Recommended Paper

A Proof of Work based on Preimage Problem of Variants of SHA-3 with ASIC Resistance

Takaki Asanuma[1,†1,a)] Takanori Isobe[1,†1,b)]

Received: March 8, 2021, Accepted: October 8, 2021

Abstract: Hashcash, which is a Proof of Work (PoW) of bitcoin, is based on a preimage problem of hash functions of SHA-2 and RIPEMD. Since these hash functions employ the Merkle-Damgard (MD) construction, a preimage can be found with a negligible amount of memory. It is well known that such calculations can be speeded up by ASIC, and this causes a serious problem from the so-called 51% attack by dedicated ASIC mining pools. To address this issue, we propose a new PoW scheme based on a preimage problem of variants of SHA-3. Unlike SHA-2 and RIPEMD, SHA-3 adopts a sponge construction as an underlying domain extension algorithm. This difference allows us to make the problem of finding a preimage very memory-consuming calculations by properly choosing parameters of sponge functions. As a result, our scheme can achieve ASIC resistance by using SHA-3 for Hashcash.

Keywords: Bitcoin, Proof of Work, Hashcash, SHA-3, Preimage attack, Sponge Construction

1. Introduction

1.1 Background

An essential issue for cryptocurrencies is how to prevent duplicate payments. A simple solution is to trade through a trusted third-party central institution such as a bank. However, this solution has the potential risk that the entire system may be shut down due to an attack on the central server or the central authority itself might attempt malicious behavior. To mitigate these risks, cryptocurrencies with a decentralized system, e.g., Bitcoin and Ethereum, are based on blockchain. The basic idea behind blockchain is to consistently retain the correct transaction data. Since the transaction data is stored in a form that everyone can see and verify its correctness as a distributed ledger, malicious users cannot manipulate the data. To maintain this property, the blockchain needs to be secure against so-called 51% attacks in which malicious users monopolize more than half of the computational resources.

Many people believe that monopolizing 51% of the computational power is impractical. However, there have actually been cases of 51% attacks on cryptocurrencies such as Bitcoin Gold [1]. The cryptocurrency used an algorithm called Proof of Work (PoW) to prevent the attack, but this did not work well enough.

PoW is a consensus algorithm in blockchain [2]. In order to prevent 51% attacks, PoW should minimize the difference in computational advantage between dedicated hardware (ASIC) and ordinary CPUs as much as possible. A solution for this purpose is a memory-hard function which consumes a large amount of memory accesses [3], [4], [5]. Since memory-consuming computations cannot be sufficiently sped up by ASIC, the advantage of mining pools equipmed with dedicated ASIC over ordinary CPUs is limited. If time and memory are a steep trade-off, then the person solving the PoW must use a large amount of memory to reduce time complexity. Therefore, a *steep time-memory tradeoff* such that the amount of computational complexity exponentially increases when the amount of memory is reduced is very important for ASIC resistance [6].

Hashcash, which is a PoW of bitcoin, is based on a preimage problem of hash functions of SHA-2 and RIPEMD [7]. Since these hash functions employ the Merkle-Damgard (MD) construction, a preimage can be found with negligible memory by a brute force search. Due to lack of a *steep time-memory tradeoff*, Hashcash does not have a resistance against the 51% attack by mining pools equipping ASIC.

1.2 Motivation

Our main purpose in this paper is to explore hash functions whose preimage problem requires a large amount of memory. By replacing such a hash function with SHA-2 and RIPEMD as underlying hash functions of Hashcash, we can develop a new PoW that has a *steep time-memory tradeoff*.

Equiphash [8] and MTP [6] are recently proposed PoW. These have the property of a steep time-memory tradeoff. However, it cannot be a drop-in-replacement for SHA-2 and RIPEMD of Hashcash, because these are not based on a problem of preimage finding of cryptographic hash functions.

[1] University of Hyogo, Kobe, Hyogo 650–0047, Japan
[†1] Presently with University of Hyogo
[a)] takaki.asanuma@gmail.com
[b)] takanori.isobe@ai.u-hyogo.ac.jp

The preliminary version of this paper was presented at the Computer Security Symposium (CSS 2020), October 2020. The paper was recommended to be submitted to the Journal of Information Processing (JIP) by the Program Chair of CSS 2020.

1.3 Our Contribution

In this paper, we propose a new PoW scheme based on a preimage problem of variants of SHA-3, which employs a sponge construction [9]. There are two methods to search the preimages on the sponge construction: a brute force search and a meet-in-the-middle (MITM) approach unlike the MD construction. Although the brute force search is feasible without memory, the MITM approach requires a large amount of memory to store intermediate values. Our idea for realizing a memory-consuming PoW is to make the MITM much more time-efficient than the brute force search by properly choosing parameters of the sponge function. In this case, mining pools cannot avoid choosing the memory-consuming MITM approach as an efficient algorithm of PoW.

Furthermore, we consider two types of memory-efficient algorithms for MITM: Floyd's MITM [10] and Nikolic-Sasaki MITM [11], and uncover parameters in order that these algorithms cannot be efficiently applied.

As a result, our PoW scheme achieves a *steep-time-memory tradeoff*, or namely the time complexity exponentially increases when the memory is reduced. For example, an instantiation of our PoW can be executed with a time complexity of 2^{96} and 2^{32} memory. If the amount of memory is reduced $\frac{1}{2^{16}}$ times, the time complexity increases to 2^{112} which is 2^{16} times higher. Thus, our scheme retains the idea of Hashcash, but achieves ASIC resistance by using SHA-3 instead of SHA-2 and RIPEMD. Besides, preimage problems of hash functions have been significantly evaluated by symmetric-key cryptography communities over the past 20 years. Thus, we can have high confidence in the difficulty of these problems. It is very important from the point of the *optimization-free* requirement. Indeed, several variants of the first version of MTP [6] were broken [12]. Finally, we show that our schemes satisfy other PoW requirements.

1.4 Organization of this Paper

In Section 2, we describe an overview of PoW, its security requirements and the problems of Hashcash. In Section 3, we explain the details of sponge functions and the possible preimage attacks. In Section 4, we reveal the parameters for the sponge function for realizing the requirement of a *steep time-memory tradeoff*. Section 5 gives instantiations of our schemes by properly adjusting the parameters. Section 6 discusses the other security requirements, and compares these with other candidates. Section 7 provides the conclusion.

2. Proof of Work

This section explains the purpose, definition, and security requirements of PoW and then discusses the problem of existing hash-based schemes.

2.1 Purpose of PoW

PoW is a problem that requires a certain amount of time to solve and is utilized as a consensus scheme for decentralization. For example, Bitcoin utilizes a preimage problem of hash functions called Hashcash as an underlying PoW. Specifically, miners/prover should find a preimage of a digest value of hash function whose first d bits are all zero as a solution for PoW. This is the preimage problem of hash functions of SHA-2 or RIPEMD.

In the blockchain, after the prover solves the PoW problem, the transaction data is connected to the blockchain. Entities can see the data on the blockchain, however, it is very difficult to tamper with it, because blocks consisting of multiple transaction data are affected by the previous block. To modify a piece of data, the attacker must modify not only the piece of data but also all subsequent blocks. In other words, the attacker needs to solve all corresponding PoW.

If the miner does not have more than 51% of all computing resources, such computation is not possible in principle. Thus, the blockchain using PoW realizes electronic commerce on peer-to-peer (P2P) networks without the need for trusted third parties.

2.2 Definition of PoW

As defined in Ref. [8], PoW has defined a problem

$$\mathcal{P} : \mathcal{R} \times \mathcal{I} \times \mathcal{S} \to \{\text{true}, \text{false}\}.$$

as hardcore predicates, where $\mathcal{R}$ is the set of parameters that determine the hardness, $\mathcal{I}$ is the set of inputs conforming to $\mathcal{R}$ and $\mathcal{S}$ is the set of possible solutions. We assume that there is an algorithm (or a family of algorithms) $\mathcal{A}_R$ that solves $\mathcal{P}_R$ on any I, or in other words finds S such that $\mathcal{P}(R, I, S) = true$.

A PoW scheme based on $\mathcal{P}$ is an interactive protocol which operates as follows:

(1) The Verifier sends a challenge input $I \in \mathcal{I}$ and parameters $R \in \mathcal{R}$ to the Prover.
(2) The Prover finds solution $\mathcal{S}$ such that $\mathcal{P}(R, I, S) = true$ and sends it to the Verifier.
(3) The Verifier computes $\mathcal{P}(R, I, S)$.

A non-interactive version (e.g., for cryptocurrencies) can be derived easily. In this setting, I contains some public value (last block hash in Bitcoin) and prover's ID. The prover publishes S so that every party can verify the proof.

2.3 Security Requirements for PoW

Biryukov and Khovratovich propose the seven security requirements of problem $\mathcal{P}$ and algorithm $\mathcal{A}$ [6].

(1) Amortization-free: Producing q outputs for $\mathcal{P}$ should be q times as expensive.
(2) Asymmetry: The solution must be short enough and verified quickly using little memory in order to prevent DoS attacks on the verifier.
(3) Steep time-memory Tradeoff: The time-space tradeoffs must be steep to prevent any price-performance reduction.
(4) Flexibility: The time and memory parameters must be tunable independently to sustain constant mining rate.
(5) Optimization-free: To avoid a clever prover obtaining an advantage over the others the advertised algorithm must be the most efficient algorithm to date.
(6) Progress-free: The algorithm must be *progress-free* to prevent centralization: the mining process must be stochastic so that the probability for finding a solution grows steadily with time and is non-zero for small time periods.
(7) Parallel-unfriendly: Parallelized implementations must be limited by the memory bandwidth.

Among these requirements, (3) is the most important for the ASIC resistance since it requires a large amount of memory and random memory access.

2.4 Problems of Hashcash

Hashcash, which is a PoW of the bitcoin, is based on a preimage problem of hash functions of SHA-2 and RIPEMD. Since these hash functions employ the Merkle-Damgard (MD) construction, a preimage can be found with a negligible amount of memory by a brute force search. Due to lack of the property of the steep time-memory tradeoff, Hashcash does not have resistance against so-called 51% attack by mining pools equipped with ASIC. In this paper, we explore hash functions requires to search the specific preimages with a large amount of memory. By replacing such function with SHA-2 and RIPEMD as underlying hash functions of Hashcash, we develop a new PoW that has a steep time-memory tradeoff.

3. Preimage Problem of Sponge Function

To address the problem of Hashcash, this paper focuses on variants of SHA-3 which are based on the sponge construction while SHA-2 and RIPEMD are based on an MD structure as a domain extension algorithm. In this section, we first look into searching the preimage of the sponge function. We then introduce the relationship between the amount of memory consumption and each parameter of several preimage attacks on sponge functions.

3.1 Sponge Function

The sponge function was introduced by Bertoni et al. as a new way of constructing a hash function based on public permutations [13]. The details of the sponge construction are shown in **Fig. 1**: The internal state S of the t-bits ($t = c + r$), consisting of the capacity of the c-bit and the bit-rate of the r-bit, is first initialized with a fixed value. After dividing the message into r-bit chunks with appropriate padding, we take the exclusive OR result of all r-bit message chunks and the bit-rate portion of the internal state.

This input process is called the "absorbing process". Once all the message chunks have been processed in this absorbing phase, the r bits are extracted from the bit-rate portion of the internal state and a substitution process is applied. Finally, these processes are repeated to complete the output of n bits. This output process is called the "squeezing process".

We use the internal permutation P modeled as a randomly chosen permutation, and then a sponge function that has been proven to be indifferentiable from a random oracle up to $2^{c/2}$ calls to P [17]. The sponge function offers collision resistance (as a truncated random oracle would) for any output length n smaller than the capacity c and (2nd) preimage resistance for any n smaller than half of c. When internal substitution is modeled as a randomly selected permutation, the sponge function is indistinguishable from a random oracle with up to $2^{c/2}$ calls to the substitution process.

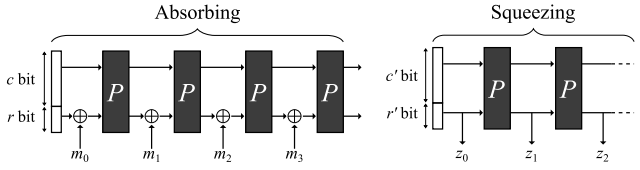

Fig. 1 Details of the sponge function.

Guo et al. extended the framework of the sponge construction in which r' is extracted during each iteration of the squeezing process, i.e., r' can be different from the bit-rate r in absorbing phase [14], [15].

PoW should satisfy the *optimization-free* requirement that the miner must use an free algorithm from further optimization. Generally speaking, it is difficult for schemes based on symmetric-key cryptography to determine whether it satisfies *optimization-free* requirement because the security of symmetric-key cryptography is not reduced on mathematical problems, unlike public-key cryptography. SHA-3 won the NIST hash competition, and no vulnerability has not been found so far despite considerable cryptanalytic efforts over 10 years. Thus, in this paper, we assume that "Sponge function" is SHA-3.

3.2 Preimage Problem of the Sponge Function

For a sponge function with capacity c, bit-rate r and r' and n-bit output, the security of collision resistance, preimage resistance, and second preimage resistance are estimated as follows [15].

$$\begin{aligned}&\text{Collision: } \min\{2^{n/2}, 2^{c/2}\}\\ &\text{2nd Preimage: } \min\{2^{n}, 2^{c/2}\}\\ &\text{Preimage: } \min\{2^{\min(n,t)}, \max\{2^{\min(n,t)-r'}, 2^{c/2}\}\}\end{aligned} \tag{1}$$

It should be noted here that the time complexity of the preimage attack is different from that of the MD construction such as RIPEMD and SHA-2, and depends on many parameters of sponge functions.

Specifically, there are two approaches for preimage attacks on the sponge function such that the brute force attack and the meet-in-the-middle (MITM) attack as shown in **Fig. 2**.

3.2.1 Brute Force

The term of $2^{\min\{n,t\}}$ is the time complexity of the exhaustive search for an n-bit output or a t-bit internal state. In this paper, we do not consider the case where n is more than 512 and the internal state length of t of SHA-3 is 1600. Thus, $\min\{n, t\} = n$ always holds. Note that the brute force attack is feasible with a negligible amount of memory.

3.2.2 MITM

The term of $\max\{2^{\min\{n,t\}-r'}, 2^{c/2}\}$ is the time complexity in MITM. In the MITM, $2^{c/2}$ intermediate values are obtained in the forward computation $f(x)$ and these are stored in memory. Then $2^{c/2}$ attempts in the backward computations $g(x)$ are required to check a collision in the intermediate values as shown in Fig. 2. MITM requires $2^{c/2}$ of time complexity and $2^{c/2}$ memory.

The term of $2^{\min\{n,t\}}$ comes from the cost for inverting the squeezing process, which is called multiblock constrained-input constrained-output problem [15]. The best known generic attack to solve this problem requires $2^{n-r'}$ computations when $t > n$ [15]. As discussed above, $\min\{n, t\} = n$ always holds. Thus, Eq. (1) is rewritten as follows.

$$\text{Preimage: } \min\{2^{n}, \max\{2^{n-r'}, 2^{c/2}\}\} \tag{2}$$

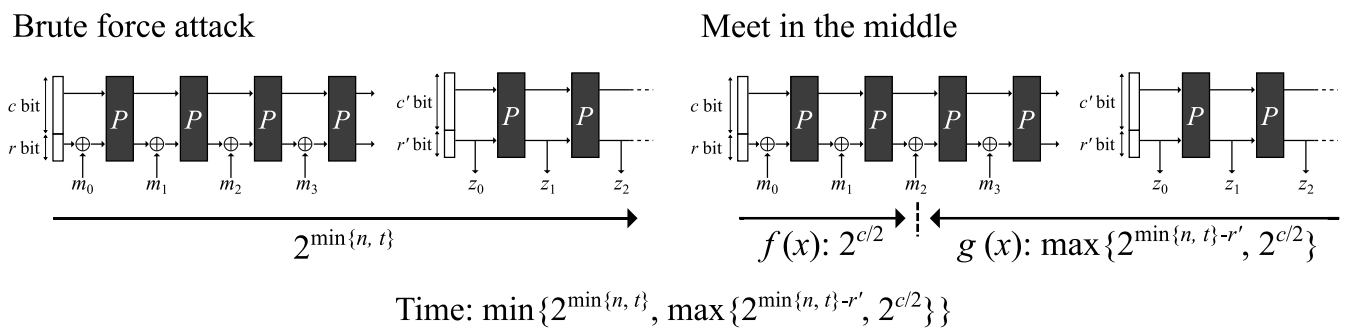

Fig. 2 Time complexity required by the preimage attack of the sponge function.

MITM follows the tradeoff $TM = N$, where T and M are time and memory complexities and N is a size of the matching point i.e., 2^c, where $T \geq 2^{c/2}$. By reducing the memory, the time complexity increases, and vice versa. The case of $T = 2^{c/2}$ is optimal for time complexity.

Thus, our basic idea is to make the mining pool choosing MITM in an efficient way by properly selecting the parameters so that MITM is much more time-efficient than the brute force attack.

3.3 Memory-Efficient MITM

There are two memory-efficient approaches for MITM: Floyd's collision finding [10] and Nikolic-Sasaki approach [11].

3.3.1 Floyd's Collision Finding

In MITM, the adversary finds a collision in two $2^{n/2}$ sets of intermediate values which are computed by forward computation $f(x)$ and backward computation $g(x)$, respectively, as shown in Fig. 2. Floyd's algorithm can be used to find a collision between the two functions by interleaving the calls to $f(x)$ and $g(x)$ during the detection of the cycle [10]. Floyd's algorithm is based on a selection function which evaluates either $f(x)$ or $g(x)$ with equal probability. Thus, a collision is an actual one for $f(x)$ and $g(x)$ with a probability $1/2$ and consequently, the search has to be repeated twice. Using Floyd's cycle-finding algorithm for MITM [10], the time complexity is almost $2^{c/2}$ with negligible memory.

3.3.2 Nikolic-Sasaki Approach

Nikolic and Sasaki proposed the memory-efficient MITM in the case where the computational cost $g(x)$ is R times higher than $f(x)$ (called unbalanced MITM) [11]. They utilize Floyd's algorithm and van Oorschot-Wiener [16]. In this paper, we call this MITM "NS-MITM" and the standard MITM "S-MITM". According to [11], if the memory is represented as 2^m, the time required for NS-MITM is estimated as follows.

$$\text{Time: } 2^{(n+m)/2} + R \times 2^{(n-m)/2} \tag{3}$$

$$\text{Memory: } 2^m$$

$$T^2M = R^2N \ (M \leq R)$$

NS-MITM can reduce the amount of memory requirements compared to S-MITM when memory size satisfies the equation of $M < N/R^2$.

4. Finding Parameters for SHA-3 based PoW

In this section, we explore parameters of the sponge function to search the preimage with a large amount of memory to efficiently solve, namely the memory-efficient approaches have no advantage over the one using a large amount of memory with respect to time complexity. In this case, mining pools equipped with dedicated ASIC cannot avoid choosing the memory-consuming MITM approach as an efficient algorithm of PoW.

4.1 Criteria for Parameter Selection

Our aim is to develop a new PoW based on a preimage problem that has a steep time-memory tradeoff for ASIC resistance. To achieve this purpose, we choose the parameters of the sponge function such that S-MITM, which is the most memory-consuming approach, is much more time-efficient than others in any time-memory tradeoff point of $N = TM$.

Specifically, we will uncover the parameters of the sponge function that satisfy the following criteria.

- **(A)** S-MITM is always more time-efficient than the brute force attack.
- **(B)** S-MITM is always more time-efficient than Floyd's algorithm [10].
- **(C)** S-MITM is always more time-efficient than NS-MITM [11].

4.2 A Parameter for (A)

The brute force attack requires 2^n computations with negligible memory while S-MITM follows the tradeoff $TM = N$ for $T \geq 2^{c/2}$. When $M = 0$, time complexity for S-MITM becomes the maximum value of $T = 2^c$.

Thus, the parameters must satisfy the following equation.

$$c \leq n$$

In this case, the time complexity of S-MITM is lower than for the brute force attack in any time-memory tradeoff point.

In addition, the term of $2^{n-r'}$ of Eq. (2) always satisfies the equation of $2^{n-r'} \leq 2^n$ when $c < 2n$. Thus, Eq. (2) is expressed as follows.

$$\text{Preimage: } \min\left\{2^n, \max\left\{2^{n-r'}, 2^{c/2}\right\}\right\} = \max\left\{2^{n-r'}, 2^{c/2}\right\}$$

4.3 A Parameter for (B)

By Floyd's algorithm, S-MITM is feasible with time complexity of $2^{c/2}$ and negligible memory [10]. As discussed by Nikolic and Sasaki [11], S-MITM is available only when the computational cost of $f(x)$ and $g(x)$ in MITM is equal (called balanced

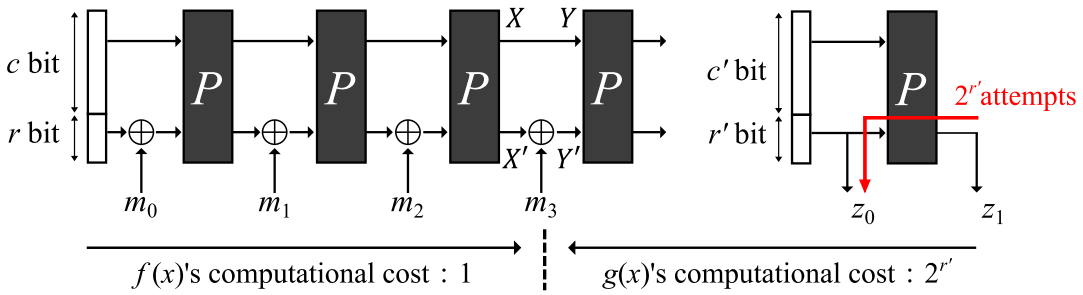

Fig. 3 Computation costs for sponge functions in the unbalanced MITM.

MITM).

To satisfy (B), we make S-MITM an *unbalance MITM* in which the computational cost of $g(x)$ is R times higher than $f(x)$. For the unbalanced MITM, the required time complexity and memory usage are estimated as follows [11].

Time: $\sqrt{RN}$

Memory: $\sqrt{N/R}$

It also follows a tradeoff of $TM = N$ where ($\sqrt{RN} \leq T$).

On the other hand, the unbalanced MITM using Floyd's algorithm requires $R \times 2^{c/2}$ time complexity because the cost of $g(x)$ is R times higher.

4.3.1 Unbalance MITM for Sponge Function

We show how to realize the unbalance MITM in the sponge function. For the sponge function, the computational cost of $g(x)$ can be different from that of $f(x)$ by choosing parameters of n and r' and fixing the meet point as the following assumptions.

Assumption 1: Parameters of n and r' satisfy the equation of $r' \leq n \leq 2r'$.

Assumption 2: The meet point of MITM is fixed to the end of the absorbing phase.

Under the assumption 1, an n-bit digest consists of two r'-bit output values z_0 and z_1 in the squeezing phase. In the backward computation $g(x)$, the miner starts the computation of P^{-1} with the value of z_1. After that, r' bits of the result of P^{-1} must be the same as z_0. Since this probability is estimated as $2^{-r'}$, the computational cost to correctly compute in the backward direction is $2^{r'}$ as shown in **Fig. 3**.

Note that if the meet point of MITM is different from the end of the absorbing process, i.e., the middle of the absorbing phase, it cannot be the unbalance MITM. This is because once the state of the end of absorbing phase is computed in the backward computation of the squeezing phase, we can mount the balanced MITM in the middle of the absorbing phase. On the other hand, under the assumption 2, to compute the matching state in the end of absorbing phase, we have to start the computation from z_1 and the result of P^{-1} must be the same as z_0 in each computation. Thus, each backward computation requires $2^{r'}$ executions of P^{-1}.

4.3.2 How to Fix the Meetpoint in PoW

To make it the unbalanced MITM problem, we should impose a condition such that the matching point is the end of the absorbing process (Assumption 2). This is achieved by limiting the degree of freedom in message values, e.g., some message blocks are predetermined or the number of message block sizes is fixed as a problem of PoW.

For example, in the case where the size of the matching state is c bits, if the degree of freedom of message blocks in the absorbing phase is less than $2^{c/2}$, the adversary has to mount MITM in which the matching point is the end of the absorbing phase as shown in Fig. 3.

4.3.3 Selecting Parameters

Under the assumptions 1 and 2, we can realize the NS-MITM. In this case, Floyd's algorithm is not directly applied. As shown in Fig. 3, the cost of $g(x)$ is $R(= 2^{r'})$ times higher than that of f(x) in the unbalanced MITM. The unbalanced MITM using Floyd's algorithm requires a time complexity of $R \times 2^{c/2} (= 2^{r'} \times 2^{c/2})$ [11].

To satisfy (B), we need to choose parameters so that S-MITM is always more time-efficient than Floyd's algorithm. The maximum value of time complexity of S-MITM in unbalanced MITM is 2^c in the memoryless case, since the unbalanced MITM also follows the trade-off $TM = N$. Therefore, parameters must satisfy the relation of $2^c \leq 2^{r'} \times 2^{c/2}$, namely $2^{\frac{c}{2}} \leq 2^{r'}$.

In summary, the parameters must satisfy the following equation.

$$\frac{c}{2} \leq r'$$
$$r' \leq n \leq 2r' \text{ (Assumption1)}$$

4.4 A Parameter for (C)

In the unbalance MITM, NS-MITM can find a preimage with more memory efficiency than S-MITM in several parameters [11]. According to Ref. [11], when memory size satisfies the equation of $M < N/R^2 = 2^c/R^2$, NS-MITM is more time-efficient than S-MITM. If we choose $R \geq 2^{c/2}$, there is no point where NS-MITM is more efficient than S-MITM. Therefore, the parameter for (C) is as follows.

$$\frac{c}{2} \leq r'$$

4.5 Summary of Parameter Selection

The parameters for (A), (B), and (C) are as follows.

$$c \leq n$$
$$\frac{c}{2} \leq r'$$
$$r' \leq n \leq 2r'$$

Among these, we want to make sure that the PoW requires as much memory as possible. Since S-MITM requires $\sqrt{N/R}$ of memory in the unbalanced MITM problem, by decreasing the value of R, we can increase the memory consumption. Thus, we

choose $R = 2^{r'} = 2^{c/2}$ under the condition of $\frac{c}{2} \leq r'$, and $c = n$ under the condition of $c \leq n$.

The proposed PoW is a preimage recovery problem of SHA-3. At least four approaches to the preimage recovery problem of SHA-3 are known: brute force attack, S-MITM, NS-MITM, and Floyd's algorithm. Among these, S-MITM is the only one in which time and memory are steep tradeoffs and can consume a large quantity of memory. Thus, if we appropriately choose the parameters and S-MITM is the most advantageous in terms of time complexity, the PoW will be a memory-hard. In summary, the parameters must meet the following conditions to satisfy the steep time-memory tradeoff and make the PoW ASIC-resistant.

$$n = c = 2r'$$

5. PoW based on Variants of SHA-3

In this section, we propose a PoW based on variants of SHA-3 that achieves a steep time-memory tradeoff. Our PoW is based on Hashcash in which underlying hash functions of RIPEMD and SHA-2 are replaced with variants of SHA-3 with parameters of $n = c = 2r'$.

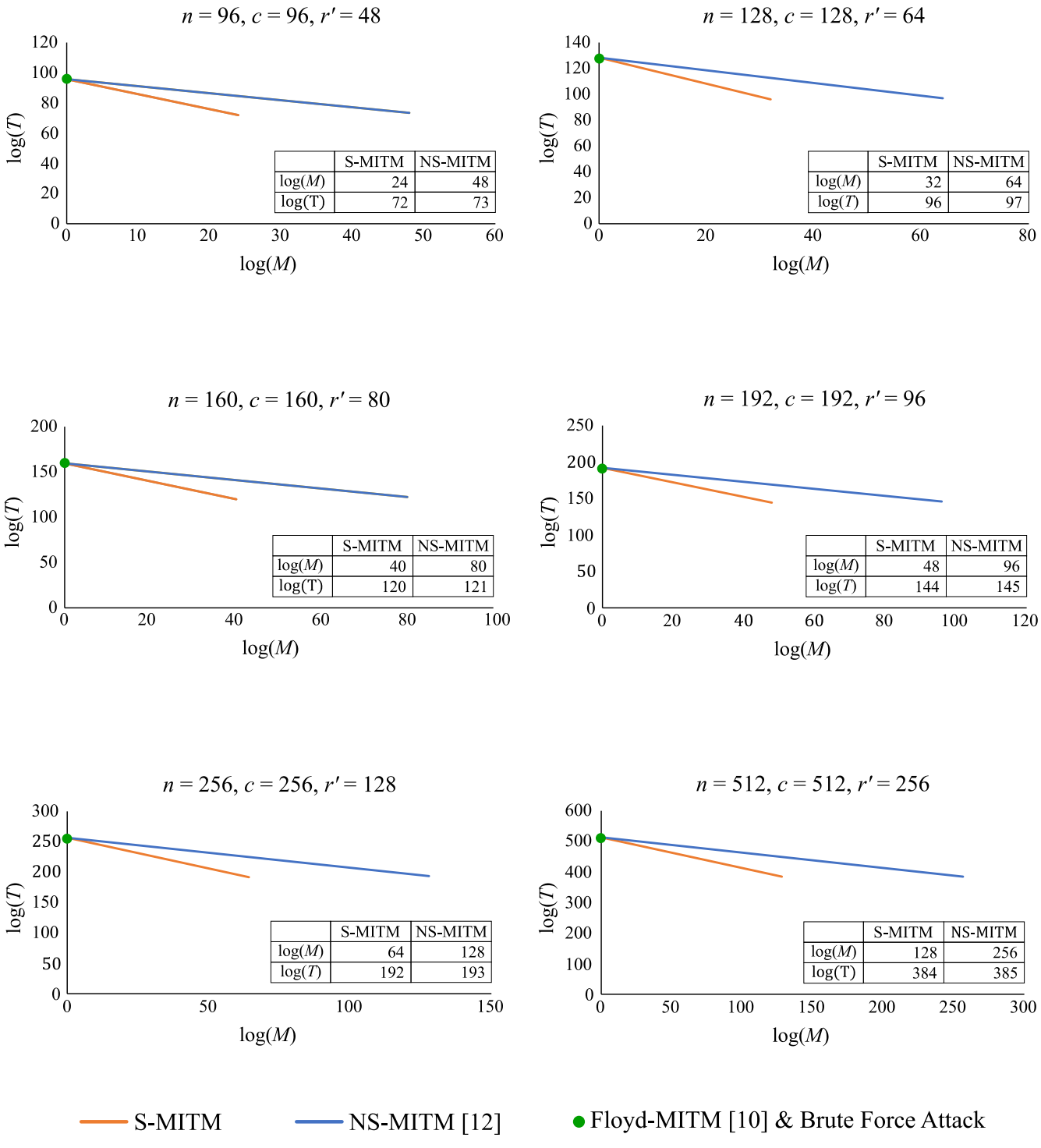

Fig. 4 Comparison of Time and Memory for $n = 96, 128, 160, 192, 256, 512$.

Table 1 Time & Memory ($n = 96$).

$\log(M)$	$\log(T)$	
	S-MITM	NS-MITM [11]
0	96	96
6	90	93
12	84	90
18	78	87
24	72	84

Table 2 Time & Memory ($n = 128$).

$\log(M)$	$\log(T)$	
	S-MITM	NS-MITM [11]
0	128	128
8	120	124
16	112	120
24	104	116
32	96	112

Table 3 Time & Memory ($n = 160$).

$\log(M)$	$\log(T)$	
	S-MITM	NS-MITM [11]
0	160	160
10	150	155
20	140	150
30	130	145
40	120	140

Table 4 Time & Memory ($n = 192$).

$\log(M)$	$\log(T)$	
	S-MITM	NS-MITM [11]
0	192	192
12	180	186
24	168	180
36	156	174
48	144	168

Table 5 Time & Memory ($n = 256$).

$\log(M)$	$\log(T)$	
	S-MITM	NS-MITM [11]
0	256	256
16	240	248
32	224	240
48	208	232
64	192	224

Table 6 Time & Memory ($n = 512$).

$\log(M)$	$\log(T)$	
	S-MITM	NS-MITM [11]
0	512	512
32	480	496
64	448	480
96	416	464
128	384	448

5.1 Instantiations

We give instantiations for $n = c = 96, 128, 160, 192, 256, 512$. **Figure 4** shows the time complexity and memory consumption of S-MITM, NS-MITM, Floyd's-MITM and brute force attack for each variant. These satisfy the following properties.

- $M = 0$: All methods require time complexity of 2^c $(= 2^n)$.
- $M > 0$: S-MITM is always more efficient than NS-MITM as shown in **Tables 1–6**. For example, for $n = 128$, S-MITM requires times complexity of 2^{96} and 2^{32} memory usage, while NS-MITM requires time complexity of 2^{97} even if 2^{64} memory is used.

Thus, S-MITM is much more time-efficient than the other methods for any point, and mining pools cannot avoid choosing the memory-consuming MITM approach as an efficient algorithm of PoW.

5.2 Steep Time-Memory Tradeoff

The required time complexity of S-MITM increases if the amount of memory usage is reduced, following the relation of $TM = N$. If memory of $\sqrt{N/R}$ is used, time complexity can be reduced to $\sqrt{RN}$, which is optimal with respect to time complexity. For example, for $n = 128$, if the memory usage of S-MITM is reduced from 2^{32} to 2^{16}, the time complexity increases from 2^{96} to 2^{112}. Thus, our scheme satisfies a steep time-memory tradeoff.

6. Discussion of Other Properties

In this section, we discuss whether our scheme satisfies other properties in addition to the steep time-memory tradeoff, and then compare our schemes with existing schemes of Hashcash, Equihash [8] and Merkle Tree Proof [6].

6.1 Amortization-Free

To satisfy the *amortization-free* requirement, the computational cost to find q solutions should be q times higher than that of finding one solution. For S-MITM, the intermediate value of the squeezing phase or the absorbing phase are stored in a table and might be re-used in the next proofs. This can easily be avoided by randomly changing target digest values or put a condition of message values in each time. These allow our scheme to achieve the *amortization-free* requirement.

6.2 Asymmetry

To satisfy the *asymmetry* requirement, the proofs require a certain amount of time and memory usage, while the verification needs only a few resources. Our scheme requires $\sqrt{NR}$ time complexity and $\sqrt{N/R}$ memory for the proof. However, the verifier only hashes the input sent by the prover and check whether the digest satisfies the condition. Thus, our scheme fully satisfies the *asymmetry* requirement.

6.3 Flexibility

To satisfy the *flexibility* requirement the time and memory must be independently adjustable. The time and memory requirements of our scheme are shown in Fig. 4. This scheme satisfies the flexiblity requirement because selecting the parameters allows us to specify the required time and memory.

6.4 Progress-Free

To satisfy the *progress-free* requirement, the number of solutions found in a given time by mining must approximate the Poisson process under a Poisson distribution. Since SHA-3 is indifferentiable from a random oracle, it can also approximate a Poisson process [17]. Therefore our scheme achieves the *progress-free* requirement.

6.5 Optimization-Free

To satisfy the *optimization-free* requirement, the miner must use an algorithm that is free from further optimization. Generally speaking, it is difficult for schemes based on symmetric-key

Table 7 Security requirements satisfied by our and other schemes.

	Hashcash	Equihash [8]	MTP [6]	Our Scheme
Amortization-free	Yes	Yes	Yes	Yes
Asymmetry	Yes	Yes	Yes	Yes
Flexibility	No	Yes	Yes	Yes
Progress-free	Yes	Yes	Yes	Yes
Steep time-memory tradeoff	No	Yes	Yes	Yes
Parallelism constraint	No	Yes	Yes	Yes
Optimization-free	Yes	Yes	Insufficient	Yes

cryptography to determine whether *optimization-free* is satisfied because the security of symmetric-key cryptography is not reduced on mathematical problems, unlike public-key cryptography. Indeed, some attacks on the first version of MTP have been proposed [12]. On the other hand, SHA-3 won the NIST hash competition, and no vulnerability has not been found so far despite considerable cryptanalytic efforts over 10 years. Therefore, we consider that our scheme based on SHA-3 sufficiently satisfies the *optimization-free* requirement. On the other hand, there is a possibility that other more effective attacks than the Standard MITM will be discovered, so further discussion is necessary.

6.6 Parallel-unfriendly

As discussed in Refs. [6], [8], parallelism is restricted in our scheme due to memory bandwidth growth in parallel implementations.

6.7 Comparison with Existing Schemes

Table 7 compares our scheme with Hashcash, Equiphash [8] and MTP [6], while Hashcash does not satisfy the properties of *flexibility*, steep time-memory trade off and parallelism. Our schemes satisfy the required properties for POW just as with Equiphash [8].

Importantly, our goal is to achieve ASIC resistance. Equiphash and MTP do not achieve this goal since these are not based on preimage problems of hash functions.

7. Conclusion

In this paper, we propose a new PoW scheme based on the preimage problem of variants of SHA-3. Unlike SHA-2 and RIPEMD, SHA-3 adopts the sponge construction as an underlying domain extension algorithm. This difference allows us to make the problem of finding a preimage very memory-consuming calculations by properly choosing parameters of sponge functions. Thus, our scheme can achieve the ASIC resistance by using SHA-3 for Hashcash.

Acknowledgments Takanori Isobe is supported by Grants-in-Aid for Scientific Research (Houga) (KAKENHI 20K21795) for Japan Society for the Promotion of Science.

References

[1] Chaudhary, K.C., Chand, V. and Fehnker, A.: Double-spending analysis of bitcoin, *Proc. PACIS 2020* (2020).
[2] Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system, Technical report, Manubot (2019).
[3] Chen, B. and Tessaro, S.: Memory-Hard Functions from Cryptographic Primitives, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*, Boldyreva, A. and Micciancio, D. (Eds.), Lecture Notes in Computer Science, Vol.11693, pp.543–572, Springer (online), DOI: 10.1007/978-3-030-26951-7_19 (2019).
[4] Boneh, D., Corrigan-Gibbs, H. and Schechter, S.E.: Balloon Hashing: A Memory-Hard Function Providing Provable Protection Against Sequential Attacks, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, Cheon, J.H. and Takagi, T. (Eds.), Lecture Notes in Computer Science, Vol.10031, pp.220–248 (online), DOI: 10.1007/978-3-662-53887-6_8 (2016).
[5] Biryukov, A., Dinu, D. and Khovratovich, D.: Argon2: New Generation of Memory-Hard Functions for Password Hashing and Other Applications, *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016*, pp.292–302, IEEE (online), DOI: 10.1109/EuroSP.2016.31 (2016).
[6] Biryukov, A. and Khovratovich, D.: Egalitarian Computing, *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*, Holz, T. and Savage, S. (Eds.), USENIX Association, pp.315–326 (online) (2016), available from ⟨https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/biryukov⟩.
[7] Back, A. et al.: Hashcash-a denial of service counter-measure (2002).
[8] Biryukov, A. and Khovratovich, D.: Equihash: Asymmetric Proof-of-Work Based on the Generalized Birthday Problem, *Ledger*, Vol.2, pp.1–30 (online) (2017), available from ⟨https://ledgerjournal.org/ojs/index.php/ledger/article/view/48⟩.
[9] Bertoni, G., Daemen, J., Peeters, M. and Van Assche, G.: Keccak specifications, *Submission to nist* (*round 2*), pp.320–337 (2009).
[10] Floyd, R.W.: Nondeterministic Algorithms, *J. ACM*, Vol.14, No.4, pp.636–644 (online), DOI: 10.1145/321420.321422 (1967).
[11] Nikolic, I. and Sasaki, Y.: A New Algorithm for the Unbalanced Meet-in-the-Middle Problem, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, Cheon, J.H. and Takagi, T. (Eds.), Lecture Notes in Computer Science, Vol.10031, pp.627–647 (online), DOI: 10.1007/978-3-662-53887-6_23 (2016).
[12] Dinur, I. and Nadler, N.: Time-Memory Tradeoff Attacks on the MTP Proof-of-Work Scheme, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II*, Katz, J. and Shacham, H. (Eds.), Lecture Notes in Computer Science, Vol.10402, pp.375–403, Springer (online), DOI: 10.1007/978-3-319-63715-0_13 (2017).
[13] Bertoni, G., Daemen, J., Peeters, M. and Van Assche, G.: Sponge functions, *ECRYPT Hash Workshop*, Vol.2007, No.9, Citeseer (2007).
[14] Andreeva, E., Mennink, B. and Preneel, B.: The parazoa family: Generalizing the sponge hash functions, *Int. J. Inf. Sec.*, Vol.11, No.3, pp.149–165 (online), DOI: 10.1007/s10207-012-0157-6 (2012).
[15] Guo, J., Peyrin, T. and Poschmann, A.: The PHOTON Family of Lightweight Hash Functions, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, Rogaway, P. (Ed.), Lecture Notes in Computer Science, Vol.6841, pp.222–239, Springer (online), DOI: 10.1007/978-3-642-22792-9_13 (2011).
[16] van Oorschot, P.C. and Wiener, M.J.: Parallel Collision Search with Cryptanalytic Applications, *J. Cryptol.*, Vol.12, No.1, pp.1–28 (online), DOI: 10.1007/PL00003816 (1999).
[17] Bertoni, G., Daemen, J., Peeters, M. and Van Assche, G.: On the indifferentiability of the sponge construction, *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp.181–197, Springer (2008).

Editor's Recommendation

Proof of Work is a widely known consensus algorithm used in cryptographic assets such as Bitcoin. This paper proposes a PoW for the original image restoration problem, which is one of the

fundamental conditions that SHA-3 and other cryptographic hash functions should meet. Since the proposed approach is a practical solution that can compete with fast mining implementations empowered by ASICs, which are considered to be the cause of the power wastage problems, we recommend that this paper be published in a journal article.

(Program Committee Chair of Computer Security Symposium 2020, Tatsuya Mori)

Takaki Asanuma was born in 1996. He received his B.E. degree from Doshisha University, Japan, in 2020. He is currently a master's student at University of Hyogo, Japan. His research interest is blockchain.

Takanori Isobe received his B.E., M.E., and Ph.D. degrees from Kobe University, Japan, in 2006, 2008, and 2013, respectively. From 2008 to 2017, he worked at the Sony Corporation. Since 2017, he has been an Associate Professor at University of Hyogo. His current research interests include information security and cryptography.