PQ-DPoL: An Efficient Post-Quantum Blockchain Consensus Algorithm

Wonwoong Kim[1][0009−0002−6245−8044],
Yeajun Kang[1][0009−0007−4005−8505],
Hyunji Kim[2][0000−0001−9828−3894],
Kyungbae Jang[1][0000−0001−5963−7127], and
Hwajeong Seo[3][0000−0003−0069−9061]

[1]Department of IT Convergence Engineering,
Hansung University, Seoul (02876), South Korea,
[2]Department of Information Computer Engineering,
Hansung University, Seoul (02876), South Korea,
[3]Department of Convergence Security,
Hansung University, Seoul (02876), South Korea,
{dnjsdndeee, etus1211, khj1594012, starj1023, hwajeong84}@gmail.com

Abstract. The advancement of quantum computers and the potential polynomial-time solution of Elliptic Curve Cryptography (ECC) using the Shor algorithm pose a significant threat to blockchain security.
This paper presents an efficient quantum-secure blockchain with our novel consensus algorithm. We integrate a post-quantum signature scheme into the transaction signing and verification process of our blockchain, ensuring its resistance against quantum attacks. Concretely, we adopt the `Dilithium` signature scheme, which is one of the selected algorithms in the NIST Post-Quantum Cryptography (PQC) standardization.
Not surprisingly, the incorporation of a post-quantum signature scheme leads to a reduction in the number of Transactions Per Second (TPS) processed by our blockchain. To mitigate this performance degradation, we introduce a new consensus algorithm that effectively combines the Proof of Luck (PoL) mechanism with a delegated approach.
We strive to build an efficient and secure blockchain for the post-quantum era by benchmarking our blockchain, adjusting the security parameters of `Dilithium`, and refining the components of the consensus algorithm.

Keywords: Consensus Algorithm · Blockchain · Post-Quantum Security · Dilithium · TEE.

1 Introduction

The ECC-based signature scheme is commonly adopted in blockchains for transaction signing and verification, thanks to its efficiency. However, the following two factors prompt us to consider replacing ECC with PQC:

1. Shor's algorithm [1] efficiently models discrete logarithm problems on elliptic curves and provides polynomial-time solutions.

2. The emergence of quantum computers capable of running Shor's algorithm is anticipated in the near future.

Grover's algorithm [2] also poses a threat to the security of hash functions by reducing the search complexity by a square root. However, there is an off-the-shelf countermeasure that increases the output size of hash functions. Furthermore, due to the significant quantum circuit depth required by Grover's algorithm [3,4,5], launching attacks becomes more challenging compared to Shor's algorithm [6]. For these reasons, we focus on the threat posed by Shor's algorithm rather than Grover's algorithm throughout this paper.

To achieve a quantum-secure blockchain, it is crucial to adopt signature schemes that are resistant to quantum attacks, such as those posed by Shor's algorithm. However, it is widely recognized that post-quantum cryptography faces performance-related challenges, including large signature sizes and slow signing/verification speeds. These challenges are particularly significant for blockchain applications due to the nature of their consensus algorithm.

Inspired by this concern, we propose Post-Quantum Delegated Proof of Luck (PQ-DPoL), an efficient and quantum-secure blockchain consensus algorithm. PQ-DPoL utilizes the post-quantum signature scheme `Dilithium` [7], which is one of the recently standardized signature schemes in NIST's PQC standardization[1].

It is important to note that utilizing post-quantum signatures in a blockchain will inevitably lead to performance degradation. To address this issue, we introduce a novel combination of consensus primitives: Proof of Luck (PoL) with a delegated approach.

Our Contribution

This paper makes several contributions, which can be summarized as follows:

1. **Post-quantum Blockchain with PQC Scheme.** After carefully considering our blockchain components, including the targeted device and consensus algorithm, we adopt the `Dilithium` PQC scheme. This decision ensures the security of our blockchain against potential quantum attacks.
2. **New Consensus Algorithm for an Efficient Blockchain.** For the first time, we present the DPoL consensus algorithm, which combines PoL with a delegated approach. DPoL algorithm enhances block generation speed by simplifying the consensus process within a TEE. With the implementation of DPoL algorithm, we achieve satisfactory performance even when utilizing a PQC scheme.
3. **Diversifying Configurations and Benchmarks.** We provide various benchmarks to ensure compatibility and optimal performance in different environments as we build our blockchain with various options (such as adjusting the number of nodes or security parameters of `Dilithium`).

[1] https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022

2 Preliminaries

2.1 Blockchain Consensus Algorithm

Blockchain is a distributed ledger technology in which nodes within a network communicate in a Peer-to-Peer (P2P) to share ledger. Blockchain is a decentralized method in which nodes each own a ledger. Therefore, nodes directly create blocks and verify transactions.

The consensus algorithm is used by nodes on the blockchain to ensure data integrity through specific procedures and to make the same decision. Each consensus algorithm has a block producer and validator. The block producer creates a block containing transactions and sends it to a validator. Validators verify that the header of a block is valid. Also, the validator performs verification by checking the signature of the transaction contained in the block. There are many types of consensus algorithms. Representatively, there are Proof of Work (PoW) used in Bitcoin and Proof of Stake (PoS) used in Ethereum. There is also Delegated Proof of Stake (DPoS), which adds delegation to PoS. In addition to this, there are Proof of Luck (PoL) and Proof of Elapsed Time (PoET) based on TEE.

In PoW, a node must perform mining to become a block producer. However, PoW has a limitation in that it consumes excessive energy during the mining process. PoS can solve the power consumption problem of PoW [8]. In PoS, nodes with sufficient stake become block producers. However, this causes a problem of the rich getting richer and the poor getting poorer, as nodes without stakes cannot create blocks.

DPoS is a consensus algorithm based on PoS [9]. In DPoS, delegates are elected through voting. These delegates perform the PoS consensus algorithm among themselves. As a result, TPS is improved, and the problem of PoS can be overcome. However, if the number of delegates is small compared to the size of the network, centralization issues can arise.

In PoET, the node that completes its assigned wait time first becomes the block producer [10]. Verification can be performed at high speed because it verifies whether the waiting time has actually been achieved through TEE. PoET provides equal opportunities to all nodes. Also, it does not cause the computing resource consumption problem that occurs in PoW.

2.2 Evaluation Metrics for Blockchain Performance

Blockchain has various elements such as network structure and consensus algorithm. Because of this, it is not easy to evaluate the performance of a blockchain using a single metric. For fair evaluation, several works have been conducted on the evaluation of blockchain performance [11,12]. Representative metrics that are mainly used include TPS and latency. In addition, there are other metrics: block verification time, decentralized level, and power consumption.

These metrics are affected by various factors of the blockchain (e.g., block size, the number of nodes and digital signature algorithm). This section describes these evaluation metrics of blockchain.

TPS TPS means how many transactions can be processed in one second. In other words, it is the number of transactions that can be confirmed in one second. TPS is the most common metric of blockchain performance currently used and represents the speed of the blockchain. Many recent blockchains are being designed in a structure that can achieve high TPS. However, TPS should not be used as the only evaluation metric of a blockchain. If TPS is improved without considering other evaluation metrics, TPS may increase, but problems (e.g. lower scalability of the blockchain) may occur. Bitcoin, the most famous cryptocurrency, has 7 TPS, Ethereum has 20 TPS, and EOS has 3000 TPS. However, since EOS is close to a centralized system. Therefore, the decentralization characteristic of blockchain is not satisfied.

Latency Latency is the time it takes from the time a transaction appears on the network until it is verified. If the latency is high, it means that a lot of time is required to process one transaction. Therefore, high latency causes performance degradation of the blockchain. Bitcoin's latency is 10 minutes, which is very large, and Ethereum's is 0.22 minutes. Also, Ripple, which aims for low latency, has a latency of about 4 seconds.

2.3 CRYSTALS-DILITHIUM

CRYSTALS-DILITHIUM[2] is a post-quantum cryptography selected by NIST as an algorithm to be standardized. `Dilithium` is a lattice-based cipher based on the Shortest Vector Problem (SVP). The SVP problem is that it is difficult to find the vector closest to an arbitrary position on the lattice in polynomial time (even on quantum computers).

Table 1 presents the sizes of `Dilithium` public keys and signatures (in bytes), `Dilithium` is divided into `Dilithium-2,3,5` depending on the level of security. The size of `Dilithium` public key ranges from 1,312 to 2,592 bytes, private key ranges from 2,528 to 4,864 bytes, and the size of the signature ranges from 2,420 to 4,595 bytes. Compared to other PQCs, `Dilithium` has larger key and signature sizes but offers faster computational speed[13].

Table 1: Details of `Dilithium`.

Scheme	NIST level	Public key	Private key	Signature
`Dilithium-2`	2	1,312	2,528	2,420
`Dilithium-3`	3	1,952	4,000	3,293
`Dilithium-5`	5	2,592	4,864	4,595

[2] https://pq-crystals.org/index.shtml

2.4 Trusted Execution Environment

TEE (Trusted Execution Environment) means the security area of the main processor. The security area ensures that code and data inside the processor can be protected in terms of confidentiality and integrity. There are various unique techniques for this. For example, Trusted Time, Monotonic Counter, Random Number Generation, and Attestation are representative. Representative platforms of TEE include Intel SGX, ARM TrustZone, RISC-V MultiZone, and AMD SEE.

Trusted Time TEE provides PRTC (Protected Real-Time Clock) based timer for trusted time service [14]. This Trusted Time Service can be used to measure the elapsed time since reading the previous timer. Timer Source Epoch can be used to detect discontinuities between Read Time points. Discontinuity means that the PRTC has been reset due to an event such as a battery replacement, or the timer has been paired with a different PRTC due to an unexpected event such as a software attack. In this case, the user MUST NOT trust the calculated period between the two Timer Readings and handle the error condition appropriately. It can be used through $sgx\_get\_trusted\_time()$, a library API function of the Intel SGX SDK.

Monotonic Counter Users can use a monotonic counter with ID, which is a unique identifier [14]. The user can create a monotonic counter through the Intel SGX SDK, increase the value, read or delete the value.

Attestation There are cases in which different TEEs must cooperate for various reasons such as communication. At this time, a function to prove each other's reliability is provided, which is called Attestation. Attestation is divided into local attestation and remote attestation. If two TEE areas exist on the same CPU, verification is possible through local attestation, and if they exist on different CPUs, verification must be performed through remote attestation.

Random Number Generation TEE can generate a random value from an Intel on-chip hardware random number generator seeded by an on-chip entropy source through a command called RDRAND.

3 PQ-DPoL

This section presents various methodologies gathered to construct our PQ-DPoL, an efficient post-quantum blockchain consensus algorithm.

Before presenting our methodologies, we provide the following remarks on the key aspects/considerations for post-quantum blockchain:

- **Key Sizes** When considering the use of Internet of Things (IoT) devices for blockchain, it is advisable to adopt PQC (Post-Quantum Cryptography) with small public and private keys.
- **Signature Size** Blockchain transactions involve user signatures. To increase the number of transactions that can be included in a block, it is recommended to adopt PQC with small signature sizes.
- **Execution Speed** PQC scheme should process a large number of transactions per second, allowing for high-speed blockchain operation.
- **Computational Complexity** While fast execution is desirable, it is important to consider the computational complexity. A PQC scheme may execute quickly on certain hardware but be slower on others, so a balance needs to be struck between computational complexity, execution time, and supported hardware devices.
- **Power Consumption** Power consumption is a concern for energy-intensive blockchains like Bitcoin. Factors such as hardware, communication transactions, and security schemes contribute to energy usage. Therefore, PQC schemes should aim for efficient energy consumption.

3.1 Post-Quantum Blockchain Using `Dilithium`

The `Dilithium` scheme provides the advantage of faster signing and verification speeds compared to many other PQC signature schemes. However, this advantage comes at the cost of large key and signature sizes (PQC schemes often involve this trade-off).

As noted earlier (Section 3), when considering IoT devices, it is recommended to use PQC schemes with small key sizes for blockchain. However, the devices targeted for our blockchain are CPUs that support TEE for the PoL consensus algorithm. This indicates that our blockchain is capable of accommodating PQC with large key sizes. Thus, the large key sizes of `Dilithium` do not present any issues for our blockchain, and we can benefit from its fast signing and verification speeds. In this regard, `Dilithium` is well-suited for our blockchain.

However, the large signature size of `Dilithium` leads to an increased transaction size. Consequently, the capacity for including transactions in a block is reduced, resulting in lower TPS.

In Section 3.3, we will present how to increase the lowered TPS using the proposed consensus algorithm, DPoL.

3.2 Construction of Trusted Execution Environment

In this work, the system is designed based on Intel SGX, one of the TEEs. TEE enforces correct operation of an algorithm or critical operation process. Representatively, in order to vote for delegation or to be elected as a delegate who can be qualified to create a block, a random value generation process through TEE is required. A monotonic counter is used to prevent concurrent invocation or a remote attestation to verify operation. Also, the Trusted Time Service (TTS)

provides a reliable verification method for whether a certain amount of waiting time has been passed.

For these reasons, TEE is required to participate in the blockchain network. In addition, even if we have a large number of CPUs with TEEs, malicious actions such as controlling the network with the majority of TEEs in the network are close to impossible due to the price of TEEs.

However, in our current implementation, the actual TEE is not used and is implemented in the form of an interface. Measuring performance by applying real TEE remains a follow-up study. The details of the functions of the TEE used are as follows.

Trusted Time TEE has a reliable time service. This allows reliable elapsed time measurements. Set the time at which the consensus algorithm starts through TEE as *round time*. *Sleep* for as much time as round time to let the time pass. After that, it verifies the block of this round and whether the time passed as much as round time.

Through this process, nodes have a deterministic block verification time, so that block creation time can be synchronized without additional operation. Because sleep is busy-wait, other work is available during this time, so time and energy are not wasted. Also, if another luckier chain is broadcasted while busy waiting, it can be changed to that chain. In NS-3 [3], this function can be provided by giving the Delay of Schedule as much as round time.

After verifying the block of the previous round, it is verified whether the round time has passed during the verification process. Through this process, nodes have a deterministic block verification time, so that block creation time can be synchronized without additional operation. By doing busy-wait, sleep is performed during the round time, and other tasks are possible during this time, so time and energy are not wasted.

Monotonic Counter A malicious user may try to gain an unfair advantage by running an algorithm concurrently on a single CPU. If the value of TEE monotonic counter is increased every time, concurrent invocation can be prevented. To do this, the monotonic counter value is stored at the time of starting the algorithm. After that, it is compared with the value of the monotonic counter when verifying the block. If the two monotonic counter values are different, it means that the algorithm was executed in parallel on the same CPU. In this case, the monotonic counter verification fails and the node cannot create a block.

Remote Attestation Algorithm 1 shows the remote attestation function.

Remote Attestation is the process of generating a *proof*, a value that nodes can verify with each other. Proof is used as a record of whether an algorithm or data has been manipulated. Because of attestation, the operation process and data in TEE cannot be manipulated.

[3] https://www.nsnam.org/

Algorithm 1: Remote Attestation.

```
1: function RemoteAttestation(nonce, luck)
2:     input ← nonce || luck
3:     proof ← Base58-Encoding(input)
4:     return proof
5: end function
```

Proof is a value generated by Base58-Encoding the concatenated data of *nonce*, which is a hash value of the block header to be generated, and *luck*, which is a random value generated through RDRAND of TEE. It is included in the block and broadcast at the end of the consensus algorithm process. The node extracts the nonce and luck by Base58-decoding the proof contained in the broadcast block. Through the extracted data, it is verified whether the value of proof has been tampered with or whether it is a value generated from the correct nonce and luck.

As a result, it is possible to prove that no block or transaction has been manipulated. It also allows nodes to communicate with each other even if they do not have a trust relationship.

Random Number Generation TEE can produce reliable random numbers, which an attacker cannot influence. A reliable random value is generated using RDRAND of Intel SGX and used as the luck value in the algorithm. This luck value is used as a value for voting or when deciding who will be the block generator in the consensus algorithm. The node with the block with the highest luck value is selected as the block producer.

3.3 Proof of Luck with Delegated Approach

PQ-DPoL, a post-quantum consensus algorithm proposed in this work, consists of *a delegation phase* and *a consensus phase*. Figure 1 shows the overview of PQ-DPoL.

In the delegation phase, the delegated node to be the delegate is selected through the voting of all nodes in the blockchain. In the consensus phase, delegated nodes take turns generating blocks. Here, the transactions included in the generated block include the signature and public key of `Dilithium`. Finally, after delegated nodes verify the block, the verified block is added to the chain.

Delegation Figure 2 shows the *delegation phase* of PQ-DPoL. The details of the delegation process are as follows:

1. **Random Number Generation** Each node generates a random number. This random number is used as a vote value.
2. **Vote** Each node broadcasts a voting transaction. A voting transaction contains a random number (vote) and the signature and public key of `Dilithium`.

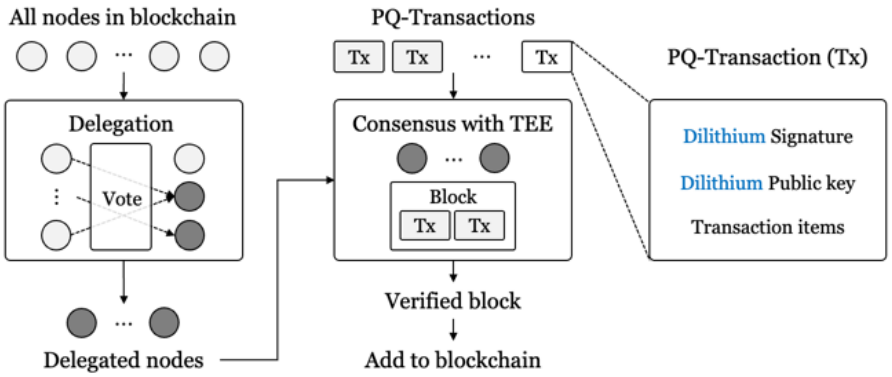

Fig. 1: Overview of PQ-DPoL.

3. **Verification** Each node receives voting transactions of other nodes and verifies the voting transaction with `Dilithium`. If the signature verification succeeds, the voting transaction is considered valid, indicating a valid vote. If the signature verification fails, it means that the transaction has been manipulated (i.e., the vote is invalid).
4. **Delegation** The number of votes received is summed up. The top n nodes with the most votes are then selected as delegated nodes.

Since we apply `Dilithium`, the TPS decreases because of the large signature size, leading to fewer transactions that can be included in one block. However, in the delegation approach, only some nodes in the blockchain perform consensus. In other words, since the number of nodes participating in the consensus is reduced, the time taken for all nodes to verify is reduced. As a result, the time required for consensus is reduced.

Thanks to the delegation phase we overcome the degraded TPS due to `Dilithium`. Further, there is the advantage of improving scalability [15], which is another important factor of blockchain.

Consensus with TEE Figure 3 shows the consensus phase of PQ-DPoL. The details of the consensus process are as follows:

1. **Block Generation** In order to generate a block, the integrity of the block must be verified. Also, PoL verifies that *round time* has passed and that the monotonic counter has not changed. If the current time is greater than round time + consensus start time, then round time has passed. And, if the monotonic counter set based on the TEE and the current monotonic counter called in the consensus process are not the same, verification is failed. After verification is completed, *luck* (random number) is generated using TEE, and *proof* is generated in the Remote Attestation process. Finally, the block is broadcast to other delegated nodes.

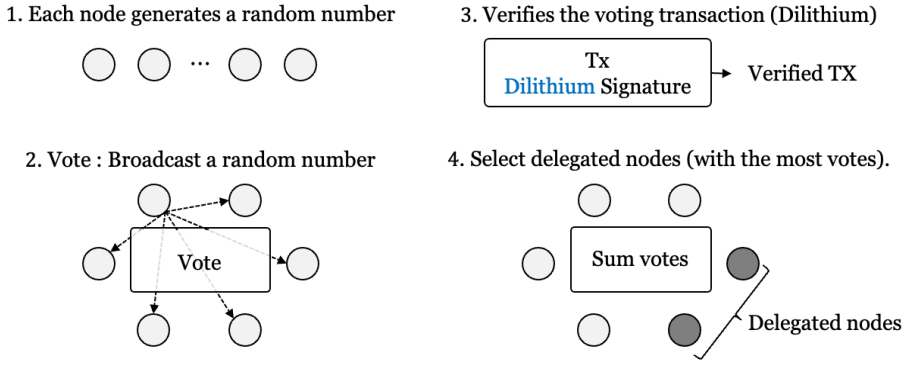

Fig. 2: Delegation phase of PQ-DPoL.

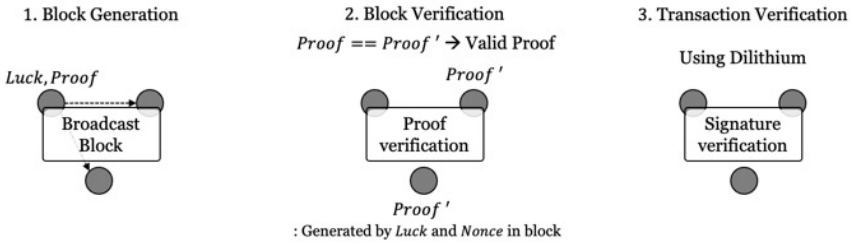

Fig. 3: Consensus phase of PQ-DPoL.

2. **Block Verification** In the block verification process, verification of nonce, the previous hash, and proof are performed. It must be verified that the Nonce generated by hashing the header of the block to be created is the same as the nonce extracted from proof (note that the nonce is the hash of the header of the current block). This process ensures the integrity of nonce.

   To verify the proof, the proof contained in the received block is compared with the *proof'* (new proof) generated in the consensus process. Here, proof' is a string encoded after concatenating luck and nonce of the block. In other words, it means verifying that the received proof is actually created from luck and nonce. Finally, the block verification process is completed by comparing the previous hash with the previous hash included in the block to be generated.

3. **Transaction Verification**

   Finally, verification for the transactions contained in the block must be performed. Delegated nodes verify the signature of transactions included in the received block. In this process, as mentioned in the delegation phase (Section 3.3), the signature verification process using `Dilithium` is performed.

In this way, TEE-based verification is performed in the consensus phase. In other words, this consensus phase is a process of verifying the values of the elements included in the block (such as luck, nonce, and proof) to ensure that a legitimate block is generated. As noted earlier, the number of transactions included in a block can be decreased due to `Dilithium` for transaction verification. However, since only delegated nodes participate in consensus, the consensus time can be reduced. Thus, by using post-quantum lightweight consensus, we can ensure security while mitigating the decrease in TPS due to the large signature size.

4 Benchmark

4.1 Environment and Evaluation Metrics

Environment C++ language and an Intel i5-8295U CPU with 16GB RAM on Ubuntu 20.04.6 LTS are used in our experiment. To build an environment close to a real blockchain network, we use NS-3, an open-source network simulator.

Performance This section presents the execution speed and performance (TPS, Latency) using `ECDSA` (`P-256`) and `Dilithium`. We use TPS and latency as performance metrics. TPS is measured identically during delegation and consensus (TPS_D, TPS_C). However, latency is defined in delegation and consensus respectively. Here are the details. Delegation latency (L_D) refers to the time from transaction generation to transaction verification. Consensus latency (L_C) is the time it takes for delegated nodes to complete block verification after a block containing a transaction is generated. We measure TPS_D, TPS_C, L_D, and L_C depending on the size of the blockchain network (the number of nodes is represented by 2^n) and the type of `Dilithium` applied.

Table 2: Execution speed comparison of `ECDSA` (`P-256`) and `Dilithium`.

Algorithm	Key Gen	Sig Gen	Sig Verify
`ECDSA (P-256)`	0.000400	0.000670	0.000350
`Dilithium-2`	0.000027	0.000100	0.000026
`Dilithium-3`	0.000047	0.000160	0.000043
`Dilithium-5`	0.000073	0.000200	0.000065

block size: 25 KB, unit: second, $n = 1$

– **Execution speed comparison of `ECDSA` and `Dilithium`** Table 2 shows the performance of the signature scheme in the blockchain network. Key generation time, signature generation time, and signature verification time of `ECDSA` and `Dilithium-2/3/5` are measured. `Dilithium` is faster than `ECDSA`. However, when the block size is 25KB, the number of transactions

included in one block is 100 and 6/4/3 in `ECDSA` and `Dilithium-2/3/5`, respectively. This is because `Dilithium` has a large key size and signature size as described in Section 2.3.

Table 3: TPS of DPoL.

n	1	2	3	4	5	6	7
`ECDSA`	713.5797	122.5398	29.8184	6.9522	0.9468	0.2341	0.1030
`Dilithium-2`	88.8721	21.0373	5.1244	1.2515	0.3302	0.0776	0.0171
`Dilithium-3`	48.4901	11.7897	2.8879	0.7128	0.1815	0.0446	0.0104
`Dilithium-5`	30.8097	7.4470	1.8600	0.4526	0.1203	0.0291	0.0066

block size: 25 KB, unit: second

– **TPS** Despite the faster signing speed, the performance of `Dilithium` in terms of TPS is relatively low compared to `ECDSA` due to the large size of `Dilithium`. As a result, the time is delayed because more transactions are included in the block to which `ECDSA` is applied, but the TPS is measured higher due to a large number of transactions.

Table 4: Latency of DPoL.

n	1	2	3	4	5	6	7
`ECDSA`	0.1401	0.8160	3.3536	14.3838	105.6131	427.1180	970.0098
`Dilithium-2`	0.0675	0.2852	1.1708	4.7940	18.1667	77.2573	350.8609
`Dilithium-3`	0.0824	0.3392	1.3850	5.6116	22.0333	89.5175	383.3946
`Dilithium-5`	0.0973	0.4028	1.6128	6.6277	24.9267	103.0436	454.1221

block size: 25 KB, unit: second

– **Latency** It is observed that the TPS decreases as the number of nodes increases (see Tables 3 and 4). Due to the limitations of the hardware environment, the performance when the number of nodes is 512 can not be measured, but based on the previous measurement results, it can be assumed that the TPS will decrease. Thus, the delegate is applied to solve the problem related to the size of `Dilithium` and the decrease in TPS as the number of nodes increases. The delegated performance measurement result is the same as 3 and 4. By reducing the number of nodes with a delegated approach, the time required for consensus is decreased (i.e., TPS increases).

5 Conclusion

In this work, we gather multiple contributions, including a post-quantum signature scheme and a novel consensus algorithm, to construct an efficient quantum-

secure blockchain. Additionally, we make a detailed attempt to ensure compatibility and optimal performance for diverse environments.

In summary of our experimental results, the TPS of DPoL applied with `Dillithium` is lower than that of `ECDSA` (since larger key and signature sizes). To improve the degraded performance, we propose DPoL, a new consensus algorithm that combines PoL and a delegation approch. Certainly, when employing `Dillithium` as the signature scheme in DPoL, although its performance may be lower than that of `ECDSA`, it still provides reasonable performance.

Finding/Adopting various methods to improve TPS can be considered for the future works. It would be advisable to replace various PQC signature schemes and conduct comparative analyses with other consensus algorithms.

6 Acknowledgements

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No.2018-0-00264, Research on Blockchain Security Technology for IoT Services, 50%) and this work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2022-0-00627, Development of Lightweight BIoT technology for Highly Constrained Devices, 50%).

References

1. P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999. 1
2. L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pp. 212–219, 1996. 2
3. M. Amy, O. Di Matteo, V. Gheorghiu, M. Mosca, A. Parent, and J. Schanck, "Estimating the cost of generic quantum pre-image attacks on sha-2 and sha-3," in *International Conference on Selected Areas in Cryptography*, pp. 317–337, Springer, 2016. 2
4. J. Lee, S. Lee, Y.-S. Lee, and D. Choi, "T-depth reduction method for efficient sha-256 quantum circuit construction," *IET Information Security*, vol. 17, no. 1, pp. 46–65, 2023. 2
5. W.-K. Lee, K. Jang, G. Song, H. Kim, S. O. Hwang, and H. Seo, "Efficient implementation of lightweight hash functions on gpu and quantum computers for iot applications," *IEEE Access*, vol. 10, pp. 59661–59674, 2022. 2
6. NIST., "Submission requirements and evaluation criteria for the post-quantum cryptography standardization process," 2016. https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf. 2
7. L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, "Crystals-dilithium: A lattice-based digital signature scheme," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 238–268, 2018. 2

8. G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014. 3
9. F. Yang, W. Zhou, Q. Wu, R. Long, N. N. Xiong, and M. Zhou, "Delegated proof of stake with downgrade: A secure and efficient blockchain consensus algorithm with downgrade mechanism," *IEEE Access*, vol. 7, pp. 118541–118555, 2019. 3
10. L. Chen, L. Xu, N. Shah, Z. Gao, Y. Lu, and W. Shi, "On security analysis of proof-of-elapsed-time (poet)," in *Stabilization, Safety, and Security of Distributed Systems: 19th International Symposium, SSS 2017, Boston, MA, USA, November 5–8, 2017, Proceedings 19*, pp. 282–297, Springer, 2017. 3
11. S. M. H. Bamakan, A. Motavali, and A. B. Bondarti, "A survey of blockchain consensus algorithms performance evaluation criteria," *Expert Systems with Applications*, vol. 154, p. 113385, 2020. 3
12. M. Salimitari and M. Chatterjee, "A survey on consensus protocols in blockchain for iot networks," *arXiv preprint arXiv:1809.05613*, 2018. 3
13. M. Raavi, P. Chandramouli, S. Wuthier, X. Zhou, and S.-Y. Chang, "Performance characterization of post-quantum digital certificates," in *2021 International Conference on Computer Communications and Networks (ICCCN)*, pp. 1–9, IEEE, 2021. 4
14. S. Cen and B. Zhang, "Trusted time and monotonic counters with intel software guard extensions platform services," *Online at: https://software. intel. com/sites/default/files/managed/1b/a2/Intel-SGX-Platform-Services. pdf*, 2017. 5
15. A. I. Sanka and R. C. Cheung, "A systematic review of blockchain scalability: Issues, solutions, analysis and future research," *Journal of Network and Computer Applications*, vol. 195, p. 103232, 2021. 9