Smart Contract-Based Checkpoint for Initial PoW Network Security

Seungmin Kim, Gyeongdeok Maeng and Heung-No Lee*
Dept. of Electrical Engineering and Computer Sciences
Gwangju Institute of Science and Technology (GIST)
Gwangju, Korea
heungno@gist.ac.kr

***Abstract*—Proof of Work (PoW) is the most widely adopted consensus mechanism on public blockchains. The PoW blockchain network achieves consensus by solving computational problems. If a network participant owns more than 50% of the total computational power in the network, they can forge the blockchain. Hence, initial PoW blockchain networks with low computing power are vulnerable to block forgery attacks. We propose a smart contract-based checkpoint method to improve security vulnerabilities in the initial blockchain networks. In our method, participants periodically record block headers in an Ethereum smart contract. The recorded checkpoint block header validates the blockchain. Participants reject blocks with blocks that differ from the recorded checkpoints. Our method ensures the integrity of blocks until the height of the most recently created checkpoint, reducing the risk of double-spending. We optimize checkpoint costs by overlapping multiple checkpoint processes in a single transaction. The interval of our checkpoint method grows with the growth of the network, making the network less dependent on checkpoints. We analyze the performance of checkpoints in mitigating attacks and demonstrate that they significantly decrease the success probability of attacks in the network.**

***Index Terms*—Checkpoint, PoW blockchain, smart contract, double spending attack**

I. Introduction

Blockchain is a decentralized ledger that maintains the integrity of data through consensus. Blockchain uses a consensus mechanism to select decision-makers in each block. [1]. The decision-makers in the blockchain record the validated transactions in block and broadcast block to the network. The consensus mechanism is critical to blockchain decision-making and therefore has a major impact on the security and scalability of a blockchain [2]. Innumerable consensus mechanisms have been studied to increase the security and scalability of blockchains [3]–[6]. Proof of Work(PoW), introduced in Bitcoin, is the most popular consensus mechanism [7]. PoW is a consensus mechanism that involves solving mathematical problems to select decision-makers. The participant who solves the problem first gets the authority to create a block. The computational works involved in decision-making make it difficult for attackers to engage in malicious behavior. Proof of Work is still the most typical consensus mechanism, although Ethereum transitioned to Proof of Stake(PoS). [8].

In a PoW network, participants with more than 50% of the total computing power have majority decision-making power. This truth means they can monopolize the network's decision-making process and manipulate the data on the blockchain [9]. It is problematic that one participant possesses more than 50% of the network's computing power in a mature PoW network. One participant possessing more than 50% of the network's computing power in a blockchain network is complicated and becomes increasingly difficult as the number of participants increases. However, in early blockchain networks, where the network size is small, possessing more than 50% is relatively easy.

Checkpoints are state values recorded at regular intervals in a blockchain network. Blockchain networks regularly store state values as checkpoints. Network participants recognize blockchains that contain checkpoints as legitimate chains. This new regulation increases the network's security by making it harder for attackers to forge blockchains. Checkpoints also serve as reference points for new network participants. The new participant connects to the blockchain network and attempts to synchronize blocks. There are many forked chains in the network, which can cause new participants to synchronize the wrong chains. New participants can use checkpoints to validate forked chains and prevent incorrect synchronization.

Related Work. There have been various implementations that have applied checkpoints and numerous research studies proposing checkpoint mechanisms. Bitcoin Core, a client software for Bitcoin, utilizes hard-coded checkpoints internally to protect the initial network from potential attack vulnerabilities [10]. Hard-coded checkpoints are inflexible and centralized to the developers. Polygon [11] executes the Proof of Stake (PoS) consensus process in Ethereum smart contracts and stores the outcomes as checkpoints. Polygon does not have its inherent proof-of-work mechanism and relies on Ethereum for network security. In [12], Wang and Kim propose an additionally distributed ledger for external validators and a method for recording checkpoints in the ledger. Their recording checkpoints to a distributed ledger incur a high network cost. [13] proposes to create a minority committee for each checkpoint, which decides and records the checkpoint. Their work assumes a synchronous network, unlike asynchronous blockchain networks.

Contributions. We propose a smart contract-based checkpoint method to improve the security of initial proof-of-work networks. The core idea of our proposed method is

to record checkpoints in the smart contracts of the Ethereum network. The Ethereum blockchain network's enormous size guarantees our checkpoints' security against tampering efforts. We design the checkpoint generation process to consider the asynchronous nature of blockchain networks. The checkpoint creation process consists of three timelines for each checkpoint generation interval. We overlap multiple checkpoint processes in a single transaction to reduce the cost of checkpoint generation. The checkpoint interval gradually increases with the total computing power of the network. As the interval between checkpoints recorded on Ethereum increases, the network becomes independent of Ethereum. We assume a selfish mining attack scenario and analyze the attack probabilities for checkpoints and typical networks. According to our analysis, our proposed method effectively reduces the attack probability compared to regular PoW networks.

Organization. The rest of this paper is structured as follows. In section II, we describe an overview of the research and background concepts related to checkpoints. Section III explains the checkpoint protocol and smart contracts. In section IV, we consider a scenario involving selfish mining attacks and compare the performance of checkpoints with a typical network. In section V, we conclude and discuss future work.

II. Background and Overview

A. PoW blockchain network attack

Attacks on proof-of-work (PoW) blockchain networks largely stem from the probabilistic nature of the PoW mechanism. PoW network follows the Longest Chain rule, which adopts the longest chain, i.e., the chain with the most accumulated work, as the legitimate chain of the network [14]. Whenever the PoW network finds a longer chain, it replaces the existing chain with the new one.

Double-spending attacks exploit the Longest Chain rule [15]. In a double-spend attack, the attacker pays through a legitimate transaction and receives a reward for the transaction. The attacker uses their computational power to create the longest blockchain that contains the forged transaction. The network recognizes the longest block, the forged blockchain, as legitimate and invalidates the legitimate transaction. The attacker obtains the forged transaction reward and reuses the currency used in the transaction. Selfish mining attack is another form of attack that exploits the Longest Chain rule [16]. In a selfish mining attack, the attacker intentionally withholds generated blocks instead of immediately broadcasting them. The attacker continues to increase the height of their private blockchain without revealing the blocks. The attacker announces the blocks after the attacker's blockchain surpasses the network's height. The transaction and block rewards on the network's chain become insignificant since the attacker's chain becomes the longest.

The attacker must catch up to the network's longest chain to execute such an attack. The attack has a probability of success of 1 if the attacker's computational power is over half of the network's total computing power since the attacker can catch up to the blockchain. However, even though the attacker's computational power does not exceed 50%, there is a possibility that they can succeed in executing the attack probabilistically [17].

B. Confirmation distance

Confirmation distance is a mechanism introduced in blockchain networks to mitigate double-spending attacks [18]. Confirmation distance refers to the number of blocks a user has to wait for a transaction to be sent and confirmed. A participant sends a transaction to the blockchain but does not immediately supply the transaction reward and waits for the confirmation distance. The participant sends the transaction reward after all the confirmation distance blocks have been generated. A double-spending attacker must forge blocks after the confirmation distance to invalidate transactions. The confirmation distance is effective against double-spending attacks employing less than 50% of the total computational power. The probability of the attacker catching up with the blocks decreases as block height increases, assuming the attacker has less than 50% computing power. To receive a transaction reward, an attacker must attempt a double-spend attack after the confirmation distance. Confirmation distance increases the difficulty of an attack by requiring the attacker to catch up to the block after the confirmation distance.

The confirmation distance in a blockchain network can vary depending on the network conditions. The confirmation distance increases with a higher rate of total computational power for an attacker and decreases with a lower rate. A longer confirmation distance means a longer transaction processing time, which can cause inconvenience for participants who require fast transaction confirmations. Furthermore, attacks with more than 50% computational power always generate longer blocks, ignoring confirmation distance. As a result, additional security protocols are necessary when the attacker can launch an attack with substantial computational power.

C. Smart contract

A smart contract is a self-executing contract where the terms and conditions are directly written into code, enabling automatic execution without the need for intermediaries [19]. These contracts operate based on predefined conditions and rules built on a blockchain platform. The smart contract code automatically executes the contract conditions, ensuring transparency and accuracy throughout the contractual process. Smart contracts are recorded and verified on the blockchain, providing high security and resistance to tampering. They reduce the need for intermediaries, streamline processes, and automate transactions, potentially saving time and cost. The self-executing nature of smart contracts is widely utilized and researched in many fields that rely on data-driven transactions [20]. Increasingly, developers and researchers are actively involved in the development and exploration of smart contracts [21]–[23].

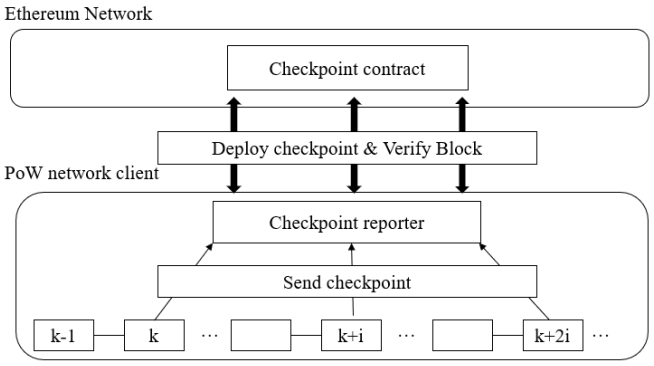

Fig. 1. The architecture of our method

III. Proposed method

In this section, we first present an architecture of our proposed checkpoint method, then introduce the details of the method.

A. Architecture

Fig. 1 depicts the architecture of our proposed method. The proposed method consists of two layers: the Ethereum network layer, which includes the checkpoint contract, and the PoW network layer, which the checkpoint aims to protect. Our method combines the Ethereum network layer with the PoW network layer to ensure the security of the PoW network. The checkpoint contract runs on the Ethereum network and serves as the administrator overseeing the management and execution of the checkpoint. The checkpoint contract defines the rules and conditions necessary to verify and record checkpoints on the Ethereum blockchain securely. The PoW network works similarly to a typical blockchain network but includes functionalities for generating, transmitting, and requesting checkpoints.

The PoW network validator generates a checkpoint when it reaches the checkpoint generation interval, which serves as a reference for the network's state. Checkpoint is nearly identical to the block header of the same block height, but they include additional data that marks them as a checkpoint. Checkpoints are generated every interval and sent to the Ethereum checkpoint contract. Participants in the network request the contract for a checkpoint when a block modification occurs such that the new block can be validated. Since checkpoint requests do not affect the state of the contract, they can be provided directly without submitting a transaction. The network client communicates with the checkpoint contract through the checkpoint reporter. The checkpoint reporter consists of an Ethereum light client that can communicate with Ethereum and is compatible with the same private key used by the PoW client.

Ethereum smart contract-based architecture provides the following advantages for the checkpoint. First, it mitigates the risk of checkpoint tampering. Ethereum is a network with many participants, and the likelihood of tampering is extremely low. Second, checkpoints operate transparently and according to predetermined rules, ensuring that checkpoint development remains decentralized and not centralized in the hands of checkpoint developers.

B. Checkpoint generation process

One of the things to consider when creating checkpoints is that Proof of Work networks operate as asynchronous distributed networks, which means that block propagation can be delayed or uncertain. We consider the asynchronous characteristics of the network during the checkpoint generation process. Despite a checkpoint created by one participant, other participants may need to be made aware of it and may still attempt to generate a checkpoint. We separate the checkpoint generation process into three timelines to reflect the network's asynchronous characteristics. Fig. 2 shows the checkpoint process timelines.

In Fig. 2, The first timeline occurs when the block height is at checkpoint time, defined as k. The decision maker creates block k but waits without creating a checkpoint. It is risky to trust a block that has just been created on an asynchronous network. Other blocks can easily replace blocks. Participants must wait until the network has transmitted block k to all participants.

The second timeline occurs at block height $k + i$, where i represents the checkpoint interval. We assume that i blocks have been generated from height k, so the propagation of block k is almost complete. The participant generates a checkpoint for block k and transmits it to the contract when the height $k + i$ has been achieved. The contract does not immediately complete the checkpoint operation for height k but instead stores the received checkpoints in a checkpoint pending pool and waits. The participant also performs the first timeline process at checkpoint $k + i$.

The third stage similarly occurs, following the completion of the second stage at height $k+i$. The second and third timelines i could differ according to the checkpoint interval adjustments described in the checkpoint interval section. However, we represent it as $k + 2i$ for ease of expression. The contract stores the checkpoints for height k in the checkpoint pending pool and waits. The first timeline for height $k + 2i$ and the second timeline for height $k + I$ execute when the network reaches height $k+2i$. During this process, participants transmit the checkpoints to contract for height $k+i$. The contract uses the checkpoint for height $k + i$ to recognize the block height, as it cannot identify the network's current block height. The contract finishes the vote for checkpoint k and decides the final checkpoint, k, after receiving the first checkpoint for height $k + i$. The final checkpoint is decided based on receiving the most votes. The checkpoint's recording guarantees the integrity of the blocks up to height k.

The actual network propagation time will probably be shorter than the checkpoint interval, but we await the propagation of the network until the next checkpoint interval. This wait causes multiple checkpoint processes to work simultaneously in an interval. Participants can merge transactions from

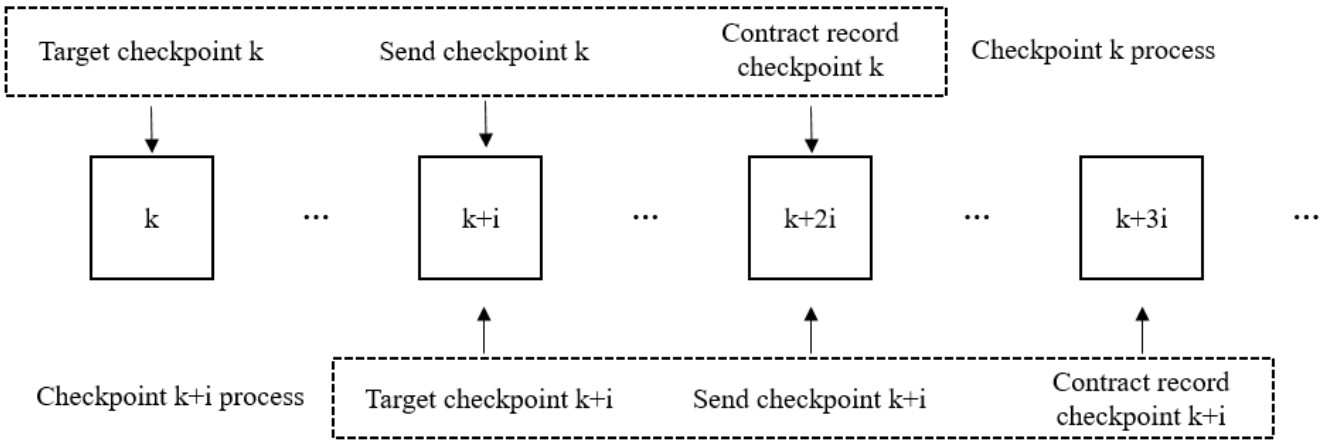

Fig. 2. Checkpoint process timeline

concurrently running checkpoints and send them in a single transaction. Checkpoint transaction merging reduces the cost of concurrently counting up to three processes by up to a third.

C. Checkpoint Smart contract

We establish a validator set that can generate checkpoints and manage the list in the contract. This list of validators forms a decentralized autonomous organization (DAO) that operates autonomously according to predefined rules within the contract. All validators have equal authority and participate in decision-making through voting. The contract records the checkpoint that receives the most votes from validators as the final checkpoint. Validator DAO can make decisions and execute processes without relying on centralized authority. The equal authority of the DAO ensures that decisions are made fairly and equitably.

The signatures of the validators determine the authenticity of a checkpoint transaction. We ensure the validators use the same private key for the PoW network and Ethereum. The validator signs the checkpoint with their private key and sends it to the contract. The contract verifies that the checkpoint's signature is included in the validators. Checkpoints with valid signatures are stored in the checkpoint pending pool. When a contract determines a checkpoint within a pending pool, the contract saves the checkpoint. The contract only stores the most recent checkpoint. The block generation process in PoW includes data from the previous block. The most recent checkpoint includes the previous one, which means the contract does not need to store the previous one. The contract calculates the attacker's computational power as the block difficulty for the spacing adjustment described in the following subsection. Algorithm 1 is a pseudo-code showing the process executed by the checkpoint contract when a checkpoint is transmitted.

Algorithm 1 Checkpoint contract process

function SUBMITCHECKPOINT(header , sig)
  $proposer, blocknumber, roothash \leftarrow header$
  if $proposer$ not in $validators$ **then**
    return $ValError$
  if $EcVerify(header, sig, proposer)$ is $false$ **then**
    return $SignError$
  if $CheckpointNumber < blocknumber$ **then**
    $checkpoint = PendingHeader.decideCheck()$
    $Interval \leftarrow CalNextInterval(checkpoint)$
  $PendingHeader.append(header)$
  return $(checkpoint, Interval)$

D. Checkpoint Interval

The smart contract calculates the checkpoint interval at the time of checkpoint creation. The checkpoint includes the block difficulty at the checkpoint's point in time, allowing us to assess the overall scale of the network through the block difficulty. Given the difficulty of block height k as $diff_k$ and block time as T_k, the total computational power of the network c_k is calculated as follows:

$$c_k = \frac{diff_k * 2^{32}}{t_k - t_{k-1}} \tag{1}$$

We find the attacker's computational power c_a to get the attacker's total computational power ratio. The c_a can be roughly gauged from the availability of computing resource rental platforms. For example, the Ethash computational power available for rent on nicehash.com in May 2023 is approximately 3 TH/s [24]. The attacker's total computational power ratio t can be expressed as follows:

$$t = \frac{c_a}{c_k} \tag{2}$$

Our checkpoints are confirmed through the three timelines and finalized two intervals later. A checkpoint interval of i will provide finality for a block up to $2i$ earlier. The confirmation distance z must be greater than $2i$ to be secure against double-spending attacks. The interval of checkpoints

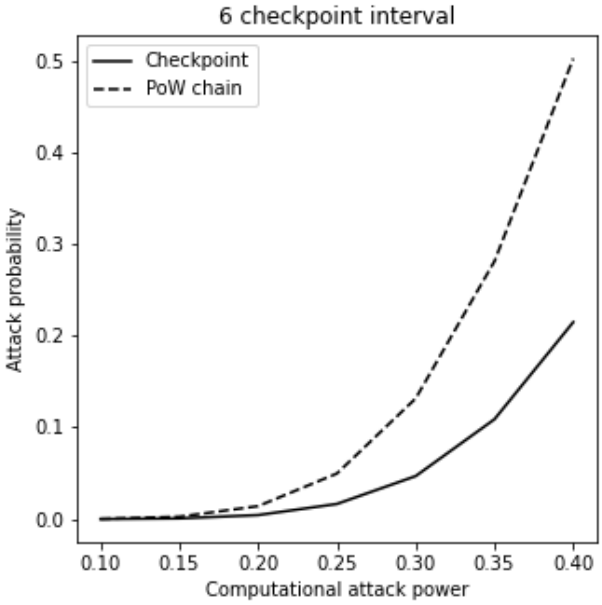

Fig. 3. probability of selfish mining attacks based on the attack power.

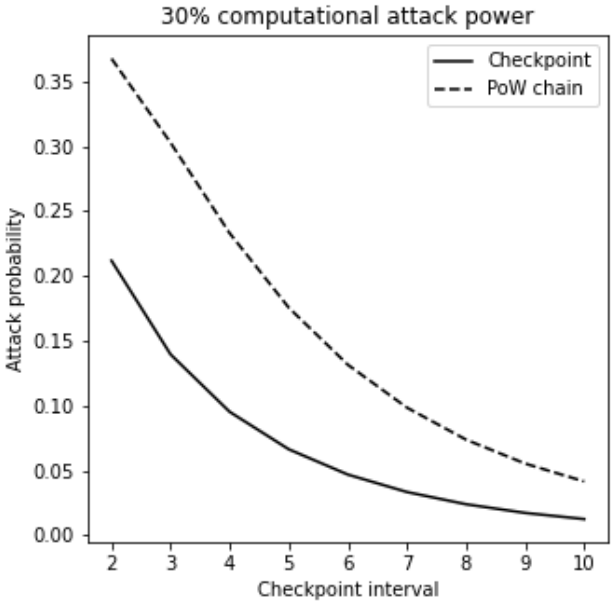

Fig. 4. probability of selfish mining attacks based on the checkpoint interval

significantly affects network performance. It provides significant completeness and security against short-term attacks but increases transaction costs.

Considering the cost of the attack, the network is sufficiently secure if the probability of a double-spend attack is less than 0.001. It means that the network is secure enough without checkpoints. We designed the interval of checkpoints to increase as the network expands. For every halving of the attack probability, the checkpoint interval doubles. This process repeats, gradually reducing the dependency on checkpoints. The attacker's probability p can be calculated from t obtained in Eq.(2) [15]. The network reaches a sufficiently large size to be secure against attacks. Given that the confirmation distance is z, and the attacker's attack probability is p, the interval i of the checkpoint can be obtained as follow:

$$i = \begin{cases} \frac{z}{2} & \text{if } p > 0.001 \\ 2^{\lceil \log_2(\frac{0.001}{p}) \rceil} \cdot z & \text{if } else \end{cases} \tag{3}$$

IV. Performance analysis

This section analyzes our proposed method's Selfish Mining Attack security performance. Our proposed method is resistant to double-spending attacks but only partially immune to selfish mining attacks. It is because Selfish Mining Attacks can occur even within confirmation distance. A typical PoW network has no checkpoints, so there is an infinite amount of time for a selfish mining attack to be attempted. In contrast to typical PoW networks, our method limits selfish mining attackers to only attacking until a checkpoint is generated.

The selfish mining attack scenario in our method is as follows. The attacker confirms that a checkpoint has been created and then prepares for a selfish mining attack before the next checkpoint. The attacker creates a block but does not broadcast it. The attacker creates and broadcasts the longest block before the checkpoint is created. Blocks and transactions created by other participants become invalid and monopolize the rewards of those blocks. The contract records the blocks generated by the attacker as checkpoints. This attack can harm network performance, but transactions are secure because the attacker's reward is limited to the mining revenue.

The probability of the selfish mining attack scenario is calculated similarly to the probability of a double-spend attack. We express the probability of a selfish attack scenario using a Poisson distribution. The attacker must create the longest chain and forge blocks before recording the checkpoint. We compute the probability distribution of blocks an attacker can generate during a checkpoint interval. We then calculate the total probability of the blocks generated by the attacker being more significant than the checkpoint interval. When the checkpoint interval is denoted as i, and the parameter λ is equal to $i * \frac{q}{1-q}$, the probability of the attacker's selfish mining attack can be computed as follows:

$$\text{attack}_i = \sum_{k=0}^{\infty} \frac{\lambda^k e^{-\lambda}}{k!} \cdot \begin{cases} 0 & \text{if } k \leq i \\ 1 & \text{if } k > i \end{cases} \tag{4}$$

We compare the performance of our method in the selfish mining attack scenario with a typical PoW network using Eq.(4). In our analysis, and we consider various attackers' computational power and the checkpoint interval. We conducted two experiments to evaluate the efficacy of checkpoints. In the first experiment, we set the checkpoint interval to 6 and analyzed the impact of varying the attacker's computational

power on the success probability of attacks. The results of this experiment are presented in Fig. 3. It shows that our method exhibits a decrease in attack probability for all levels of attack power, with the reduction rate gradually increasing. It showcases a maximum reduction of approximately 60% in the attack probability. In the second experiment, we maintained the attacker's computational power at a fixed rate of 30% and observed the changes in the attack success probability by adjusting the checkpoint interval. Fig. 4 depicts the outcomes of this experiment. Similarly, Fig. 4 shows that our method reduces the attack success probability across all checkpoint intervals.

As a result, the two experiments show that checkpoint serves as an effective solution to prevent selfish mining attacks. The consistent decrease in attack probability for all situations is convincing evidence of the performance of our method.

V. Conclusion

In this paper, we present a smart contract-based checkpoint mechanism for Proof of Work (PoW) networks, addressing the challenges associated with network security and vulnerability to attacks. We record checkpoints in Ethereum smart contracts to reduce the probability of checkpoints being forged, as Ethereum is exceedingly difficult to forge. Since blockchain networks are asynchronous, we proposed a checkpoint generation process composed of three timelines that execute each checkpoint interval. To reduce checkpoint costs, We merge concurrently running processes and submit them as a single transaction. The checkpoint interval increases gradually as the PoW network expands. When the checkpoint interval is large enough, the network moves away from its dependence on checkpoints and becomes independent of Ethereum. We performed an attack probability analysis on a selfish mining attack to show the effectiveness of our method. The proposed checkpoint significantly lowers the probabilities of attacks, especially in the initial stages of PoW networks, compared to typical PoW networks. In our future work, we will apply our checkpoint method to existing blockchain networks and research methods to reduce the cost associated with checkpoints.

Acknowledgment

This work was supported by the MSIT, Korea, under the ITRC (Information Technology Research Center) support Program (IITP-2023-2021-0-01835) supervised by the IITP (Institute for Information & Communications Technology Planning Evaluation.)

References

[1] M. Platt, J. Sedlmeir, D. Platt, J. Xu, P. Tasca, N. Vadgama, and J. I. Ibañez, "The energy footprint of blockchain consensus mechanisms beyond proof-of-work," in *2021 IEEE 21st International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, 2021, pp. 1135–1144.

[2] S. Velliangiri and P. Karthikeyan, "Blockchain technology: Challenges and security issues in consensus algorithm," in *2020 International Conference on Computer Communication and Informatics (ICCCI)*, 2020, pp. 1–8.

[3] S. Chen, H. Mi, J. Ping, Z. Yan, Z. Shen, X. Liu, N. Zhang, Q. Xia, and C. Kang, "A blockchain consensus mechanism that uses proof of solution to optimize energy dispatch and trading," *Nature Energy*, vol. 7, no. 6, pp. 495–502, 2022.

[4] Y. Wang, H. Peng, Z. Su, T. H. Luan, A. Benslimane, and Y. Wu, "A platform-free proof of federated learning consensus mechanism for sustainable blockchains," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 12, pp. 3305–3324, 2022.

[5] "Proof-of-contribution consensus mechanism for blockchain and its application in intellectual property protection," *Information Processing Management*, vol. 58, no. 3, p. 102507, 2021.

[6] P. Fernando, K. Dadallage, T. Gamage, C. Seneviratne, A. Madanayake, and M. Liyanage, "Proof of sense: A novel consensus mechanism for spectrum misuse detection," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 12, pp. 9206–9216, 2022.

[7] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized business review*, p. 21260, 2008.

[8] E. Kapengut and B. Mizrach, "An event study of the ethereum transition to proof-of-stake," *Commodities*, vol. 2, no. 2, pp. 96–110, 2023. [Online]. Available: https://www.mdpi.com/2813-2432/2/2/6

[9] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 3–16. [Online]. Available: https://doi.org/10.1145/2976749.2978341

[10] "Bitcoin core 0.11.0," https://bitcoin.org/en/release/v0.11.0, 2015.

[11] "Matic whitepaper," https://github.com/maticnetwork/whitepaper/, 2020.

[12] K. Wang and H. S. Kim, "Reducing confirmation reversal probability of pow blockchains using checkpoints," in *2022 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2022, pp. 1–9.

[13] D. Karakostas and A. Kiayias, "Securing proof-of-work ledgers via checkpointing," in *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2021, pp. 1–5.

[14] S. Sayeed and H. Marco-Gisbert, "Assessing blockchain consensus and security mechanisms against the 512019. [Online]. Available: https://www.mdpi.com/2076-3417/9/9/1788

[15] M. Rosenfeld, "Analysis of hashrate-based double spending," *arXiv preprint arXiv:1402.2009*, 2014.

[16] Q. Bai, X. Zhou, X. Wang, Y. Xu, X. Wang, and Q. Kong, "A deep dive into blockchain selfish mining," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6.

[17] J. Jang and H.-N. Lee, "Profitable double-spending attacks," *Applied Sciences*, vol. 10, no. 23, 2020. [Online]. Available: https://www.mdpi.com/2076-3417/10/23/8477

[18] Y. Kawase and S. Kasahara, "Transaction-confirmation time for bitcoin: A queueing analytical approach to blockchain mechanism," in *Queueing Theory and Network Applications*, W. Yue, Q.-L. Li, S. Jin, and Z. Ma, Eds. Cham: Springer International Publishing, 2017, pp. 75–88.

[19] N. Szabo, "Formalizing and securing relationships on public networks," *First monday*, 1997.

[20] W. Zou, D. Lo, P. S. Kochhar, X.-B. D. Le, X. Xia, Y. Feng, Z. Chen, and B. Xu, "Smart contract development: Challenges and opportunities," *IEEE Transactions on Software Engineering*, vol. 47, no. 10, pp. 2084–2106, 2021.

[21] S. Ahmadisheykhsarmast and R. Sonmez, "A smart contract system for security of payment of construction contracts," *Automation in Construction*, vol. 120, p. 103401, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092658052030981X

[22] A. Khatoon, "A blockchain-based smart contract system for healthcare management," *Electronics*, vol. 9, no. 1, 2020. [Online]. Available: https://www.mdpi.com/2079-9292/9/1/94

[23] H. Hasan, E. AlHadhrami, A. AlDhaheri, K. Salah, and R. Jayaraman, "Smart contract-based approach for efficient shipment management," *Computers Industrial Engineering*, vol. 136, pp. 149–159, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360835219304140

[24] "Nicehash marketplace," https://www.nicehash.com/marketplace, 2023.