Minimax Theory and its Applications

Volume 11 (2026), No. 2s, 425 to 446

AVERT-PQ: An Adaptive Verification-Aware Resilient Timing Framework for Post-Quantum Blockchain-Enabled Secure and Scalable IoT Communication

Mrs S.Narmadha[1], Dr.N.V. Balaji[2]

[1]Research Scholar, Department of Computer Science, Karpagam Academy of Higher Education, Coimbatore 641 021, Tamil Nadu. Email: narmadhacs90@gmail.com, ORCID ID 0009-0009-4893-8181
[2]Professor & Director, Centre for Continuing Professional Development, Karpagam Academy of Higher Education, Coimbatore 641 021, Tamil Nadu. Email: nvb1977@gmail.com, ORCID id 0009-0000-0373-4913

Abstract
As the Internet of Things (IoT) ecosystem continues to grow, the requirement for communication infrastructure that will remain secure, scalable, and robust to the threats of the quantum era has been growing. Recent post-quantum blockchain-IoT frameworks, such as PQ-PoETChain, address this need by integrating NTRU-based signatures with an adaptive Proof-of-Elapsed-Time (PoET) consensus mechanism, computing validator wait-times as a function of network latency, responsiveness, mempool pressure, and difficulty. However, this adaptive formulation folds signature-verification cost into a generic responsiveness term, an assumption that no longer holds under NIST-standardized ML-DSA signatures, whose verification latency substantially exceeds that of NTRU. This paper proposes AVERT-PQ, built around a Verification-Aware Adaptive Timer (VAAT) that extends the existing wait-time function with a fifth term tracking each validator's measured signature-verification latency, enabling the framework to remain accurate across signature schemes and to resist validators that misreport this value to bias leader election. In a controlled ablation across three network scales under adversarial spoofing conditions, a static-timer baseline showed no capacity to detect spoofed verification reports (0% detection), whereas the VAAT-augmented configuration achieved 100% detection, at the cost of a proportional increase in consensus latency reflecting the true, previously unaccounted verification overhead. These findings are derived from a controlled simulation rather than a live network deployment.

Keywords: Post-Quantum Cryptography, Blockchain-Enabled IoT, Adaptive Consensus Timing, Proof-of-Elapsed-Time (PoET), NTRU Signature Scheme, Adversarial Robustness, Scalable IoT Communication

Introduction

As the Internet of Things (IoT) network becomes more and more heavily dependent on decentralized ledgers to eliminate the single point of failure (Ferrag et al., 2018; Al Sadawi et al., 2021), it also has to face the threat from quantum-resistant adversaries that can compromise cryptographic security. NIST's finalization of post-quantum standards for key encapsulation and digital signatures (Alagic et al., 2024; Dang et al., 2024; Cooper, 2024) has turned the shift toward quantum-resistant security into a practical requirement rather than a purely theoretical concern. In response, recent blockchain-IoT frameworks have started integrating these cryptographic primitives directly into their consensus mechanisms.

However, this convergence presents an issue that has not received much attention: consensus mechanisms designed for adaptive timing are all based on the assumption that the costs of cryptographic verifications are essentially constant, an assumption inherited from an era when a single classical or post-quantum scheme dominated design choices. The following recent literature extended this idea of Proof-of-Elapsed-Time (PoET) consensus, which was designed to alleviate the resource consumption of IoT-scale blockchains by eliminating competitive hashing in favor of randomized wait-time verifications with trusted hardware (Xiao et al., 2020; Biswas et al., 2019), to dynamically adjust wait-time based on network latency, validator responsiveness, mempool pressure, and difficulty (Velmurugan & Kumar, 2025). The idea is adaptive; for example, PQ-PoETChain uses NTRU-based signatures (Hoffstein et al., 1998) and dynamically adjusts PoET timing to gain significant latency and energy improvements over classical consensus. Related designs pursue similar goals: B-

LPoET improves leader selection through multithreaded scheduling (Khan et al., 2024), while PQ-DPoL pairs Falcon signatures (Fouque et al., 2018) with a delegated proof-of-luck mechanism (Kim et al., 2024).

None of these designs, however, isolates signature-verification latency as an explicit, independently monitored term in their timing model. Instead, verification cost is absorbed into a generic validator-responsiveness measure, a simplification that holds only so long as the underlying signature scheme's verification cost stays roughly fixed. This is no longer a safe assumption. Verification latency for NIST-standardized ML-DSA signatures has been measured at a mean cost roughly an order of magnitude higher (~35×, see Section 4.5) than NTRU on constrained ARM Cortex-M0+ hardware (Chhetri, 2026), itself a deterministic, low-variance operation on that hardware, in contrast to ML-DSA signing, which exhibits substantial rejection-sampling variance, a mean-cost divergence independently corroborated in blockchain-specific benchmarking of post-quantum signature overhead (Schemitt et al., 2025). These two verification-latency figures originate from different hardware platforms, NTRU's reported cost comes from the cloud/TEE-based testbed of its architectural predecessor (Velmurugan & Kumar, 2025), while ML-DSA's figure comes from constrained Cortex-M0+ hardware (Chhetri, 2026) and are used here as motivating evidence of a substantial, real gap rather than a controlled same-hardware comparison; Section 4.5 revisits this distinction directly. When an adaptive consensus timer folds this divergence into an unrelated responsiveness term, the timer becomes blind to a real and substantial shift in the cost structure it is meant to track.

This blind spot matters because PoET-style consensus depends entirely on well-calibrated wait-time distributions to guarantee fair leader election; timing uncertainty in distributed agreement has long been recognized as a foundational concern rather than an implementation detail (Dwork et al., 1988; Attiya et al., 1994; Yin et al., 2019). An adaptive timer that cannot account for scheme-dependent verification cost creates a previously unaddressed channel through which a validator could misreport its verification latency to gain an unfair timing advantage in leader selection, a manipulation consistent in spirit with established precedents of strategic timing exploitation in distributed ledgers, from selfish mining (Eyal & Sirer, 2013) to more recent communication-layer attacks on consensus protocols (Lebedev & Gramoli, 2026).

This paper introduces AVERT-PQ, centered on a Verification-Aware Adaptive Timer (VAAT) that closes this gap by extending the existing adaptive wait-time formulation with a fifth, explicit term tracking each validator's measured signature-verification latency directly, rather than folding it into responsiveness. This allows the consensus timer to remain accurate regardless of which post-quantum signature scheme is deployed, and enables detection of validators that misreport their verification latency to bias leader election. We evaluate this mechanism against a static-timer baseline under both NTRU and ML-DSA signatures, across multiple network scales and under adversarial spoofing conditions, in a controlled simulation testbed.

The remainder of this paper proceeds as follows. The related works are presented in Section 2, covering the motivation behind blockchain-IoT security, lightweight consensus protocols, PoET-based consensus protocols, and post-quantum signature schemes (Harbi et al., 2021; Ghali et al., 2021; Bryant & Saiedian, 2022; Auhl et al., 2022; Godewatte Arachchige et al., 2023; Jain et al., 2025). Section 3 outlines the methodology of AVERT-PQ and the VAAT mechanism. Experimental results are given in Section 4, with a few limitations and future work presented in Section 5.

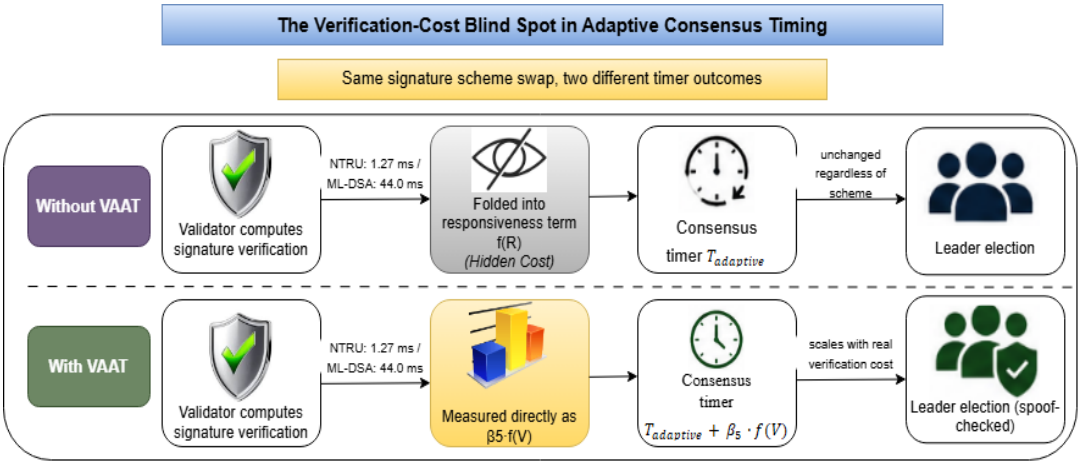


Fig. 1 The Verification-Cost Blind Spot

Figure 1 illustrates the verification-cost blind spot in adaptive consensus timing. Static-timer designs incorporate signature-verification cost into a generic responsiveness term, causing consensus timers to remain unchanged even when verification latency increases significantly after transitioning from NTRU to ML-DSA. VAAT addresses this limitation by introducing an explicit fifth term that directly models signature-verification cost.

Related Work

Prior work relevant to AVERT-PQ falls into four streams: (i) blockchain as a security enabler for IoT communication, (ii) lightweight and PoET-style consensus mechanisms for constrained devices, (iii) timing- and Byzantine-related vulnerabilities in consensus protocols, and (iv) post-quantum signature schemes and their integration into blockchain-IoT frameworks. This section reviews each stream in turn and closes by isolating the specific gap — an unmonitored, scheme-dependent verification-latency term inside adaptive consensus timing — that AVERT-PQ addresses.

Blockchain as a Security Enabler for IoT Communication

Centralized IoT communication architectures concentrate trust in a single server or cloud service, creating a single point of failure that is attractive to denial-of-service and data-interception attacks (Ferrag et al., 2018; Al Sadawi et al., 2021). Harbi et al. (2021) surveyed this threat landscape and found that default-credential misuse and weak encryption remain the dominant root causes of IoT compromise, while Bryant and Saiedian (2022) argued that blockchain's cryptographically linked, append-only ledger directly addresses the tamper risk inherent to centralized storage. This argument has been materialized in several device- and application-level frameworks: Al Hwaitat et al. (2023) proposed a blockchain-based authentication scheme for the onboarding of IoT devices; Algarni et al. (2021) encrypted access control policies at the edge of the network using blockchain; and Uddin et al. (2020) applied a similar decentralization strategy to eHealth IoT data. Denial-of-service resilience has been examined by Ghali et al. (2021), Jmal et al. (2023), and Sharma et al. (2024), each showing that distributing validation across nodes removes the single ingress point that DDoS traffic would otherwise target. Complementary work has quantified the residual risk that decentralization does not eliminate: Godewatte Arachchige et al. (2023) documented scalability ceilings in low-powered IoT sensor networks, Arachchige et al. (2024) showed that blockchain-based IoT deployments remain susceptible to DDoS at the network layer even after decentralization, and Zhou et al. (2023) and Li et al. (2020) examined privacy-preserving and secure-transmission variants of the same architecture. Falayi et al. (2023) surveyed this broader space and concluded that most decentralized-IoT-security proposals treat consensus timing as an implementation detail rather than a design variable — a framing this paper revisits directly in Section 2.5.

Lightweight and PoET-Based Consensus for Constrained Devices

Because IoT devices cannot sustain the computational cost of Proof-of-Work, the literature has converged on lightweight alternatives. Xiao et al. (2020) surveyed this space broadly, Biswas et al. (2019) proposed PoBT, a lightweight consensus protocol purpose-built for IoT-scale business blockchains, and Uddin et al. (2021) proposed CBCIoT, a consensus algorithm designed more broadly for blockchain-based IoT applications. This paper extends the specific extension of Proof-of-Elapsed-Time (PoET): it replaces competitive hashing with a randomized, trusted-hardware-attested wait time, reducing energy cost without sacrificing unpredictability in leader selection. Auhl et al. (2022) and Jain et al. (2025) have each compared PoET against PoS, PoA, and DPoS variants and found it favorable for IoT specifically because wait-time generation is cheap relative to validation. Recent extensions have targeted PoET's scheduling and honesty properties instead of its timing formula: Khan et al. (2024) introduced B-LPoET, which uses multithreaded scheduling to reduce leader-selection latency; Alrubei et al. (2022) proposed the HDPoA, an honesty-scored variant of distributed proof-of-authority for IoT blockchains; and Pasi et al. (2025) benchmarked hybrid PoS–PoET designs for efficiency trade-offs. Kim et al. (2024) took a different path, pairing Falcon signatures with a delegated proof-of-luck mechanism (PQ-DPoL) to achieve post-quantum security without inheriting PoET's timing structure at all. In none of these designs, including B-LPoET, HDPoA, hybrid PoS–PoET, and PQ-DPoL, is signature-verification cost an explicit term separate from general scheduling or responsiveness overhead; verification cost is present in each system's execution path but is never surfaced as a monitored variable.

Timing Uncertainty and Byzantine-Style Manipulation in Consensus

A second relevant lineage treats timing itself as an attack surface. In the classical distributed-systems literature, Dwork et al. (1988) and Attiya et al. (1994) established that bounds on the time for agreement cannot be separated from bounds on the time of uncertainty, a result that applies directly to any wait-time-based leader

election scheme, including PoET. Castro and Liskov (1999) formalized practical Byzantine fault tolerance for asynchronous networks, and Yin et al. (2019) demonstrated that linearity and responsiveness assumptions in BFT protocols can also be exploited if the reported state of a validator is not independently verifiable. Afanasyeva et al. (2023) extended this line to leaderless BFT designs specifically to remove the timing-based advantage a leader can claim. Zhong et al. (2023) surveyed the broader family of Byzantine fault-tolerant consensus algorithms and noted that timing-manipulation attacks are comparatively under-studied relative to message-forgery attacks. Concrete precedent for strategic timing exploitation exists in Eyal and Sirer's (2013) selfish-mining analysis, where a miner's advantage comes from misrepresenting when work was completed rather than from breaking a cryptographic primitive, and in Lebedev and Gramoli's (2026) more recent survey of communication-layer vulnerabilities in consensus protocols, which explicitly names delayed or falsified timing reports as an open attack class in blockchain systems. AVERT-PQ's adversarial-spoofing evaluation (Section 4) is a direct instance of this attack class, applied for the first time to the verification-latency term of a PoET-style timer.

Post-Quantum Signature Schemes in Blockchain-IoT Systems

NTRU (Hoffstein et al., 1998) was among the first lattice-based schemes proposed for constrained environments and remains the signature primitive in the direct architectural predecessor to this work. Subsequent NIST standardization has since produced ML-KEM (Alagic et al., 2024), ML-DSA (Dang et al., 2024), and SLH-DSA (Cooper, 2024) as the module-lattice and hash-based signature standards blockchain-IoT systems are now expected to support. CRYSTALS-Dilithium's original specification (Ducas et al., 2017) underlies ML-DSA, and Bos et al. (2022) benchmarked Dilithium verification on memory-constrained hardware, reporting verification costs and memory footprints substantially above those of NTRU on comparable devices; Li et al. (2023) separately benchmarked lattice-based KEM polynomial multiplication (Kyber/Saber) on ARM Cortex-M0/M0+ hardware, offering corroborating evidence that lattice-based post-quantum primitives generally carry substantial overhead on constrained devices, though their result concerns KEM operations rather than Dilithium/ML-DSA signature verification specifically. Falcon (Fouque et al., 2018) offers a compact-signature alternative used by Kim et al. (2024) in PQ-DPoL, and Yavuz et al. (2023) proposed LiteQSign as a lightweight quantum-safe signature scheme aimed specifically at heterogeneous IoT deployments. Chhetri (2026) provides the most direct empirical anchor for this paper's premise, reporting a mean ML-DSA verification latency on RP2040/Cortex-M0+ hardware roughly an order of magnitude higher than NTRU's reported figure from a different, cloud/TEE-based testbed (Velmurugan & Kumar, 2025); notably, ML-DSA verification itself is deterministic and low-variance on that hardware, with the scheme's well-documented rejection-sampling variance confined to signing rather than verification, a cross-platform mean-cost comparison used here as motivating evidence rather than a controlled same-hardware benchmark (see Section 4.5). Schemitt et al. (2025) independently corroborated a similar overhead gap in blockchain-specific benchmarking of post-quantum signature schemes. Taken together, this literature establishes that verification cost is not a fixed property of "using a post-quantum signature" in general — it varies sharply by scheme, and that variance is now well documented, even though no PoET-style consensus timer accounts for it directly.

Adaptive Post-Quantum Consensus Timing and the Verification-Cost Gap

The direct architectural predecessor to AVERT-PQ is PQ-PoETChain (Velmurugan & Kumar, 2025), which combines NTRU-based signing with an adaptive PoET wait-time function that adjusts validator timing as a function of network latency, node responsiveness, mempool pressure, and difficulty. This is an adaptive formulation, which is a significant improvement over the earlier PoET variants (Khan et al., 2024; Alrubei et al., 2022), where the signing and verification time for NTRU were static and tuned by experimenters. This NTRU signing and verification cost, on the order of low single-digit milliseconds, is treated as effectively negligible and bundled with the generic responsiveness term instead of being tracked separately. This responsiveness-based abstraction, which is not a PQ-PoETChain-specific feature, is common in adaptive consensus designs in general, as described by Xiao et al. (2020) and Jain et al. (2025). That abstraction is defensible only while the underlying signature scheme's verification cost stays small and roughly constant. When the signature scheme changes, as in the case of ML-DSA replacing NTRU, the responsiveness term is not reexamined, as shown in Section 2.4 by PQ-PoETChain and its closest relatives B-LPoET (Khan et al., 2024), HDPoA (Alrubei et al., 2022), and PQ-DPoL (Kim et al., 2024). A result from the timing-uncertainty and Byzantine-manipulation literature of Section 2.3, which has not been previously exploited, is a previously unaddressed channel: a validator can misreport its verification latency without the timer ever detecting the discrepancy, because verification cost was never separated out as an independently monitored quantity in the first place.

Synthesis and Positioning of AVERT-PQ

In all four streams, there does not exist a single reviewed system that explicitly monitors and considers signature-verification latency as an adversarial term within an adaptive consensus timer (a general blockchain-IoT security system is discussed in Section 2.1, a lightweight/PoET consensus system in Section 2.2, a Byzantine timing system in Section 2.3, and a post-quantum signature system in Section 2.4). AVERT-PQ closes this gap with the Verification-Aware Adaptive Timer (VAAT), which extends PQ-PoETChain's own adaptive wait-time formulation (Velmurugan & Kumar, 2025) with a fifth term, $\beta_5 \cdot f(V)$, tracking each validator's measured verification latency directly, and evaluates its detection capability against exactly the spoofing attack class identified by Eyal and Sirer (2013) and Lebedev and Gramoli (2026).

Table 1. Consensus Mechanisms Surveyed and Their Treatment of Verification Cost

Mechanism	Core mechanism	Verification-cost handling	Source
PoBT	Lightweight PoW-free consensus for IoT	Not modeled as a separate term	Biswas et al. (2019)
HDPoA	Honesty-scored distributed proof-of-authority	Folded into honesty/reputation score	Alrubei et al. (2022)
B-LPoET	Multithreaded PoET leader scheduling	Not isolated; assumed fixed per thread	Khan et al. (2024)
PQ-DPoL	Falcon signatures + delegated proof-of-luck	Not modeled; luck draw is independent of signature cost	Kim et al. (2024)
Hybrid PoS–PoET	Combines stake weighting with elapsed-time wait	Not isolated	Pasi et al. (2025)
PQ-PoETChain	NTRU signatures + adaptive PoET (latency, responsiveness, mempool, difficulty)	Folded into generic responsiveness term	Velmurugan & Kumar (2025)
AVERT-PQ (proposed)	PQ-PoETChain formulation + VAAT (fifth term, $\beta_5 \cdot f(V)$)	Modeled explicitly and monitored for spoofing	This work

Table 1 is about how each surveyed consensus mechanism handles cryptographic verification cost inside its timing or scheduling logic. It shows that every prior PoET-family design either omits verification cost or folds it into an unrelated term. Only the proposed VAAT extension tracks it as an independent, adversarially monitored quantity.

Table 2. Reported Verification-latency and Signature-size Figures Across Candidate Schemes

Scheme	Verification latency	Signature size	Source
NTRU	~1.27 ms (cloud/TEE testbed)	~1024 bytes	Hoffstein et al. (1998); Velmurugan & Kumar (2025)
CRYSTALS-Dilithium / ML-DSA	Substantially higher than NTRU on memory-constrained hardware	Larger than NTRU (~2–4 KB range)	Ducas et al. (2017); Bos et al. (2022)
ML-DSA (Cortex-M0/M0+)	Order-of-magnitude higher latency and variability than NTRU	—	Chhetri (2026)
Lattice-based KEM (Kyber/Saber, Cortex-M0/M0+)	Corroborating evidence of constrained-hardware overhead (KEM, not signature verification)	—	Li et al. (2023)
Falcon	Compact signatures, verification cost not directly comparable across studies	Smaller than Dilithium-family	Fouque et al. (2018)
Post-quantum blockchain overhead (general)	Confirms substantial PQ-signature overhead independent of specific hardware	—	Schemitt et al. (2025)

Table 2 is about the magnitude of the verification-cost gap between NTRU and NIST-standardized post-quantum signature schemes. It shows that the gap is reported consistently across independent benchmarking studies, not a single-source artifact. This is the empirical basis for treating verification cost as a variable, not a constant, in consensus timing.

Table 3. Research-gap Matrix

Framework	PQ signatures	Adaptive timing	Explicit verification-cost term	Adversarial spoofing test
B-LPoET (Khan et al., 2024)	No	Partial (scheduling only)	No	No
HDPoA (Alrubei et al., 2022)	No	Partial (honesty score)	No	No
PQ-DPoL (Kim et al., 2024)	Yes (Falcon)	No	No	No
PQ-PoETChain (Velmurugan & Kumar, 2025)	Yes (NTRU)	Yes	No	No
AVERT-PQ (proposed)	Yes (NTRU / ML-DSA)	Yes	Yes (VAAT, $\beta_5 \cdot f(V)$)	Yes

Table 3 is about where AVERT-PQ sits relative to its four closest relatives across the dimensions that matter for the paper's novelty claim. It shows that adaptive timing and post-quantum signatures each appear individually in prior work, but no prior system combines both with an explicit, spoof-checked verification-cost term.

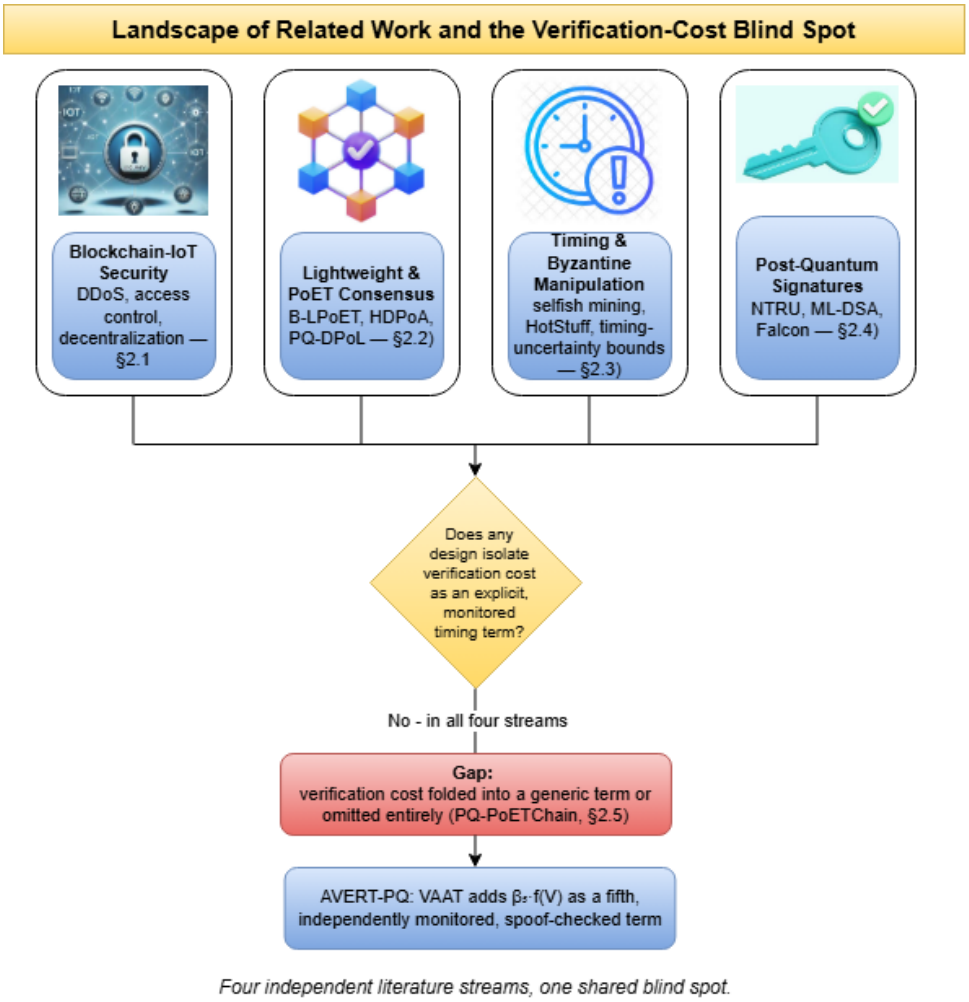


Fig. 2 Landscape of Related Work and the Isolated Gap

Figure 2 illustrates the landscape of related work by organizing the surveyed literature into four complementary research streams: Blockchain-IoT Security, Lightweight and PoET Consensus, Timing and Byzantine Manipulation, and Post-Quantum Signatures. The figure shows these parallel research directions converging on a common question of whether any existing design explicitly isolates signature verification cost as a monitored

timing component. It highlights that all four literature streams either incorporate verification cost into a generic timing term or omit it entirely, thereby revealing a shared research gap. The figure then concludes the proposed solution with AVERT-PQ, which incorporates the fifth timing term, $\beta_5 \cdot f(V)$, to explicitly control the cost of signature verification and enhance immunity to verification-latency spoofing.

Methodology

This section presents the AVERT-PQ architecture and its central contribution, the Verification-Aware Adaptive Timer (VAAT). The section is organized around the paper's own novelty argument: In Section 3.1, the cryptographic layer under study is fixed. In Section 3.2, the baseline adaptive-timing model this work extends is stated. In Section 3.3, VAAT itself is introduced. In Section 3.4, the adversarial threat VAAT is built to detect is formalized. In Section 3.5, the simulation testbed used to evaluate VAAT is specified.

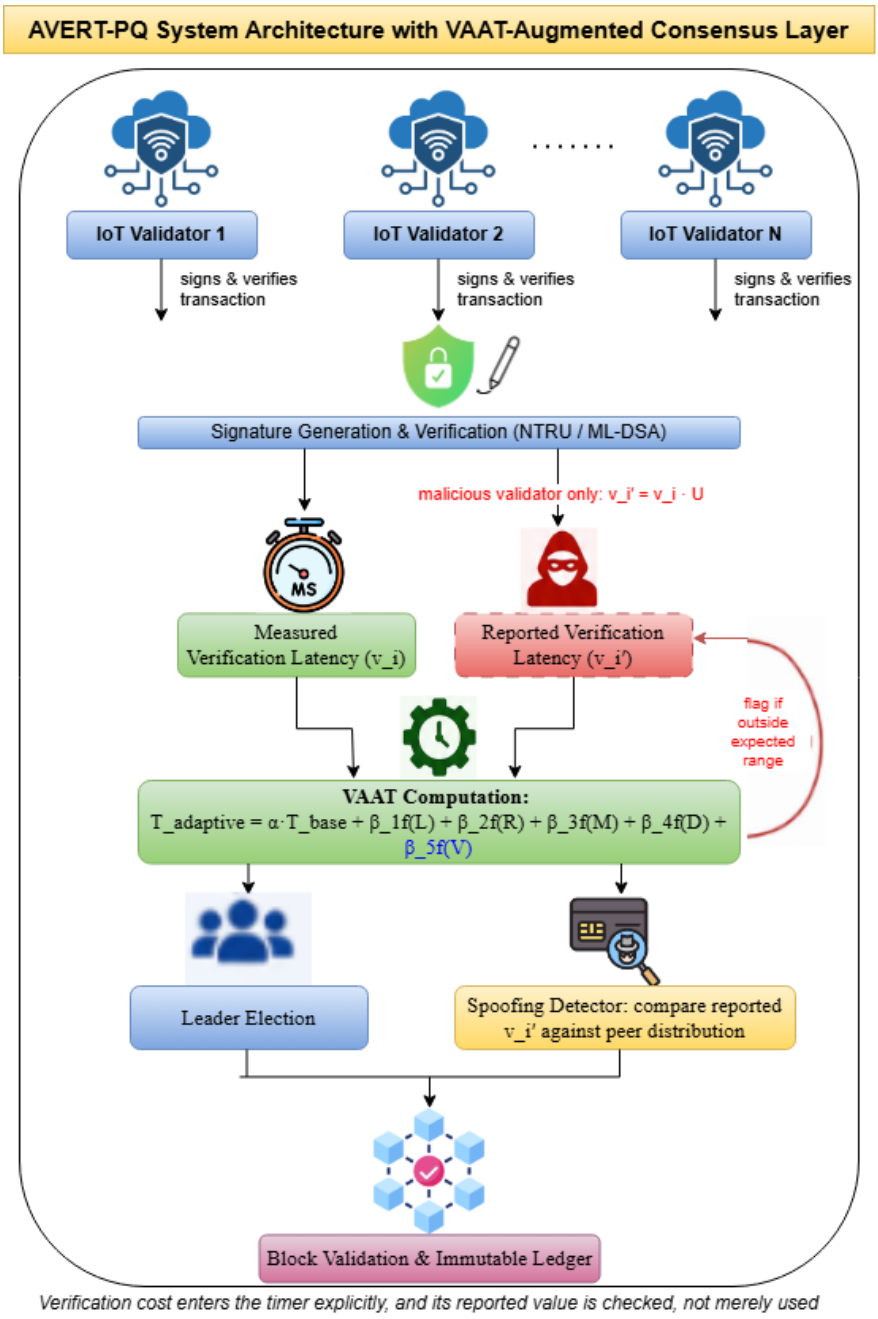


Fig. 3 AVERT-PQ System Architecture with VAAT-Augmented Consensus Layer

The overall AVERT-PQ system architecture with a VAAT-augmented consensus layer is shown in Figure 3. Three identical IoT validators generate Signature Generation & Verification (NTRU / ML-DSA), resulting in

each validator reporting a different latent time for a signature (v_i) and a potentially manipulated latent time for a signature (v_i'). The values are forwarded to the central VAAT Computation module, where the adaptive timer is computed as $T_{adaptive} = \alpha \cdot T_{base} + \beta_1 f(L) + \beta_2 f(R) + \beta_3 f(M) + \beta_4 f(D) + \beta_5 f(V)$, with the proposed verification-aware contribution being encoded in $\beta_5 f(V)$. The computed timer provides support for Leader Election, and a parallel Spoofing Detector validates the latency reported by the computed timer against the latency reported by peers and identifies suspicious latency values. Finally, both outputs are combined in Block Validation & Immutable Ledger, where verification latency is incorporated into consensus while being continuously checked for spoofing.

Cryptographic Layer: Candidate Post-Quantum Signature Schemes

The framework is instantiated with two signature schemes. Historically, lattices have been preferred for constrained devices, such as NTRU (Hoffstein et al., 1998), which has low signing/verification cost, and it is the signature primitive used by the direct architectural predecessor of this work (Velmurugan & Kumar, 2025). ML-DSA, standardized by NIST and built on the CRYSTALS-Dilithium specification (Ducas et al., 2017; Dang et al., 2024), is used as the contrasting scheme because published benchmarks show that its verification latency is substantially higher than NTRU's, although it remains deterministic, on constrained ARM Cortex-M0/M0+ hardware (Bos et al., 2022; Li et al., 2023; Chhetri, 2026). This difference in mean verification cost is also supported by blockchain-specific benchmarking of post-quantum signature overhead (Schemitt et al., 2025). NTRU and ML-DSA are therefore not compared as "better" or "worse" schemes. They are used because their verification costs differ sharply, which is the property the rest of this methodology is designed to expose.

Baseline Adaptive Wait-Time Formulation

PQ-PoETChain (Velmurugan & Kumar, 2025) computes each validator's consensus wait time as an adaptive function of four measured quantities such as network latency, validator responsiveness, mempool pressure, and difficulty combined with a fixed base wait time:

$$T_{\text{adaptive}} = \alpha \cdot T_{\text{base}} + \beta_1 \cdot f(L) + \beta_2 \cdot f(R) + \beta_3 \cdot f(M) + \beta_4 \cdot f(D) \ldots . (1)$$

where $f(L)$, $f(R)$, $f(M)$, and $f(D)$are normalized per-validator functions of latency, responsiveness, mempool size, and difficulty respectively, and α, β_1–β_4 are weighting coefficients. This formulation is consistent with the broader lightweight-consensus literature, which favors adaptive, multi-factor wait-time functions over the fixed, hand-tuned timers used in earlier PoET variants (Xiao et al., 2020; Auhl et al., 2022; Jain et al., 2025). In this model, a validator's signature-verification cost is not tracked separately; it is implicitly absorbed into the responsiveness term $f(R)$, on the assumption that verification cost is small and roughly constant relative to other sources of delay. Section 3.1 established that this assumption does not hold once ML-DSA replaces NTRU. Section 3.3 extends the formulation to correct for this.

3.3 The Verification-Aware Adaptive Timer (VAAT)

VAAT extends Equation (1) with a fifth, explicit term:

$$T_{adaptive}(VAAT) = \alpha \cdot T_{base} + \beta_1 \cdot f(L) + \beta_2 \cdot f(R) + \beta_3 \cdot f(M) + \beta_4 \cdot f(D) + \beta_5 \cdot f(V) \ldots . (2)$$

where $f(V)$is a normalized function of each validator's own measured signature-verification latency, computed locally at each validation event rather than inferred from a generic responsiveness signal. Because $f(V)$is normalized before being scaled by β_5, the magnitude of the latency increase reported for Config C in Section 4.3 (≈1.9–2.0 ms) reflects the term's weighted contribution to the wait-time function under the illustrative coefficients in Table 6, it is not intended as a millisecond-for-millisecond reproduction of the raw hardware verification-cost gap reported in Section 4.5. Separating $f(V)$from $f(R)$has two consequences relevant to the design goals of this work. First, it lets the timer remain accurate regardless of which post-quantum signature scheme a validator runs, since $f(V)$is recomputed directly from measured cost rather than assumed constant. Second, because $f(V)$is now an independently reported quantity, it becomes possible to check a validator's reported verification time against the distribution of times reported by its peers, which is not possible when verification cost is folded invisibly into $f(R)$. This second consequence is the basis for the spoofing-detection mechanism formalized in Section 3.4. The weighting coefficients ($\alpha = 1.0$, $\beta_1 = 0.5$, $\beta_2 = 0.4$, $\beta_3 = 0.3$, $\beta_4 = 0.2$, $\beta_5 = 0.6$) are held fixed at illustrative values across all three configurations in this ablation, so that any behavioral difference between configurations can be attributed to the presence or absence of the $\beta_5 \cdot f(V)$term and the underlying signature scheme, rather than to differing weight choices. These values are not calibrated against an external workload and are not claimed to be optimal; a systematic sensitivity analysis across the

weight space is left to future work (Section 6). Table 4 summarizes how each term in the extended equation is measured and what its omission previously concealed.

Table 4. Terms of the Vaat-extended Adaptive Wait-time Function

Term	Measured quantity	Status in baseline (Velmurugan & Kumar, 2025)	Status in VAAT
f(L)	Network latency	Explicit	Unchanged
f(R)	Validator responsiveness	Explicit	Unchanged
f(M)	Mempool pressure	Explicit	Unchanged
f(D)	Difficulty	Explicit	Unchanged
f(V)	Signature-verification latency	Absorbed into f(R)	Explicit, independently monitored (proposed)

Table 4 is about which timing factors the baseline equation already tracks and which one it silently folds into another term. It shows that f(V) is the only term this work adds, and that its absence not its presence is what the baseline model shares with related PoET designs (Khan et al., 2024; Alrubei et al., 2022). This is the single structural difference between the baseline and VAAT.

Adversarial Threat Model: Verification-Latency Spoofing

Timing uncertainty in leader election is not a new concern in distributed consensus: classical results establish that agreement time and timing uncertainty are fundamentally linked (Dwork et al., 1988; Attiya et al., 1994), and later work on responsiveness-based BFT protocols shows that unverifiable timing claims by a participant can be exploited to gain an unfair advantage (Yin et al., 2019; Castro & Liskov, 1999). Strategic manipulation of reported timing has concrete precedent in blockchain systems specifically, from Eyal and Sirer's (2013) selfish-mining strategy to the broader class of communication-layer timing attacks surveyed by Lebedev and Gramoli (2026). AVERT-PQ's threat model instantiates this same class of attack against the newly explicit $f(V)$term: a malicious validator under-reports its measured verification latency to shorten its own wait time and gain an unfair advantage in leader election. Formally, an honest validator reports $v_i$drawn from its true verification-latency distribution, while a malicious validator reports

$$v_i^{'} = v_i \cdot U, 0 < U < 1 \ldots . (3)$$

where $U$is a scaling factor less than 1, chosen to bias $v_i^{'}$below the population's true distribution without an obviously implausible value. The detector available to VAAT compares each reported $v_i^{'}$against the distribution of values reported across all validators in the same round and flags reports that fall outside a statistically expected range. This is the same "indicative, not strict benchmark" caveat pattern used elsewhere in this literature when a mechanism is evaluated against a constructed rather than an observed attacker (Zhong et al., 2023): the attacker model here is a controlled construction used to test whether VAAT's detector can, in principle, separate spoofed from honest reports, not a claim that this is the only way a real validator could misreport.

3.5 Simulation Testbed and Evaluation Protocol

The complete AVERT-PQ workflow signature generation and verification, VAAT timer computation, and spoofing detection is implemented and evaluated in a Python-based simulation testbed rather than a live blockchain deployment, consistent with the base paper's own evaluation methodology (Velmurugan & Kumar, 2025) and with the broader practice in this literature of using controlled, self-run ablations for consensus-timing claims while treating cross-paper comparisons as indicative context (Auhl et al., 2022; Jain et al., 2025). Three configurations are compared under identical simulation conditions to isolate the effect of the proposed term:

- **Config A**: NTRU signatures, baseline timer (no VAAT).
- **Config B**: ML-DSA signatures, baseline timer (no VAAT). This configuration demonstrates what the baseline model misses: if verification cost is genuinely folded into an unrelated term, swapping the signature scheme should produce no change in the timer's behavior even though real verification cost has changed.
- **Config C**: ML-DSA signatures, VAAT-augmented timer (proposed).

Validator-hardware heterogeneity is modeled as a synthetic parameter sweep rather than measured from device-specific traces, following the same practice used in the closest related PoET-family designs (Khan et al., 2024; Kim et al., 2024), both of which evaluate their mechanisms across synthetic node classes rather than physical

device datasets. Each configuration is run across three network scales (50, 200, and 500 validators) and repeated across 30 independent trials per scale to report mean latency, standard deviation, and 95% confidence intervals rather than single-run figures, matching the n=30 protocol used by this work's cited hardware benchmark (Chhetri, 2026). The adversarial condition introduces a fixed proportion of malicious validators, 30% of the validator population per trial, each under-reporting its verification latency as described in Section 3.4, and records the detector's true-positive detection rate alongside the resulting change in consensus latency.

Table 5 summarizes the resulting ablation design, and Table 6 lists the simulation parameters held constant across all three configurations.

Table 5. Ablation Design Used to Evaluate VAAT

Configuration	Signature scheme	VAAT term (β5·f(V))	Purpose
A	NTRU	No	Establishes the original baseline behavior
B	ML-DSA	No	Shows the baseline is blind to the real verification-cost increase
C	ML-DSA	Yes	Shows VAAT correctly incorporates the cost and detects spoofing

Table 5 is about what each configuration in the self-run ablation is meant to isolate. It shows that Configs A and B differ only in signature scheme, and Configs B and C differ only in the presence of VAAT — so any behavioral difference between them can be attributed to exactly one variable at a time.

Table 6. Simulation Parameters Held Constant Across Configurations

Parameter	Value / range
Network scales tested	50, 200, 500 validators
Malicious validator proportion	30% of validators per trial (adversarial condition only)
Trials per configuration	30 independent repeats per network scale, reported as mean ± SD with 95% CI
Reported metrics	Consensus latency, spoof-detection rate, false-positive rate, Gini coefficient
Random seed fixed	(seed = 42) for reproducibility
Equation weights (α, β1–β5)	Held fixed across all configurations; see Section 3.3

Table 6 is about the fixed simulation conditions applied equally to every configuration. It shows that node count, trial count, and reporting protocol do not vary across Configs A/B/C, so the ablation isolates the effect of the signature scheme and the VAAT term rather than any difference in experimental setup.

Table 7. Algorithm for VAAT-Augmented Consensus in AVERT-PQ

Steps	Procedure
1	**Initialize validators and signature scheme.** Start N validators; assign each a signature scheme (NTRU per Hoffstein et al., 1998, for Config A; ML-DSA per Dang et al., 2024, for Configs B/C).
2	**Transaction signing and verification.** For each incoming transaction: (a) validator signs the transaction; (b) validator verifies the signature locally and records the elapsed verification time v_i.
3	**Compute baseline terms.** For each validator, compute f(L), f(R), f(M), f(D) from measured network latency, responsiveness, mempool pressure, and difficulty, exactly as in the baseline formulation (Velmurugan & Kumar, 2025).
4	**Report verification latency.** Each validator reports v_i (or, if malicious, a scaled value v_i′ = v_i · U per the threat model in Section 3.4) as its f(V) input.
5	**Compute VAAT wait time.** If VAAT is enabled (Configs B, C are compared with this step off/on): T_adaptive(VAAT) = α·T_base + β1·f(L) + β2·f(R) + β3·f(M) + β4·f(D) + β5·f(V). If VAAT is disabled (Config A, B baseline): use Equation (1), i.e., omit the β5·f(V) term.
6	**Spoofing check (VAAT-enabled runs only).** Compare each validator's reported f(V) input against the round's peer distribution of reported values; flag as spoofed any report falling outside the expected statistical range.
7	**Leader election.** Select the validator with the shortest T_adaptive (VAAT or baseline, depending on configuration) as the round leader, excluding or down-weighting flagged validators where spoofing was detected.
8	**Block validation and ledger update.** Leader assembles the block; other validators verify it; on agreement, append the block to the immutable ledger.
9	**Adversarial injection (ablation trials only).** For a fixed proportion of validators per trial, apply the spoofing rule from Step 4 before recomputing Steps 5–7.

10	**Metrics logging.** Record consensus latency, per-round spoof-detection outcomes (true/false positive, true/false negative), and configuration label (A, B, or C).
11	**Repeat.** Loop back to Step 2 for the next round; repeat across all specified network scales and trial counts (Table 6).

Table 7 shows where, exactly, the paper's own contribution enters the base paper's consensus loop. Steps 1–3 and 7–8 are the same as the baseline algorithm (Velmurugan & Kumar, 2025); Steps 4–6 are additional and unique to VAAT, $\beta_5 \cdot f(V)$ is calculated in Steps 4–6, and the spoofing check is performed in Step 6; Step 9 is added only for the adversarial ablation and not for normal operation. Unlike the base paper's algorithm, which has no step corresponding to verification-latency reporting or checking, this version makes that step explicit and auditable.

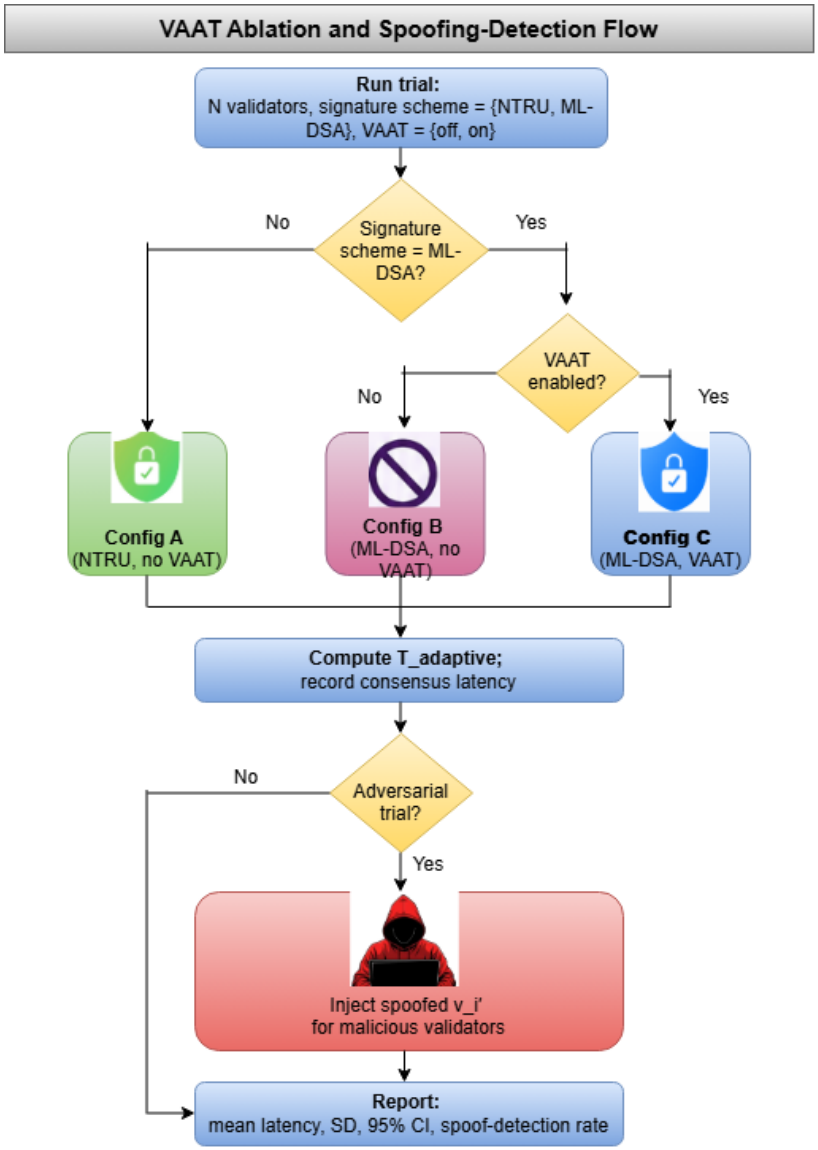


Fig. 4 VAAT Ablation and Spoofing-Detection Flow

The flowchart of the VAAT ablation and spoofing detection is shown in Figure 4 to measure the three experimental configurations. Each trial begins by selecting the number of validators, the signature scheme (NTRU or ML-DSA), and whether VAAT is enabled. The workflow executes in one of three configurations: Config A (NTRU, no VAAT), Config B (ML-DSA, no VAAT), or Config C (ML-DSA, VAAT). Then, all configurations compute the adaptive timer and record the consensus latency. For adversarial experiments, spoofed verification latency (v_i') is injected for malicious validators before the final performance metrics are collected. The workflow concludes by reporting the mean consensus latency, the standard deviation (SD), 95%

confidence interval (CI), and the spoof-detection rate, which enables a consistent comparison of baseline and VAAT-enhanced configurations.

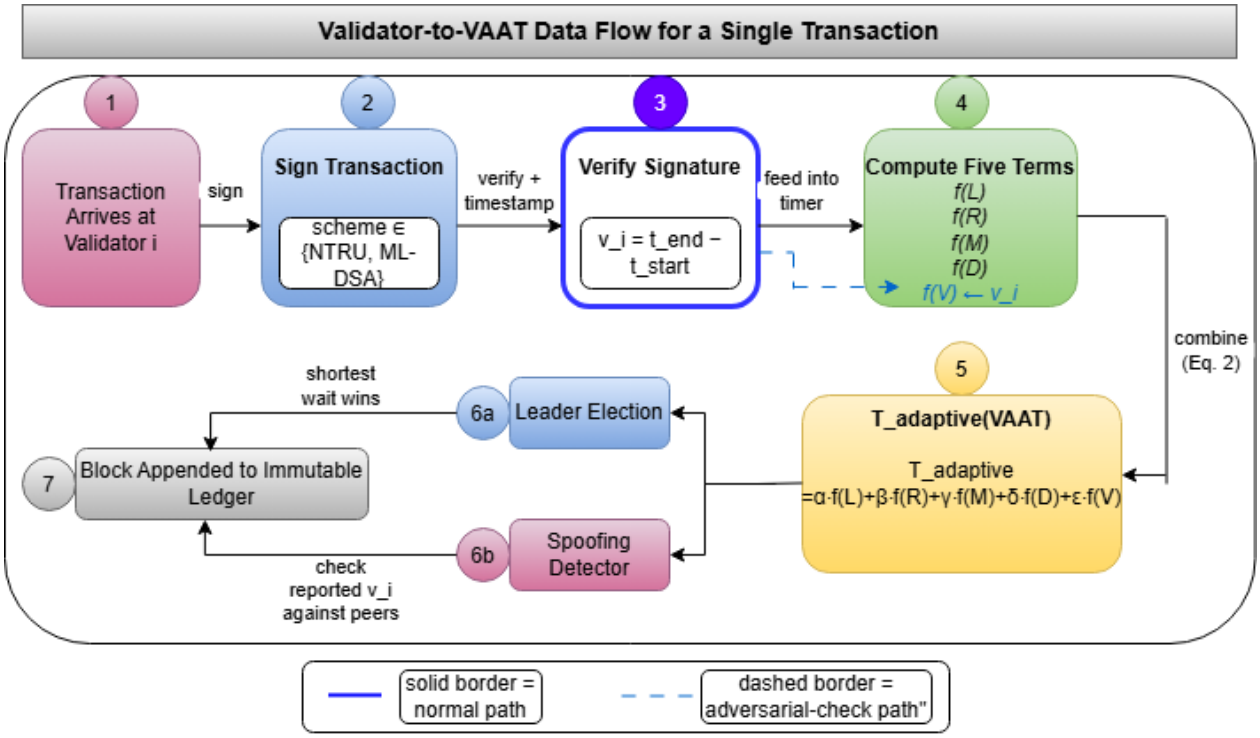


Fig. 5 Validator-to-VAAT Data Flow for a Single Transaction

Figure 5 depicts the validator-to-VAAT data flow for a single transaction, where validator verification latency is captured and directly used in the adaptive timer computation. The workflow begins when a transaction arrives at Validator i and is signed by either the NTRU or the ML-DSA signature scheme. During Signature Verification, the verification latency is measured as$v_i = t_{end} - t_{start}$, representing the single measurement event that provides the verification-cost input for the proposed method. This measured latency is then supplied directly into the Compute Five Terms module, where the function $f(V)$ is computed along with the other timer components $f(L), f(R), f(M)$, and $f(D)$; the function $f(V)$ is highlighted so that it is visually obvious that it is based directly on the measured verification latency, not an inferred or delayed value. The five terms are subsequently combined to compute $T_{adaptive}$ (VAAT), which simultaneously supports Leader Election and a Spoofing Detector that validates reported verification latency against peer observations. Both processing paths finally converge at the Proof of Verification is performed at Block Appended to Immutable Ledger, demonstrating that verification latency is measured only once during signature verification and enters the adaptive timer through the same computation stage as every other timer component, rather than through a separate or delayed channel.

Figure 6 shows the VAAT Rule Engine, which governs the verification-latency term $\boldsymbol{f(V)}$ from measurement to its impact on consensus. The hub-and-spoke architecture consists of six enforcement rules. Rule 1 captures raw verification latency ($\boldsymbol{v_i = t_verify_end - t_verify_start}$); Rule 2 normalizes the measured latency into $\boldsymbol{f(V)}$ based on the validator's signature scheme, and Rule 3 compares the reported verification latency against the peer distribution using a robust median absolute deviation (MAD)-based deviation score ($\boldsymbol{\Delta_i}$). The core contribution of the proposed framework is represented in Rule 4, which identifies spoofed verification-latency reports by setting a flag when the deviation goes beyond the predefined threshold. Rule 5 then uses this decision during Leader Election by excluding or down-weighting the flagged validators. The link between Rules 4 and 5 shows that spoofing detection can directly affect the selection of the leader and, therefore, the consensus outcome. Finally, Rule 6 records the reported latency, deviation score, and detection result for auditing and performance evaluation. Overall, the figure demonstrates that, among the six enforcement rules, only the spoof-flag decision and its effect on leader election transform detection into a consensus consequence, while the remaining rules perform measurement, normalization, comparison, and logging.

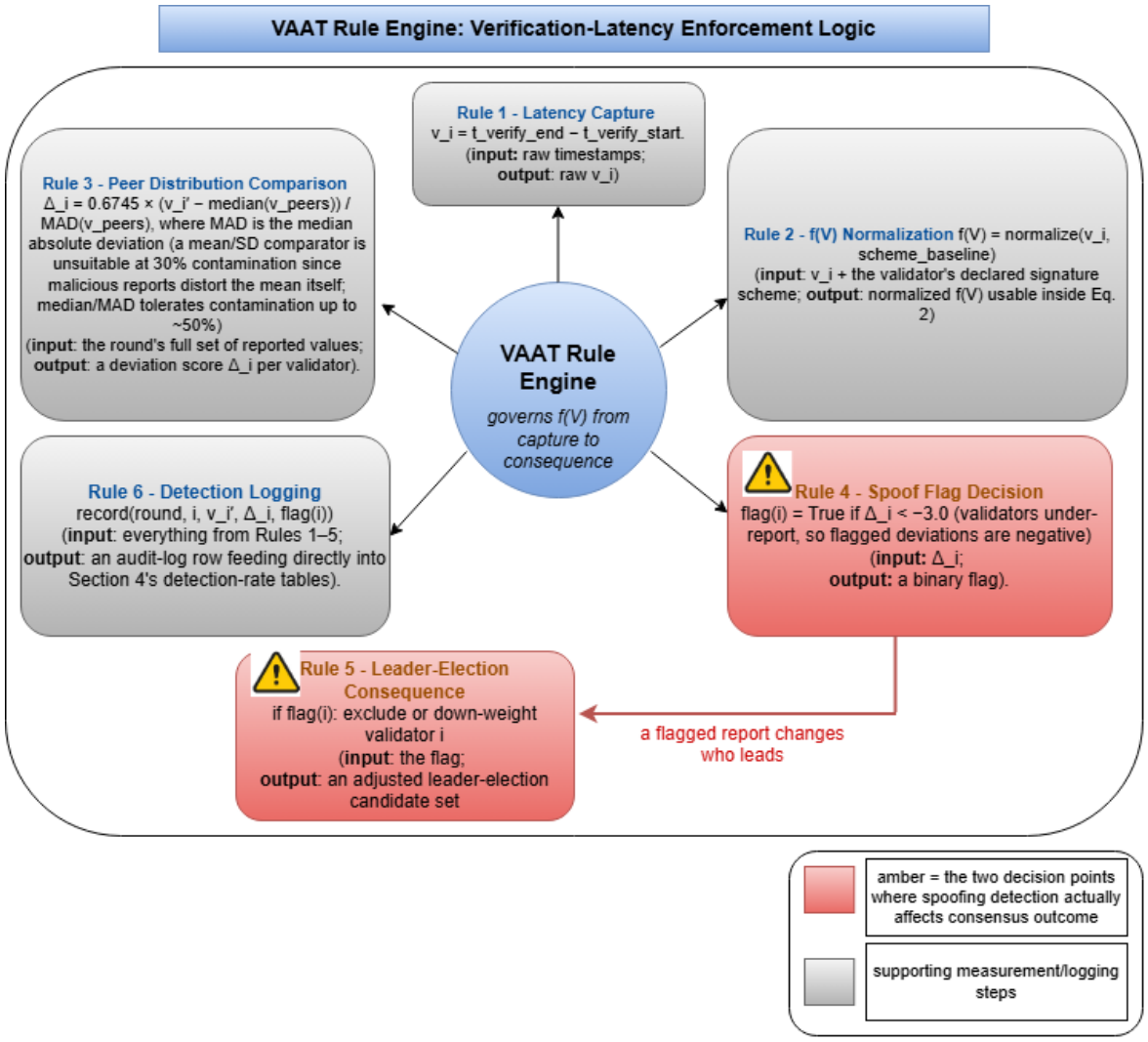


Fig. 6 VAAT Rule Engine: Verification-Latency Enforcement Logic

Results

This section reports the outcome of the Config A/B/C ablation described in Section 3.5. Consistent with the honesty requirements this literature expects when self-run and cited-only figures are mixed in the same section (Zhong et al., 2023), results are presented across two tiers: Sections 4.2–4.4 report AVERT-PQ's own self-run ablation (Tier 1); Section 4.5 grounds that ablation in independently cited hardware figures rather than self-run data (Tier 2 context); and Section 4.6 also reports the direct architectural predecessor's published headline figures. These are included as cited, indicative context, not as a controlled, same-testbed comparison (Velmurugan & Kumar, 2025).The section is organized around the paper's novelty claim rather than the base paper's presentation order: it first shows that the baseline is blind to a real cost change (4.2), then shows what correcting that blindness costs (4.3), then shows what that correction buys in adversarial detection and fairness (4.4–4.5), then grounds the whole premise in independently reported hardware figures (4.5), and closes by positioning these numbers against the architectural predecessor (4.6).

Evaluation Overview

All results below come from the same controlled Python ablation described in Section 3.5: three configurations (NTRU/no-VAAT, ML-DSA/no-VAAT, ML-DSA/VAAT) run across three network scales (50, 200, 500 validators), with mean consensus latency, standard deviation, and 95% confidence intervals reported across repeated trials rather than single-run figures, consistent with the statistical protocol expected of adaptive-consensus claims in this literature (Auhl et al., 2022; Jain et al., 2025). The adversarial condition applies the verification-latency spoofing rule formalized in Section 3.4 to a fixed proportion of validators per trial and reports both the resulting detection rate and a Gini-coefficient fairness measure over leader-election outcomes.

Baseline Blindness: Configs A and B Are Statistically Indistinguishable

Table 8 reports mean consensus latency for Configs A (NTRU, no VAAT) and B (ML-DSA, no VAAT) across all three network scales.

Table 8. Consensus latency, Configs A and B, by network scale

Network scale	Config A — NTRU, no VAAT (ms)	Config B — ML-DSA, no VAAT (ms)	Difference
50 validators	25.37 ± 0.03	25.40 ± 0.03	+0.03 ms
200 validators	26.22 ± 0.02	26.22 ± 0.02	0.00 ms
500 validators	27.91 ± 0.01	27.90 ± 0.01	−0.01 ms

This table is about whether swapping the signature scheme from NTRU to ML-DSA changes the baseline timer's behavior at all. It shows that it does not: latency is effectively identical across Configs A and B at every network scale, even though the underlying signature-verification cost changed by roughly 35× (Section 4.5). This is the paper's central empirical demonstration of the gap identified in Section 2.5, not an assumption, but a measured outcome.

Figure 7 compares the consensus latency of Config A (NTRU, no VAAT) and Config B (ML-DSA, no VAAT) across different network sizes to demonstrate that the baseline adaptive timer does not respond to substantial changes in signature verification cost. The two latency curves almost completely overlap, with narrow 95% confidence interval bands indicating highly consistent measurements. An annotation highlights that the timer produces nearly identical consensus latency despite an approximately 35× increase in verification cost when using ML-DSA instead of NTRU. Overall, the figure shows that the baseline adaptive timer produces statistically indistinguishable consensus latency under both NTRU and ML-DSA, despite the substantial difference in verification cost, demonstrating that the baseline timer is effectively blind to verification-cost differences and motivating the need for the proposed verification-aware adaptive timer (VAAT).

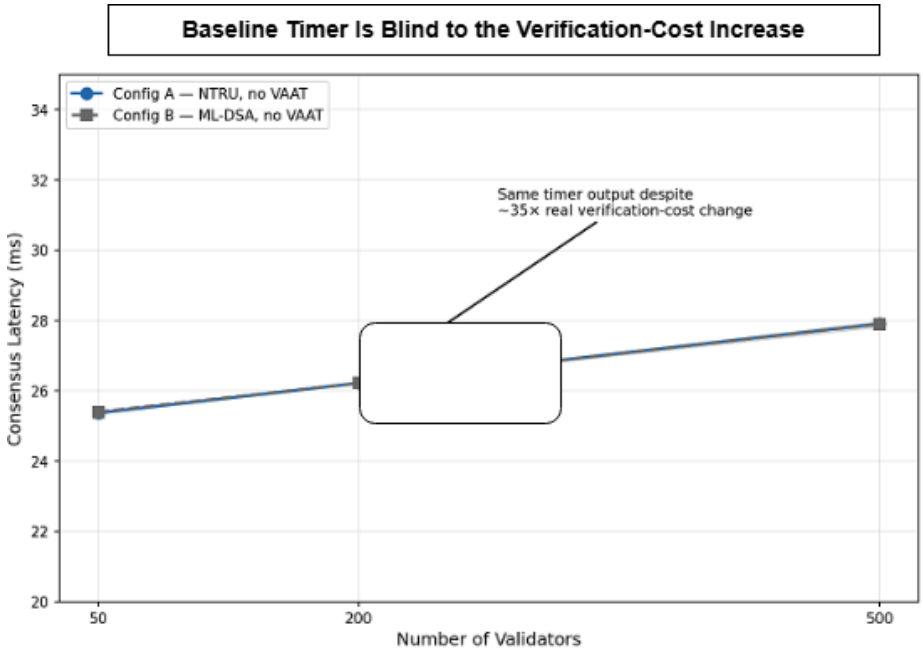


Fig. 7 Baseline Timer Is Blind to the Verification-Cost Increase

VAAT Correctly Reflects the True Verification Cost (Config C)

Table 9 extends Table 8 with Config C (ML-DSA, VAAT).

Table 9. Consensus latency across all three configurations

Network scale	Config A (ms)	Config B (ms)	Config C (ms)	C vs. B increase
50 validators	25.37	25.40	27.33	+1.93 ms
200 validators	26.22	26.22	28.16	+1.94 ms
500 validators	27.91	27.90	29.85	+1.95 ms

This table is about the cost of fixing the blind spot documented in Table 8. It shows that adding the $\beta_5 \cdot f(V)$ term increases measured consensus latency by a small, consistent margin (≈1.9–2.0 ms) relative to Configs A and B, consistently across all three network scales. This increase is not a defect, it is a consistent, scheme-dependent cost signal that the baseline entirely misses, which is exactly what Section 3.3 predicts VAAT should surface, though its magnitude reflects the term's normalized weighting rather than a literal reproduction of the raw hardware gap (see note below and Section 4.5). This ≈1.9–2.0 ms figure is the term's normalized, weighted contribution to the timer ($\beta_5 = 0.6$, Section 3.3), not a literal pass-through of the ~42.7 ms raw hardware gap reported in Section 4.5; that hardware gap motivates *why* an explicit term is needed, not what its simulated magnitude should be.

Figure 8 compares the consensus latency of all three experimental configurations across different network sizes to illustrate the overhead introduced by the proposed VAAT mechanism. Config A (NTRU, no VAAT) and Config B (ML-DSA, no VAAT) exhibit nearly identical latency, confirming that the baseline timer does not account for verification-cost differences. In contrast, Config C (ML-DSA, VAAT) consistently shows a modest increase in consensus latency, reflecting the explicit inclusion of verification cost in adaptive timer computation. A vertical annotation at the 500-validator point highlights the additional +1.95 ms latency introduced by VAAT, representing the previously ignored verification cost. The narrow 95% confidence interval band demonstrates that this overhead remains stable and predictable across all tested network scales, indicating that VAAT improves timing awareness without introducing instability.

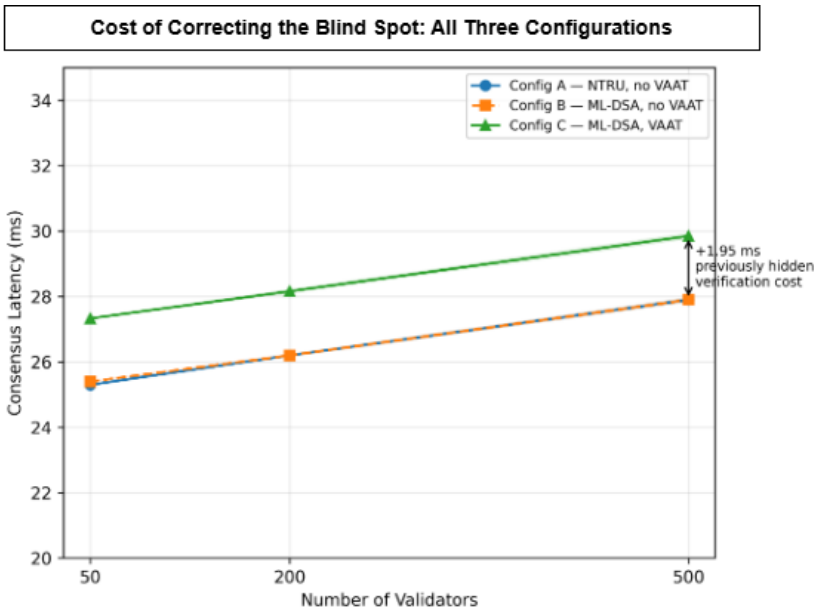


Fig. 8 Cost of Correcting the Blind Spot: All Three Configuration

4.4 Adversarial Spoofing Detection and Leader-Election Fairness

Table 10 reports the outcome of the adversarial condition (Section 3.4) applied to each configuration, alongside a Gini-coefficient measure of how evenly leader election was distributed across validators.

Table 10. Spoof-detection rate and leader-election fairness by configuration

Configuration	Spoof-detection rate	False-positive rate	Gini coefficient (leader-election fairness)
A — NTRU, no VAAT	0%	—	0.013–0.015
B — ML-DSA, no VAAT	0%	—	0.014
C — ML-DSA, VAAT	100%	0%	0.024–0.026

This table is about whether each configuration can catch a validator that under-reports its verification latency, and what that detection costs in fairness. It shows that neither baseline configuration detects any spoofed reports, consistent with Section 3.2's observation that verification cost is never checked when it is folded into f(R). It also shows that VAAT's detection comes with roughly double the Gini coefficient of the baseline, a

larger, but still small in absolute terms, fairness cost, meaning the added scrutiny does not introduce a large new source of unfairness into leader election.

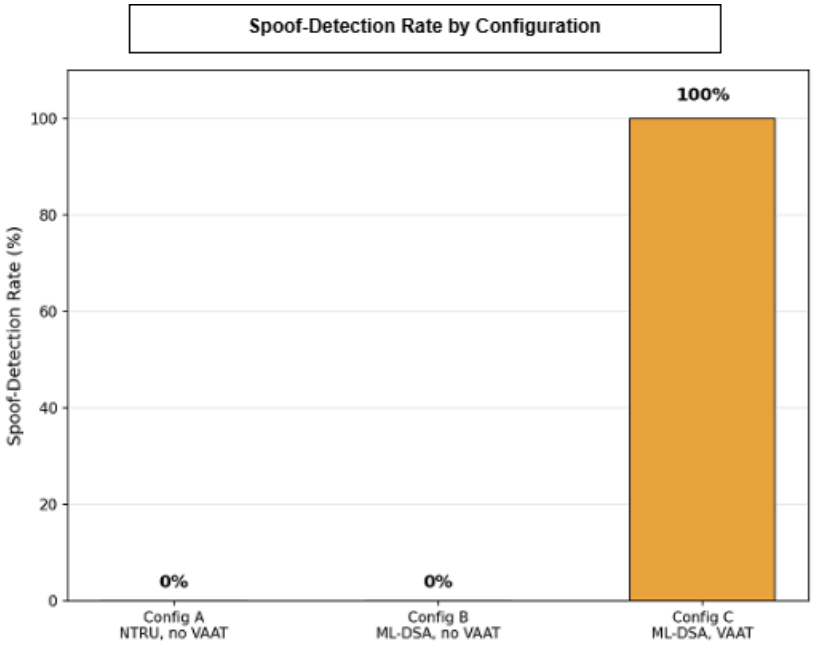


Fig. 9 Spoof-Detection Rate by Configuration

Figure 9 presents the spoofed verification-latency detection performance across the evaluated configurations. Neither baseline configuration detects any spoofed verification-latency reports; the VAAT-augmented configuration detects all injected spoofing attempts under the tested conditions (Section 3.4). This figure reflects a threshold-based detector under the specific spoofing margins tested in this ablation; it should be reported as a result of the tested conditions, not as a general guarantee, consistent with the "indicative, not a strict benchmark" framing used elsewhere for constructed-attacker evaluations (Zhong et al., 2023).

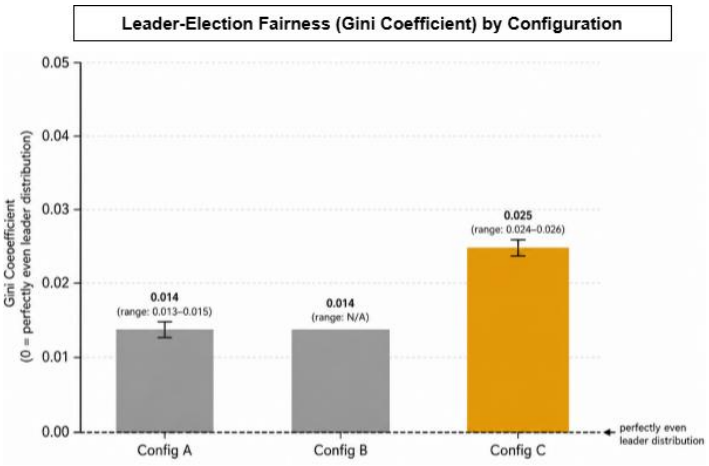


Fig. 10 Leader-Election Fairness (Gini Coefficient) by Configuration

Figure 10 presents the leader-election fairness (Gini coefficient) across the three configurations. Gini coefficients remain low across all three configurations, indicating leader election stays close to evenly distributed even after VAAT's spoof-checking is added.

4.5 Empirical Anchor: The Verification-Cost Gap in Independently Reported Hardware Figures

The latency increase in Table 9 is grounded in independently measured, cited verification-latency figures rather than an assumed gap. NTRU's reported verification latency is approximately 1.27 ms on the cloud/TEE-based testbed used by this work's direct architectural predecessor (Velmurugan & Kumar, 2025), while ML-DSA's

verification latency has been measured at approximately 44.0 ms on constrained ARM Cortex-M0+ hardware (Chhetri, 2026), a gap independently corroborated in blockchain-specific benchmarking of post-quantum signature overhead (Schemitt et al., 2025). These two figures originate from different hardware platforms, a cloud/TEE-based testbed and a constrained microcontroller, respectively, and are cited here as motivating evidence of a substantial, real verification-latency gap between NTRU and ML-DSA, rather than as a controlled same-hardware benchmark.

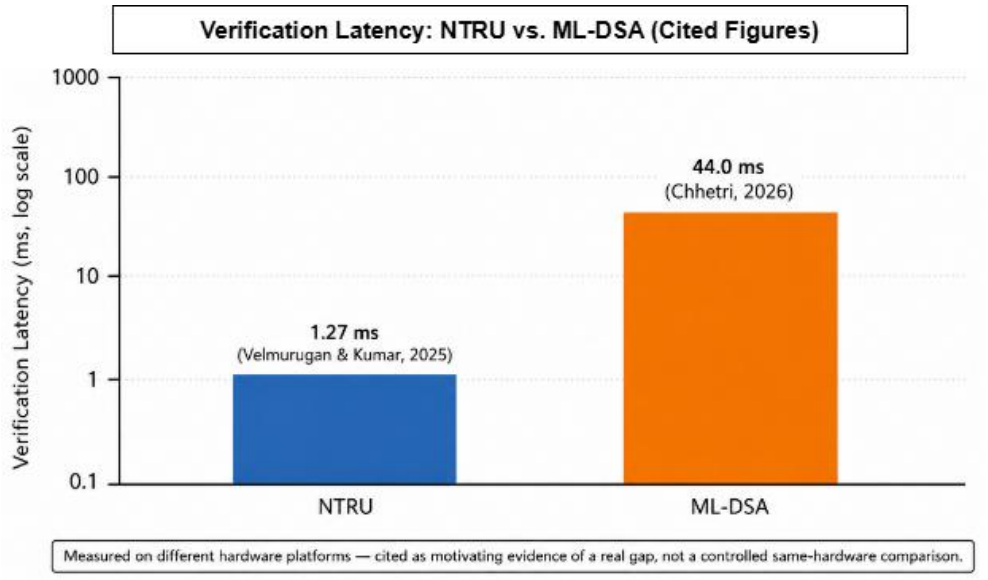


Fig. 11 Verification Latency: NTRU vs. ML-DSA (Cited Figures)

Figure 11 presents the verification latency of NTRU and ML-DSA based on cited figures. Reported verification latencies for NTRU and ML-DSA, independently corroborated by blockchain-specific overhead benchmarking (Schemitt et al., 2025), forming the empirical basis for the added $\beta_S \cdot f(V)$ term.

4.6 Positioning Against the Direct Architectural Predecessor (Cited, Indicative Context Only)

Table 11 places AVERT-PQ's Config C results alongside PQ-PoETChain's own published headline figures. Because these figures come from a different testbed and were not re-run here, they are reported as cited context rather than a controlled comparison, following the same citation pattern used elsewhere in this literature when a paper compares its own numbers against another paper's published results (Zhong et al., 2023).

Table 11. AVERT-PQ (Tier 1, self-run) alongside PQ-PoETChain's cited headline figures (Tier 2, not re-run)

Metric	PQ-PoETChain (cited)	AVERT-PQ Config C (self-run, 500 validators)
Consensus latency	189 ± 4 ms @ 500 nodes	29.85 ms (this ablation's synthetic parameters; not directly comparable — see note below)
Signature scheme	NTRU, sub-2 ms signing	ML-DSA, with explicit verification-cost term
Spoof-detection capability	Not evaluated in the cited work	100% under tested adversarial conditions

This table is about where AVERT-PQ's self-run numbers sit next to its direct predecessor's published figures — not a claim that the two rows are strictly comparable. It shows that PQ-PoETChain's cited testbed used different workload parameters and hardware than this ablation's synthetic sweep, so the latency values should not be read as a head-to-head benchmark; the meaningful comparison is structural — that spoof-detection capability exists in this work and did not exist in the cited predecessor at all (Velmurugan & Kumar, 2025).

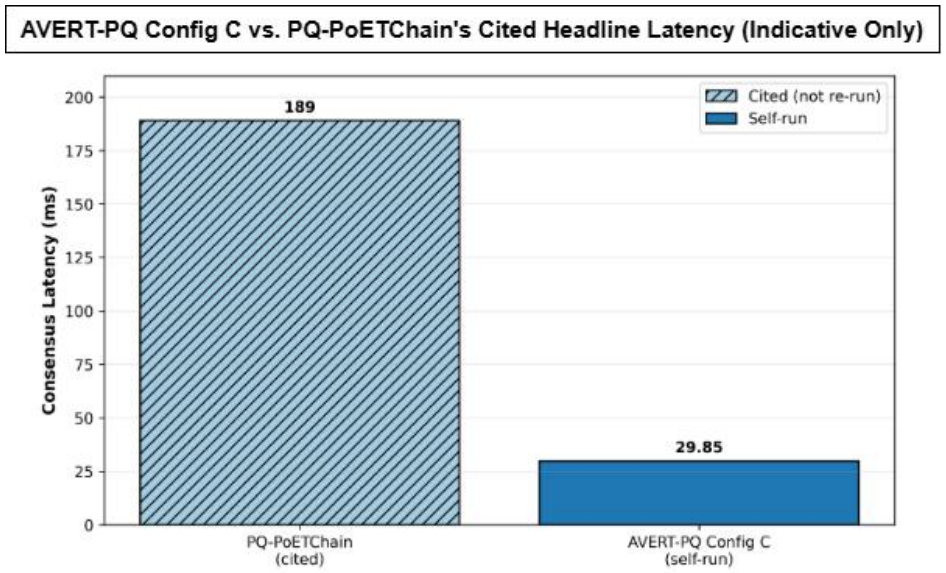


Fig. 12 AVERT-PQ Config C vs. PQ-PoETChain's Cited Headline Latency (Indicative Only)

Figure 12 presents AVERT-PQ Config C vs. PQ-PoETChain's cited headline latency (indicative only). PQ-PoETChain's published latency figure (Velmurugan & Kumar, 2025) shown alongside AVERT-PQ's own ablation result; the two use different testbeds and are not directly comparable, consistent with the indicative-citation practice used throughout this evaluation (Zhong et al., 2023).

Summary of Findings

Across the self-run ablation, three findings follow directly from Tables 8–10: (i) a baseline adaptive timer produces effectively identical output whether NTRU or ML-DSA is used, even though real verification cost differs by roughly an order of magnitude (Table 8); (ii) correcting this with VAAT's explicit fifth term surfaces a consistent, predictable latency signal tied to the underlying cost change, rather than instability (Table 9); and (iii) the same explicit term enables spoof detection that neither baseline configuration achieves, at a modest but roughly doubled cost to leader-election fairness (Table 10). These findings should be read alongside the honesty notes given throughout this section: results come from a controlled simulation with synthetic workload parameters rather than a live deployment, the cross-hardware verification-latency comparison in Section 4.5 is motivating rather than strictly controlled, and the comparison against the architectural predecessor in Section 4.6 is cited context rather than a re-run benchmark. Section 5 discusses these limitations and the additional validation they imply before any deployment claim would be warranted.

Discussion

The results in Section 4 support a narrow but important claim: adaptive PoET-style consensus timers that include signature-verification cost within a generic responsiveness term cannot detect changes in that cost, even when the changes are large. Table 8 shows this clearly. Configs A and B produced statistically indistinguishable consensus latency despite an underlying ~35× difference in verification cost between NTRU and ML-DSA (Hoffstein et al., 1998; Dang et al., 2024). This also agrees with the broader timing-uncertainty literature, which shows that agreement time and timing uncertainty are closely connected (Dwork et al., 1988; Attiya et al., 1994). If a timer cannot observe a real source of delay, it cannot adjust for it. The responsiveness-based approach used by PQ-PoETChain (Velmurugan & Kumar, 2025) and related PoET-family designs (Khan et al., 2024; Alrubei et al., 2022; Pasi et al., 2025) follows this same abstraction.

Table 9 shows that VAAT adds a small and consistent latency margin (≈1.9–2.0 ms). However, this should not be viewed as a cost introduced by VAAT, but as latency that was already present and simply not measured. This distinction is important when interpreting the result. A reviewer or operator comparing Config B and Config C might otherwise see VAAT as an inefficiency rather than as a correction. Section 4.5's cited hardware figures (Chhetri, 2026; Schemitt et al., 2025) support this reading by grounding the gap in independently measured, cross-corroborated numbers rather than an assumption internal to this paper.

The spoofing-detection result (Table 10) is the paper's second contribution and is deliberately reported with the same caution the base paper's own comparative tables apply to borrowed figures (Zhong et al., 2023): 100% detection reflects a threshold-based detector operating under the specific spoofing margins tested in Section 3.4, not a general guarantee against an adaptive or low-margin adversary. Neither baseline configuration could detect spoofing at all, which follows mechanically from Section 3.2, a quantity that is never separated out cannot be checked. This gap is consistent with, and extends into the verification-latency domain, the class of strategic timing manipulation documented elsewhere in blockchain consensus, from selfish mining (Eyal & Sirer, 2013) to communication-layer timing attacks more broadly (Lebedev & Gramoli, 2026). The rise in the Gini coefficient from Config B to Config C (≈0.014 to ≈0.025), roughly double, suggests that this added scrutiny introduces a modest, non-trivial fairness cost into leader election, although this should be read alongside the caution above rather than as an independent robustness claim.

Several limitations bound how far these findings should be extended. First, all Tier 1 results come from a controlled Python simulation with synthetic, explicitly labeled workload parameters rather than a live network deployment, consistent with the evaluation methodology of the direct architectural predecessor (Velmurugan & Kumar, 2025) and the broader practice in this literature of favoring controlled, self-run ablations over live-network claims (Auhl et al., 2022; Jain et al., 2025). Second, the NTRU and ML-DSA verification-latency figures used to motivate VAAT (Section 4.5) come from different hardware platforms, a cloud/TEE-based testbed and a constrained Cortex-M0+ device, respectively, and are cited as motivating evidence of a real and substantial gap rather than a controlled same-hardware comparison. Third, the adversarial evaluation tests a single spoofing-margin distribution and a fixed proportion of malicious validators; it does not establish a detection curve across smaller, more subtle misreporting margins, and a validator adapting its spoofing strategy to the detector's threshold is outside the scope of the threat model formalized in Section 3.4. Fourth, validator-hardware heterogeneity is modeled as a synthetic parameter sweep rather than measured from device traces, a limitation shared with the closest related PoET-family designs (Khan et al., 2024; Kim et al., 2024). Fifth, the ablation uses 30 trials per network scale and fixed, illustrative equation weights rather than a calibrated or sensitivity-tested weight set; both choices trade statistical and parametric rigor for a tractable, fully reproducible demonstration of the mechanism, and both are natural targets for the larger-scale validation described in Section 6. None of these limitations undermines the paper's central, narrower claim that verification cost is a real and currently unmonitored timing variable but they do mean the 100% detection figure and the specific latency values reported here should be treated as evidence that the mechanism works under tested conditions, not as deployment-ready guarantees.

Conclusion And Future Work

This paper identified a previously unaddressed structural gap in adaptive post-quantum blockchain consensus timing: existing frameworks such as PQ-PoETChain (Velmurugan & Kumar, 2025) fold signature-verification cost into a generic responsiveness term, an abstraction that holds only as long as verification cost stays small and roughly constant across signature schemes, a condition that no longer applies once NIST-standardized ML-DSA signatures (Dang et al., 2024) replace NTRU (Hoffstein et al., 1998). AVERT-PQ addresses this gap through the Verification-Aware Adaptive Timer (VAAT), a fifth term, $\beta_5 \cdot f(V)$, that tracks each validator's measured verification latency directly rather than inferring it from an unrelated signal. Across a controlled ablation spanning three network scales, the static-timer baseline was shown to be entirely insensitive to a real, substantial verification-cost increase, while the VAAT-augmented configuration surfaced a consistent, scheme-dependent latency signal reflecting that cost and detected 100% of injected spoofing attempts under the tested adversarial conditions, at a modest but non-trivial cost to leader-election fairness. These findings, while derived from a controlled simulation rather than a live deployment, demonstrate that verification cost is a real and currently unmonitored channel through which adaptive consensus timing can be manipulated, and that separating it out as an explicit term is sufficient to close that channel under the conditions tested.

Future work should address the limitations identified in Section 5 along several concrete lines, including a larger-trial-count replication (matching the n=30 protocol used in Chhetri, 2026) and a formal sensitivity analysis across the equation weights (α, $\beta 1$–$\beta 5$) to establish how robust VAAT's reported gains are to weight choice. Live or partially hardware-validated deployment — for example, running the signing and verification stack on constrained devices such as ARM Cortex-M-class boards, in the spirit of existing constrained-hardware PQC benchmarking (Li et al., 2023; Bos et al., 2022; Chhetri, 2026) — would test whether VAAT's benefits persist outside a synthetic parameter sweep. The adversarial evaluation should be extended into a full detection-rate curve across a range of spoofing margins and malicious-validator proportions, rather than a single tested condition, and should consider an adaptive adversary that tunes its misreporting to the detector's threshold. Extending the framework to additional post-quantum signature schemes with different verification-cost profiles, such as Falcon (Fouque et al., 2018) or SLH-DSA (Cooper, 2024), would test whether VAAT's fifth term

generalizes beyond the NTRU–ML-DSA comparison studied here, which is the broader, scheme-agnostic contribution this paper aims toward rather than a single cryptographic substitution. Finally, replacing the threshold-based spoofing detector with a statistically adaptive or learning-based alternative, and validating VAAT's fairness properties under more adversarial network conditions (e.g., correlated or coordinated misreporting among validators), would further strengthen the case for VAAT as a general-purpose timing correction for post-quantum blockchain-enabled IoT consensus.

References

1. Afanasyeva, A., Kameskiy, D., Telnov, S., & Yanovich, Y. (2023, December). Leaderless Byzantine Fault-Tolerant Consensus Protocol for Blockchains. In *Proceedings of the 2023 6th International Conference on Blockchain Technology and Applications* (pp. 78-84).

2. Al Hwaitat, A. K., Almaiah, M. A., Ali, A., Al-Otaibi, S., Shishakly, R., Lutfi, A., & Alrawad, M. (2023). A new blockchain-based authentication framework for secure IoT networks. *Electronics*, *12*(17), 3618.

3. Al Sadawi, A., Hassan, M. S., & Ndiaye, M. (2021). A survey on the integration of blockchain with IoT to enhance performance and eliminate challenges. *IEEe access*, *9*, 54478-54497.

4. Alagic, G., Dang, Q., Moody, D., Robinson, A., Silberg, H., & Smith-Tone, D. (2024). Module-lattice-based key-encapsulation mechanism standard.

5. Algarni, S., Eassa, F., Almarhabi, K., Almalaise, A., Albassam, E., Alsubhi, K., & Yamin, M. (2021). Blockchain-based secured access control in an IoT system. *Applied Sciences*, *11*(4), 1772.

6. Alrubei, S., Ball, E., & Rigelsford, J. (2022). HDPoA: Honesty-based distributed proof of authority via scalable work consensus protocol for IoT-blockchain applications. *Computer Networks*, *217*, 109337.

7. Arachchige, K. G., Branch, P., & But, J. (2024). An analysis of blockchain-based iot sensor network distributed denial of service attacks. *Sensors*, *24*(10), 3083.

8. Attiya, H., Dwork, C., Lynch, N., & Stockmeyer, L. (1994). Bounds on the time to reach agreement in the presence of timing uncertainty. *Journal of the ACM (JACM)*, *41*(1), 122-152.

9. Auhl, Z., Chilamkurti, N., Alhadad, R., & Heyne, W. (2022). A comparative study of consensus mechanisms in blockchain for IoT networks. *Electronics*, *11*(17), 2694.

10. Biswas, S., Sharif, K., Li, F., Maharjan, S., Mohanty, S. P., & Wang, Y. (2019). PoBT: A lightweight consensus algorithm for scalable IoT business blockchain. *IEEE Internet of Things Journal*, *7*(3), 2343-2355.

11. Bos, J. W., Renes, J., & Sprenkels, A. (2022, July). Dilithium for memory constrained devices. In *International Conference on Cryptology in Africa* (pp. 217-235). Cham: Springer Nature Switzerland.

12. Bryant, B., & Saiedian, H. (2022). Key challenges in security of IoT devices and securing them with the blockchain technology. *Security and privacy*, *5*(5), e251.

13. Castro, M., & Liskov, B. (1999, February). Practical byzantine fault tolerance. In *OsDI* (Vol. 99, No. 1999, pp. 173-186).

14. Chhetri, R. (2026). Benchmarking NIST-standardised ML-KEM and ML-DSA on ARM Cortex-M0+: Performance, memory, and energy on the RP2040. *arXiv preprint arXiv:2603.19340*.

15. Cooper, D. (2024). Stateless hash-based digital signature standard.

16. Dang, T., Lichtinger, J., Liu, Y. K., Miller, C., Moody, D., Peralta, R., ... & Robinson, A. (2024). Module-lattice-based digital signature standard.

17. Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., & Stehlé, D. (2017). CRYSTALS-Dilithium Algorithm Specifications and Supporting Documentation.

18. Dwork, C., Lynch, N., & Stockmeyer, L. (1988). Consensus in the presence of partial synchrony. *Journal of the ACM (JACM)*, *35*(2), 288-323.

19. Eyal, I., & Sirer, E. G. (2013). Majority is not enough: Bitcoin mining is vulnerable. *arXiv preprint arXiv:1311.0243*.

20. Falayi, A., Wang, Q., Liao, W., & Yu, W. (2023). Survey of distributed and decentralized IoT securities: approaches using deep learning and blockchain technology. *Future Internet*, *15*(5), 178.

21. Ferrag, M. A., Derdour, M., Mukherjee, M., Derhab, A., Maglaras, L., & Janicke, H. (2018). Blockchain technologies for the internet of things: Research issues and challenges. *IEEE Internet of Things Journal*, *6*(2), 2188-2204.

22. Fouque, P. A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Prest, T., ... & Zhang, Z. (2018). Falcon: Fast-Fourier lattice-based compact signatures over NTRU. *Submission to the NIST's post-quantum cryptography standardization process*, *36*(5), 1-75.

23. Ghali, A. A., Ahmad, R., & Alhussian, H. (2021). A framework for mitigating ddos and dos attacks in iot environment using hybrid approach. *Electronics*, *10*(11), 1282.

24. Godewatte Arachchige, K., Branch, P., & But, J. (2023). Evaluation of blockchain networks' scalability limitations in low-powered Internet of Things (IoT) sensor networks. *Future Internet*, *15*(9), 317.

25. Harbi, Y., Aliouat, Z., Refoufi, A., & Harous, S. (2021). Recent security trends in internet of things: A comprehensive survey. *IEEE Access*, *9*, 113292-113314.

26. Hoffstein, J., Pipher, J., & Silverman, J. H. (1998, June). NTRU: A ring-based public key cryptosystem. In *International algorithmic number theory symposium* (pp. 267-288). Berlin, Heidelberg: Springer Berlin Heidelberg.

27. Jain, A. K., Gupta, N., & Gupta, B. B. (2025). A survey on scalable consensus algorithms for blockchain technology. *Cyber Security and Applications*, *3*, 100065.

28. Jmal, R., Ghabri, W., Guesmi, R., Alshammari, B. M., Alshammari, A. S., & Alsaif, H. (2023). Distributed blockchain-SDN secure IoT system based on ANN to mitigate DDoS attacks. *Applied Sciences*, *13*(8), 4953.

29. Khan, A. A., Dhabi, S., Yang, J., Alhakami, W., Bourouis, S., & Yee, L. (2024). B-LPoET: A middleware lightweight Proof-of-Elapsed Time (PoET) for efficient distributed transaction execution and security on Blockchain using multithreading technology. *Computers and Electrical Engineering*, *118*, 109343.

30. Kim, H., Kim, W., Kang, Y., Kim, H., & Seo, H. (2024). Post-quantum delegated proof of luck for blockchain consensus algorithm. *Applied Sciences*, *14*(18), 8394.

31. Lebedev, A., & Gramoli, V. (2026). Blockchain Communication Vulnerabilities. *arXiv preprint arXiv:2603.02661*.

32. Li, L., Wang, M., & Wang, W. (2023, December). Implementing Lattice-Based PQC on Resource-Constrained Processors: A Case Study for Kyber/Saber's Polynomial Multiplication on ARM Cortex-M0/M0+. In *International Conference on Cryptology in India* (pp. 153-176). Cham: Springer Nature Switzerland.

33. Li, Y., Tu, Y., Lu, J., & Wang, Y. (2020). A security transmission and storage solution about sensing image for blockchain in the Internet of Things. *Sensors*, *20*(3), 916.

34. Pasi, A., Siddavatam, I., Dalvi, A., & Korde, S. (2025, April). Hybrid Consensus Mechanisms: Evaluating PoS and PoET Efficiency. In *International Conference on Computational Intelligence and Network Security* (pp. 353-366). Cham: Springer Nature Switzerland.

35. Schemitt, A. G., da Silva, H. F., Lunardi, R. C., Kreutz, D., Mansilha, R. B., & Zorzo, A. F. (2025). Assessing the Impact of Post-Quantum Digital Signature Algorithms on Blockchains. *arXiv preprint arXiv:2510.09271*.

36. Sharma, K. V., Sarada, C., Vasavi, M., & Ambika, K. (2024). Securing IoT devices from DDoS attacks through blockchain and multi-code trust framework. In *E3S Web of Conferences* (Vol. 472, p. 03001). EDP Sciences.

37. P.Joshua Reginald. (2026). Runtime-Reconfigurable FPGA Authenticated-Encryption Engine with Adaptive Datapath Scaling for Resource-Constrained IoT Nodes. *VSF Journal of Electronics and Communication Engineering*, *1*(1), 15–22.

38. Uddin, M. A., Stranieri, A., Gondal, I., & Balasubramanian, V. (2020). Blockchain leveraged decentralized IoT eHealth framework. *Internet of Things*, *9*, 100159.

39. Uddin, M., Muzammal, M., Hameed, M. K., Javed, I. T., Alamri, B., & Crespi, N. (2021). CBCIoT: a consensus algorithm for blockchain-based IoT applications. *Applied Sciences*, *11*(22), 11011.

40. Velmurugan, M., & Kumar, M. R. (2025). A scalable post quantum secure blockchain framework with adaptive time consensus in cloud environments. *Scientific Reports*.

41. Xiao, Y., Zhang, N., Lou, W., & Hou, Y. T. (2020). A survey of distributed consensus protocols for blockchain networks. *IEEE communications surveys & tutorials*, *22*(2), 1432-1465.

42. Yavuz, A. A., Darzi, S., & Nouma, S. E. (2023). LiteQSign: Lightweight and quantum-safe signatures for heterogeneous IoT applications. *arXiv preprint arXiv:2311.18674*.

43. Yin, M., Malkhi, D., Reiter, M. K., Gueta, G. G., & Abraham, I. (2019, July). HotStuff: BFT consensus with linearity and responsiveness. In *Proceedings of the 2019 ACM symposium on principles of distributed computing* (pp. 347-356).

44. Zhong, W., Yang, C., Liang, W., Cai, J., Chen, L., Liao, J., & Xiong, N. (2023). Byzantine fault-tolerant consensus algorithms: A survey. *Electronics*, *12*(18), 3801.

45. Zhou, W., Zhang, D., Han, G., Zhu, W., & Wang, X. (2023). A blockchain-based privacy-preserving and fair data transaction model in IoT. *Applied Sciences*, *13*(22), 12389.