrow$ {0:Blake, 1:BMW, ..., 14:SHA2-512}
$num\_hash = size(Pool)$

$x \leftarrow B|n$
$H \leftarrow Hashes[x \mod num\_hash]$
for $i \leftarrow 1$ **to** 15 **do**
  $x \leftarrow H(x)$
  $H \leftarrow Hashes[x \mod num\_hash]$

$v \leftarrow x < th$
return v

Similar to the per-block performance variation in Variable-15R, the computation time to perform a single instance Variable-15N computation would be different for each nonce value. However, the performance of a PoW platform should be determined by the throughput (i.e., the number of tried nonce values per given amount of time). We measure P_y(Variable-15N) over the period of 1 hour using random nonce values. Figs. 9a-9c also show the hashrates of Variable-15N. P_{CPU}(Variable-15N) has almost no difference with other hashrates on the CPU because per-nonce performance variation would even out over a large number of PoW computations with different nonce values. On FPGA, Variable-15N also suffers from the load imbalance similar to Variable-15R. However, P_{FPGA}(Variable-15N) is better than the mean value of P_{FPGA}(Variable-15R), only 15.7% lower than P_{FPGA}(Variable-15P). On VarChain, multiple instances of Variable-15N computations are computed simultaneously, each of which has a different combination of hash functions in its chain. This mixture of different hash chains would mitigate the load imbalance in the individual chains per nonce.

Interestingly, P_{GPU}(Variable-15N) is considerably low (2.68% of P_{GPU}(Variable-15P)). This is due to the *control flow divergence* on GPU platforms [60]. GPUs achieve the high computation efficiency by executing multiple threads in parallel if the threads go through the same control path. When computing PoW algorithms, GPU threads are assigned with PoW computation tasks that have different nonce values. The control paths of the threads do not have significant divergence from each other if all of them use the same hash sequence. On Variable-15N, however, the computation task for each nonce value would use a different hash chain sequence. This divergence results in a severe performance degradation on GPU. Therefore, Variable-15N is not an effective way to achieve ASIC-resistance, but it can be used to exclude GPU-based platforms from mining.

D. DISCUSSION

Table 3 summarizes the evaluated PoW mechanisms and their ASIC-disadvantage levels. Regardless of their hash chain types or the current availability of the ASIC-based mining systems, multi-hash PoW mechanisms have low degrees of ASIC-resistance. Variable-15R has a better ASIC-resistance than other multi-hash PoW mechanisms, but its AD levels are much lower than those of the memory-hard Ethash PoW algorithm. Given that there exist ASIC-based mining systems for Ethash as well, it is difficult to consider the ASIC-resistance level of Variable-15R is strong enough to preclude ASIC-based mining.

TABLE 3. ASIC-resistance of the PoW mechanisms discussed in this paper.

PoW mechanism	Hash chain type	AD_{CPU}	AD_{GPU}	Example PoW implementations	ASIC availability[a]
Fixed-15	Fixed sequence	1.05	0.94	X11, X12, X13, X14, X15, X17 C11, Tribus, Qubit, NIST5	Yes
Variable-15P	Permutation without repeats	0.72	0.97	X11EVO, X16S, TimeTravel	No
Variable-15R	Permutation with repeats	0.76	1.33	X16R	No
Variable-15N	Permutation with repeats (different chain per nonce)	0.77	0.03	Quark[b]	No[c]
Memory-hard(Ethash)	-	2.83	2.32	Ethash, CryptoNight, Scrypt	Yes

[a] Based on ASIC-based mining platforms that have been introduced to the retail market by September 2018.
[b] Variable-15N is a more generalized approach than Quark with a larger variability in the hash function sequence.
[c] ASIC mining system for Quark is available, but not for Variable-15N.

A new PoW mechanism can be designed to amplify the degree of load imbalance in Variable-15R or X16R to impose a higher performance disadvantage on ASIC. For example, a PoW mechanism simply selects only a few hash functions out of the candidates on each block, which greatly reduces the ASIC utilization. However, in such a case, the unused hash modules can be power-cycled to reduce the energy consumption of the platform. In terms of the energy efficiency, an ASIC still has an advantage. Evaluating the ASIC-resistance in terms of the energy efficiency is not the focus of this work. Most of the computing cores or hash modules are almost fully occupied while computing the multi-hash PoW algorithms discussed in this work.

We observed low levels of ASIC-resistance in many variants of Multi-hash PoW mechanisms. It means that someone can easily create ASICs for such PoW mechanisms that provide superior computing efficiency than CPUs or GPUs.

VI. CONCLUSION

In this work, we evaluated ASIC-resistance levels of multi-hash PoW mechanisms. Even after the most basic form of multi-hash PoW mechanisms that use a fixed hash chain is defeated by ASICs, many variants of multi-hash PoW mechanisms are continuously being developed. We implemented several multi-hash PoW mechanisms on FPGA to assess their performance on ASICs. Our evaluation includes PoW mechanisms that are already shown to be not resistant to ASICs as well as mechanisms that still claim ASIC-resistance. The results from this work revealed that PoW mechanisms based on multiple hash functions have little ASIC-resistance in terms of the performance gap against the general-purpose computing platforms.

ASIC-resistance is challenging to achieve because the profitability of ASIC-based mining is determined by many factors, not only by the performance. Even the memory-hard PoW mechanisms that favor general-purpose computing systems in terms of the achievable performance are not safe from ASIC-based mining. Blockchain networks that require ASIC-resistance in their consensus protocol should develop a PoW mechanism with a stronger ASIC-resistance or consider different types of consensus mechanisms, such as PoS.

REFERENCES

[1] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten, "SoK: Research perspectives and challenges for Bitcoin and cryptocurrencies," in *Proc. IEEE Symp. Secur. Privacy*, May 2015, pp. 104–121.

[2] M. Swan, *Blockchain: Blueprint for a New Economy*. Newton, MA, USA: O'Reilly Media, 2015.

[3] F. Tschorsch and B. Scheuermann, "Bitcoin and beyond: A technical survey on decentralized digital currencies," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 2084–2123, 3rd Quart., 2016.

[4] S. Nakamoto. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[5] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," Ethereum, Tech. Rep. EIP-150 REVISION, 2014. [Online]. Available: http://yellowpaper.io/

[6] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the Internet of Things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[7] S. Underwood, "Blockchain beyond Bitcoin," *Commun. ACM*, vol. 59, no. 11, pp. 15–17, 2016.

[8] M. Iansiti and K. R. Lakhani, "The truth about blockchain," *Harvard Bus. Rev.*, vol. 95, no. 1, pp. 118–127, 2017.

[9] M. B. Taylor, "The evolution of bitcoin hardware," *Computer*, vol. 50, no. 9, pp. 58–66, 2017.

[10] A. Beikverdi and J. S. Song, "Trend of centralization in bitcoin's distributed network," in *Proc. IEEE/ACIS 16th Int. Conf. Softw. Eng., Artif. Intell., Netw. Parallel/Distrib. Comput. (SNPD)*, Jun. 2015, pp. 1–6.

[11] *Hashrate Distribution—An Estimation of Hashrate Distribution Amongst the Largest Mining Pools*. Accessed: Sep. 19, 2018. [Online]. Available: https://www.blockchain.com

[12] *The Antbleed Backdoor*. Accessed: Sep. 27, 2018. [Online]. Available: https://www.antbleed.com/

[13] BITMAIN. *ANTMINER S9i, The World's Most Power-Efficient Bitcoin Miner*. Accessed: Sep. 19, 2018. [Online]. Available: https://shop.bitmain.com/promote/antminer_s9i_asic_bitcoin_miner/

[14] E. Duffield. *X11 White Paper*. Accessed: Sep. 19, 2018. [Online]. Available: https://github.com/dashpay/dash/wiki/Whitepaper

[15] *X14 POW/POS Cryptocurrency With Block Explorer, Stats, and IRC Chat!* Accessed: Sep. 19, 2018. [Online]. Available: https://github.com/webcoinx14/Webcoin

[16] *X17 Algorithm—List of All X17 Coins and Miners for NVIDIA & AMD*. Accessed: Sep. 19, 2018. [Online]. Available: https://coinguides.org/x17-algorithm-coins/

[17] *RevolverCoin Resources*. Accessed: Sep. 19, 2018. [Online]. Available: http://revolvercoin.org/resources/

[18] L. Pighetti. (2018). *X16S—Sixteen Shuffled Algorithms Designed for Small Miners*. Accessed: Sep. 19, 2018. [Online]. Available: https://github.com/Pigeoncoin/brand/blob/master/X16S-whitepaper.pdf

[19] T. Black and J. Weight. *X16R—ASIC Resistant by Design*. Accessed: Sep. 19, 2018. [Online]. Available: https://ravencoin.org/wp-content/uploads/2018/03/X16R-Whitepaper.pdf

[20] *Quark Mining Guide*. Accessed: Sep. 19, 2018. [Online]. Available: http://www.quarkcoins.com/mining-quarkcoin.html

[21] *Time Travel (TimeTravel10 | Bitcore) Algorithm, Coins, Miners and Hashrate*. Accessed: Sep. 19, 2018. [Online]. Available: https://coinguides.org/time-travel-coins/

[22] C. Percival, "Stronger key derivation via sequential memory-hard functions," Tarsnap, Vancouver, BC, Canada, Tech. Rep., 2009.

[23] N. V. Saberhagen. (2013). *CryptoNote V.2.0*. [Online]. Available: https://cryptonote.org/whitepaper.pdf
[24] M. M. S. Aly *et al.*, "Energy-efficient abundant-data computing: The N3XT 1,000x," *Computer*, vol. 48, no. 12, pp. 24–33, Dec. 2015.
[25] J. Emanuel. *'Loaded' PoW: A New Direction in Proof-of-Work Algorithms*. Accessed: Sep. 27, 2018. [Online]. Available: https://medium.com/@jeffrey.emanuel/loaded-pow-a-new-direction-in-proof-of-work-algorithms-ae15ae2ae66a
[26] *ProgPoW—A Programmatic Proof of Work*. Accessed: Sep. 27, 2018. [Online]. Available: https://github.com/ifdefelse/ProgPOW
[27] *What We Need to Know About Proof of Work (POW)*. Accessed: Sep. 27, 2018. [Online]. Available: https://www.reddit.com/r/Monero/comments/8bshrx/-what_we_need_to_know_about_proof_of_work_pow/
[28] I. Bentov, A. Gabizon, and A. Mizrahi, "Cryptocurrencies without proof of work," in *Financial Cryptography and Data Security*, J. Clark, S. Meiklejohn, P. Y. Ryan, D. Wallach, M. Brenner, and K. Rohloff, Eds. Berlin, Germany: Springer, 2016, pp. 142–157.
[29] BitFury Group. (2015). *Proof of Stake Versus Proof of Work Version 1.0*. [Online]. Available: https://bitfury.com/content/downloads/pos-vs-pow-1.0.2.pdf
[30] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Advances in Cryptology*, J. Katz and H. Shacham, Eds. Cham, Switzerland: Springer, 2017, pp. 357–388.
[31] *Delegated Proof-of-Stake Consensus*. Accessed: Sep. 19, 2018. [Online]. Available: https://bitshares.org/technology/delegated-proof-of-stake-consensus/
[32] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *Proc. 3rd Symp. Oper. Syst. Design Implement.* Berkeley, CA, USA: USENIX Assoc., 1999, pp. 173–186.
[33] J. Blocki and H.-S. Zhou, "Designing proof of human-work puzzles for cryptocurrency and beyond," in *Theory Cryptography*, M. Hirt and A. Smith, Eds. Berlin, Germany: Springer Berlin Heidelberg, 2016, pp. 517–546.
[34] L. V. Ahn, M. Blum, N. J. Hopper, and J. Langford, "CAPTCHA: Using hard ai problems for security," in *Proc. 22nd Int. Conf. Theory Appl. Cryptograph. Techn. (EUROCRYPT)*, Berlin, Germany: Springer-Verlag, 2003, pp. 294–311.
[35] S. King, "Primecoin: Cryptocurrency with prime number proof-of-work," Primecoin, Tech. Rep., 2013. [Online]. Available: http://primecoin.io/bin/primecoin-paper.pdf
[36] P. Rogaway and T. Shrimpton, "Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance," in *Fast Software Encryption*, B. Roy and W. Meier, Eds. Berlin, Germany: Springer, 2004, pp. 371–388.
[37] "Announcing request for candidate algorithm nominations for a new cryptographic hash algorithm (SHA-3) family," NIST, Gaithersburg, MD, USA, Tech. Rep. E7-21581, 2007.
[38] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche, "The Keccak SHA-3 submission," NIST, Gaithersburg, MD, USA, Tech. Rep. Version 3, 2011. [Online]. Available: https://keccak.team/files/Keccak-submission-3.pdf
[39] J.-P. Aumasson, W. Meier, R. C.-W. Phan, and L. Henzen, *The Hash Function BLAKE*. New York, NY, USA: Springer, 2015.
[40] D. Gligoroski, V. Klima, S. J. Knapskog, M. El-Hadedy, and J. Amundsen, "Cryptographic hash function blue midnight wish," in *Proc. 1st Int. Workshop Secur. Commun. Netw.*, May 2009, pp. 1–8.
[41] P. Gauravaram *et al.*, "Grøstl—A SHA-3 candidate," NIST, Gaithersburg, MD, USA, Tech. Rep. Version no. 2.0, 2008. [Online]. Available: http://www.groestl.info
[42] H. Wu, "The hash function JH," NIST, Gaithersburg, MD, USA, Tech. Rep. 42, 2011. [Online]. Available: https://www3.ntu.edu.sg/home/wuhj/research/jh/jh_round3.pdf
[43] N. Ferguson *et al.*, "The Skein hash function family," NIST, Gaithersburg, MD, USA, Tech. Rep. Version 1.3, 2010. [Online]. Available: http://www.skein-hash.info/sites/default/files/skein1.3.pdf
[44] C. D. Canniére, H. Sato, and D. Watanabe, "Hash function Luffa: Specification," NIST, Gaithersburg, MD, USA, Tech. Rep., 2008. [Online]. Available: https://ehash.iaik.tugraz.at/uploads/e/ea/Luffa_Specification.pdf
[45] D. J. Bernstein, "CubeHash specification (2.B.1)," NIST, Gaithersburg, MD, USA, Tech. Rep., 2009. [Online]. Available: http://cubehash.cr.yp.to/submission2/spec.pdf
[46] E. Biham and O. Dunkelman, "The SHAvite-3 hash function," NIST, Gaithersburg, MD, USA, Tech. Rep., 2009. [Online]. Available: http://www.cs.technion.ac.il/~orrd/SHAvite-3/Spec.01.02.09.pdf
[47] G. Leurent, "The SIMD hash function," NIST, Gaithersburg, MD, USA, Tech. Rep. [Online]. Available: https://who.rocq.inria.fr/Gaetan.Leurent/simd.html
[48] R. Benadjila *et al.*, "SHA-3 proposal: ECHO," NIST, Gaithersburg, MD, USA, Tech. Rep., 2010. [Online]. Available: http://crypto.rd.francetelecom.com/echo/doc/echo_description_2-0.pdf
[49] Ö. Küçük, "The hash function Hamsi," NIST, Gaithersburg, MD, USA, Tech. Rep., 2009.
[50] S. Halevi, W. E. Hall, and C. S. Jutla, "The hash function 'Fugue,'" NIST, Gaithersburg, MD, USA, Tech. Rep., 2009. [Online]. Available: https://researcher.watson.ibm.com/researcher/files/us-csjutla/fugue_Oct09.pdf
[51] E. Bresson *et al.*, "Shabal, a submission to NIST's cryptographic hash algorithm competition," NIST, Gaithersburg, MD, USA, Tech. Rep., 2008.
[52] *The Firs X11 Mining ASIC iBeLink DM384M ASIC DASH Miner*. Accessed: Sep. 19, 2018. [Online]. Available: https://cryptomining-blog.com/7117-the-first-x11-mining-asic-ibelink-dm384m-asic-dash-miner/
[53] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 2, pp. 203–215, Feb. 2007.
[54] T. Pruvot. *CPUMiner-Multi*. Accessed: Sep. 27, 2018. [Online]. Available: https://github.com/tpruvot/cpuminer-multi
[55] *Ccminer.org—Crypto Currencies Mining Solutions*. Accessed: Sep. 27, 2018. [Online]. Available: http://ccminer.org/
[56] K. Gaj, E. Homsirikamol, and M. Rogawski, "Fair and comprehensive methodology for comparing hardware performance of fourteen round two SHA-3 candidates using FPGAs," in *Cryptographic Hardware and Embedded Systems*, S. Mangard and F.-X. Standaert, Eds. Berlin, Germany: Springer, 2010, pp. 264–278.
[57] P. S. Barreto and V. Rijmen, "The WHIRLPOOL hashing function," Tech. Rep., 2003.
[58] "Secure hash standard (SHS)," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep. FIPS PUB 180-4, 2015.
[59] D. Sanchez, G. Michelogiannakis, and C. Kozyrakis, "An analysis of on-chip interconnection networks for large-scale chip multiprocessors," *ACM Trans. Archit. Code Optim.*, vol. 7, no. 1, pp. 4:1–4:28, May 2010. [Online]. Available: http://doi.acm.org/10.1145/1756065.1736069
[60] W. W. L. Fung, I. Sham, G. Yuan, and T. M. Aamodt, "Dynamic warp formation and scheduling for efficient GPU control flow," in *Proc. IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2007, pp. 407–420.

HYUNGMIN CHO received the B.S. degree in computer science engineering from Seoul National University, South Korea, in 2005, and the M.S. and Ph.D. degrees in electrical engineering from Stanford University in 2010 and 2015, respectively. He was a Research Scientist at Intel Labs, Santa Clara, CA, USA. He is currently an Assistant Professor with the Department of Computer Engineering, Hongik University, South Korea. His research interests include reliable computer systems and accelerator architectures for high-performance computing.