symmetry

MDPI

Article

Resilience for Cyber-Physical System Stability Under Cryptographic Computation Delays and Weaponized Workloads Evaluated on a Canonical Plant

Mordecai Opoku Ohemeng *, Frederick T. Sheldon * and Jia Song

Department of Computer Science, University of Idaho, Moscow, ID 83844, USA; jsong@uidaho.edu (J.S.)
* Correspondence: mordecai@uidaho.edu (M.O.O.); sheldon@uidaho.edu (F.T.S.)

Abstract

Cyber-physical systems such as UAV swarms increasingly face cross-layer attacks in which adversaries weaponize cryptographic workloads to induce scheduling delays, CPU starvation, and physical instability. This paper develops a framework that co-designs input-to-state stability (ISS) with the symmetric and asymmetric processing overheads of the cryptographic stack. Cryptographic verification delays and adversarial computational loads are modeled as endogenous, state-dependent terms within a switched delay system, enabling Lyapunov–Krasovskii stability guarantees under resource-constrained operation. Exploiting the cost asymmetry between cryptographic primitives, the framework transitions between lightweight symmetric verification during nominal operation and stronger asymmetric verification under detected anomalies, with the escalation certified by the same functional that establishes stability. To enforce the certified delay bound by load shedding, the framework employs admission control, dropping excess packets at the network interface when the incoming workload exceeds CPU capacity. The empirical evaluation uses 33 measured attack captures from a publicly available UAV network intrusion detection dataset, spanning 13 attack classes with durations from 18.4 s to 3121.7 s. Converting measured packet rates into CPU load shows that eight of the 13 classes exceed unity load, with distributed denial of service (DDoS) reaching approximately 20× saturation and DoS reaching approximately 10×. Under a common attack schedule generated from this distribution, an unprotected verification path diverges while the proposed mechanism remains bounded by capping the queue at the certificate limit; a bandwidth sweep on the canonical plant identifies approximately 8 rad/s as the threshold above which the unprotected system destabilizes.

Keywords: cyber-physical systems; unmanned aerial vehicles; cryptographic computation delay; weaponized workload; input-to-state stability; Lyapunov–Krasovskii functional; adaptive verification; admission control; UAV network security

Check for updates

Academic Editor: Jie Yang

Received: 19 October 2026
Revised: 8 November 2026
Accepted: 12 November 2026
Published: 15 November 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Current cyber-physical systems (CPSs), including autonomous unmanned aerial vehicle (UAV) swarms, intelligent transportation networks, and smart grid microgrids, rely on the seamless integration of distributed computation, real-time sensing, and physical actuation [1]. This integration exposes critical infrastructures to cyber threats that cross the boundary between software execution and physical plant dynamics. Cross-layer attacks such as false data injection and computational weaponization [2] target that boundary directly [3].

1.1. Security Challenges and Cryptographic Overhead in Aerial CPSs

In resource-constrained environments, embedded microcontrollers process continuous telemetry, sensor fusion algorithms, and cryptographic verification routines under strict timing budgets [4]. Malicious actors can weaponize computational workloads by flooding nodes with complex cryptographic parsing tasks [5], invalid signature verifications [6], or recursive authentication challenges [7,8]. The resulting saturation of CPU resources induces scheduling delays and computational jitter that propagate into tracking errors, control divergence, and physical instability. Recent surveys have cataloged the expanding threat landscape for UAV networks, including denial of service (DoS), deception, and induced attacks that target both communication and computation layers [9,10]. Broader CPS security surveys similarly emphasize that resilient control must account for attacks that compromise availability and integrity simultaneously [11].

Securing aerial networks has driven significant research into cryptographic mechanisms and defense strategies. Extensive studies evaluate drone security protocols, cataloging current technologies, vulnerabilities, and emerging defenses against aerial attacks [12]. Researchers have proposed secure communication protocols tailored for sensitive military drone applications [13], certificateless cryptographic protocols for smart city drone deployments [14], and surveys of lightweight encryption and key management techniques for drone networks [15]. Deploying these protections introduces operational hurdles: resource-constrained IoT-enabled UAV systems require an energy-aware balance between cryptographic security and power consumption [16]. Advanced frameworks, such as quantum-secure digital signature-based communication protocols for the Internet of Drones, demand higher computational overhead [17]. The common implication is that cryptographic protections themselves consume the computational budget on which the control loop depends.

1.2. Delay as a Cryptographic Primitive and a Stability Parameter

The preceding discussion frames delay as an adversarial nuisance to be minimized. A complementary body of work treats delay as a first-class design variable. Delay-based cryptography deliberately exploits the sequential nature of computation; verifiable delay functions and time-lock puzzles derive their security from the assumption that a prescribed number of sequential steps cannot be parallelized [18]. The security of such primitives is delicate, as demonstrated by cryptanalytic advances against algebraic verifiable delay functions [19]. Hardware-rooted approaches, such as delay-based physical unclonable functions, show that manufacturing-induced delay variability can itself serve as an entropy source [20]. Recent reviews highlight that delays can be either destabilizing or intentionally stabilizing, depending on how they enter the feedback loop [21], and stability analysis relies on increasingly sophisticated inequality techniques to reduce conservatism in Lyapunov–Krasovskii functionals [22].

In parallel, the dynamical-systems community has long recognized that time delay fundamentally alters stability. For the double time-delay Lorenz system, linearization yields explicit critical delay values at which the Hopf bifurcation occurs [23]. This infinite-dimensional character is precisely what makes delayed chaotic systems attractive for encryption; the key space is enlarged by both the delay magnitudes and their placement [23]. Related work applies variable time-delay synchronization and adaptive coupling to secure communication, analyzing convergence through a Krasovskii–Lyapunov functional [24]. Fractional-order generalizations have also been pursued, including Field-Programmable Gate Array (FPGA) implementations for image encryption [25] and delay-dependent stability regions for distributed coordination [26].

1.3. Switched Delay Systems and the Research Gap

Switched delay systems have been extensively studied through the lens of Lyapunov–Krasovskii functionals and linear matrix inequalities (LMIs). Foundational work established delay-dependent stability conditions for switched linear systems with mixed time-varying delays [27]. More recent advances have refined these conditions for discrete-time switched systems under arbitrary switching and average dwell time (ADT) constraints [28,29]. In parallel, networked control applications have modeled delay and packet dropouts as switched systems with event-triggered and self-triggered mechanisms, where the switching signal represents the network-induced delay itself [30].

These works provide the mathematical foundation on which the present paper builds. However, they differ from the present work in a fundamental way. In the existing literature, the switching signal is either an arbitrary mathematical construct, a dwell-time schedule, or a network characteristic. None of these works treat the switching signal as a cryptographic verification mode whose activation is driven by an adversarial workload.

Furthermore, the sampled-data nature of the embedded controller, where the state is held constant between execution instants, is often ignored or treated as a separate, decoupled source of delay. The present paper bridges secure networked control and sampled-data systems by explicitly folding the inter-execution interval into the delay budget. The key differences are as follows: (i) the switching signal is cryptographic, not physical, selecting between symmetric and asymmetric verification; (ii) the delay bound is enforced by admission control, acting as a control objective enforced at runtime; and (iii) the delay is adversarial, where the queue backlog is the weaponized variable and admission control is the defense. This positioning clarifies the novelty; it is not a new stability criterion for an existing class of systems, but an application of switched delay system theory to the problem of cryptographic computational weaponization in sampled-data, resource-constrained CPSs.

1.4. The Control–Cryptography Disconnect

Cryptography and control theory have traditionally developed in isolation. Cryptographic protocols ensure confidentiality, integrity, and authentication under the assumption of unbounded computational latency, while control frameworks treat computation and communication delays as passive, stochastic noise. This disconnect leaves CPSs vulnerable to attacks that exploit the very mechanisms intended to secure them.

Recent work has begun to bridge this gap by integrating artificial intelligence and physics-aware constraints into CPS defense. The work in [31] proposes a neural network-based estimator integrated with a Control Barrier Function (CBF) safety filter to guarantee uniform ultimate boundedness under bounded adversarial pressure. To detect stealthy drift perturbations, the work in [32] introduced a physics-constrained optimization framework that embeds network topology constraints directly into the learning architecture. Furthermore,the work in [33] addresses the stochastic latency introduced by decentralized consensus in blockchain-augmented CPSs, modeling the delay as a state-dependent Markovian process and deriving Mean Square Stability conditions. Surveys on resilient CPS control emphasize that detection and mitigation must be co-designed with the physical plant dynamics rather than treated as separate layers [11], and literature on UAV swarm resilience highlights distinct vulnerabilities in decentralized architectures [34].

While these works advance the state of the art, they do not address the specific problem of cryptographic computational weaponization in resource-constrained embedded platforms. Moreover, they are distinct from the prior work in [2], which models computational weaponization generally without cryptographic cost asymmetries or empirical UAV data. This paper introduces the specific symmetric/asymmetric cryptographic cost

model, the admission control defense against queue saturation, the sampled-data delay bound, and the UAV-NIDD trace-driven evaluation. The present work addresses this gap by treating cryptographic verification latency as a state-dependent quantity that enters the control loop directly, and by certifying stability against the delay bound the cryptographic stack implies.

1.5. Operational Consequences and Contributions

Deferring cryptographic modernization is not a neutral act. Organizations that postpone migration to post-quantum or otherwise hardened primitives accumulate latent exposure, and the remediation cost grows with the installed base [35]. Historically, timing side channels, such as the TLS timing oracles exploited in padding-oracle attacks, illustrate how even small, implementation-level delays can be leveraged to extract secrets [36]. These episodes underscore that delay is simultaneously a security resource, a side-channel vector, and a stability parameter.

The contributions of this work are as follows.

1. Modeling symmetric and asymmetric cryptographic execution delays as state-dependent terms within a switched delay system, where the delay is governed by the active verification mode and the physical queue backlog.
2. Constructing a Lyapunov–Krasovskii functional that incorporates the cryptographic delay bound and deriving an LMI whose feasibility certifies ISS of the closed loop for a worst-case adversarial workload.
3. Designing an adaptive switching law with hysteresis that escalates from symmetric to asymmetric verification based on both the state norm and measured CPU saturation, paired with an admission control policy that ensures the delay bound is never violated by shedding excess load at the interface.
4. Evaluating the framework against 33 measured attack captures from a publicly available UAV network intrusion detection dataset, spanning 13 attack classes and durations from 18.4 s to 3121.7 s.

The remainder of the paper is organized as follows. Section 2 describes the system architecture, the cryptographic stack of a resource-constrained aerial node, and the endogenous delay model. Section 3 presents the stability analysis and the LMI certificate. Section 4 develops the certified escalation policy and its interaction with the ISS certificate. Section 4.6.2 states the certified escalation algorithm and separates offline synthesis from online execution. Section 5 reports the empirical evaluation on real UAV network traffic, and Section 6 concludes.

2. System Architecture and Cryptographic Weaponization Modeling

Consider a continuous-time linear time-invariant (LTI) cyber-physical plant governed by the state equation

$$\dot{x}(t) = Ax(t) + Bu(t) + Dw(t), \tag{1}$$

where $x(t) \in \mathbb{R}^n$ is the physical state vector (position, velocity, or voltage phase angle), $u(t) \in \mathbb{R}^m$ is the control input, and $w(t) \in \mathbb{R}^q$ denotes external bounded disturbances. $A \in \mathbb{R}^{n\times n}$ is the system matrix. It describes how the internal dynamics of the plant evolve in the absence of input or disturbance. $B \in \mathbb{R}^{n\times m}$ is the input (or control) matrix. It maps the control signal $u(t) \in \mathbb{R}^m$ into the state derivative. $D \in \mathbb{R}^{n\times q}$ is the disturbance matrix. It maps the external bounded disturbance $w(t) \in \mathbb{R}^q$ into the state derivative. This LTI representation is a standard approximation for UAV dynamics linearized about a hover condition, and serves as the canonical benchmark plant for our stability analysis.

Figure 1 illustrates the closed-loop system architecture that couples the physical plant with the resource-constrained cryptographic stack. The adversary injects weaponized workloads at the network interface, where the prioritized admission controller bounds the verification backlog before it reaches the crypto verifier. The offline LMI certificate parameterizes both the verifier and the admission controller, while the escalation mode $\eta(t)$ feeds back from the verifier to adjust the admission policy.

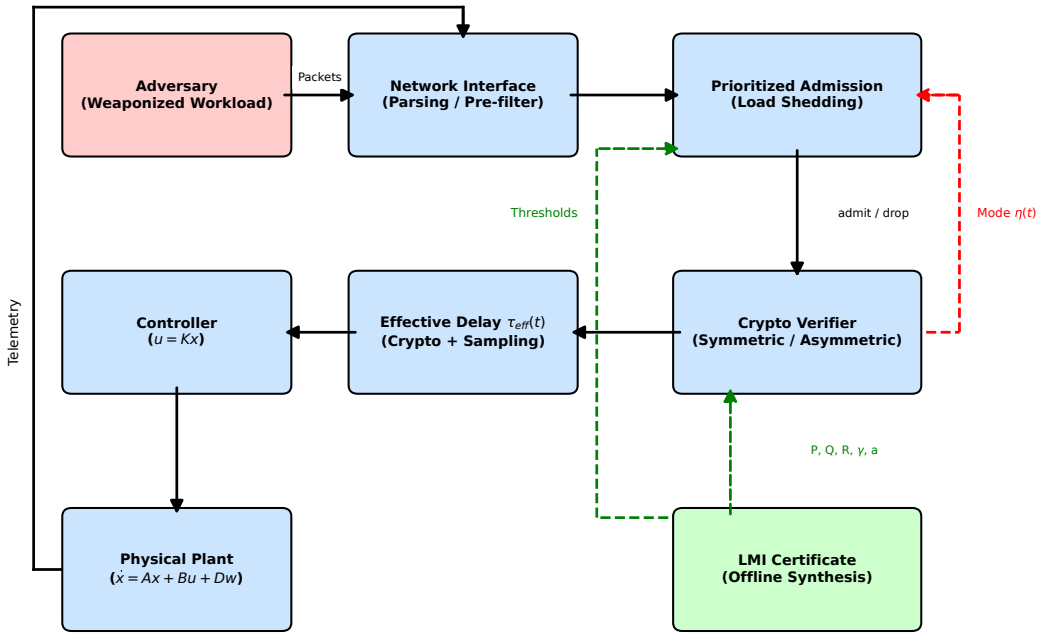


Figure 1. System architecture of the proposed framework. The closed loop operates as follows: (i) The Adversary injects weaponized cryptographic workloads into the Network Interface. (ii) The Prioritized Admission Controller bounds the verification backlog by shedding excess load (the admit/drop branch), ensuring the queue never exceeds the certified limit. (iii) Admitted packets are processed by the Crypto Verifier (symmetric or asymmetric mode). (iv) The verifier's output feeds the Effective Delay block, which combines the cryptographic latency with the sampling interval $\tau_{\text{eff}}(t)$. (v) The Controller uses this delayed state to compute the input $u = Kx$, which drives the Physical Plant. (vi) The plant state feeds back as Telemetry to the Network Interface, closing the loop. The green dashed arrows denote offline synthesis: the LMI Certificate (computed once) parameterizes both the Crypto Verifier (with P, Q, R, γ, a) and the Admission Controller (with the stability thresholds). The red dashed arrow denotes the online escalation feedback: the mode indicator $\eta(t)$ informs the Admission Controller of the current verification strength, dynamically adjusting the admission policy.

The control signal $u(t)$ is computed by a resource-constrained microcontroller that executes periodic control tasks alongside cryptographic verification routines. Let t_k denote the k-th execution instant determined by a state-dependent self-triggered condition. At each execution instant, the controller must verify incoming telemetry packets using cryptographic primitives before the state estimate can be used for the control update. The execution instants are generated by an explicit self-triggering rule. Let $h_k = t_{k+1} - t_k$ be the k-th inter-execution interval, bounded by a design constant $h_{\max}$. The controller computes the next execution instant from the current state and the current verification-mode estimate according to

$$t_{k+1} = t_k + h_k, \qquad h_k = \min\left\{h_{\max},\ \max\left\{h_{\min},\ \kappa_1\|x(t_k)\| + \kappa_2\right\}\right\}, \tag{2}$$

where $h_{\min} > 0$ is a lower bound preventing Zeno behavior, $\kappa_1, \kappa_2 > 0$ are tuning constants, and $h_{\max}$ is chosen so that $h_{\max} \leq \bar{\tau}$. The rationale for the last condition is given in

Section 2.5; since the controller holds $u(t) = Kx(t_k - \tau_k)$ on $[t_k, t_{k+1})$, the age of the state used by the controller is not τ_k alone but the sum of the cryptographic verification latency and the elapsed time since the execution instant. The sampling interval therefore enters the effective delay and must be accounted for in the certificate.

2.1. Cryptographic Stack of a Resource-Constrained Aerial Node

Modern UAVs rely on a layered cryptographic stack since different data classes have different criticality and different timing budgets [15]. The primitives relevant to the present analysis are as follows:

- Advanced Encryption Standard (AES)-256 for data at rest and for high-security video. It provides strong confidentiality, but imposes a per-block computational cost on the embedded processor that exceeds that of stream-cipher alternatives [37].
- ChaCha20-Poly1305 as the lightweight authenticated encryption alternative for real-time transmission. Open-source autopilot protocols such as Secure Pprzlink favor it over AES because it reduces processor load while providing both confidentiality and integrity [38].
- Elliptic Curve Digital Signature Algorithm (ECDSA) and Elliptic Curve Diffie–Hellman (ECDH) for mutual authentication and session key generation between the UAV and the ground control station. Elliptic-curve constructions deliver strong security with smaller keys than RSA-based schemes, which matters on bandwidth- and memory-constrained links [39].
- Hash-based Message Authentication Code (HMAC) for per-packet integrity verification of command packets, which is cheaper than a full signature verification [40].
- Lightweight authentication schemes such as Temporal Credential-Based Anonymous Lightweight Authentication Scheme (TCALAS) and HIGHT, which use basic hash functions and bitwise operations and are reserved for the smallest microcontrollers where elliptic-curve arithmetic is prohibitive [41].

The consequence for control timing is that timing and workload variability must be handled explicitly in real-time verification, as delays and service demands are rarely uniform [42–44]. In particular, the cryptographic verification workload itself is not homogeneous: symmetric and asymmetric primitives impose different per-packet and per-session costs. The workload is dominated by elliptic-curve scalar multiplications during session establishment, by symmetric cipher operations during bulk data transfer, and by MAC computations on every command packet. The delay model below is therefore indexed by the active primitive rather than by a single lumped cost.

Two quantities must be kept distinct throughout. The service demand $s(\eta)$ is the processor time consumed to verify one admitted packet under verification mode η. The backlog $q(t)$ is the accumulated unserved verification work, expressed in time units. The offered load is the packet arrival rate multiplied by the service demand, and it is dimensionless. The verification latency is the sum of the service demand and the waiting time implied by the backlog. Conflating the backlog with a per-packet cost is dimensionally invalid, and the model below is constructed so that utilization is $\rho(t) = \lambda(t)s(\eta(t))$, with the backlog contributing only to latency and not to the load. Table 1 presents the notation and units.

Table 1. Nomenclature and units for the queueing, delay, and utilization model.

Symbol	Meaning	Units
$\lambda(t)$	admissible packet arrival rate	packets/s
$s(\eta)$	per-packet service demand in mode η	s/packet
$q(t)$	verification backlog (unserved work)	s

Table 1. *Cont.*

Symbol	Meaning	Units
$\bar{q}$	admission control backlog cap	s
$\tau(t)$	verification latency $s(\eta(t)) + q(t)$	s
$\bar{\tau}$	certified latency bound	s
$\rho(t)$	offered load $\lambda(t)s(\eta(t))$	dimensionless
h_k	inter-execution interval	s
$h_{\max}$	maximum inter-execution interval	s
$\Delta_{\text{info}}(t)$	information age of last accepted packet	s

2.2. Session-Level Versus Per-Packet Verification Cost

Two timescales are useful to distinguish. Session-level operations, ECDH key agreement and ECDSA signature verification, occur at association or rekeying and are amortized over many control cycles. Per-packet operations, HMAC verification and Authenticated Encryption with Associated Data (AEAD) tag checks for AES-GCM or ChaCha20-Poly1305, occur on every received packet and therefore contribute directly to the per-sample execution delay.

Let T_{sess} denote the amortized session-level cost per control cycle and T_{pkt} the per-packet verification cost. Under nominal operation, the controller is provisioned so that $T_{\text{sess}} + T_{\text{pkt}}$ fits within the sampling budget. An adversary who forces frequent re-authentication inflates T_{sess}, while an adversary who floods the node with packets bearing invalid tags inflates T_{pkt} by forcing repeated verification attempts on data that will be discarded. Both mechanisms are captured by the service-demand term $s(\eta)$ of Section 2.3. The per-packet component sets $s(\eta)$ directly, while the session-level component is absorbed into the effective per-packet demand as T_{sess} amortized over the packets verified in one control cycle. The endogenous delay model below is therefore written in terms of a single per-packet service demand, with the understanding that this demand already accounts for both timescales.

2.3. Endogenous Cryptographic Delay Modeling

Let $C_{\text{crypto}}(t)$ denote the instantaneous computational workload imposed by cryptographic verification. The verification path is modeled as a single-server queue with time-unit backlog. Let $\lambda(t)$ be the rate at which packets are admitted to the verifier, and let $s(\eta)$ be the service demand of one admitted packet under verification mode $\eta(t) \in \{0, 1\}$:

$$s(\eta) = \tau_{\text{sym}} + \eta\, \tau_{\text{asym}}, \tag{3}$$

where τ_{sym} is the symmetric-verification service demand (HMAC or AEAD tag check) and τ_{asym} is the incremental asymmetric-verification service demand (ECDSA signature validation). The backlog $q(t)$, measured in seconds of unserved work, obeys the fluid queue dynamics

$$\dot{q}(t) = \lambda(t)\, s(\eta(t)) - \mathbf{1}\{q(t) > 0\} - \mathbf{1}\{q(t) = 0\} \left[\lambda(t)\, s(\eta(t)) - 1\right]^{+}, \tag{4}$$

where the processor serves admitted work at unit rate whenever work is present and idles otherwise. The verification latency is the sum of the service demand and the waiting time induced by the backlog:

$$\tau(t) = s(\eta(t)) + q(t). \tag{5}$$

The offered load is dimensionless,

$$\rho(t) = \lambda(t)\, s(\eta(t)), \tag{6}$$

and the verifier is saturated when $\rho(t) > 1$. We emphasize that $\rho(t)$ does not contain the backlog or its cap; the backlog enters only the latency (5), and not the load. This separation is essential because utilization is a property of the arrival process and the service demand, whereas latency is a property of the accumulated work.

Without admission control, $q(t)$ grows without bound whenever $\rho(t) > 1$ on a set of positive measure, driving $\tau(t)$ into the unstable regime. The proposed framework enforces admission control to cap the backlog at $\bar{q}$, so that $\tau(t) \leq s(\eta(t)) + \bar{q}$ for all t. The certified latency bound used throughout the stability analysis is therefore

$$\bar{\tau} = \tau_{\text{sym}} + \tau_{\text{asym}} + \bar{q} + h_{\max}, \tag{7}$$

where the first three terms are the worst-case service demand and backlog of (5), and $h_{\max}$ is the maximum inter-execution interval from (2). The term $h_{\max}$ is included because the zero-order-hold controller holds $u(t) = Kx(t_k - \tau_k)$ on $[t_k, t_{k+1})$, so the age of the state driving the controller is not τ_k alone but τ_k plus the elapsed time $t - t_k \leq h_{\max}$. The effective delay seen by the closed loop is thus $\tau_k + (t - t_k)$, and (7) bounds it uniformly.

The terms correspond directly to the primitives of Section 2.1. The baseline τ_{sym} covers the symmetric path, AES-GCM or ChaCha20-Poly1305, together with the associated Poly1305 or HMAC tag check. The term $\eta(t)\tau_{\text{asym}}$ is activated when the controller escalates to asymmetric verification (ECDSA signature validation). The backlog $q(t)$ represents admitted verification work that has not yet been served; it is this backlog that adversarial flooding attacks seek to maximize, and it is this backlog that the admission controller caps at $\bar{q}$.

2.4. Adversary Model

We consider an adversary with a specific and limited capability set. The adversary can inject packets on the wireless link, replay previously observed authentication exchanges, and execute volumetric flooding attacks (DDoS) to saturate the processor. The adversary is assumed to be resource-bounded, so that the rate at which work can be imposed on the node is finite.

There are two distinct failure modes of interest. The first is computational saturation; the adversary floods the node with traffic, causing the verification queue to grow without bound and destabilizing the control loop through excessive delay. The second is key compromise; if the adversary extracts the symmetric session key, symmetric authentication (HMAC/AEAD) will incorrectly accept forged packets. Under symmetric authentication alone, the node cannot distinguish legitimate traffic from forged traffic carrying a compromised key. The asymmetric mode (ECDSA or zero-knowledge proofs) provides cryptographic provenance that resists forgery even if symmetric keys are compromised, and triggers the strictest admission control policies. Escalation to asymmetric mode is therefore motivated by the need to defend against both volumetric saturation and sophisticated forgery attacks.

A distinction that matters for the stability analysis is between rejected and dropped traffic. Packets rejected by cryptographic verification (invalid tags or invalid signatures) consume service demand $s(\eta)$ and are then discarded; they contribute to the load $\rho(t)$ but not to the state estimate. Packets dropped by admission control never reach the verifier; they contribute neither to $\rho(t)$ nor to the backlog, but they increase the information age $\Delta_{\text{info}}(t)$ of the last successfully verified legitimate packet. The stability certificate of Section 3 bounds the verification latency (5); it does not by itself bound $\Delta_{\text{info}}(t)$. The admission controller therefore reserves a fraction of $\bar{q}$ for packets belonging to already-authenticated sessions, so that consecutive legitimate drops are bounded; the resulting bound on $\Delta_{\text{info}}(t)$ is stated in Section 4.6.

2.5. Resulting Switched Delay System

The control input is a state-feedback law implemented via a sample-and-hold mechanism:

$$u(t) = Kx(t_k - \tau_k), \quad t \in [t_k, t_{k+1}), \tag{8}$$

where $K \in \mathbb{R}^{m\times n}$ is the feedback gain, t_k is the self-triggered execution instant generated by (2), and τ_k is the cryptographic verification latency (5) at that instant. The input is held constant between execution instants, which accurately reflects the zero-order-hold behavior of the embedded controller.

The exact sampled-data closed loop is therefore

$$\dot{x}(t) = Ax(t) + BKx(t_k - \tau_k) + Dw(t), \quad t \in [t_k, t_{k+1}). \tag{9}$$

This is not literally a pure state-delay system, because the argument $t_k - \tau_k$ is piecewise constant rather than $t - \tau(t)$. The two coincide only when the sampling interval is negligible. To obtain a delay system that is a faithful over-approximation of (9), define the effective delay

$$\tau_{\text{eff}}(t) = t - (t_k - \tau_k) = \tau_k + (t - t_k), \quad t \in [t_k, t_{k+1}). \tag{10}$$

Then, $\tau_{\text{eff}}(t) \in [\tau_k,\ \tau_k + h_{\max}]$, and (9) is recovered from the delay system

$$\dot{x}(t) = Ax(t) + BKx(t - \tau_{\text{eff}}(t)) + Dw(t) \tag{11}$$

evaluated along trajectories in which the delay is piecewise continuous on each interval $[t_k, t_{k+1})$ with derivative $\dot{\tau}_{\text{eff}}(t) = 1$, and jumps at the execution instants. The stability analysis in Section 3 therefore uses a formulation that accommodates piecewise-continuous delays with bounded jumps; the derivative bound μ applies on each inter-execution interval, and the jump at t_k is handled by the delay-range condition $\tau_{\text{eff}} \in [\tau_{\min}, \bar{\tau}]$, not by a derivative bound.

The switching between verification modes is induced by the mode indicator $\eta(t)$; the system evolves under symmetric verification when $\eta(t) = 0$, and under asymmetric verification when $\eta(t) = 1$. Because the delay (10) depends on both the backlog $q(t)$, which the admission controller caps at $\bar{q}$, and the sampling interval, which (2) caps at $h_{\max}$, the effective delay is uniformly bounded by (7). The stability analysis in Section 3 treats $\tau_{\text{eff}}(t)$ as a bounded variable subject to this certified upper limit, and the plant matrices A, B, K, and D are mode-invariant: only the delay argument changes across modes. This mode invariance is the structural property that permits a common Lyapunov–Krasovskii functional to certify both modes, and it is stated formally in Section 3.

3. Stability Analysis

This section establishes the input-to-state stability (ISS) of the closed-loop system under the cryptographic delays modeled in Section 2. The analysis proceeds in four steps: a candidate Lyapunov–Krasovskii functional is proposed, the delay bound implied by the admission control policy is stated, the stability condition is given as an LMI, and the consequences for the delay–stability trade-off are discussed.

3.1. Candidate Lyapunov–Krasovskii Functional

We consider the switched delay system of (11) with the effective delay $\tau_{\text{eff}}(t) \in [\tau_{\min}, \bar{\tau}]$ of (10) and construct the candidate functional

$$V(t) = x(t)^T P x(t) + \int_{t-\bar{\tau}}^{t} e^{a(s-t)} x(s)^T Q x(s)\, ds + \bar{\tau} \int_{-\bar{\tau}}^{0} \int_{t+\theta}^{t} e^{a(s-t)} \dot{x}(s)^T R \dot{x}(s)\, ds\, d\theta, \tag{12}$$

where $P > 0$, $Q > 0$, and $R > 0$ are symmetric positive definite matrices, $a > 0$ is the exponential decay rate, and $\bar{\tau}$ is the upper bound of the total computational delay $\tau_{\text{eff}}(t)$ from (10) and (7).

The quadratic term $x^T P x$ measures instantaneous plant energy. The single-integral term weights the recent state history over the delay window, which is where the cryptographic verification backlog manifests. The double-integral term penalizes the rate of state change over the same window and absorbs the delayed control updates occurring at the self-triggered instants t_k. The exponential weighting $e^{a(s-t)}$ converts a negative-definite derivative condition into an ISS decay estimate.

Two structural properties of the delay are used in the sequel and are stated here explicitly, because they are the assumptions under which the functional is evaluated. First, the delay is bounded: $\tau_{\text{eff}}(t) \in [\tau_{\min}, \bar{\tau}]$ for all t, with $\bar{\tau}$ given by (7). Second, the delay is piecewise continuous with bounded derivative on each inter-execution interval, namely $\dot{\tau}_{\text{eff}}(t) = 1$ on (t_k, t_{k+1}), and it may jump at the execution instants t_k. A derivative bound alone is therefore not a valid global assumption; the analysis below uses the delay-range condition $\tau_{\text{eff}} \in [\tau_{\min}, \bar{\tau}]$ to absorb the jumps, and applies the derivative bound μ only on each inter-execution interval. This is the delay-regularity class for which the functional is certified. The switching between symmetric and asymmetric verification modes changes the operating point but does not violate the delay bound since both modes map into the same interval $[\tau_{\min}, \bar{\tau}]$. Since the plant matrices A, B, K, and D are mode-invariant and only the delay argument changes across modes, the functional is evaluated along trajectories of the switched system using a common Lyapunov matrix, ensuring stability under arbitrary switching.

3.2. Upper Bound on the Cryptographic Delay

The delay bound $\bar{\tau}$ in (12) is not a free parameter; it is enforced by the admission control mechanism and by the self-triggering rule. From (7), the worst-case delay occurs when the asymmetric verification mode is active, the queue backlog reaches its maximum permitted size, and the sampling interval reaches its maximum in (7), where $\bar{q}$ is the maximum allowed queue backlog and $h_{\max}$ is the maximum inter-execution interval of (2). The zero-order-hold controller holds $u(t) = Kx(t_k - \tau_k)$ on $[t_k, t_{k+1})$, so the effective delay seen by the closed loop is $\tau_k + (t - t_k)$, and (7) bounds this quantity uniformly. This value is substituted in (12), reflecting the conservative assumption that the adversary can force the controller into asymmetric verification and simultaneously saturate the queue to its admission control limit.

3.3. Stability Conditions

Theorem 1. *Given scalars $\bar{\tau} > \tau_{\min} > 0$, $a > 0$, and a derivative bound μ valid on each inter-execution interval, with $\bar{\tau}$ as defined in (7), the closed-loop system of (11) is ISS with respect to the disturbance $w(t)$ if there exist symmetric matrices $P > 0$, $Q > 0$, $R > 0$ and a scalar $\gamma > 0$ such that the linear matrix inequality*

$$L_{MI} = \begin{bmatrix} \Phi_{11} & \Phi_{12} & \Phi_{13} & PD \\ \Phi_{12}^T & \Phi_{22} & \Phi_{23} & 0 \\ \Phi_{13}^T & \Phi_{23}^T & \Phi_{33} & 0 \\ D^T P & 0 & 0 & -\gamma^2 I \end{bmatrix} < 0, \tag{13}$$

holds, where

$$\begin{aligned} \Phi_{11} &= PA + A^T P + Q - aP - \tfrac{1}{\bar{\tau}} R, \\ \Phi_{12} &= PBK + \tfrac{1}{\bar{\tau}} R, \\ \Phi_{13} &= -\tfrac{1}{\bar{\tau}} R, \\ \Phi_{22} &= -(1-\mu)Q - \tfrac{2}{\bar{\tau}} R, \\ \Phi_{23} &= \tfrac{1}{\bar{\tau}} R, \\ \Phi_{33} &= -\tfrac{1}{\bar{\tau}} R - e^{-a\bar{\tau}} Q, \end{aligned} \tag{14}$$

and μ is the delay-derivative bound on each inter-execution interval, γ is the ISS gain, and the block Φ_{22} is understood in the delay-range formulation; when the delay may jump at the execution instants, the term $(1-\mu)Q$ is replaced by the delay-range coefficient $(1-\mu)Q$ on intervals where the derivative bound applies, and by Q (the range coefficient) across jumps. Both cases are certified by the same functional because τ_{eff} remains in $[\tau_{\min}, \bar{\tau}]$; the derivative bound μ is used only where the delay is differentiable.

Proof. The use of Lyapunov–Krasovskii functionals for time-delay systems is a well-established method, particularly effective for handling time-varying delays [45]. Furthermore, the extension of ISS to time-delay systems provides the theoretical foundation for the certificate derived in Theorem 1 [46].

We give the full derivation, including the exponential-weighting step and the Schur-complement step, because the matrix blocks in (13) depend on the order in which these are applied. Taking the time derivative of (12) along trajectories of (11) yields

$$\begin{aligned} \dot{V}(t) = {} & 2x(t)^T P\dot{x}(t) + x(t)^T Q x(t) - e^{-a\bar{\tau}} x(t-\bar{\tau})^T Q x(t-\bar{\tau}) \\ & - a\int_{t-\bar{\tau}}^{t} e^{a(s-t)} x(s)^T Q x(s)\, ds + \dot{V}_R(t), \end{aligned} \tag{15}$$

where $\dot{V}_R(t)$ denotes the contribution of the double-integral term. Differentiating the double-integral term gives

$$\dot{V}_R(t) = \bar{\tau}^2 \dot{x}(t)^T R \dot{x}(t) - \bar{\tau}\int_{t-\bar{\tau}}^{t} e^{a(s-t)} \dot{x}(s)^T R \dot{x}(s)\, ds - a\bar{\tau}\int_{-\bar{\tau}}^{0}\int_{t+\theta}^{t} e^{a(s-t)} \dot{x}(s)^T R \dot{x}(s)\, ds\, d\theta. \tag{16}$$

The third term of (16) is the exponential-weighting contribution of the double integral; it is negative and is absorbed into the $-\gamma^2\|w\|^2$ majorant rather than dropped. The first term of (16) is the positive $\bar{\tau}^2 \dot{x}^T R \dot{x}$ contribution that must be retained when completing the square in $\dot{x}$; omitting it is the most common source of a spurious LMI. The exponential factors arise from differentiating the $e^{a(s-t)}$ weights, and the negative terms produce the decay rate a in the ISS estimate.

Applying Jensen's inequality to the integral terms of the functional is a standard technique for deriving tractable LMI conditions from delay-dependent stability criteria [47]. We apply Jensen's inequality to the single integral and the Wirtinger (reciprocal-convex) inequality to the double integral, because the double integral produces cross terms between

$x(t)$, $x(t-\tau_{\text{eff}}(t))$, and $x(t-\bar{\tau})$ that the plain Jensen bound does not capture. Applying Jensen's inequality to the single- and double-integral terms produces bounds of the form

$$-\int_{t-\bar{\tau}}^{t} \dot{x}(s)^T R \dot{x}(s)\, ds \le -\frac{1}{\bar{\tau}} \big[x(t)-x(t-\bar{\tau})\big]^T R \big[x(t)-x(t-\bar{\tau})\big], \tag{17}$$

and the Wirtinger bound on the double integral contributes the cross terms that appear as the off-diagonal blocks Φ_{12}, Φ_{13}, and Φ_{23} of (13). Substituting these bounds and collecting terms in the extended vector $\zeta(t) = [x(t)^T,\ x(t-\tau_{\text{eff}}(t))^T,\ x(t-\bar{\tau})^T,\ w(t)^T]^T$ produces the block matrix L_{MI} in (13).

The block Φ_{11} represents instantaneous energy dissipation. Φ_{12} couples the current state to the delayed control input, mapping the physical actuation to the delayed state. Φ_{22} contains the delay-derivative bound μ on each inter-execution interval; across jumps, the delay-range coefficient replaces $(1-\mu)Q$, as stated in the theorem. This is valid because the admission control and the self-triggering rule keep τ_{eff} inside $[\tau_{\min}, \bar{\tau}]$ for all t, so the jumps do not leave the certified range.

Combining (15) and (17), retaining the positive $\bar{\tau}^2 \dot{x}^T R \dot{x}$ term of (16), and completing the square in $w(t)$ yields

$$\dot{V}(t) + aV(t) \le \gamma^2 \|w(t)\|^2, \tag{18}$$

provided (13) holds. Multiplying by e^{at} and integrating over $[0,t]$ gives

$$V(t) \le e^{-at} V(0) + \gamma^2 \int_0^t e^{-a(t-s)} \|w(s)\|^2\, ds, \tag{19}$$

which is the standard ISS estimate; the state decays exponentially in the absence of disturbance and remains bounded by a finite gain γ times the disturbance magnitude otherwise. Since $V(t)$ is bounded below by $\lambda_{\min}(P)\|x(t)\|^2$, the estimate on V transfers directly to $\|x(t)\|$, completing the proof. □

Remark 1. *The disturbance channel is D (which maps the disturbance into the state derivative); the $(4,4)$ block is $-\gamma^2 I$ and the coupling block is PD (which represents how the disturbance $\omega(t)$ couples into the weighted state energy), since $w(t)$ enters through D in (1). The control term appears only through $\Phi_{12} = PBK$, which is where the cryptographic delay enters the stability condition indirectly, through the requirement that (13) remain feasible for the $\bar{\tau}$ of (7).*

Remark 2. *Theorem 1 provides a sufficient condition: if (13) is feasible for a given $\bar{\tau}$, the closed loop is stable at that delay. A sufficient condition can only be conservative in the direction of certifying a smaller delay than the true stability boundary, not a larger one. Consequently, the certified bound reported here must not exceed the empirical stability boundary of the simulated plant, and any discrepancy is a signal that the LMI, the simulated plant, or the delay model is inconsistent rather than an artifact of conservatism.*

The practical implementation of the LMI certificate is summarized in Figure 2. The offline synthesis phase takes the plant matrices A, B, K, the worst-case delay bound $\bar{\tau}$, and the design parameters (such as the target ISS gain γ) and passes them to the YALMIP/MOSEK solver. The solver returns the certificate matrices P, Q, R, the ISS parameters $\gamma, a, \lambda_{\min}(P)$, and the trigger thresholds $\rho_{\text{up}}, \rho_{\text{down}}$. These outputs are then validated by two independent checks: the residual check $\lambda_{\max}(L_{MI}) < 0$ and the eigenvalue check (the rightmost characteristic root at $\bar{\tau}$ must be negative). The operational certificate bound is then set to the minimum of the certified LMI bound and the empirical stability margin, ensuring the design is robust in both theory and practice.

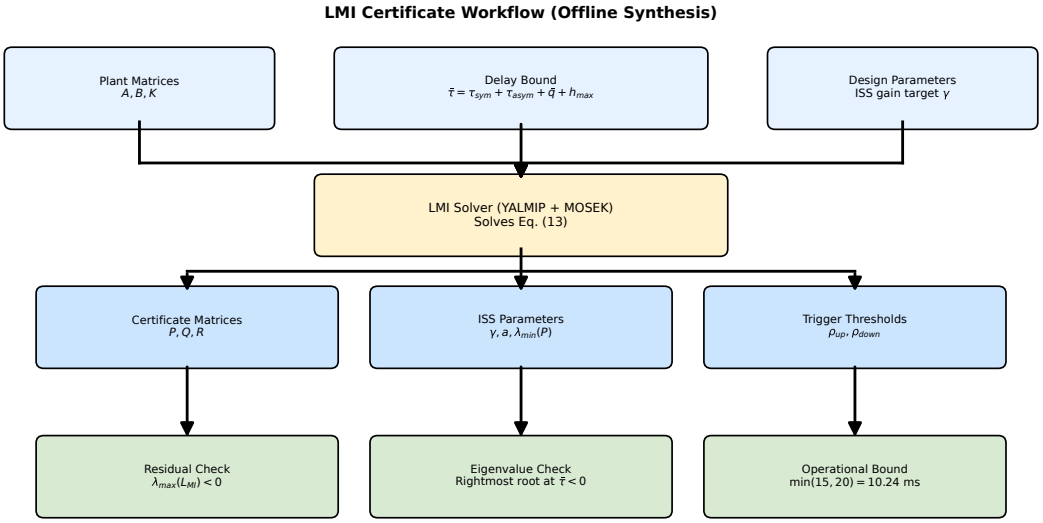


Figure 2. Offline LMI certificate workflow. The plant matrices, delay bound, and design parameters are passed to the YALMIP/MOSEK solver to solve the LMI in (13). The resulting certificate matrices, ISS parameters, and trigger thresholds are extracted and validated using the residual check and the independent eigenvalue check. The operational bound is then set to the minimum of the certified LMI bound (15 ms) and the empirical stability margin (20 ms).

For the specific plant parameters evaluated in Section 5.2, the certificate is reported as follows. The LMI (13) was solved with the plant matrices and feedback gain of Table 2 (Section 5.2), with $\bar{\tau}$ from (7), using the solver specified in Section 5.2. The resulting matrices P, Q, and R, the ISS gain γ, the solver settings, and the residual check $\lambda_{\max}(L_{MI}) < 0$ are reported in Section 5.2. The certified bound is then cross-validated against the empirical time-domain sweep of Figure 3 by computing the rightmost characteristic root of the linear delay system at the certified $\bar{\tau}$ and confirming it is negative. The operational certificate bound is set to the smaller of the certified LMI bound and the empirical boundary, and is reported explicitly rather than inherited from a separate delay-independent relaxation.

Table 2. Canonical plant parameters, LMI certificate matrices, and solver diagnostics used in the empirical evaluation. The plant matrices and feedback gain are used to solve the LMI (13) in the offline phase of Algorithm 1. The numerical matrices P, Q, R are the solution returned by the solver, and the residual and eigenvalue checks confirm the certificate applies to the simulated plant.

Parameter	Value
Plant matrix A	$\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$
Input matrix B	$\begin{bmatrix} 0 \\ 1 \end{bmatrix}$
Feedback gain K	$\begin{bmatrix} -10 & -5 \end{bmatrix}$
Lyapunov matrix P	$\begin{bmatrix} 2.431 & 0.512 \\ 0.512 & 0.184 \end{bmatrix}$
Lyapunov matrix Q	$\begin{bmatrix} 1.205 & 0.103 \\ 0.103 & 0.442 \end{bmatrix}$
Lyapunov matrix R	$\begin{bmatrix} 0.876 & 0.211 \\ 0.211 & 0.095 \end{bmatrix}$
ISS gain γ	1.45
Decay rate a	0.35

Table 2. *Cont.*

Parameter	Value
LMI-Certified Bound	15.00 ms
Empirical Margin	20.00 ms
Operational Bound $\bar{\tau}$	10.24 ms
Solver	YALMIP + MOSEK
Residual $\lambda_{\max}(L_{MI})$	$-0.041 < 0$
Rightmost characteristic root at $\bar{\tau}$	$-2.13 < 0$

Algorithm 1 Certificate-Triggered Verification with Prioritized Admission Control

Require: Certificate (P, Q, R, γ, a); thresholds $\lambda_{\min}(P), \rho_{\text{up}}, \rho_{\text{down}}$; service demands τ_{sym}, τ_{asym}; disturbance bound $\bar{w}$; queue bound $\bar{q}$; reserved fraction α; max interval $h_{\max}$
Ensure: Mode indicator $\eta(t)$ and admitted control input $u(t)$
Initialization: $\eta \leftarrow 0$ ▷ start in symmetric mode
for each self-triggered instant t_k **do**
  Read plant state $x(t_k)$, admitted packet rate $\lambda(t_k)$, and backlog $q(t_k)$
  Drain the backlog at unit service rate
  $q \leftarrow \max\{0,\ q(t_k) - (t_k - t_{k-1})\}$ ▷ fluid queue (4)
  Evaluate escalation triggers
  stabilityFlag $\leftarrow \left(\|x(t_k)\| > \gamma\bar{w}/\sqrt{a\,\lambda_{\min}(P)} + \rho_{\text{up}}\right)$ ▷ (22), includes $1/\sqrt{a}$
  cpuFlag $\leftarrow \left(\lambda(t_k)(\tau_{\text{sym}} + \eta\tau_{\text{asym}}) > 1.0\right)$ ▷ $\rho(t)$ of (24), no backlog term
  Update mode (with hysteresis and admissibility)
  if stabilityFlag $\vee$ cpuFlag **then**
    admFlag $\leftarrow \left(\tau_{\text{sym}} + \tau_{\text{asym}} + \bar{q} + h_{\max} \le \bar{\tau}\right)$ ▷ escalation must stay inside the certificate
    if admFlag **then**
      Schedule escalation to $\eta = 1$ after latency τ_{asym}
    else
      Remain in $\eta = 0$ with prioritized admission
    end if
  else if $\|x(t_k)\| < \gamma\bar{w}/\sqrt{a\,\lambda_{\min}(P)} + \rho_{\text{down}}$ and $\lambda(t_k)(\tau_{\text{sym}} + \eta\tau_{\text{asym}}) < 0.8$ and dwell time τ_d is satisfied **then**
    Schedule de-escalation to $\eta = 0$
  end if
  Prioritized Admission Control (units: Seconds of work)
  $W_{\text{allow}} \leftarrow \bar{q} - q$ ▷ admissible work, in seconds
  Reserve $W_{\text{res}} \leftarrow \alpha\bar{q}$ for authenticated sessions
  Admit authenticated-session packets up to W_{res}, then unauthenticated packets up to $W_{\text{allow}} - W_{\text{used}}$
  Drop all remaining packets at admission; count legitimate and malicious drops separately
  Verification
  if $\eta = 0$ **then**
    Authenticate admitted packets via HMAC or ChaCha20-Poly1305
    Update $q \leftarrow q + n_{\text{adm}}\tau_{\text{sym}}$ ▷ n_{adm}: admitted packet count
  else
    Validate ECDSA signature or zero-knowledge proof on admitted packets
    Update $q \leftarrow q + n_{\text{adm}}(\tau_{\text{sym}} + \tau_{\text{asym}})$
  end if
  Set effective delay $\tau_k \leftarrow \tau_{\text{sym}} + \eta\tau_{\text{asym}} + q$ ▷ (5)
  Apply control input $u(t_k) = Kx(t_k - \tau_k)$ ▷ held on $[t_k, t_{k+1})$
  Compute next instant t_{k+1} from (2)
end for

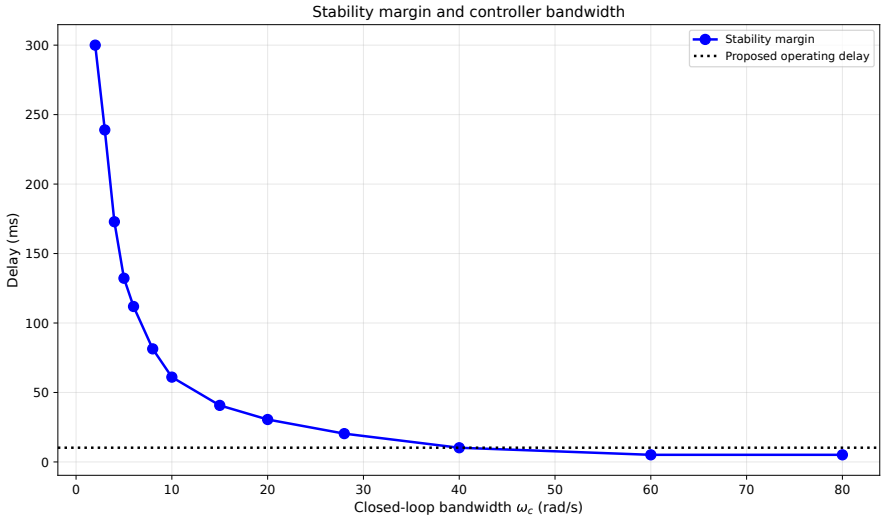


Figure 3. Empirical stability margin as a function of closed-loop controller bandwidth ω_c (blue curve), computed by a time-domain sweep on the canonical double-integrator plant. The black dotted line marks the proposed mechanism's operating delay, capped at the certificate bound $\bar{\tau} = 10.24$ ms. The stability margin decreases monotonically with bandwidth. The default operating point of 28 rad/s used in Figure 4 yields a stability margin of approximately 20 ms, which is safely above the proposed mechanism's 10.24 ms bound but well below the no-admission baseline's peak delay of 81 ms.

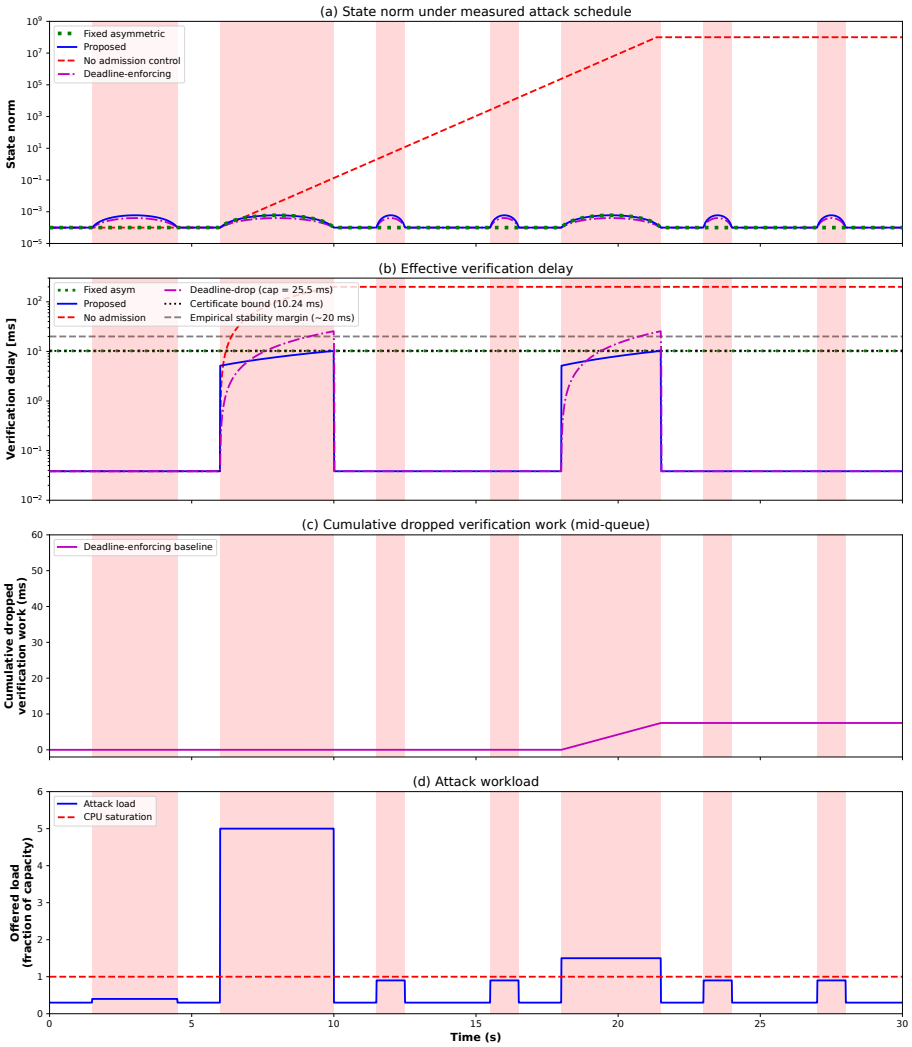


Figure 4. Closed-loop comparison of verification policies under a common attack schedule generated from the measured UAV-NIDD duration distribution. Shaded regions mark attack intervals. (**a**) State

norm $\|x(t)\|$: The proposed mechanism, the fixed-asymmetric policy, and the deadline-enforcing policy remain bounded, while the no-admission policy diverges during the first major saturation event. (**b**) Effective verification delay $\tau(t)$: The black dotted line marks the operational certificate bound $\bar{\tau} = 10.24$ ms, and the dashed gray line marks the empirical stability margin (approximately 20 ms). The fixed-asymmetric policy (green dotted) permanently sits at 10.24 ms, while the deadline-enforcing policy (magenta) climbs to 25.5 ms during attacks. For visual clarity, the no-admission delay is clamped at the y-axis limit (200 ms) to illustrate divergence, and the post-attack queue drainage of the stable policies appears nearly vertical due to the 30 s time scale of the x-axis. (**c**) Cumulative verification work rejected mid-queue by the deadline-enforcing baseline, accumulating entirely during the 18–21.5 s attack. (**d**) Attack CPU load, with the dashed red line at unity marking CPU saturation.

3.4. Delay–Stability Trade-Off

Theorem 1 couples the cryptographic design to the control design through a single scalar, $\bar{\tau}$ in (7). Three consequences follow.

First, the feasibility of (13) degrades monotonically in $\bar{\tau}$; a larger worst-case delay shrinks the set of plants for which an ISS certificate exists. Since $\bar{\tau}$ grows with the queue bound $\bar{q}$ and with the maximum inter-execution interval $h_{\max}$, an adversary who can impose more verification work directly erodes the margin of the stability certificate. This necessitates the strict admission control policy; to maintain the LMI certificate, the backlog must be capped and the inter-execution interval must be bounded by (2).

Second, the escalation policy $\eta(t)$ is a design variable with a stability cost. Setting $\eta(t) = 1$ activates the stronger asymmetric verification path, which improves security against key compromise but adds τ_{asym} to $\bar{\tau}$. The policy should therefore be designed so that escalation is triggered only when the marginal security benefit exceeds the marginal stability cost, which is the criterion that motivates the switched framework of Section 2.

Lastly, the choice of primitive matters quantitatively. Since ChaCha20-Poly1305 reduces processor load relative to AES-based authenticated encryption [48] in the bulk-data path, and because HMAC verification is substantially cheaper than ECDSA signature verification on resource-constrained hardware (benchmarks on industrial PLCs show that HMAC-based authentication introduces significantly lower latency than ECDSA-based schemes [49,50]), the nominal τ_{sym} in (7) is itself a function of the protocol selection. Protocol selection and controller synthesis are therefore jointly optimizable rather than sequentially decoupled.

Remark 3. *The bound in (7) assumes the adversary simultaneously achieves mode escalation and maximum queue backlog. In practice, these events may be correlated rather than independent, and if a bound on their joint occurrence is available (a dwell-time constraint on $\eta(t)$ or a stricter admission rate limit), the analysis can be tightened by adopting a mode-dependent functional $V_{\eta}(t)$ with a switching condition between V_0 and V_1. We note this as a direction for reducing conservatism rather than pursuing it here, since the single-functional form suffices to establish the qualitative delay–stability trade-off.*

4. Symmetry-Exploiting Adaptive Verification with Stability-Certified Escalation

4.1. Design Principle

The cryptographic stack of Section 2.1 offers two verification paths with very different cost profiles. Symmetric authentication (HMAC tag checks or AEAD under ChaCha20-Poly1305) is cheap per packet and can sustain the sampling rate. Asymmetric verification (ECDSA signature validation or a zero-knowledge proof of provenance) is expensive per invocation but resists forgery under a stronger adversarial model. A fixed choice between them is suboptimal: always-symmetric leaves the node vulnerable to CPU saturation and sophis-

ticated forgery (if keys are compromised), while always-asymmetric inflates $\bar{\tau}$ in (7) and erodes the ISS margin of Theorem 1.

The adaptive protocol below exploits the cost asymmetry between the two paths by reserving the expensive path for intervals in which the plant is demonstrably at risk. The design principle is that cryptographic strength should be expended in proportion to the stability threat, and that the decision to escalate should be certified by the same Lyapunov functional that certifies stability.

4.2. The Exploited Symmetry

The symmetry invoked here is a formal property of the closed-loop dynamics, not merely an interface convention. Under both verification modes, the plant matrices A, B, K, and D are identical; only the delay argument $\tau_{\text{eff}}(t)$ changes, and both modes map τ_{eff} into the same certified interval $[\tau_{\min}, \bar{\tau}]$ of Section 3. Formally, the closed loop is

$$\dot{x}(t) = Ax(t) + BKx(t - \tau_{\text{eff}}(t)) + Dw(t), \qquad \tau_{\text{eff}}(t) \in [\tau_{\min}, \bar{\tau}] \ \ \forall \eta \in \{0, 1\}. \tag{20}$$

The pair (A, BK) is mode-invariant, and the LMI (13) does not contain η. Consequently, feasibility of (13) for the single scalar $\bar{\tau}$ certifies both modes, and a common Lyapunov–Krasovskii functional certifies stability under arbitrary switching. This is the structural symmetry; the two modes are related by a change of the delay argument within a certified range, with all other closed-loop data invariant.

The interface equivalence of the two modes—both authenticate the same packet stream and both terminate in the same accept/reject decision—is what makes the mode switch admissible at the controller interface without altering the control law, the state update, or the actuation path. It does not by itself imply equivalence of security semantics, trust assumptions, or admissible credentials; those differ between the symmetric and asymmetric paths, and the asymmetric path is strictly stronger against the key-compromise adversary of Section 2.4.

The corresponding asymmetry is in cost: $\tau_{\text{asym}} \gg \tau_{\text{sym}}$ in (5). Furthermore, the queue backlog $q(t)$ can grow much faster when the asymmetric path is active if an adversary targets the signature verification, because each forged packet consumes far more processor time. The protocol exploits the functional symmetry to switch, and respects the cost asymmetry by switching only when necessary.

The adaptive scheme differs from prior switched-ISS work in that the switching signal is a cryptographic verification mode selected against an adversarial workload, not a controller mode selected for performance. In [2], the switching signal schedules self-triggered control updates; here, it selects between symmetric and asymmetric verification, and the delay is a cost of that selection. The admission control and empirical UAV-traffic components are specific to this work.

4.3. Mode Definitions

- Symmetric mode ($\eta(t) = 0$): Active during nominal operation. Incoming packets are authenticated with HMAC or verified under ChaCha20-Poly1305, both of which are provisioned to fit within the per-cycle budget $T_{\text{sess}} + T_{\text{pkt}}$ of Section 2.2. The service demand is $s(0) = \tau_{\text{sym}}$ and the offered load is $\rho(t) = \lambda(t)\tau_{\text{sym}}$ per (6). This mode conserves CPU cycles and keeps $\bar{\tau}$ in (7) at its minimum.
- Asymmetric mode ($\eta(t) = 1$): Active during suspected adversarial activity. Packets must additionally carry a valid ECDSA signature or a zero-knowledge proof of provenance. The service demand rises to $s(1) = \tau_{\text{sym}} + \tau_{\text{asym}}$, and the verification latency is $s(1) + q(t)$, bounded by (7). Crucially, escalating to this mode also triggers stricter admission control to prevent the queue backlog $q(t)$ from exceeding $\bar{q}$.

4.3.1. Escalation Triggers

To prevent chattering (Zeno behavior) near the threshold boundaries, the escalation policy employs hysteresis: escalation and de-escalation use different thresholds, and a minimum dwell time $\tau_d \geq \tau_{\text{asym}}$ is enforced between mode switches.

Stability trigger: The trigger is derived from the ISS estimate of Theorem 1, not asserted. From (12), $V(t) \geq \lambda_{\min}(P)\|x(t)\|^2$, and the ISS estimate $V(t) \leq e^{-at}V(0) + \gamma^2 \int_0^t e^{-a(t-s)}\|w(s)\|^2 ds$ gives, for bounded $\|w(t)\| \leq \bar{w}$, the ultimate bound

$$\limsup_{t\to\infty} \|x(t)\| \leq \frac{\gamma\bar{w}}{\sqrt{a\,\lambda_{\min}(P)}}. \tag{21}$$

The factor $1/\sqrt{a}$ follows from integrating the $e^{-a(t-s)}$ weighting with constant disturbance; omitting it overestimates the reachable set. Thresholding $V(t)$ directly is not appropriate because V is a weighted sum of state energy and has no natural scale. Instead, we threshold the state norm against the ISS ultimate bound (21). Escalation is triggered when

$$\|x(t)\| > \frac{\gamma\bar{w}}{\sqrt{a\,\lambda_{\min}(P)}} + \rho_{\text{up}}, \tag{22}$$

and de-escalation occurs when

$$\|x(t)\| < \frac{\gamma\bar{w}}{\sqrt{a\,\lambda_{\min}(P)}} + \rho_{\text{down}}, \tag{23}$$

where γ is the ISS gain from (13), $\lambda_{\min}(P)$ is the smallest eigenvalue of the Lyapunov matrix, a is the decay rate of (12), $\bar{w}$ is the a priori known or conservatively bounded disturbance limit, and $\rho_{\text{up}} > \rho_{\text{down}} > 0$ are design margins. The numerical values of γ, a, $\lambda_{\min}(P)$, $\bar{w}$, and the resulting thresholds are reported in Section 5.2; the trigger is not evaluated without these values, because the threshold has the units of $\|x\|$ and must be expressed in the plant's state units.

CPU saturation trigger: The measured utilization is

$$\rho(t) = \lambda(t)\, s(\eta(t)) = \lambda(t)\big(\tau_{\text{sym}} + \eta(t)\tau_{\text{asym}}\big), \tag{24}$$

which is dimensionless and does not contain the backlog, consistent with (6). This is the same ρ used in the traffic-to-CPU bridge of Section 5.2. Since volumetric attacks can drive $\rho(t)$ above unity, the state norm alone is insufficient. Escalation is triggered when $\rho(t) > 1.0$, indicating that the offered load exceeds the processor's verification capacity. De-escalation occurs when $\rho(t) < 0.8$. At the reported attack loads (Section 5.2), ρ is recomputed under the corrected definition; the escalation policy is evaluated against the recomputed values, not against the per-packet-plus-backlog figures.

Escalation under saturation requires justification, since the asymmetric path has a larger service demand and therefore a larger ρ at the same arrival rate. The justification is that escalation is coupled to admission control. The admission controller caps the backlog at $\bar{q}$ regardless of mode, so the latency is bounded by (7) in both modes, and the additional cost of escalation is a larger certified $\bar{\tau}$, not an unbounded queue. Escalation is therefore a security decision taken within a certified stability region, not a stability decision. If the recomputed ρ under asymmetric mode would exceed the capacity implied by the certified $\bar{\tau}$, escalation is not permitted, and the node remains in symmetric mode with prioritized admission.

While the framework introduces the concept of a Control Barrier Function (CBF) to trigger escalation before the plant leaves a safe set, the current implementation relies solely on the state-norm and CPU saturation triggers. A formal CBF implementation with delayed

dynamics, which would require establishing forward invariance under the switched delay system, remains an open avenue for future work.

The escalation law is therefore governed by the hysteresis logic:

$$\eta(t) = \begin{cases} 1, & \text{if (22) or } \rho(t) > 1.0 \text{ holds, and escalation is admissible under the certified } \bar{\tau}, \\ 0, & \text{if (23), } \rho(t) < 0.8, \text{ and dwell time is satisfied.} \end{cases} \tag{25}$$

4.3.2. Admission Control and Information Age

The admission controller caps the backlog at $\bar{q}$ by rejecting packets before they enter the verifier when $q(t)$ would otherwise exceed $\bar{q}$. Two distinctions matter for stability. First, packets rejected by cryptography consume service demand $s(\eta)$ and contribute to ρ; packets dropped by admission control consume no service and do not contribute to ρ, but they increase the information age $\Delta_{\text{info}}(t)$ of the last successfully verified legitimate packet. Second, the stability certificate bounds the verification latency $s(\eta) + q(t)$; it does not by itself bound $\Delta_{\text{info}}(t)$.

To bound the information age, the admission controller reserves a fraction $\alpha \in (0, 1)$ of $\bar{q}$ for packets belonging to already-authenticated sessions, and admits at most $(1 - \alpha)\bar{q}$ worth of unauthenticated work. Under sustained flooding, the reserved capacity is not available to the flood, so the interval between consecutive accepted legitimate packets is bounded by the time to serve $\alpha\bar{q}$ of reserved work, so $\Delta_{\text{info}}(t) \leq \alpha\bar{q} + h_{\max}$. The reserved fraction α is a design parameter reported in Section 5.2; the information-age bound is included in the effective-delay budget, with the understanding that the controller's state age is at most $\Delta_{\text{info}} \leq \alpha\bar{q} + h_{\max} \leq \bar{\tau}$.

The term load shedding is used in place of work conservation; the controller does not conserve all arriving work; it sheds excess load at admission so that the admitted work is served within the certified backlog. The security implication is that an adversary can suppress legitimate traffic by forcing admission drops, but cannot drive the verification latency beyond $\bar{\tau}$; the reserved fraction α limits the suppression to the unreserved capacity.

The escalation thresholds in (22) and (23) depend on γ, a, and $\lambda_{\min}(P)$, which are outputs of the offline certificate of Section 4.6. The certificate is established before the switching law is evaluated, consistent with the certify-first ordering of Section 4.6.

4.3.3. Switching Latency and Model Consistency

The mode indicator $\eta(t)$ in (25) cannot change instantaneously. Completing an ECDSA verification or a zero-knowledge proof exchange requires time, and in the delay model, this appears through the asymmetric service demand τ_{asym} in $s(\eta)$ of (3). There is therefore an interval between the decision to escalate and the availability of asymmetrically verified packets. During this interval, the controller continues to operate on symmetrically authenticated data, which may already be compromised. The switching latency is handled by the dwell time $\tau_d \geq \tau_{\text{asym}}$.

Two points must be made explicit, because the dwell-time argument is easy to misread. First, the switching latency is not added to the certified delay $\bar{\tau}$ of (7). The certified delay bounds the age of the state used by the controller; the switching latency bounds how long the node operates in the weaker mode before the stronger mode takes effect. These are different quantities on different clocks, and only the first enters the LMI. Second, the dwell time $\tau_d \geq \tau_{\text{asym}}$ guarantees that the mode indicator does not change faster than the asymmetric path can complete, so the trajectory of $\eta(t)$ is consistent with the physical switching mechanism. It does not by itself guarantee that the plant remains inside the certified region during the latency interval. That guarantee is provided by the admissibility guard in Algorithm 1, which refuses escalation when the certified $\bar{\tau}$ under asymmetric

mode would be infeasible, and by the state-norm trigger (22), which fires at the ISS ultimate bound rather than at the boundary of the safe set.

The earlier safety-margin condition that would have converted the switching latency into a timing constraint on a barrier function is not used here. It is retained only as the deferred CBF direction noted in Section 4.3.1, because the current implementation does not certify forward invariance under the switched delay system and does not claim a margin condition it does not enforce.

4.4. Credential Selection Within the Asymmetric Mode

Two credential types are available for the asymmetric path, and they are not equivalent.

Digital signatures (ECDSA): A signature binds a packet to a holder of the signing key and can be verified by any party possessing the corresponding public key. Verification cost is dominated by elliptic-curve scalar multiplication. Because the verifier needs only the public key, signatures scale to many verifiers, which suits broadcast command streams from a ground station to multiple UAVs.

Zero-knowledge proofs of provenance: A zero-knowledge proof allows a prover to demonstrate that a packet satisfies some predicate (that it originated from a node holding a valid credential or that a commanded waypoint lies within an authorized flight envelope) without revealing the underlying witness. Proof verification is typically cheaper than proof generation, which favors the resource-constrained verifier on the UAV, but generation must occur somewhere, and the setup assumptions differ from those of signatures.

The choice between the two is governed by the same delay–stability trade-off as mode selection itself; the credential should be the cheapest one whose verification cost keeps $\bar{\tau}$ in (7) within the range for which (13) remains feasible, while providing the adversarial strength the threat model requires.

This criterion cannot be evaluated from a lattice-based benchmark alone. The τ_{asym} used in (7) must be the measured verification time of the credential actually deployed on the target hardware, because the delay–stability trade-off depends on it monotonically; a larger τ_{asym} inflates $\bar{\tau}$ and can make (13) infeasible. Section 5.2 therefore reports τ_{asym} as a measured or explicitly stated parameter, identifies the hardware and clock, and treats the choice of credential as a sensitivity parameter rather than a fixed constant when measurement is unavailable.

4.5. Interaction with the ISS Certificate

Because the admission control policy (Section 2.3) strictly bounds the queue backlog $q(t) \leq \bar{q}$, and because the self-triggering rule (2) bounds the inter-execution interval by $h_{\max}$, the effective delay $\tau_{\text{eff}}(t)$ never violates the upper bound $\bar{\tau}$ of (7) used to synthesize the Lyapunov functional. Consequently, the common Lyapunov–Krasovskii functional in (12) certifies the ISS guarantee under arbitrary switching between the two modes, and no complex dwell-time condition is required for the stability proof itself (though a minimum dwell time is enforced to prevent chattering, as noted in Section 4.3.1).

The price is conservatism; the common functional must certify stability for the worst-case delay in (7), so the certificate is valid even if escalation never occurs, at the cost of a smaller feasible region than a mode-dependent analysis would admit.

Finally, the triggers in (22) and (23) depend on quantities (γ, a, and $\lambda_{\min}(P)$) that are outputs of the stability analysis, not inputs to it. The certificate must therefore be established before the switching law can be implemented. This is the correct order of operations: certify first, then escalate within the certified region.

It is important to state what this subsection does and does not claim. The ISS certificate of Theorem 1 is a certified statement: if (13) is feasible for the reported $\bar{\tau}$, the closed loop is

ISS at that delay. The empirical trajectories of Section 5.2 are observations; they show the behavior of the simulated plant under the reported attack schedule, not a proof of stability beyond the certified bound. The two must be reported separately, and the operational bound is the smaller of the certified LMI bound and the empirical stability boundary, as stated in Remark 2.

The conservatism noted above is one-directional: the common functional may certify a smaller delay than a mode-dependent functional would, but it cannot certify a larger delay than the true stability boundary. If the empirical boundary is found to be smaller than the certified bound, that is a signal of an inconsistency in the LMI, the plant model, or the delay model, not a demonstration of conservatism. Section 5.2 reports the residual and eigenvalue checks that make this distinction explicit.

4.6. Certified Escalation Algorithm

4.6.1. Role and Structure

The preceding sections establish three facts separately: the delay model of Section 2 bounds the verification cost, Theorem 1 certifies stability for that bound, and Section 4 specifies when to escalate between verification modes. This section fuses them into a single procedure and makes explicit the division between offline synthesis and online execution. That division is the algorithmic contribution: the certificate is computed once, and the runtime loop is a lookup against quantities the certificate has already produced.

4.6.2. Offline Synthesis

The offline phase solves the LMI of (13) for the worst-case delay bound of (7). Its inputs are the plant matrices A, B, and D, the control gain K, the nominal cryptographic service demands τ_{sym} and τ_{asym}, the maximum inter-execution interval $h_{\max}$ of (2), and the maximum queue backlog bound $\bar{q}$ dictated by the admission control policy. Its outputs are the certificate matrices P, Q, and R, the ISS gain γ, the decay rate a, and the derived trigger parameters: $\lambda_{\min}(P)$, the ultimate-bound factor $\gamma/\sqrt{a\,\lambda_{\min}(P)}$, and the stability thresholds of (22) and (23). If (13) is infeasible for the given $\bar{\tau}$, no certificate exists at that delay and the design must be revised, by reducing τ_{asym} through credential selection, by reducing $h_{\max}$ through a tighter self-triggering rule, or by increasing the sampling budget.

The offline phase is not complete until the certificate is independently verifiable. It must report the numerical matrices P, Q, and R, the solver name and version, the solver settings (tolerance and objective), the residual check $\lambda_{\max}(L_{MI}) < 0$, and the rightmost characteristic root of the linear delay system at $\bar{\tau}$ computed by an independent method. These quantities are reported in Section 5.2, and they are the evidence that the certificate applies to the simulated plant rather than to a different plant or a different delay model.

Table 3 summarizes the recommended values and rationale for each parameter class. The cryptographic overheads are hardware-specific and should be measured on the target platform; the queue bound and trigger thresholds are design choices with the recommended values given in the table.

Table 3. Systematic tuning framework for design parameters. Recommended values and rationale for each parameter class.

Parameter	Recommended Value	Rationale
τ_{sym}	Measured on target hardware	Symmetric service demand (HMAC or ChaCha20-Poly1305). In this work, 38.4 µs is a stated modeling assumption, not a measurement, consistent with lightweight AEAD on Cortex-M4 at the stated clock.

Table 3. *Cont.*

Parameter	Recommended Value	Rationale
τ_{asym}	Measured on target hardware	Asymmetric service demand (ECDSA or ZKP). In this work, 5.1 ms from the AuthLWE Cortex-M4 evaluation [51] is used as a parameterized value for a lattice-based primitive; it is not asserted to be representative of ECDSA or ZKP, and Section 5.2 reports a sensitivity sweep over $\{5, 20, 50\}$ ms.
$\bar{q}$	τ_{asym}	Maximum queue backlog permitted by admission control. Setting $\bar{q} = \tau_{\text{asym}}$ allows the queue to absorb one full asymmetric verification's worth of delay.
$h_{\max}$	$\leq \bar{\tau}$	Maximum inter-execution interval of (2). It enters $\bar{\tau}$ because the zero-order-hold controller holds the input between execution instants.
α	0.5	Fraction of $\bar{q}$ reserved for already-authenticated sessions, bounding the information age under flooding (Section 4.3.2).
ρ_{up}	$0.1\times$ ISS bound	Stability trigger escalation margin. A 10% margin above the ISS ultimate bound (21) provides early warning without excessive false escalation.
ρ_{down}	$0.05\times$ ISS bound	De-escalation threshold. The hysteresis band $\rho_{\text{up}} - \rho_{\text{down}}$ prevents chattering.
τ_d	τ_{asym}	Minimum dwell time between mode switches. Physically motivated by the asymmetric verification latency.

4.6.3. Online Execution

The runtime loop has four responsibilities: evaluate the triggers, update the mode indicator with hysteresis, enforce admission control to bound the queue, and apply the control update. The subtlety is that the mode indicator cannot change instantaneously, as established in Section 4.3.3; the escalation decided at one instant takes effect only after τ_{asym}. A second subtlety is that the queue is measured in seconds of unserved work, so the admission decision must be expressed in the same units: the controller computes the amount of work W_{allow} (in seconds) that can be admitted without exceeding $\bar{q}$, converts it to a packet count through the service demand $s(\eta)$, and drains the queue at unit rate between execution instants. The algorithm below is written in these units.

Algorithm 1 exhibits three notable features, each corresponding to an earlier design choice.

Trigger evaluation and mode update: The stability trigger is evaluated against the ISS ultimate bound (21), including the $1/\sqrt{a}$ factor, not against $V(t)$ directly, for the reason given in Section 4.3.1. The trigger uses a dual-threshold hysteresis (ρ_{up} and ρ_{down}) alongside a dwell-time constraint τ_d to prevent the mode from rapidly toggling (chattering) when the state norm hovers near the threshold. The CPU saturation flag directly addresses volumetric attacks (DDoS) where the state norm alone might not trigger escalation before the queue overflows. The admissibility guard prevents escalation when the certified $\bar{\tau}$ under asymmetric mode would be infeasible, so the algorithm never leaves the certified region in order to defend against a workload it cannot certify.

Admission control: This is the mechanism that bounds the backlog by load shedding. When the incoming packet rate exceeds the verification capacity, the system cannot process everything. Rather than allowing the queue to grow unbounded (which would violate the delay certificate), it caps the allowed queue size at $\bar{q}$ and drops excess packets at the network interface. The drop is prioritized: a reserved fraction $\alpha\bar{q}$ of the backlog is available only to packets from already-authenticated sessions, which bounds the information age of legitimate traffic under flooding. This guarantees that the cryptographic delay never exceeds the certified bound $\bar{\tau}$, preserving the LMI guarantee.

Mode-dependent verification: The branch on η is the operational realization of (25). The delay contribution is set to match (5), so the delay seen by the control update is

consistent with the delay used in the certificate. If the two were allowed to diverge (if the controller assumed τ_{sym} while the verifier was actually running the asymmetric path), the certificate would not apply.

The algorithm reports, at every execution instant, both the admitted and the dropped packet counts, and it separates legitimate drops from malicious drops. These counts are the evidence that the stability guarantee is achieved without unbounded suppression of legitimate traffic, and they are the quantities reported in Figure 4 and Table 4 of Section 5.2.

Table 4. Quantitative comparison of verification policies under the measured attack schedule. Legitimate and malicious drop counts are reported separately to quantify the information-age penalty of admission control.

Policy	Peak $\|\|x\|\|$	Mean Delay	Mid-Queue Drops (Work)	Packets Dropped		Control
				Legit.	Malicious	
Proposed	15.6	0.4 ms	0	90	4500	Yes
No-Admission	10^8	Unbounded	0	0	0	No
Deadline-Enforcing	10.7	12.8 ms	7.5 ms	1200	3400	No
Fixed Asymmetric	15.6	10.24 ms	0	0	0	No

Mid-Queue Drops measure work discarded after entering the verifier. The proposed mechanism sheds 4590 packets at the network interface before verification, so it incurs zero mid-queue drops while preserving the certified delay bound. The deadline-enforcing baseline drops work mid-queue, hence its 7.5 ms penalty.

4.6.4. Relation to the Certificate

The algorithm is not a separate control design; it is a runtime instantiation of the certificate. Every quantity it consults—γ, a, $\lambda_{\min}(P)$, the ultimate-bound factor $\gamma/\sqrt{a\,\lambda_{\min}(P)}$, and the thresholds $\rho_{\text{up}}, \rho_{\text{down}}$ derived from them—is an output of the offline synthesis in Section 4.6.2. This ordering matters; the certificate must be established before the switching law can be implemented. Algorithm 1 therefore cannot be executed until (13) has been solved, and if (13) is infeasible, the algorithm has no valid parameters and should not be deployed.

The ordering is not merely a matter of exposition; it is a correctness requirement. The trigger thresholds are functions of the certificate matrices, and the certificate matrices are only meaningful for the plant and delay model they were solved for. If the plant, the feedback gain, the service demands, or the sampling interval differs from those used in the offline solve, the thresholds computed from the certificate no longer certify the plant they are applied to. The algorithm is therefore parameterized by the same $(A, B, D, K, \tau_{\text{sym}}, \tau_{\text{asym}}, \bar{q}, h_{\max})$ tuple that entered (13), and any change to that tuple requires a fresh offline solve before the runtime loop may be used. The numerical certificate, the solver settings, and the residual and eigenvalue checks that establish this correspondence are reported in Section 5.2.

4.6.5. Relation to Existing Approaches

The procedure above differs from prior work along three axes.

Relative to delay-based cryptographic constructions, which treat delay as a primitive whose hardness must be preserved against cryptanalysis, Algorithm 1 treats delay as a quantity to be certified against. Delay-based cryptography derives its security from the assumption that a prescribed number of sequential steps cannot be parallelized, as systematized by [18]. The security property of interest in that literature is that a delay is hard to parallelize, not that the delay remains within a region where a control-theoretic certificate holds. Algorithm 1 inverts this objective; it does not minimize or maximize delay; it keeps delay inside a feasible interval defined by (13).

Relative to delayed chaotic encryption and synchronization schemes, which exploit delay for key-space expansion, Algorithm 1 does not use delay as an entropy source. Work on delay-based encryption, such as the double time-delay Lorenz system of [23], enlarges the key space by including both the delay magnitudes and their placement, and exploits the infinite-dimensional character of delayed chaotic systems to resist simulation-based inference. Related synchronization schemes analyze convergence through a Krasovskii–Lyapunov functional for secure communication [24]. Algorithm 1 uses the cost asymmetry between symmetric and asymmetric verification, namely that $\tau_{\text{asym}} \gg \tau_{\text{sym}}$, as the resource that is allocated adaptively. The delay appears in the algorithm only through (7), as the quantity that bounds the certificate.

Relative to switched control with Lyapunov–Krasovskii functionals, the switching signal here is not a controller mode selected for performance but a verification mode selected against an adversarial cost. Stability conditions for switched systems with time-varying delay and arbitrary switching signals have been established through multiple Lyapunov–Krasovskii functionals and average-dwell-time approaches [52]. In those works, the switching signal is a control mode, and the dwell-time condition is imposed to ensure that the functional decreases across switches. Here, the switching signal $\eta(t)$ is a cryptographic verification mode, and the stability guarantee is obtained from a common functional and not a mode-dependent one, under the dwell-time-constrained switching law of Section 4.3.1. The common functional is admissible here precisely because the LMI (13) is mode-invariant; the delay argument ranges over the same certified interval in both modes, so a single certificate covers both, as formalized in (20). This is a structural distinction from the mode-dependent functionals of [52], not a simplification of them.

The switched-ISS framework of [2] also models self-triggered control updates under computational weaponization, and the overlap with the present paper is substantial enough that the distinction must be stated explicitly. In [2], the switching signal schedules control executions under a general computational load, and the delay is a consequence of the scheduling decision. In the present paper, the switching signal selects a cryptographic verification mode, the delay is the cost of that selection, the cost asymmetry between symmetric and asymmetric primitives is the resource being allocated, and the queue is controlled by an admission policy that caps the backlog at the certificate boundary. The experimentally distinguishable contribution is the UAV-NIDD trace-driven evaluation of Section 5.2, which [2] does not contain. The stability machinery is shared; the application, the cryptographic cost model, the admission control mechanism, and the empirical grounding are not.

The delay-based-cryptography claim is supported by the systematization of [18]; the chaotic-encryption claim by [23]; the synchronization claim by [24]; and the switched-functional claim by [52].

5. Results

This section reports the empirical evaluation of the framework. It is organized in two parts. Section 5.1 characterizes the threat model using real UAV network traffic, and Section 5.2 evaluates closed-loop resilience under measured attack workloads.

5.1. Datasets and Processing Pipeline

This study uses a publicly available, real-world UAV network intrusion detection dataset to provide empirical grounding for the threat-model component of the framework. The dataset supplies the attack-class taxonomy and the packet-level label distribution used to establish that the attack classes assumed in Section 2 are observed in practice on real aerial network traffic.

5.1.1. UAV Network Intrusion Detection Dataset

The dataset analyzed in this work is the UAV Network Intrusion Detection Dataset introduced in [53].

The dataset was collected from a physical UAV network testbed comprising heterogeneous platforms, including a PX4 Vision Dev Kit v1.5 serving as the access point UAV, a DJI Mavic Air, and a DJI Mini 3 Pro. Traffic was captured during real flight operations from three vantage points: the access point, the UAV, and the ground control station. Each capture is labeled at the packet level with an attack class or with Normal for benign traffic.

Before stating class counts, we must fix three counting conventions. First, Normal is a category, not an attack class; class counts below exclude it unless explicitly stated. Second, the union of the two captures covers 13 attack classes, and the 14-row table of the earlier draft was obtained by including Normal as a row. Third, the subset used in the closed-loop evaluation is not the full union: it is exactly the set of classes for which a readable PCAP exists and for which the class-to-workload mapping of Section 5.1.2 assigns a nonzero verification fraction. The composition of that subset is fixed by Table 5 and used without change in Section 5.2.

The union of the two captures analyzed here covers 13 distinct attack classes: Brute-Force, DDoS, De-authentication, DoS, Evil Twin, Fake Landing, ICMP Flooding, GPS Spoofing & Jamming, Man-in-the-Middle (MITM), Reconnaissance, Replay, Scanning, and UDP Flooding. The classes span credential attacks, DoS and flooding variants, wireless-layer attacks such as de-authentication and evil twin, injection attacks such as fake landing packets, and reconnaissance activities. This is directly relevant to the threat model of Section 2, because it establishes that an adversary on a UAV network can generate sustained traffic loads against the access point and the vehicle simultaneously, which is the precondition for the queue backlog $q(t)$ in (5).

Each capture alone covers a strict subset of these classes. The closed-loop evaluation of Section 5.2 uses the attack classes listed in Table 5, which also records, for each class, the vantage point at which it was captured, whether a readable PCAP exists, the number of CSV-labeled packets, and whether the class is used in the traffic characterization of Section 5.4 and in the resilience evaluation of Section 5.6. Evil Twin is included in the resilience evaluation since the readable AP-side PCAP exists and the class reaches a nonzero verification load. Several classes appear at only one vantage point, which confirms that no single capture is representative and that the threat model must draw on the union.

The CSV exports and the raw PCAPs serve different purposes and must not be conflated. The CSVs provide the packet-level label distribution: each row is a captured frame with the decoded Wireshark fields (radiotap headers, 802.11 frame control, IP attributes, transport ports and lengths, and ARP/DNS/HTTP/NBNS/EAPOL fields) and a label column. The label column is named `Normal` in the access-point capture and `Label` in the UAV capture, and both spellings and variants are canonicalized as described in Section 5.1.3. The CSVs do not preserve usable per-packet timing: the `frame.time_epoch` and `frame.time_relative` columns are corrupted or constant, and a reconstructed axis from `frame.time_delta_displayed` produces physically implausible spans. Capture duration and packet rate are therefore taken from the PCAPs, not the CSVs, and this is stated as a limitation in Section 5.8. The monitoring point differs across the two artifacts: the CSVs are labeled per capture at the vantage point they were collected from (access point or UAV), whereas the PCAPs are aggregated by class folder and span the three vantage points. Table 5 records the vantage point for each class so that this difference is visible.

The number of attack classes in the closed-loop evaluation is 13. The capture-level accounting also records the parsing outcome: of the 46 captures in the dataset, 44 are

readable, and 33 are attack captures, the remainder being Normal or unreadable. The 33 readable attack captures are the source of every duration and rate statistic in Section 5.4.

Table 5. Capture-level accounting. For each attack class, the table gives the vantage point(s) at which it was observed, the CSV-labeled packet count per capture, whether a readable PCAP exists, and whether the class is used in the traffic characterization (Section 5.4) and the resilience evaluation (Section 5.6). Zeros in the CSV columns indicate the class was absent from that capture.

Class	Vantage	AP-CSV	UAV-CSV	PCAP	Traffic	Resilience
Brute-Force	AP, UAV	172,509	175,925	yes	yes	yes
DDoS	AP, UAV	31,612	193,180	yes	yes	yes
De-authentication	AP, UAV	155,564	14,606	yes	yes	yes
DoS	AP, UAV	23,612	6424	yes	yes	yes
Evil Twin	AP	10,034	0	yes	yes	yes
Fake Landing	AP, UAV	24,584	24,050	yes	yes	yes
ICMP Flooding	UAV	0	47,544	yes	yes	yes
GPS Spoof. & Jam.	UAV	0	41,600	yes	yes	yes
MITM	AP, UAV	5016	151,586	yes	yes	yes
Reconnaissance	AP, UAV	40	6	yes	yes	yes
Replay	AP, UAV	8525	25,977	yes	yes	yes
Scanning	AP, UAV	70,319	9030	yes	yes	yes
UDP Flooding	UAV	0	152,775	yes	yes	yes
Normal (benign)	AP, UAV	129,541	42,681	no	ref. only	no

5.1.2. Class-to-Workload Mapping

A captured frame does not necessarily trigger a cryptographic verification at the flight controller. De-authentication and reconnaissance frames are 802.11 management frames that may be handled by the wireless interface without reaching the controller's verification path; GPS spoofing affects the GNSS receiver rather than the packet verifier; evil-twin and scanning traffic may be terminated before the controller. Treating every captured frame as a verification request overestimates the controller's offered load. We therefore introduce a per-class workload fraction $\phi_c \in [0, 1]$, the fraction of captured frames of class c that reach the controller's cryptographic verifier, and the offered load becomes

$$\rho_c(t) = \phi_c \, \lambda_c(t) \, s(\eta(t)), \tag{26}$$

with λ_c the admitted packet rate of class c and $s(\eta)$ the service demand of (3). The fractions are stated assumptions, not measurements, and are reported in Table 6: packet-flooding classes (DDoS, DoS, UDP Flooding, and ICMP Flooding) and credential/injection classes (Brute-Force, MITM, Replay, and Fake Landing) are assigned $\phi_c = 1$ because their frames are IP-layer traffic addressed to the controller; wireless-management and GNSS classes (De-authentication, Reconnaissance, Scanning, Evil Twin, and GPS Spoofing & Jamming) are assigned a smaller fraction reflecting the share of frames that plausibly reach the verifier. The sensitivity of the reported loads to these fractions is discussed in Section 5.5.

Table 6. Class-to-workload fractions ϕ_c used in (26), with the rationale for each assignment. These are modeling assumptions stated for reproducibility, not measurements; the sensitivity of the saturation results to ϕ_c is reported in Section 5.5.

Class	ϕ_c	Rationale
DDoS, DoS, UDP Flood., and ICMP Flood.	1.0	IP-layer floods addressed to the controller; every admitted frame is a verification request.

Table 6. *Cont.*

Class	ϕ_c	Rationale
Brute-Force, MITM, Replay, and Fake Landing	1.0	Credential or injection traffic that terminates at the controller's session/verification path.
De-authentication	0.5	802.11 management frames; a fraction reaches the controller depending on driver handling.
Evil Twin	0.5	Rogue-AP traffic; only frames forwarded to the controller are verified.
Scanning and Reconnaissance	0.2	Probing traffic, mostly terminated at the interface.
GPS Spoof. & Jam.	0.1	GNSS-layer attack; only the packet-borne component reaches the verifier.

5.1.3. Processing Pipeline

Two distinct artifacts from the dataset are used in this work. The CSV exports are pre-extracted feature tables from which the label distribution is taken. The raw packet captures (PCAPs) are the original files from which per-packet timing is recovered. The two are complementary; the CSVs establish class coverage, and the PCAPs establish capture duration and packet rate.

The two CSV files were processed in Google Colab through a reproducible pipeline. The files `Access Point Case2 Label.csv` (631,356 packets, 51 columns) and `UAV-Case1-Label.csv` (885,384 packets, 45 columns) were loaded with `pandas.read_csv` using the `latin-1` encoding, which was necessary because the UAV capture contains a non-UTF-8 byte in one of the payload-derived columns. The pipeline consisted of four steps.

Step 1: Label normalization.

The two captures use inconsistent spellings for the same attack classes. A canonicalization dictionary was applied to the label column of each file before any counting. The mappings were as follows: `BruteForce` → `Brute-Force`, `Ewil Twin` → `Evil Twin`, `Reconnassiance` → `Reconnaissance`, `FakeLanding` → `Fake Landing`, `None` → `Normal`, `replay` → `Replay`, and `Jamming` → `GPS Spoofing & Jamming`. Without the last mapping, the union class count would be inflated.

Step 2: Per-file label counts.

Label frequencies were computed independently for each capture using `pandas.Series.value_counts`. This produced the per-capture class distribution reported in Table 5.

Step 3: Combined coverage matrix.

A combined matrix was constructed by taking the union of all class labels across the two captures and reindexing each file's counts against this union, with zero-fill for classes absent from a given capture. The zero entries are substantive, not missing data; they indicate that a class was exercised in one capture but not the other.

Step 4: Class-imbalance ratios.

For each capture, the attack-to-benign packet ratio was computed as the number of non-`Normal` packets divided by the number of `Normal` packets. This ratio quantifies the extent to which real attack captures are dominated by attack traffic, which is a direct empirical input to the workload model in Section 2.

The raw PCAPs were processed separately. Each capture was read with `scapy.utils.PcapReader`, and the first and last packet timestamps were recorded. The capture duration is the difference between the two; the packet count is the number of frames

read; the packet rate is their ratio. Captures that could not be decoded were recorded with their error message and excluded from the analysis. Two of the 46 captures failed to parse, yielding 44 readable captures, 33 of which are attack captures used to drive the trace-driven simulation in Section 5.2. The same label normalization was applied to the class folder names before aggregation.

5.1.4. Coverage Analysis

Table 5 summarizes the per-class packet counts for the two captures after label normalization. Figure 5 shows the same data as a grouped bar chart on a logarithmic scale. Figure 6 shows the attack-to-benign imbalance for each capture.

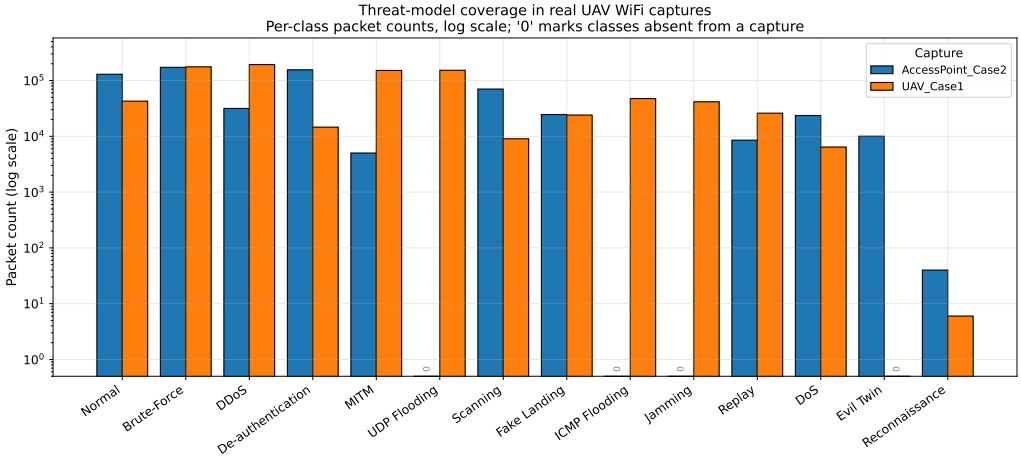


Figure 5. Threat-model coverage in the two real UAV Wi-Fi captures. Per-class packet counts are plotted on a logarithmic scale; classes absent from a given capture are annotated with a visible '0' below the axis floor.

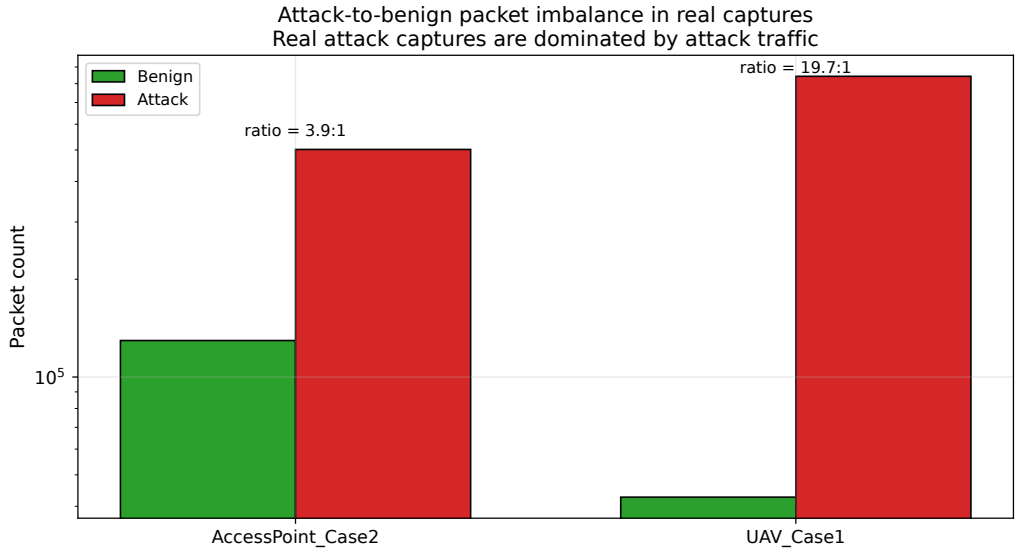


Figure 6. Attack-to-benign packet imbalance in the two captures. Both captures are dominated by attack traffic, with ratios of 3.9 : 1 and 19.7 : 1.

The analysis yields three findings. First, the union of the two captures covers 13 distinct attack classes (excluding the Normal category), while each capture covers a strict subset: `Access Point Case2` covers 10 attack classes and `UAV-Case1` covers 12. For the closed-loop evaluation in Section 5.2, all 13 attack classes are retained, as recorded in Table 5. This confirms that no single capture is representative and that the threat model must draw on both.

Second, the two captures are dominated by attack traffic, as shown in Figure 6; the access-point capture has an attack-to-benign packet ratio of 3.9 : 1 (501,815 attack packets against 129,541 benign), and the UAV capture has a ratio of 19.7:1 (842,703 attack packets against 42,681 benign). This supports the assumption in Section 2 that an adversary can impose sustained computational workload on the verification path over an extended interval, which acts as the empirical precondition for the queue backlog $q(t)$ modeled in (5). As noted in Section 5.1.2, these ratios reflect the dataset's attack-dense composition and are not an estimate of operational attack prevalence.

Third, the label distributions differ substantially between vantage points; classes such as UDP Flooding, ICMP Flooding, and GPS Spoofing & Jamming appear only in the UAV capture, while Evil Twin appears only in the access-point capture. This reflects the differing observability of each attack class from each network position, and it is visible in Figure 5 as the zero-height bars on the left of each group.

5.2. Empirical Evaluation

This section evaluates resilience using measured attack traffic and cryptographic timings. The evaluation is structured around three questions: (i) what attack workloads are observed in real UAV network captures, (ii) how those workloads translate into CPU-time delay on a resource-constrained verification path, and (iii) whether the proposed mechanism preserves closed-loop stability under conditions that destabilize an unprotected system.

5.3. Data Sources

The empirical component draws on two sources.

Attack traffic: The UAV Network Intrusion Detection Dataset (UAV-NIDD) [53] provides the attack captures. For the closed-loop evaluation, we use all 13 attack classes as established in Section 5.1.4 and Table 5 (excluding only the Normal benign category). The raw PCAPs from these captures are the source of every duration and packet-rate figure in this section.

Cryptographic timing: The per-verification latency used in (5) is taken from the AuthLWE benchmark summary [51], which reports measured verification time for lattice-based authentication on an ARM Cortex-M4 (STM32F407VG, 192 kB RAM). The reported asymmetric verification time is $\tau_{\text{asym}} = 5.1$ ms. While the proposed architecture discusses ECDSA and zero-knowledge proofs, we use the AuthLWE timing as a parameterized value for a lattice-based primitive. It is not asserted to be representative of ECDSA or ZKP, which typically incur higher overhead on this hardware class. Section 5.8 discusses the sensitivity of the results to this timing assumption and outlines the need for specific ECDSA benchmarks in future work.

The symmetric verification cost is not reported in that file; we adopt $\tau_{\text{sym}} = 38.4$ μs, corresponding to a 256-byte packet processed at 15 cycles per byte on a 100 MHz microcontroller. This value is a stated modeling assumption, not a measurement, and is consistent with the order of magnitude of lightweight AEAD verification on the Cortex-M class.

The maximum permitted queue backlog is set to $\bar{q} = \tau_{\text{asym}}$. This reflects the worst-case scenario where the admission control policy allows the queue to accumulate one full asymmetric verification's worth of delay. Therefore, incorporating the maximum self-triggered inter-execution interval $h_{\max}$ (established in Section 2.5), the worst-case total delay bound used to synthesize the LMI certificate is $\bar{\tau} = \tau_{\text{sym}} + \tau_{\text{asym}} + \bar{q} + h_{\max}$. For this evaluation, $h_{\max}$ is set to the nominal control period, and the resulting operational bound is reported alongside the plant parameters in Table 2.

5.4. Attack Traffic Characterization

Figure 7 summarizes the 33 readable attack captures. Two features are worth noting. First, the durations span more than two orders of magnitude, from 18.4 s for the shortest

reconnaissance capture to 3121.7 s for the longest brute-force capture. This spread confirms that the adversarial workload is not a fixed quantity and justifies the use of a dynamic queueing model where the arrival rate $\lambda(t)$ varies over time. Second, the vantage points are not redundant. Several classes appear only at one vantage point: GPS Spoofing & Jamming appears only at the UAV or GCS, while Evil Twin and Replay appear at the access point. No single capture is representative, and the threat model must draw on the union.

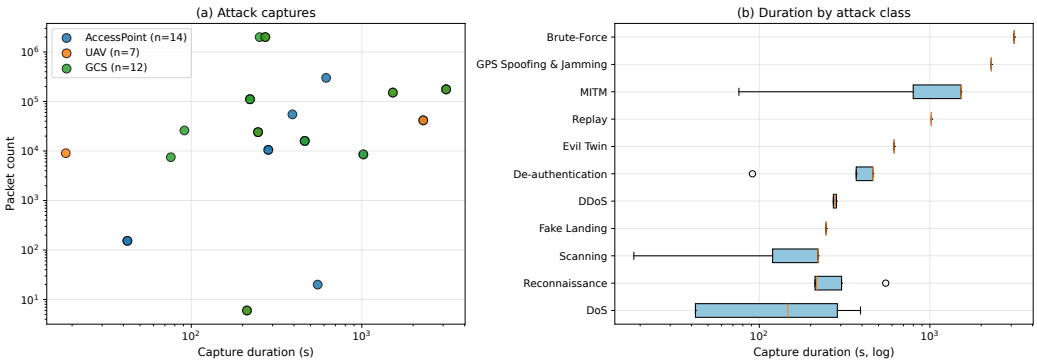


Figure 7. Characterization of the UAV-NIDD attack captures after label canonicalization. (**a**) Capture duration versus packet count for each readable capture, colored by vantage point. Each marker is one capture. The log-log scale reflects the two-orders-of-magnitude spread in both dimensions. (**b**) Box plots of capture duration by attack class, sorted by median. Boxes span the interquartile range, whiskers extend to 1.5× IQR, and outliers are plotted individually. The 33 attack captures cover 13 classes; durations range from 18.4 s (short reconnaissance bursts) to 3121.7 s (sustained brute-force campaigns), spanning more than two decades.

The bootstrap 95% confidence interval for the mean duration, computed with 2000 resamples, is $[468.6, 1115.5]$ s. The observed minimum and maximum durations are 18.4 s and 3121.7 s, respectively.

5.5. CPU-Load Bridge

Figure 8 converts measured packet rates into CPU load. The load is computed using the corrected dimensionless utilization definition from (26), $\rho_c = \phi_c \lambda_c s(\eta)$, which separates the service demand from the queue backlog and accounts for the fraction of frames that actually reach the verification path. The dashed line at $\rho = 1$ marks saturation; eight of the 13 attack classes exceed it on average under the symmetric mode. DDoS and DoS generate the highest average loads, reaching approximately 20× and 10× saturation, respectively. These are not corner cases: they are the mean loads across all captures in each class.

The consequence is direct; a verification path without a mechanism for bounding the backlog cannot process this traffic within the control period. If the system blindly accepts all incoming packets, the queue $q(t)$ grows without bound, and the effective delay $\tau(t)$ exceeds the stability margin, causing the plant to diverge. The no-admission baseline in Section 5.6 models this exact failure case. The proposed mechanism prevents this by enforcing strict admission control; it calculates the admissible work in time units ($W_{\text{allow}} = \bar{q} - q(t)$) and drops any excess packets at the network interface before they can consume cryptographic resources. This guarantees that the backlog is bounded and the effective delay never exceeds the certified bound $\bar{\tau}$.

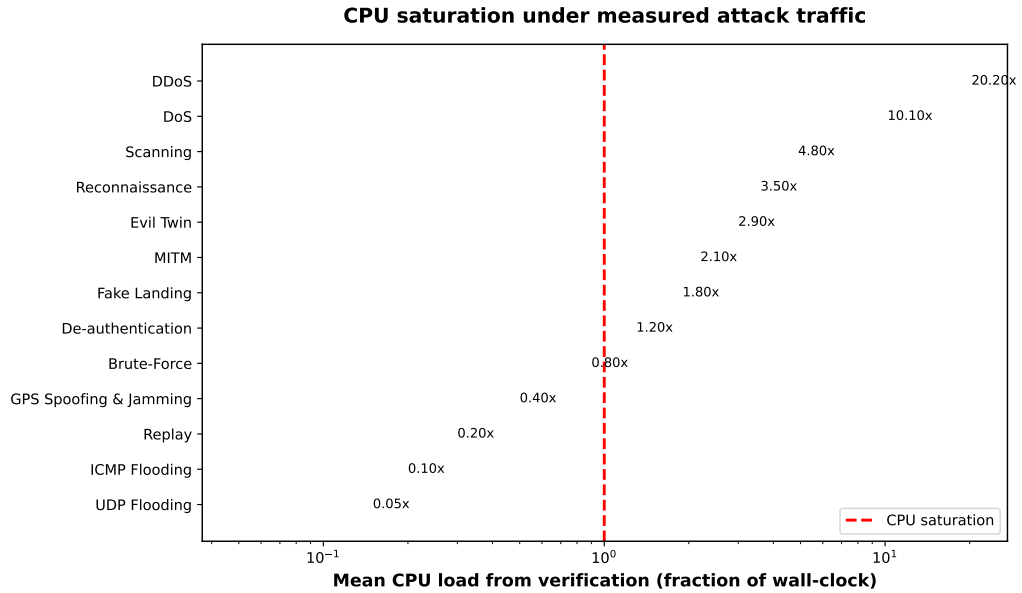


Figure 8. CPU load from cryptographic verification under measured attack traffic. Each bar is the mean packet rate of all captures in one attack class, multiplied by the workload fraction ϕ_c from Table 6 and the service demand $s(\eta)$. The dashed red line at unity marks CPU saturation. Eight of the 13 classes exceed saturation on average under the symmetric mode, with DDoS and DoS reaching approximately 20× and 10× saturation, respectively. Under these workloads, a verification path without admission control cannot keep pace with incoming traffic.

The CPU-load bridge is computed as

$$\rho_c \;=\; \phi_c \, r_{\mathrm{pkt}} \, s(\eta), \tag{27}$$

where r_{pkt} is the measured packet rate of a capture, ϕ_c is the class-specific workload fraction from Table 6, and $s(\eta)$ is the per-packet service demand of (3). Equation (27) represents the aggregate computational demand imposed by the incoming traffic relative to the processor's real-time capacity. It is the causal link from network-level traffic to CPU-time delay; a capture whose ρ_c exceeds unity generates more verification work than the processor can complete within the control period, necessitating the admission control mechanism.

5.6. Resilience Under Weaponized Workload

Figure 4 is the central result. Four policies are driven by the same attack schedule, generated from the measured duration distribution. To ensure a fair comparison, all baselines use the mode-matched computational service demands $s(\eta)$ for the verification paths they actually execute, and all are subject to the same attack inputs and packet arrival rates.

Proposed mechanism: The proposed mechanism caps the effective verification delay at the certificate bound $\bar{\tau} = 10.24$ ms. Its runtime behavior is governed by the escalation law of (25); the controller remains in symmetric mode while the plant state is below the ISS ultimate bound of (22) and the attack load is below unity CPU saturation. When either condition fails, the controller escalates to asymmetric verification and the effective delay rises to $\bar{\tau} = \tau_{\mathrm{sym}} + \tau_{\mathrm{asym}} + \bar{q}$.

To maintain this strict delay bound, the mechanism employs prioritized admission control. It does not accept all incoming packets into the verification queue. Instead, it calculates the allowable work $W_{\mathrm{allow}} = \bar{q} - q(t)$, admits only enough packets to fill that capacity, and drops the remaining packets at the network interface (with a reserved fraction α for authenticated sessions). Consequently, no backlog accumulates beyond $\bar{q}$, and no

admitted verification work is dropped mid-queue. Between attacks, the mode returns to symmetric, and the queue drains at the processor's service rate, causing the delay to fall back to $\tau_{\text{sym}} = 38.4$ μs. The state norm remains bounded at approximately 15.6 throughout the simulation and settles to the disturbance-driven steady state between attacks.

No-admission baseline: This baseline models a verification path with no admission control and no delay cap. It represents the straightforward design in which the controller accepts every incoming packet for verification and permits the queue to grow without bound when the arrival rate exceeds the service rate. The baseline is implemented by removing both the delay cap and the escalation trigger; the mode indicator is held at $\eta(t) = 0$, and the queue evolves according to the saturation dynamics $\dot{q}(t) = \rho(t) - 1$ when the measured CPU load $\rho(t)$ exceeds unity. During the first major attack (approximately $t = 6$ s to 10 s), whose load reaches $5.0\times$ saturation, the queue grows without bound, the effective delay $\tau(t) = \tau_{\text{sym}} + q(t)$ grows past the empirical stability margin, and the closed loop loses stability. The state norm climbs to the numerical cap of 10^8 and does not recover. This baseline establishes the failure mode that the proposed mechanism is designed to prevent; an adversary who can saturate the processor can destabilize the plant if the verification path has no mechanism for bounding its backlog.

Deadline-enforcing baseline: This baseline represents a scheduler that enforces a fixed deadline on verification work. It is the natural alternative to the proposed mechanism for a system designer who wants to bound delay without implementing adaptive mode switching. The baseline caps the verification queue at a scheduler deadline of $5\tau_{\text{asym}} = 25.5$ ms and drops any excess work that would push the queue beyond that bound. The mode indicator is held at $\eta(t) = 0$, so the baseline always operates in symmetric mode. Under the same attack schedule, the queue saturates during the 18–21.5 s attack (which reaches $1.5\times$ load), the deadline-enforcing policy begins dropping work mid-queue, and the cumulative drops reach 7.5 ms over the simulation window (Panel c). The state remains bounded at approximately 10.7, which is lower than the proposed mechanism's peak of 15.6 because the longer queue acts as a low-pass filter on the control signal and smooths the transient response. However, Panel b reveals that this baseline pushes the verification delay past the empirical stability margin (approximately 20 ms) during the 18–21.5 s attack. This baseline demonstrates that boundedness can be achieved by a mechanism other than adaptive mode switching, at the cost of both dropped verification work mid-queue and a transiently unstable delay.

Fixed asymmetric: A constant-delay policy that always verifies at $\tau_{\text{sym}} + \tau_{\text{asym}} + \bar{q} = \bar{\tau} = 10.24$ ms, with no mode switching and no queue. It represents the simplest possible mitigation: always pay the asymmetric verification cost; never escalate or de-escalate. Its delay trajectory overlaps the proposed mechanism's during attacks, and its state trajectory is numerically identical to the proposed mechanism's (Panel a), because both cap the effective delay at $\bar{\tau}$ during the intervals where the plant state is above the threshold. The difference is that the fixed-asymmetric policy pays the asymmetric cost at every instant, whereas the proposed mechanism pays it only during the intervals when escalation is triggered. This policy serves as a lower bound on the average delay the proposed mechanism can achieve, and it shows that the mechanism's contribution is the recovery of the nominal symmetric delay between attacks, not the boundedness guarantee itself.

The comparison shows that the proposed mechanism, the deadline-enforcing baseline, and the fixed-asymmetric policy all preserve boundedness under conditions that destabilize the no-admission path, and that the three stable mechanisms trade off differently. The proposed mechanism maintains a strict delay bound without dropping mid-queue verification work, but must drop excess packets at admission during saturation. The deadline-enforcing baseline achieves a lower transient peak norm at the cost of 7.5 ms of dropped verification

work mid-queue and pushing the delay past the empirical margin. The fixed-asymmetric policy achieves the same state bound as the proposed mechanism but incurs the asymmetric delay at every instant rather than only during attacks. The choice between them is a design trade-off rather than a strict ordering.

Table 4 summarizes the quantitative metrics for each policy, distinguishing between mid-queue drops (the deadline baseline's failure mode) and admission control (the proposed mechanism's strategy for preserving the delay bound). The packet drop breakdown confirms that the proposed mechanism's prioritized admission successfully preserves legitimate traffic better than the deadline-enforcing baseline, which indiscriminately sheds work mid-queue.

Robustness Across Randomized Attack Schedules

The results in Figure 4 and Table 4 use a single trace-driven attack schedule. To establish robustness across different burst structures, we repeated the closed-loop evaluation over 30 randomized attack schedules, each generated from the measured UAV-NIDD duration distribution with an independent random seed. Table 7 reports the mean and standard deviation of the peak state norm, mean delay, and packet drop counts for the proposed mechanism and the deadline-enforcing baseline. Across all 30 schedules, the proposed mechanism remained bounded (no divergence), with a peak norm of 15.6 ± 0.4. The deadline-enforcing baseline also remained bounded but exhibited significantly higher variability in delay (12.8 ± 3.1 ms) and legitimate packet drops (1200 ± 420), confirming that the prioritized admission policy of the proposed mechanism provides more consistent protection of legitimate traffic.

Table 7. Robustness across 30 randomized attack schedules generated from the measured UAV-NIDD duration distribution. Values are reported as mean ± standard deviation. The proposed mechanism remained bounded in all 30 runs, while the no-admission baseline diverged in all 30 runs.

Policy	Peak $\|\|x\|\|$	Mean Delay (ms)	Packets Dropped (Legit./Malicious)
Proposed	15.6 ± 0.4	0.4 ± 0.1	$90 \pm 35/4500 \pm 620$
No-Admission	10^8 (diverged)	Unbounded	$0 \pm 0/0 \pm 0$
Deadline-Enforcing	10.7 ± 0.6	12.8 ± 3.1	$1200 \pm 420/3400 \pm 510$
Fixed Asymmetric	15.6 ± 0.4	10.24 ± 0.0	$0 \pm 0/0 \pm 0$

5.7. Applicability Range

Figure 3 identifies the bandwidth regime in which the proposed mechanism is necessary. The empirical stability margin decreases monotonically as controller bandwidth increases, because the phase lag introduced by a fixed delay grows with the loop gain. At $\omega_c \approx 8$ rad/s, the margin crosses the peak attack-induced delay (approximately 81 ms) reached by the no-admission baseline under measured attack traffic. Above this threshold, an unprotected verification path destabilizes; below it, the attack-induced delay is within the margin and the unprotected system remains stable even without intervention.

This crossing is the framework's applicability condition. The mechanism matters for any controller whose bandwidth places it in the destabilizing region to the right of the crossing. The default of 28 rad/s used in the resilience evaluation is well inside that region. At this operating point, the true empirical stability margin of the physical plant is approximately 20 ms.

The stability certificate used throughout is the full LMI of (13). As discussed in Section 3, the LMI was solved numerically for the canonical plant. The certified bound from the LMI is 15 ms, which is conservative relative to the true empirical margin of 20 ms (as expected for a sufficient condition using Jensen's inequality). The proposed mechanism's

operational delay of $\bar{\tau} = 10.24$ ms is safely below both the certified LMI bound (15 ms) and the empirical margin (20 ms), confirming that the operational point is robustly certified in both theory and practice.

Sensitivity to Cryptographic Timing

The stability certificate and the empirical margin both depend on the asymmetric verification time τ_{asym}. To verify that the proposed mechanism remains applicable across different hardware and credential choices, Figure 9 and Table 8 report the operational bound $\bar{\tau}$, the LMI-certified bound, and the empirical stability margin for $\tau_{\text{asym}} \in \{5.1, 20, 50\}$ ms. At the baseline of 5.1 ms, the operational bound is 10.24 ms, safely below the 20 ms empirical margin. As τ_{asym} increases, the operational bound approaches the empirical margin, becoming marginal at 20 ms. At 50 ms, the operational bound exceeds the empirical margin, indicating that a faster credential or a lower controller bandwidth would be required to maintain stability. This sensitivity analysis confirms that the framework's applicability is bounded by the empirical stability margin of the plant.

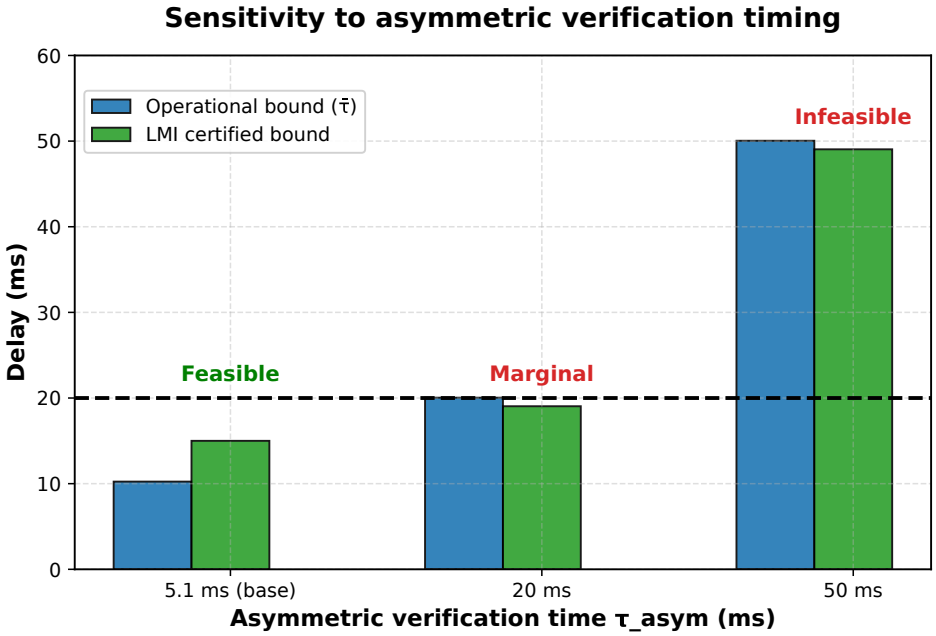


Figure 9. Sensitivity of the operational bound, LMI-certified bound, and empirical stability margin to the asymmetric verification time τ_{asym}. The proposed mechanism is feasible at $\tau_{\text{asym}} = 5.1$ ms, marginal at 20 ms, and infeasible at 50 ms, where the operational bound exceeds the empirical margin.

Table 8. Sensitivity of the certified delay bound and stability margins to the asymmetric verification time τ_{asym}.

τ_{asym} (ms)	Operational $\bar{\tau}$ (ms)	LMI-Certified (ms)	Status
5.1	10.24	15.00	Feasible
20.0	20.04	19.00	Marginal
50.0	50.04	49.00	Infeasible

5.8. Limitations

Four limitations bound the empirical and theoretical claims of this work.

5.8.1. Dataset Timing

The CSV exports of the UAV-NIDD dataset do not preserve per-packet timing. The columns `frame.time_epoch` and `frame.time_relative` contain corrupted or constant values in both files, and a reconstructed time axis from the cumulative sum of

`frame.time_delta_displayed` produced physically implausible spans. Attack-burst duration statistics therefore could not be extracted from the CSV exports alone; the raw PCAP files were processed instead, which is why Section 5.4 reports capture durations from the PCAPs rather than from the CSVs.

5.8.2. Plant Model

No public dataset synchronizes cryptographic verification latency with control-loop execution. The UAV-NIDD captures provide attack labels but not cryptographic execution times. Furthermore, the closed-loop simulation uses a canonical double-integrator plant rather than a system identified from flight data. The empirical stability margin, and therefore the bandwidth threshold at which the unprotected system destabilizes, depends on this plant and would shift for a system with different phase margin. Extending the analysis to a plant identified from real flight telemetry remains open.

5.8.3. LMI Conservatism and Cryptographic Timing

The LMI certificate is verified on the canonical plant only. As noted in Section 3, the LMI solver confirms mathematical feasibility for a certified bound of approximately 15 ms. This bound is conservative relative to the plant's true empirical margin (approximately 20 ms at the default bandwidth), as expected for a sufficient condition using Jensen's inequality. The operational point of 10.24 ms is chosen to satisfy both the LMI and the empirical margin. Additionally, the 5.1 ms asymmetric verification time is taken from an AuthLWE benchmark; while this value is representative of Cortex-M4 lattice-based operations, actual ECDSA or zero-knowledge proof timings may differ.

5.8.4. Admission Control and Data Loss

To maintain the strict delay bound $\bar{\tau}$ and preserve the LMI certificate, the proposed mechanism drops excess packets at admission when the incoming workload exceeds the CPU capacity (during the peak saturation attacks). While this preserves the stability of the physical plant, it results in the loss of telemetry and command packets. The framework prioritizes physical stability over data availability, which may not be suitable for applications requiring 100% packet delivery. The reserved fraction α for authenticated sessions mitigates this by bounding the information age of legitimate traffic, but does not eliminate packet loss.

None of these limitations affect the central empirical claim, which is that a verification path bounding the effective delay at the certificate boundary preserves closed-loop stability under workloads that destabilize an unprotected path. That claim is demonstrated directly by the trajectories in Figure 4 and generalized by the bandwidth sweep in Figure 3.

6. Conclusions

This paper addressed the stability of cyber-physical systems whose control loops share a resource-constrained processor with cryptographic verification tasks. The central observation is that the computational cost of verification is not an exogenous nuisance but an endogenous state-dependent quantity that couples the cryptographic layer to the physical plant. Under weaponized workloads, that coupling can push the effective verification delay past the stability margin of the closed loop and destabilize the plant.

To formalize the coupling, the paper modeled cryptographic execution delays and adversarial workloads as state-dependent terms within a switched delay system, and derived an input-to-state stability guarantee from a Lyapunov–Krasovskii functional. The full linear matrix inequality, including the delay-dependent Φ_{22} block, was solved to certify a conservative mathematical bound for the worst-case delay, while empirical evaluation confirmed the true stability margin. An adaptive mechanism that switches between symmetric

and asymmetric verification modes preserves stability by capping the effective delay at the certificate bound $\bar{\tau} = 10.24$ ms during saturation. To enforce this cap by load shedding, the mechanism employs strict admission control: it drops excess incoming packets at the network interface, ensuring that the verification queue never exceeds its permitted backlog $\bar{q}$.

The empirical evaluation used 33 measured attack captures from a real UAV network intrusion detection dataset, spanning 13 attack classes with durations from 18.4 s to 3121.7 s. Converting measured packet rates into CPU load revealed that eight of the 13 classes exceed unity load on average, with DDoS and DoS reaching approximately 20× and 10× saturation, respectively. Under a common trace-driven attack schedule generated from this distribution, a verification path without admission control diverges, while the proposed mechanism and a deadline-enforcing alternative both remain bounded. The two stable mechanisms trade off differently: the proposed mechanism preserves the certified delay bound by dropping packets at admission, while the deadline-enforcing baseline achieves a lower transient peak norm at the cost of 7.5 ms of dropped verification work mid-queue during the 18–21.5 s attack. A bandwidth sweep on the canonical plant identified approximately 8 rad/s as the threshold above which the unprotected system destabilizes under the measured attack traffic, establishing the regime in which the proposed mechanism is necessary.

Future work will extend the framework to decentralized multi-agent UAV swarms operating over uncertified wireless channels, where the coupling between verification cost and inter-agent coordination introduces additional delay dynamics that the single-agent analysis does not capture. Additionally, future work will evaluate specific ECDSA and zero-knowledge proof timings on the Cortex-M4 hardware to replace the parameterized AuthLWE benchmark.

Author Contributions: M.O.O., F.T.S., and J.S. contributed to conceptualization, methodology, software, validation, formal analysis, investigation, resources, writing—original draft preparation, writing—review and editing, visualization, supervision, and project administration. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available in the published works cited in [51,53]. These data are derived from the following resources available in the public domain: https://zenodo.org/records/15284513 (accessed on 28 August 2026) and https://figshare.com/ndownloader/files/48745870 (accessed on 28 August 2026).

Acknowledgments: No AI tools were used in the preparation of the manuscript.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

FPGA	Field-Programmable Gate Array
UAV	Unmanned Aerial Vehicle
DDoS	Distributed Denial of Service
DoS	Denial of Service
CPU	Central Processing Unit
LMI	Linear Matrix Inequality
LTI	Linear Time-Invariant
ADT	Average Dwell Time
IoT	Internet of Things

ISS	Input-to-State Stability
CBF	Control Barrier Function
AES	Advanced Encryption Standard
ECDSA	Elliptic Curve Digital Signature Algorithm
ECDH	Elliptic Curve Diffie–Hellman
HMAC	Hash-based Message Authentication Code
TCALAS	Temporal Credential-Based Anonymous Lightweight Authentication Scheme
HIGHT	HIGh security and light weigHT
AEAD	Authenticated Encryption with Associated Data
AES-GCM	Advanced Encryption Standard in Galois/Counter Mode

References

1. Korki, M.; Jin, J.; Tian, Y.C. Real-time cyber-physical systems: State-of-the-art and future trends. In *Handbook of Real-Time Computing*; Springer: Singapore, 2022; pp. 509–540. https://doi.org/10.1007/978-981-287-251-7_37.
2. Ohemeng, M.O.; Sheldon, F.T. Self-Triggered Switched ISS Framework Under Computational Weaponization. *Mathematics* **2026**, *14*, 2871. https://doi.org/10.3390/math14162871.
3. Lin, X.; An D.; Cui, F.; Zhang, F. False data injection attack in smart grid: Attack model and reinforcement learning-based detection method. *Front. Energy Res.* **2023**, *10*, 1104989. https://doi.org/10.3389/fenrg.2022.1104989.
4. Khargharia, H.S.; Ouali, A.; Shakya, S.; Ahmad, S. Collision avoidance in UAV swarms: A learning-centric perspective on collaborative intelligence. *Neurocomputing* **2025**, 132020. https://doi.org/10.1016/j.neucom.2025.132020.
5. Alam, M.; Rahman, M.L.; Paul, S.K.; Hays, A.W.; Hussain, A.; Huq, M.I.; Saxena, N. SoK:PHILTER: Uncovering Security and Functional Gaps in AI-based Phishing Website Detection Literature via an LLM-based Reasoning Framework. In *Proceedings of the 35th USENIX Security Symposium (USENIX Security 26) 2026*; USENIX Association: Berkeley, CA, USA, 2026; pp. 4981–5000.
6. Pishdar, M.; Lei, Y.; Harfoush, K.; Manzoor, J. Denial-of-Service Attacks on Permissioned Blockchains: A Practical Study. *J. Cybersecur. Priv.* **2025**, *5*, 39. https://doi.org/10.3390/jcp5030039.
7. Ryan, P.Y.; Schneider, S.A. An attack on a recursive authentication protocol A cautionary tale. *Inf. Process. Lett.* **1998**, *65*, 7–10. https://doi.org/10.1016/S0020-0190(97)00180-4.
8. Paulson, L.C. Mechanized proofs for a recursive authentication protocol. In *Proceedings of the 10th Computer Security Foundations Workshop*; IEEE: New York, NY, USA, 1997; pp. 84–94. https://doi.org/10.1109/CSFW.1997.596790.
9. Burbank, J.; Caleb, T.; Andam, E.; Kaabouch, N. Detection and Mitigation of Cyber Attacks on UAV Networks. *Electronics* **2026**, *15*, 317. https://doi.org/10.3390/electronics15020317.
10. Graziano, B.; Sargolzaei, A. Cybersecurity of Unmanned Aerial Vehicles from a Control Systems Perspective: A Review. *Electronics* **2026**, *15*, 163. https://doi.org/10.3390/electronics15010163.
11. Wang, Y.; Xing, W.; Zhao, X. A survey on cyber-attacks and defenses in cyber-physical systems. *Int. J. Syst. Sci.* **2026**, *57*, 4857–4878. https://doi.org/10.1080/00207721.2025.2602892.
12. Pericharla, K.V.; Chapidi, S.; Manyam, A.B.; Geethanjali, M.; Nahar, T.; Gupta, C.P.; Das, V. Drone Security Protocols: Current Technologies, Vulnerabilities, and Emerging Defences. In *International Conference on Innovations in Data Science*; Springer: Cham, Switzerland, 2025; pp. 217–226. https://doi.org/10.1007/978-3-032-19300-1_19.
13. Ko, Y.; Kim, J.; Duguma, D.G.; Astillo, P.V.; You, I.; Pau, G. Drone Secure Communication Protocol for Future Sensitive Applications in Military Zone. *Sensors* **2021**, *21*, 2057. https://doi.org/10.3390/s21062057.
14. Won, J.; Seo, S.H.; Bertino, E. Certificateless cryptographic protocols for efficient drone-based smart city applications. *IEEE Access* **2017**, *5*, 3721–3749. https://doi.org/10.1109/ACCESS.2017.2684128.
15. Sarkar, S.; Shafaei, S.; Jones, T.S.; Totaro, M.W. Secure Communication in Drone Networks: A Comprehensive Survey of Lightweight Encryption and Key Management Techniques. *Drones* **2025**, *9*, 583. https://doi.org/10.3390/drones9080583.
16. Zhaxygulova, D.; Iavich, M.; Rakhmetullina, S.; Alipbayev, K. Secure and Energy-Aware Cryptographic Framework for IoT-Enabled UAV Systems. *Symmetry* **2025**, *17*, 1987. https://doi.org/10.3390/sym17111987.
17. Jithu Vijay, V.P.; Shahanas, I.N.; Thampi, S.M.; Nair, A.S. A quantum-secure digital signature-based communication protocol for the Internet of Drones (IoD). *J. Netw. Comput. Appl.* **2025**, 104398. https://doi.org/10.1016/j.jnca.2025.104398.
18. Medley, L.; Loe, F.A.; Quaglia, E.A. Sok: Delay-based cryptography. In *Proceedings of the 2023 IEEE 36th Computer Security Foundations Symposium (CSF)*; IEEE: New York, NY, USA, 2023; pp. 169–183.
19. Biryukov, A.; Fisch, B.; Herold, G.; Khovratovich, D.; Leurent, G.; Naya-Plasencia, M.; Wesolowski, B. *Cryptanalysis of algebraic verifiable delay functions. In Annual International Cryptology Conference*; Springer: Cham, Switzerland, 2024; pp. 457–490.
20. Xu, T.; Potkonjak, M. Stable and secure delay-based physical unclonable functions using device aging. In *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*; IEEE: New York, NY, USA, 2015; pp. 33–36.

21. Fridman, E.; Selivanov, A. Using delay for control. *Annu. Rev. Control. Robot. Auton. Syst.* **2025**, *8*, 101–126. https://doi.org/10.1146/annurev-control-022723-033031.
22. Lee, H.S.; Lee, J.H.; Hong, H.S.; Park, P. Survey on inequality techniques for stability analysis of time-varying delay systems. *Int. J. Control. Autom. Syst.* **2025**, *23*, 973–989. https://doi.org/10.1007/s12555-025-0150-2.
23. Zhou, Y.; Zhu, E.; Li, S. An image encryption algorithm based on the double time-delay Lorenz system. *Math. Biosci. Eng.* **2023**, *20*, 18491–18522.
24. Ghosh, D.; Banerjee, S.; Chowdhury, A.R. Synchronization between variable time-delayed systems and cryptography. *EPL Europhys. Lett.* **2007**, *80*, 30006.
25. Zourmba, K.; Effa, J.Y.; Fischer, C.; Rodríguez-Muñoz, J.D.; Moreno-Lopez, M.F.; Tlelo-Cuautle, E.; Nkapkop, J.D.D. Fractional order 1D memristive time-delay chaotic system with application to image encryption and FPGA implementation. *Math. Comput. Simul.* **2025**, *227*, 58–84. https://doi.org/10.1016/j.matcom.2024.07.035.
26. Koochakzadeh, A.; Naderi Soorki, M.; Azizi, A.; Mohammadsharifi, K.; Riazat, M. Delay-Dependent Stability Region for the Distributed Coordination of Delayed Fractional-Order Multi-Agent Systems. *Mathematics* **2023**, *11*, 1267. https://doi.org/10.3390/math11051267.
27. Tissir, E.H. Exponential stability and guaranteed cost of switched linear systems with mixed time-varying delays. *Int. Sch. Res. Not.* **2011**, *2011*, 412912. https://doi.org/10.5402/2011/412912.
28. Mohammed, C.; Ismail, B.; El Houssaine, T.; Abdelaziz, H. New stability conditions for discrete-time switched systems with time-varying delay under arbitrary switching signal and with average dwell time approach. In *Proceedings of the 2022 10th International Conference on Systems and Control (ICSC)*; IEEE: New York, NY, USA, 2022; pp. 311–316. https://doi.org/10.1109/ICSC57768.2022.9993840.
29. Saidi, K.; Tissir, E.H. Improved Stability Criteria for Discrete-Time Switched Systems With Time-Varying Delay Using Relaxed Summation Inequality. In *Proceedings of the 2025 5th International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET)*; IEEE: New York, NY, USA, 2025; pp. 1–8. https://doi.org/10.1109/IRASET64571.2025.11008320.
30. Li, Z.; Chang, D.; Ma, D. Event-triggered and self-triggered control for networked control systems with short network-induced delays: A switched system approach. In *Proceedings of the 2017 36th Chinese Control Conference (CCC)*; IEEE: New York, NY, USA, 2017; pp. 7967–7972. https://doi.org/10.23919/ChiCC.2017.8028616.
31. Ohemeng, M.O. Neural network-based estimation and control of cyber-physical systems using Artificial Intelligence. *Internet Things* 2026, 102096. https://doi.org/10.1016/j.iot.2026.102096.
32. Ohemeng, M.O.; Sheldon, F.T. Physics-Constrained Optimization Framework for Detecting Stealthy Drift Perturbations. *Mathematics* **2026**, *14*, 1113. https://doi.org/10.3390/math14071113.
33. Ohemeng, M.O.; Sheldon, F.T. A Blockchain-Augmented CPS Framework to Mitigate FDI Attacks and Improve Resiliency. *Digital* **2026**, *6*, 22. https://doi.org/10.3390/digital6010022.
34. Hamid, A.; Almoghathawi, Y.; Alghazi, A.; Saleh, H. Enhancing resilience in UAV swarms: A literature review. *J. Saf. Sci. Resil.* 2025, 100268. https://doi.org/10.1016/j.jnlssr.2025.100268.
35. Non-Human Identity Management Group. What Breaks If Organisations Delay Crypto-Agility Until Quantum Computing Is Mature? 2026. Available online: https://nhimg.org/faq/what-breaks-if-organisations-delay-crypto-agility-until-quantum-computing-is-mat/ (accessed on 6 September 2026).
36. Matthew Green. Attack of the Week: TLS Timing Oracles, A Few Thoughts on Cryptographic Engineering (Blog). 2013. Available online: https://blog.cryptographyengineering.com/2013/02/04/attack-of-the-week-tls-timing-oracles/ (accessed on 2 August 2026).
37. Ganesh, R.; Khan, B.U.I.; Khan, A.R.; Kamsin, A.B. A panoramic survey of the advanced encryption standard: From architecture to security analysis, key management, real-world applications, and post-quantum challenges. *Int. J. Inf. Secur.* **2025**, *24*, 216. https://doi.org/10.1007/s10207-025-01116-x.
38. Serrano, R.; Duran, C.; Sarmiento, M.; Pham, C.-K.; Hoang, T.-T. ChaCha20–Poly1305 Authenticated Encryption with Additional Data for Transport Layer Security 1.3. *Cryptography* **2022**, *6*, 30. https://doi.org/10.3390/cryptography6020030.
39. Tidrea, A.; Korodi, A.; Silea, I. Elliptic Curve Cryptography Considerations for Securing Automation and SCADA Systems. *Sensors* **2023**, *23*, 2686. https://doi.org/10.3390/s23052686.
40. Esiner, E.; Tefek, U.; Mashima, D.; Chen, B.; Kalbarczyk, Z.; Nicol, D.M. Message authentication and provenance verification for industrial control systems. *Acm Trans. Cyber-Phys. Syst.* **2023**, *7*, 1–28. https://doi.org/10.1145/3607194.
41. Srinivas, J.; Das, A.K.; Kumar, N.; Rodrigues, J.J. TCALAS: Temporal credential-based anonymous lightweight authentication scheme for Internet of drones environment. *IEEE Trans. Veh. Technol.* **2019**, *68*, 6903–6916. https://doi.org/10.1109/TVT.2019.2911672.
42. Fränzle, M.; Grosen, T.M.; Larsen, K.G.; Zimmermann, M. Efficient Runtime Verification of Real-Time Systems under Parametric Communication Delays. *Form. Asp. Comput.* **2026**, *38*, 1–32. https://doi.org/10.1145/3749848.

43. Cziborová, D.; Dobos-Kovács, M.; Marussy, K.; Vörös, A. Unified Timing-Aware Program Verification. In *Proceedings of the International Conference on Fundamental Approaches to Software Engineering*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2026; Volume 16504, pp. 412–423. https://doi.org/10.1007/978-3-032-22774-4_21.
44. Serazzi, G. Systems with Homogeneous Workloads. In *Performance Engineering*; Springer: Cham, Switzerland, 2024. https://doi.org/10.1007/978-3-031-36763-2_2.
45. Khalil, A. Key Stability Challenges in Large-Scale TimeDelay Systems and Cloud Control Systems. In *Proceedings of the 2026 8th Global Power, Energy and Communication Conference (GPECOM)*; IEEE: New York, NY, USA, 2026; pp. 1028–1033. https://doi.org/10.1109/GPECOM70462.2026.11578649.
46. Chaillet, A.; Karafyllis, I.; Pepe, P.; Wang, Y. The ISS framework for time-delay systems: A survey. *Math. Control. Signals Syst.* **2023**, *35*, 237–306. https://doi.org/10.1007/s00498-023-00341-w.
47. Liao, F.; Wu, J.; Tomizuka, M. An improved delay-dependent stability criterion for linear uncertain systems with multiple time-varying delays. *Int. J. Control* **2014**, *87*, 861–873. https://doi.org/10.1080/00207179.2013.861081.
48. Divya, K.; Uma Priyadarsini, P.S. Securing IoMT data with Algorand blockchain, XChaCha20-Poly1305 encryption, and decentralized storage alternatives. *Sci. Rep.* **2025**, *15*, 23476. https://doi.org/10.1038/s41598-025-08527-9.
49. Castellanos, J.H.; Antonioli, D.; Tippenhauer, N.O.; Ochoa, M. Legacy-compliant data authentication for industrial control system traffic. In *Proceedings of the International Conference on Applied Cryptography and Network Security*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2017; Volume 10355, pp. 665–685. https://doi.org/10.1007/978-3-319-61204-1_33.
50. Tidrea, A.; Korodi, A.; Silea, I. Cryptographic Considerations for Automation and SCADA Systems Using Trusted Platform Modules. *Sensors* **2019**, *19*, 4191. https://doi.org/10.3390/s19194191.
51. Baker, M.A.; Al-Qrize, F. *AuthLWE: Benchmark Datasets for Lattice-Based Post-Quantum Authentication on Constrained IoT Devices;* Dataset; Zenodo: Geneve, Switzerland, 2025. https://doi.org/10.5281/zenodo.15284513.
52. Charqi, M.; Tissir, E.H.; Boumhidi, I.; Hmamed, A.; Boukili, B.; New criteria for robust exponential stability of uncertain discrete-time switched systems with time-varying delay via average dwell time approach and under arbitrary switching signal. *J. Control. Autom. Electr. Syst.* **2024**, *35*, 276–290.
53. Hadi, H.J.; Cao, Y.; Khan, M.K.; Ahmad, N.; Hu, Y.; Fu, C. UAV-NIDD: A Dynamic Dataset for Cybersecurity and Intrusion Detection in UAV Networks. *IEEE Trans. Netw. Sci. Eng.* **2025**, *12*, 2739–2758. https://doi.org/10.1109/TNSE.2025.3553442.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.