Evaluation of the BLS12-381 Signature Aggregation Scheme for Multi-Signature Validation Efficiency in Proof-of-Stake Blockchain Systems

Rasyid Rizky Susilo Nurdwiputro - 18223114
Information System & Technology Study Program
School of Electrical Engineering and Informatics
Institut Teknologi Bandung, Ganesha Street No. 10 Bandung
E-mail: rasyid.rsn@gmail.com, 18223114@std.stei.itb.ac.id

***Abstract*—The scalability of Proof-of-Stake (PoS) blockchains is fundamentally constrained by the computational latency and storage overhead inherent in multi-signer verification processes. This paper evaluates the BLS12-381 signature aggregation scheme as a cryptographic solution to this scalability bottleneck. By engineering a vanilla Node.js simulation framework, we empirically benchmarked the execution time and data footprint of traditional linear validation against aggregated verification across simulated network scales of up to 1000 validators. The results indicate that traditional linear verification scales at O(n), demanding nearly 76.6 seconds to validate 1000 nodes. Conversely, the BLS12-381 protocol successfully compresses N signatures and public keys into a constant 144-byte aggregated payload, yielding a 99.9% reduction in storage overhead at peak scale. Most notably, the final pairing check of the aggregated signature executed in a constant O(1) time of approximately 60 milliseconds, isolating the O(n) complexity purely to lightweight point additions. These findings empirically validate that BLS12-381 signature aggregation significantly mitigates the multi-signature scalability bottleneck, providing a highly efficient and mathematically robust consensus foundation for next-generation decentralized networks.**

Keywords—component; BLS12-381, Signature Aggregation, Proof-of-Stake, Blockchain Scalability, Elliptic Curve Cryptography.

I. Introduction

The rapid growth of decentralized systems has accelerated the transition of blockchain consensus mechanisms from the energy-intensive Proof-of-Work (PoW) model to the more efficient Proof-of-Stake (PoS) paradigm. In PoS networks, consensus relies on the participation of numerous validator nodes, each of which must attest to the validity of a proposed block through a digital signature. Although this multi-validator approach enhances decentralization and fault tolerance, it also creates significant computational and storage overhead during signature verification.

In conventional digital signature schemes such as ECDSA and RSA, each validator produces an independent signature that must be stored and verified separately. As the number of validators increases, the network must process and maintain a growing collection of signatures, causing verification costs and storage requirements to scale linearly with the validator set size. This scalability issue can reduce transaction throughput, increase latency, and enlarge block sizes, limiting the efficiency of PoS systems.

To address these challenges, this research investigates the implementation and performance of a signature aggregation framework based on the Boneh–Lynn–Shacham (BLS) scheme over the BLS12-381 pairing-friendly elliptic curve. Through signature aggregation, multiple validator signatures can be combined into a single constant-size aggregated signature, significantly reducing communication and storage overhead. Furthermore, aggregated verification can decrease the number of verification operations required compared with traditional individual signature verification.

Despite these advantages, the bilinear pairing operations used in BLS12-381 are computationally more expensive than the elliptic curve operations employed by ECDSA. Therefore, the main objective of this study is to empirically evaluate the performance of aggregated BLS verification relative to individual BLS signature verification. Using a standalone implementation in vanilla Node.js, the research measures execution latency and signature size reduction across various validator set sizes. The results are expected to identify the point at which signature aggregation becomes more efficient than individual verification, providing practical evidence for the adoption of BLS aggregation in scalable PoS blockchain systems.

II. Background

A. Proof-of-Stake Consensus Mechanism

Proof-of-Stake (PoS) has emerged as a widely adopted consensus mechanism for modern blockchain networks, offering a more energy-efficient alternative to Proof-of-Work (PoW) [4]. Instead of relying on computational mining, PoS selects block proposers and validators according to the amount of cryptocurrency they commit as collateral, commonly referred to as their stake.

In a typical PoS protocol, a selected validator proposes a new block containing a set of transactions and broadcasts it to other validators. These validators examine the proposed state

transition and issue cryptographic attestations in the form of digital signatures. A block is considered finalized once it receives approval from a supermajority of the active voting stake, commonly exceeding two-thirds of the total stake [7]. While this mechanism improves decentralization and fault tolerance, it also generates a large number of signatures that must be transmitted, stored, and verified for every block.

When the validator set grows, the communication and verification overhead associated with these signatures can become a significant scalability challenge. Consequently, efficient signature management techniques are required to maintain high throughput and low latency in PoS-based blockchain systems [8].

B. Elliptic Curve Cryptography & Bilinear Pairings

Elliptic Curve Cryptography (ECC) forms the cryptographic foundation of many modern blockchain protocols. Compared with traditional RSA-based systems, ECC provides equivalent security while requiring substantially smaller key sizes. ECC is defined over a finite field (F_q), where an elliptic curve is commonly represented by the short Weierstrass equation:

$$y^2 = x^3 + ax + b$$

Cryptographic operations utilize the abelian group structure formed by the points on the curve. A private key is defined as a scalar integer sk chosen randomly from a prime-order field Z_r, while the corresponding public key PK is generated via scalar multiplication of sk with a base generator point G, expressed as $PK = sk \cdot G$. The security of this paradigm relies entirely on the intractability of the Elliptic Curve Discrete Logarithm Problem (ECDLP).

To enable signature aggregation, ECC is extended through Pairing-Based Cryptography (PBC) [1]. A bilinear pairing is a mapping between two elliptic-curve groups and a target multiplicative group:

$$e : G_1 \times G_2 \rightarrow G_T$$

The mapping function exhibits three fundamental mathematical properties:

1) *Bilinearity*: For any points $P \in G_1$, $Q \in G_2$, and scalar fields $a, b \in Z_r$, the pairing function allows scalars to be factored out or shifted dynamically:

$$e\,(a \cdot P, b \cdot Q) = e\,(P, Q)^{ab} = e\,(ab \cdot P, Q) = e\,(P, ab \cdot Q)$$

2) *Non-degeneracy*: If g_1 and g_2 are generators of G_1 and G_2 respectively, then $e\,(g_1, g_2) \neq 1 \in G_T$.

3) *Computability*: There exists an efficient algorithmic implementation (such as Miller's algorithm combined with final exponentiation) to compute $e\,(P, Q)$ in a practical timeframe.

C. BLS12-381 Curve

The performance and mathematical validity of bilinear pairings are heavily dependent on the parameters of the underlying elliptic curve. The BLS12-381 curve is a pairing-friendly elliptic curve specifically engineered to maximize pairing efficiency while maintaining a robust security baseline [3]. Constructed within the Barreto-Lynn-Scott family of curves, the numerical designators denote its structural properties: an embedding degree (k) of 12, and a base field size of approximately 381 bits.

Mathematically, the curve is defined over a prime field F_q where the prime q is exactly 381 bits in length, and the curve equation is instantiated as:

$$y^2 = x^3 + 4$$

The embedding degree of 12 ensures a mathematically optimal balance between the group size of G_1 (defined over the base field F_q) and G_2 (defined over the extension field F_{q^2}). This structural alignment allows BLS12-381 to yield a targeted security level of 128 bits, providing an estimated 128-bit security level against currently known classical attacks on elliptic-curve and finite-field discrete logarithm problems in extension fields, while offering significantly faster pairing evaluation times than older curve standards like Barreto-Naehrig (BN) curves [3].

D. Signature Aggregation Concept

The Boneh-Lynn-Shacham (BLS) cryptographic signature scheme leverages the algebraic properties of the BLS12-381 curve to achieve non-interactive signature aggregation [2], [5]. Unlike traditional ECDSA schemes which generate non-combinable outputs, the mathematical architecture of BLS operates deterministically through the following primitive workflows:

1) *Key Generation*: A validator generates a private key $sk \in Z_r$. The public key PK is computed as a point on the G_2 group by multiplying the private key with the G_2 base generator g_2:

$$PK = sk \cdot g_2 \in G_2$$

2) *Signing*: To attest to a message m, the arbitrary binary string of the message is mapped onto a coordinate point on the G_1 curve using a deterministic hash-to-curve function, yielding $H(m) \in G_1$. The individual signature S is generated by scalar multiplication:

$$S = sk \cdot H(m) \in G_1$$

3) *Aggregation*: Given a set of n independent validators signing the exact same message m, an aggregator module combines the individual public keys $(PK_1, PK_2, \ldots, PK_n)$ and individual signatures

$(S_1, S_2, ..., S_n)$ through simple elliptic curve point addition. Due to the closure property of elliptic curve groups, the summation compresses the data into a single aggregated signature S_{agg} and a single aggregated public key PK_{agg} [8]:

$$S_{agg} = \sum_{i=1}^{n} S_i \in G_i$$

$$PK_{agg} = \sum_{i=1}^{n} PK_i \in G_2$$

4) *Aggregated Verification*: Rather than executing n sequential validation routines, a verifier utilizes the bilinearity property of the curve to confirm the integrity of all n attestations simultaneously. The verifier validates the entire consensus block by checking a single pairing equation:

$$e(g_2, S_{agg}) = e(PK_{agg}, H(m))$$

This identity holds true because the algebraic transformations map perfectly on both sides of the equation:

$$e(g_2, \sum S_i) = e(g_2, \sum(sk_i \cdot H(m))) = e(g_2, H(m))^{\sum sk_i}$$

$$e(PK_{agg}, H(m)) = e(\sum(sk_i \cdot g_2), H(m)) = e(g_2, H(m))^{\sum sk_i}$$

Consequently, signature aggregation reduces the number of verification operations from n individual signature verifications to a constant number of pairing evaluations for a common-message setting, while also compressing storage requirements into a single aggregated signature [8].

III. METHODOLOGY

A. System Architecture

The proposed cryptographic framework is designed to simulate a high-throughput, multi-signer verification environment typical of scalable Proof-of-Stake (PoS) blockchain networks. The architecture isolates the computational mechanics of the BLS12-381 signature scheme into a structured, three-tiered pipeline: the Key Initialization Phase, the Attestation and Aggregation Phase, and the Centralized Verification Phase. The operational flow of the system ensures that multi-signer validation scales efficiently by reducing the number of verification operations through signature aggregation while maintaining a constant-size aggregated signature.

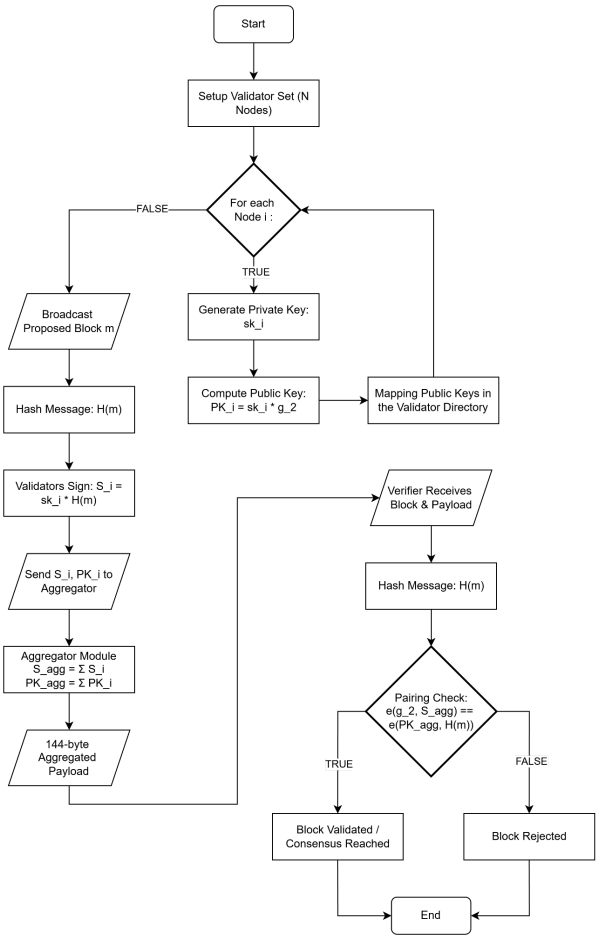

Fig. 1. System Architecture Flowchart

1) *Key Initialization Phase*: The system establishes a simulated validator set consisting of n independent nodes. Each validator node i independently generates its cryptographic identity. A private key (sk_i) is sampled as a cryptographically secure random integer within the scalar field of the curve, denoted as $sk_i \in Z_r$. The corresponding public key (PK_i) is computed by performing scalar multiplication of the private key with the base generator point g_2 of the second source group (G_2), resulting in $PK_i = sk_i \cdot g_2 \in G_2$. Public keys are mapped and indexed within a public validator directory to represent the active consensus set.

2) *Attestation and Aggregation Phase*: Upon the creation of a proposed block or state transition message m, the message is broadcasted to all active validators. Every participating node attests to the validity of m by first mapping the binary string of the message onto a point on the first source group (G_1) using a secure hash-to-curve function, yielding $H(m) \in G_1$. Each validator then generates an

individual digital signature (S_i) by multiplying the hashed message point with its private key, expressed as:

$$S_i = sk_i \cdot H(m) \in G_1$$

An aggregator module subsequently collects the individual signatures and public keys from the participating nodes. Utilizing the algebraic properties of elliptic curve point addition, the aggregator combines the n distinct signatures into a single aggregated signature ($S_{agg} \in G_1$) and computes a corresponding aggregated public key ($PK_{agg} \in G_2$) via the following equations:

$$S_{agg} = \sum_{i=1}^{n} S_i$$

$$PK_{agg} = \sum_{i=1}^{n} PK_i$$

The study assumes that all validator public keys are registered and trusted prior to aggregation. Protection mechanisms against rogue-key attacks, such as Proof of Possession (PoP), are outside the scope of this performance evaluation.

3) *Verification Phase*: The core efficiency of the architecture is realized in the verification tier, where any node auditing the blockchain ledger can validate the collective attestations of the entire validator set simultaneously. Instead of sequentially parsing n individual signatures, the verifier evaluates a constant number of bilinear pairing equations. The system computes and checks the mathematical equivalence of two pairing functions:

$$e(g_2, S_{agg}) = e(PK_{agg}, H(m))$$

If the pairing identity holds true, the cryptographic integrity of the block and the authenticity of all contributing signers are verified in a single aggregate verification procedure, significantly reducing the verification overhead associated with validating individual signatures separately.

B. Implementation Setup

The experimental simulation is constructed to isolate and evaluate the cryptographic performance of BLS12-381 signature aggregation against sequential validation methods. To ensure that the recorded benchmarks reflect the pure algorithmic overhead of the cryptographic primitives rather than network latencies, consensus delays, or framework-specific abstraction layers, the entire evaluation framework is developed using vanilla JavaScript within the Node.js runtime environment. Writing the core consensus and evaluation pipeline without heavy external software architectures guarantees execution transparency and prevents high-level runtime optimizations from obscuring the underlying mathematical complexities of the protocol.

The implementation setup adheres to the following configurations and specifications:

1) *Runtime and Language Environment*: The simulation is executed on the Node.js long-term support (LTS) platform using native ECMAScript modules. The choice of a vanilla programming approach allows for direct control over data structures, loop iterations, and synchronous execution blocks, which are critical for capturing unbiased processing metrics during the benchmarking cycles.
2) *Cryptographic Primitives*: While the higher-level logic of validator identity management, signature collection, and aggregation loops is written from scratch, the low-level elliptic curve mathematics, such as scalar point multiplication on G_1 and G_2, point addition, and the computation of bilinear pairings, are handled via a lightweight, zero-dependency cryptographic library, specifically `@noble/bls12-381`, optimized for the BLS12-381 curve. This ensures mathematical correctness and cryptographic validity while maintaining a minimal operational footprint.
3) *Measurement Metrics and Instrumentation*: High-resolution execution timing is instrumented using the native Node.js Performance Measurement API (`performance.now()`), providing high-resolution timing measurements when isolating the exact duration of signature generation, aggregation, and verification phases. To quantify storage and data propagation efficiency, public keys and signatures are serialized into raw byte arrays, and their physical footprint is measured directly using the `Buffer.byteLength()` property.
4) *Hardware Baseline*: To maintain experimental consistency and minimize background noise from concurrent operating system threads, all simulation instances are executed sequentially on a single, isolated hardware baseline (e.g., AMD Ryzen 7 7435HS, 16 GB RAM, running a win32-based operating system). Each distinct test case is repeated across multiple independent iteration loops to compute reliable statistical averages and filter out transient CPU frequency scaling anomalies.

C. Experimental Design

To systematically evaluate the performance overhead and efficiency gains of the BLS12-381 signature aggregation protocol, a controlled experimental methodology is designed. The experiment aims to benchmark the aggregated verification model against the traditional linear verification model (representing an unaggregated multi-signature verification workflow) across varying network scales.

1) Experimental Variables

The experiment is governed by one primary independent variable: the scale of the validator set, denoted as N. To simulate both small-scale consensus committees and large-scale decentralized networks, the testing parameters for N are predefined at

intervals of 10, 50, 100, 250, 500, and 1000 simulated nodes.

The dependent variables measured during the execution of each validator set scale are:

- *Total Execution Latency (ms)*: The absolute time required to validate the attestations of N validators.
- *Cryptographic Data Footprint (bytes)*: The total physical memory size of the signatures that must be stored or propagated across the network.

2) Testing Scenarios

For each defined scale of N, the system executes two distinct testing scenarios to establish a comparative baseline:

- *Scenario A*: Linear Verification (Baseline). The system simulates a traditional validation flow where N distinct signatures are verified sequentially against N public keys. The performance timer records the cumulative latency of executing N individual BLS signature verification operations. The data footprint is calculated as *N x (size of a single signature + size of a single public key)*.
- *Scenario B*: Aggregated Verification (Proposed). The system executes the BLS12-381 aggregation protocol. The performance timer is divided into two sub-metrics: Aggregation Time (the time required to aggregate N public keys and N signatures) and Verification Time (the time required to perform the aggregated signature verification procedure using a constant number of bilinear pairing evaluations). The Total Execution Latency is computed as the sum of these two sub-metrics. The data footprint is measured as the size of the aggregated signature S_{agg} and the aggregated public key PK_{agg}.

3) Execution Protocol and Data Normalization

To mitigate runtime anomalies, such as Node.js garbage collection pauses or CPU cold-start delays, a rigorous execution protocol is enforced. Before data collection begins, the simulation performs a "warm-up" phase of 50 unrecorded iterations to stabilize the JavaScript Just-In-Time (JIT) compiler. Subsequently, for each scale of N, both Scenario A and Scenario B are executed for 100 independent iterations. The highest and lowest 5% of the recorded latencies are discarded as statistical outliers. The final benchmark data presented in the results represents the normalized arithmetic mean of the remaining iterations.

IV. Results and Analysis

The experimental simulation successfully benchmarked the performance of BLS12-381 signature aggregation against traditional linear verification across six scaling validator sets (N = 10, 50, 100, 250, 500, 1000). The recorded metrics isolate the algorithmic execution time and the physical data footprint, providing empirical evidence of the cryptographic trade-offs inherent in multi-signer consensus models.

TABLE I. BLS12-381 Signature Aggregation Benchmark Results

N	Linear (ms)	Aggregation Time (ms)	Aggregated Verification (ms)	Total Aggregated Route (ms)	Linear Payload (bytes)	Aggregated Payload (bytes)
10	606.5166	83.1739	59.4482	142.6221	1440	144
50	2976.8724	379.3512	59.0941	438.4453	7200	144
100	6047.861	753.5167	60.891	814.4078	14400	144
250	18743.0171	2367.6877	73.4224	2441.1101	36000	144
500	30829.8428	3839.5897	61.2334	3900.8231	72000	144
1000	76614.2229	9427.853	75.1072	9502.9602	144000	144

A. Computational Time Evaluation

The execution latency data vividly illustrates the O(n) computational bottleneck of linear verification. As the validator set expanded from 10 to 1000 nodes, the linear verification time grew proportionally, surging from 606.52 ms to a staggering 76614.22 ms (approximately 76.6 seconds). Such latency would become a significant performance concern in large-scale PoS environments.

In contrast, the Aggregated Verification protocol (Scenario B) demonstrated a massive reduction in total execution latency. For N = 1000, the total aggregated route (which includes both the mathematical aggregation of keys/signatures and the final pairing check) was completed in 9,502.96 ms, performing approximately 8.1 times faster than the linear verification approach.

The most critical finding from the performance evaluation lies in the Single Agg Verify metric. While the initial aggregation process (point addition on the elliptic curve) scales linearly based on N, the actual verification of the resulting aggregated signature remains strictly constant at O(1). Regardless of whether the signature represents 10 or 1000 validators, the bilinear pairing verification consistently executed in ~59 to ~75 ms. This proves that once a block proposer aggregates the signatures, every other node auditing the blockchain only spends ~60 ms to cryptographically verify the consensus of the entire network.

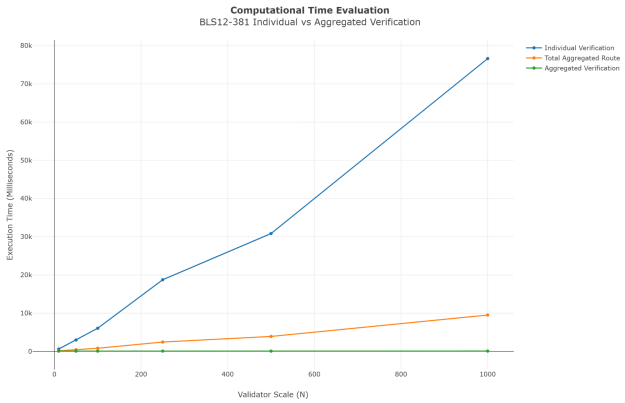

Fig. 2. Computational time evaluation of individual linear verification vs BLS12-381 aggregated verification.

B. Data Size Efficiency

The storage footprint analysis further justifies the superiority of BLS12-381 aggregation. In the traditional linear model, each validator appends a 48-byte signature and a 96-byte public key (totaling 144 bytes per attestation) to the block. Consequently, the storage overhead scales linearly, reaching 72,000 bytes for 500 validators and peaking at 144,000 bytes for 1000 validators.

By employing signature aggregation, the mathematical properties of the BLS curve allow N signatures to be compressed into a single valid point on the G_1 curve. The experimental data confirms that the aggregated data footprint remains strictly constant at 144 bytes (comprising a 48-byte compressed signature and a 96-byte compressed aggregated public key), regardless of the validator scale. At N = 1000, this yields a phenomenal 99.9% reduction in storage overhead.

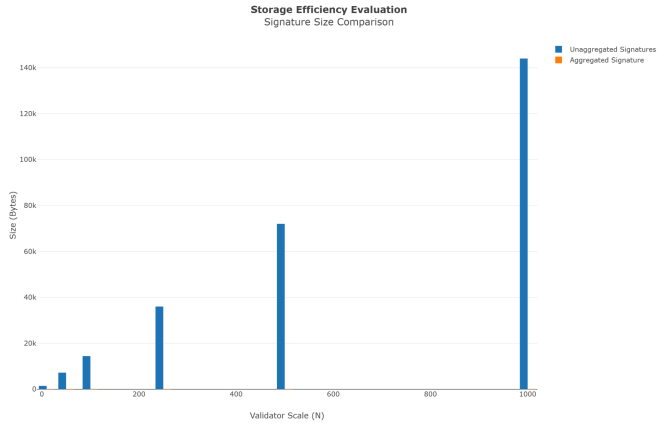

Fig. 3. Storage overhead comparison of aggregated payloads vs unaggregated signatures.

C. Blockchain Scalability

The empirical results generated from this simulation directly answer the scalability problem formulated in Section I. In modern PoS architectures like Ethereum 2.0, where thousands of validators must attest to a block's validity, linear signature verification is practically and economically infeasible.

The data confirms that BLS12-381 signature aggregation resolves both the bandwidth/storage limitations and the CPU execution bottlenecks. By reducing the data footprint to a constant 144 bytes, network bandwidth consumption for block propagation is minimized, and the archival storage requirements for ledger history are drastically reduced. Furthermore, by transforming the O(n) validation latency into an O(1) pairing check of ~60 ms, the network can verify consensus attestations highly efficiently, allowing the consensus mechanism to scale seamlessly as the validator set grows.

V. CONCLUSION

This paper presented an empirical performance evaluation of the BLS12-381 signature aggregation scheme to address the inherent scalability bottlenecks of multi-signer verification in Proof-of-Stake (PoS) blockchains. Through a controlled, vanilla Node.js simulation, the execution latency and cryptographic data footprint were benchmarked across varying validator scales ranging from 10 to 1000 nodes.

The experimental findings clearly demonstrate the unsustainability of traditional linear verification, which exhibited an O(n) latency growth, requiring approximately 76.6 seconds and 144,000 bytes of storage to process 1000 unaggregated credentials. In stark contrast, the implementation of BLS12-381 signature aggregation successfully compressed the multi-signature data into a single, constant 144-byte aggregated payload, achieving a 99.9% reduction in storage overhead for 1000 validators. Furthermore, while the aggregation process itself scales linearly, the ultimate verification of the aggregated signature proved to operate in constant O(1) time, consistently executing in approximately 60 milliseconds regardless of the network scale.

These empirical results mathematically and practically justify the adoption of BLS12-381 signature aggregation in modern decentralized architectures. Based on the performance evaluation, it is recommended to consider the following aspects when designing scalable PoS consensus mechanisms:

1) *Adopt Signature Aggregation for Large-Scale Sets*: As demonstrated, networks anticipating validator sets exceeding hundreds of nodes must transition from linear verification to aggregation schemes to prevent network latency and ledger bloat. The constant 144-byte payload ensures seamless bandwidth scaling.
2) *Implement Rogue-Key Attack Mitigations*: While the pairing check operates in O(1) time, the mathematical integrity relies on the assumption that all public keys are distinct and verified. Developers must pair this aggregation scheme with robust registration protocols, such as requiring a Proof of Possession (PoP), to prevent rogue-key vulnerabilities during the O(n) aggregation phase.
3) *Optimize the Final Pairing Check*: Although the verification time is constant (~60 ms), bilinear pairings remain computationally heavy operations.

For resource-constrained client nodes, implementing hardware-accelerated elliptic curve pairings or WebAssembly (WASM) optimizations is highly recommended to further minimize the absolute verification latency.

These findings provide blockchain architects with practical insights to strike the optimal balance between computational efficiency, storage minimization, and robust cryptographic security. Potential options for future research include investigating the integration of Zero-Knowledge proofs to further compress the aggregation phase or evaluating the protocol's resilience under adversarial network latency conditions.

Source Code Repository At Github

https://github.com/rasyidrizky/bls12-381-signature-experiment

Video Link at Youtube

https://youtu.be/EWLjt3CHFss

Acknowledgment

The author would like to express sincere gratitude to Dr. Ir. Rinaldi Munir, M.T. for his invaluable guidance, insightful lectures, and continuous support throughout the II4021 Cryptography course, which provided the fundamental knowledge and inspiration necessary to complete this research. Special appreciation is also extended to the course teaching assistants, Ahmad Naufal Ramadhan and Diero Arga Purnama, for their continuous technical assistance, assignment coordination, and invaluable structural clarifications during the development and implementation phases in this course.

References

[1] D. Boneh, B. Lynn, and H. Shacham, “Short signatures from the Weil pairing,” Journal of Cryptology, vol. 17, no. 4, pp. 297–319, 2004, doi: https://doi.org/10.1007/s00145-004-0314-9.

[2] D. Boneh, S. Gorbunov, R. Wahby, H. Wee, and Z. Zhang, “BLS Signatures,” datatracker.ietf.org, 2020. https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-bls-signature.

[3] S. Bowe, “BLS12-381: New zk-SNARK Elliptic Curve Construction,” electriccoin.co, Mar. 2017. https://electriccoin.co/blog/new-snark-curve/.

[4] “Proof-of-stake (PoS) | ethereum.org,” ethereum.org, 2023. https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/.

[5] B. Edgington, “Upgrading Ethereum: BLS Signatures,” eth2book.info, 2023. https://eth2book.info/capella/part2/building_blocks/signatures/.

[6] P. Miller, “noble-bls12-381: Fast, secure & zero-dependency BLS12-381 signature implementation in vanilla JS,” GitHub, 2023. https://github.com/paulmillr/noble-bls12-381.

[7] V. Buterin et al., “Combining GHOST and Casper,” arXiv.org, Mar. 2020. https://arxiv.org/abs/2003.03052.

[8] D. Boneh, M. Drijvers, and G. Neven, "Compact Multi-Signatures for Smaller Blockchains," in Advances in Cryptology – ASIACRYPT 2018, 2018, pp. 435–464, doi: https://doi.org/10.1007/978-3-030-03329-3_15.

STATEMENT

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Juni 2026

Rasyid Rizky Susilo N. 18223114