ICJK
Journal of Reliable and Secure Computing
http://dx.doi.org/10.62762/JRSC.2026.615829

RESEARCH ARTICLE

Paying for Trust at the Edge: Secure, Lightweight, Deadline-Aware Named Data Networking for Edge-Cloud Compute Orchestration

Mehbub Alam[1,*]

[1] Department of Computer Science and Engineering, Indian Institute of Information Technology Raichur, Raichur 584102, India

Abstract

Edge nodes are among the most resource-constrained components of computing infrastructure, yet security verification competes directly with application execution for limited compute resources. In Named Data Networking (NDN), each retrieved object carries a producer signature whose verification consumes edge CPU cycles, while existing cryptographic and orchestration studies rarely quantify its impact on service reliability. This paper develops Sled-Ndn, a secure edge–cloud framework over NDN that incorporates a Security Verification Engine (SVE) into the edge compute budget, explicitly coupling verification occupancy with named-function execution. Sled-Ndn combines Dacsir, which prices verification into deadline-aware cache-assisted routing; Lsav, which amortises one Ed25519 signature over a Merkle tree covering B Data packets and re-validates cached objects using symmetric tokens within a trust domain; and Drep, which dynamically provisions and reclaims cloud burst workers according to edge utilisation. Cryptographic costs measured on a host CPU were scaled to individual tiers using modelling factors and incorporated into a discrete-event simulator. At 6,000 Interests per second, Sled-Ndn achieves a deadline satisfaction ratio of 0.978, compared with 0.952 for the same datapath using per-packet RSA-2048 and 0.600 for vanilla NDN, while reducing mean per-request verification latency from 7,150.5 to 258.4 μs and recovering 94.5 % of the insecure upper-bound utility. Replacing RSA-2048 with Ed25519 alone yields little improvement, indicating that reliability is more sensitive to verification frequency than to the cryptographic primitive in the evaluated configurations. These results demonstrate that verification should be treated as a first-class resource cost and jointly optimised with workload placement and deadline requirements, while noting that system-level results are simulation-based and per-tier scaling factors are estimated.

Keywords: secure edge computing, trustworthy edge infrastructure, cryptographic verification overhead,

Submitted: 16 July 2026
Revised: 16 September 2026
Accepted: 19 September 2026
Published: 21 September 2026

Vol. 2, **No.** 3, 2026.
10.62762/JRSC.2026.615829

***Corresponding author:**
✉ Mehbub Alam
mehbub@iiitr.ac.in

Citation
Alam, M. (2026). Paying for Trust at the Edge: Secure, Lightweight, Deadline-Aware Named Data Networking for Edge-Cloud Compute Orchestration. *Journal of Reliable and Secure Computing*, 2(3), 212–232.

resource-efficient security, named data networking, deadline-aware service reliability, secure edge-cloud orchestration, signature amortisation.

1 Introduction

Edge computing places computation on resource-constrained nodes that operate outside the physical and administrative protections of conventional data centres [1]. Security mechanisms on these nodes therefore compete with application workloads for limited computing capacity, creating a fundamental tension between security and reliable service execution [2, 3]. This tension is particularly important in Named Data Networking (NDN), where data objects are individually secured and authenticated. An NDN forwarder resolves hierarchical content names, keeps per-request state, and secures the object rather than the channel [4, 5]. If the result of a named computation is itself a named Data object, the Content Store (CS) becomes a memo table for computation, the Pending Interest Table (PIT) collapses concurrent identical requests into one execution [6], and the Forwarding Information Base (FIB) becomes a placement mechanism. This underpins named function execution [7–10] and explains why NDN recurs in edge and fog proposals [11–13].

This interaction sharpens the security question. In NDN, a Data packet carries a producer signature, and the validation policy in force determines which nodes check it and when [14, 15]. Deployments that validate at the consumer and at caching forwarders, which is the policy the baselines in this study implement, are affordable in wide-area distribution because validation runs on capable routers. At the edge, however, the situation is different: the node performing validation is also the node hosting the computation. Cycles spent checking a signature are cycles unavailable to the named function the node was deployed to run. Security and computation therefore draw on one budget, and the user observes the outcome only as a missed deadline.

Three bodies of literature bear on this problem, but their interaction remains largely unexamined. Information-centric networking established the architecture and its caching behaviour [16–18], and the named-computation line extended naming to execution [19], but generally treats signed Data as given without accounting for the cost of its verification on a node that also executes the computation. Edge and cloud orchestration produced placement policies optimised against an explicit cost vector [1, 20–25], which are the closest antecedents to our utility model; these vectors carry compute, bandwidth and energy terms but no cryptographic term. NDN security has been surveyed thoroughly [26–29], with the attack surface and verification cost implications characterised in detail [30]. The verification burden is usually addressed by substituting cheaper primitives [31] or adopting content-object security protocols on constrained hardware [32]. A parallel line in secure distributed computing bounds or removes a recurring structural cost instead [33], and recent work has further demonstrated that cryptographic validation at the edge can be made efficient through public-key mechanisms designed for resource-limited nodes [34]. No formulation represents the fact that verification consumes the same constrained edge resources computation requires. Under light load this is invisible; under high load and tight deadlines it decides whether the service meets its objective, because verification contracts the service rate that determines where the system saturates.

A natural response is to replace RSA with an elliptic-curve scheme. Our measurements indicate that this addresses the wrong half of the operation. Ed25519 signing is roughly an order of magnitude cheaper than RSA-2048 signing (410.0 against 4,437.8 μs at the edge tier), which matters for producers, but under any validation policy that checks a signature more than once per delivered object, verification is the operation that recurs, and a single Ed25519 verification (851.7 μs) is in fact about 2.9 times slower than a single RSA-2048 verification (289.8 μs) in the evaluated implementation. We present this as an observation about the evaluated configurations rather than a universal statement, since the ratio depends on the library, exponent and platform.

The proposed Sled-Ndn is organised around one modelling decision: the Security Verification Engine (Sve) is placed inside the edge compute budget, so its occupancy is subtracted from the cycle rate available to named-function execution, and verification is priced in the utility function alongside compute and transport. Dacsir performs deadline-aware, cache-assisted placement that accounts for this cost. Lsav amortises one Ed25519 signature over a Merkle tree covering B Data packets and mints a symmetric Edge Trust Token (Ett) so that re-serving a cached object inside a trust domain costs one message authentication code. Drep provisions and reclaims

cloud burst workers from measured edge utilisation. Cryptographic costs were measured on a host CPU and scaled to the edge, fog, cloud and IoT tiers; the system was evaluated in a discrete-event simulator against vanilla NDN, an IP edge stack and two controlled ablations.

1.1 Contributions

The main contributions are as follows.

- A secure edge-cloud NDN architecture is proposed that integrates computation, caching, cryptographic verification and cloud elasticity within a unified framework.
- A security-resource coupling model is designed in which verification occupancy reduces the computational capacity available for edge execution, and verification time is explicitly priced in the utility function.
- Dacsir is developed to incorporate verification cost into the placement decision, while Lsav reduces critical-path public-key verifications through Merkle-tree amortisation and intra-domain re-validation.
- Drep is developed to provision and reclaim bounded cloud burst capacity based on measured edge utilisation, including verification occupancy, using a hysteresis mechanism.
- An evaluation methodology is developed that combines cryptographic measurements on a host CPU with discrete-event simulation and seed-matched comparisons of six systems, including two controlled ablations. The results show that, in the evaluated configurations, deadline satisfaction and service delay depend more strongly on verification frequency than on replacing the cryptographic primitive.

1.2 Paper Organisation

The remainder of this paper is organised as follows. Section 2 reviews related work on NDN, edge computing, and security. Section 3 presents the system model and mathematical formulation. Section 4 describes the Sled-Ndn architecture and its proposed mechanisms. Section 5 presents the evaluation methodology. Section 6 discusses the experimental results. Section 7 discusses the implications, limitations, and future directions. Section 8 concludes the paper.

2 Related Work

2.1 NDN and in-network computation

Content-Centric Networking [4] and NDN [5] replaced host addresses with hierarchical names and introduced the CS, PIT and FIB triple that defines the forwarding plane. Surveys mapped the design space [16, 17]; Yi et al. [6] argued that per-Interest state enables adaptive forwarding rather than burdening it; and caching received sustained attention [18, 35] under Zipf-distributed demand [36]. Naming was then extended from content to computation by Named Function Networking [7], unikernel execution with demand-driven placement [8], parameterised remote invocation [9], distributed computing over ICN [10], and a Content Store restructured for computation reuse [19].

The existing NDN and named-computation literature establishes that the NDN forwarding plane can support computation and that in-network caching can convert repeated computations into cache hits. However, it does not account for the operating cost of NDN's security mechanisms on the constrained nodes where these functions execute. Security is therefore typically treated as an architectural property rather than as a consumer of the resources required for computation.

2.2 Edge and cloud computing

The case for edge computing and surveys of mobile edge computing frame offloading in host-centric terms [1, 20, 21]. ICN-specific treatments arrived through the Internet of Things (IoT), with NDN measured on constrained hardware [37] and edge data processing developed over it [12, 38]; resolution strategies for named functions [11] and breadcrumb-based discovery of edge compute resources [13] addressed where a computation should run. On the allocation side, joint communication, computation and caching formulations, and latency-minimising collaborative placement schemes, optimise assignment under delay constraints [22–25].

These formulations provide the structural basis for our model: an explicit cost vector, a deadline constraint, and a placement decision minimising cost subject to it. What remains unresolved is that none of the cost vectors contains a cryptographic term, so a policy derived from them is blind to the fact that a placement decision also determines where the associated verification work is performed. In NDN this matters particularly, because a cache hit still requires

verification: a policy maximising hit ratio can increase total verification load even while decreasing compute load.

2.3 Security and trust in NDN

NDN's data-centric security has been surveyed comprehensively [26, 27], with trust schematisation supplying the key-to-name binding [15]. Interest flooding and cache poisoning have been characterised [28] and mitigation mechanisms proposed [29], while the per-packet verification burden has been stated explicitly [14]. Merkle's construction for amortising one signature over many messages [39] is standard in cryptography, underpins certificate transparency [40] and hash-based signatures [41], and has been applied to sign packet streams with a single signature [33]. However, the evaluated literature does not, to our knowledge, combine this construction with an NDN edge trust model that explicitly accounts for the resources consumed by verification.

The existing literature establishes threat coverage and key management, but does not establish the affordability of security mechanisms on constrained nodes. Performance is typically characterised through per-operation microbenchmarks, which provide a necessary input to a systems-level analysis but do not capture the effect of repeated verification on application execution. Similarly, substituting an elliptic-curve scheme for RSA can reduce signing cost and key size, but does not by itself address the recurring verification cost imposed by the validation policy. Security attacks specific to NDN and their implications for the per-packet verification burden that forwarders must sustain under adversarial load have been reviewed comprehensively [30]; taken together, these results motivate treating verification frequency as a design variable rather than a fixed architectural cost.

2.4 Secure and reliable edge computing

A separate line treats security and reliability in resource-constrained distributed infrastructures directly. Information-centric edge frameworks that co-locate computation and forwarding at the edge demonstrate that in-network reuse can reduce task completion time substantially, yet do not model the security verification cost that such nodes must also bear [42]. Cryptographic data validation at the edge has been shown to impose non-trivial processing overhead on resource-limited nodes, with the cost of public-key operations identified as the primary bottleneck [34], and a blockchain-enhanced federated intrusion-detection framework for smart homes, integrating Proof of Stake, practical Byzantine fault tolerance and Proof of Authority consensus, reports a 52.8% computational overhead relative to conventional federated learning [43], an explicit example of a security mechanism drawing on resources that the workload would otherwise use. At the infrastructure layer, a systematic survey of multi-access edge computing security and privacy analyses the threat vectors arising from running security mechanisms on resource-constrained MEC nodes, and proposes mitigations that must be weighed against their computational cost [44].

Two aspects carry over: the framing that edge security must be evaluated against the resources actually available rather than asymptotic cost, and the strategy of reducing or bounding recurring structural costs rather than accelerating an individual operation. What the evaluated literature does not supply is a model in which such costs and the application workload contend for the same resource budget, allowing the trade-off to be quantified rather than argued. Addressing this gap requires three elements: shared capacity for verification and computation, a utility function that prices verification time against deadline attainment, and a placement rule that accounts for both. Sled-Ndn combines these elements. The Sve is placed inside the edge compute budget, so verification and named-function execution explicitly contend for capacity (Section 3.3); τ_{sec} is included in the utility function, and Dacsir accounts for this cost when selecting an execution site (Section 4.2); and Lsav reduces the number of critical-path verifications rather than their unit cost (Section 4.3). The evaluation further separates these effects through controlled ablations that vary the cryptographic primitive and the verification structure independently.

3 System Model and Mathematical Formulation

3.1 Network topology and naming

We consider the three-tier hierarchy shown in Figure 1. Let $\mathcal{E}$ denote the set of edge nodes ($|\mathcal{E}| = 16$), $\mathcal{A}$ the set of aggregation or fog nodes ($|\mathcal{A}| = 4$), and c the cloud. Each edge node $j \in \mathcal{E}$ has a parent $a(j) \in \mathcal{A}$ and a peer set $\mathcal{P}(j) \subset \mathcal{E}$ of siblings within the same trust domain. Every node runs an NDN forwarder with a Content Store, a Pending Interest Table, and a Forwarding Information Base, co-located with a compute pool and

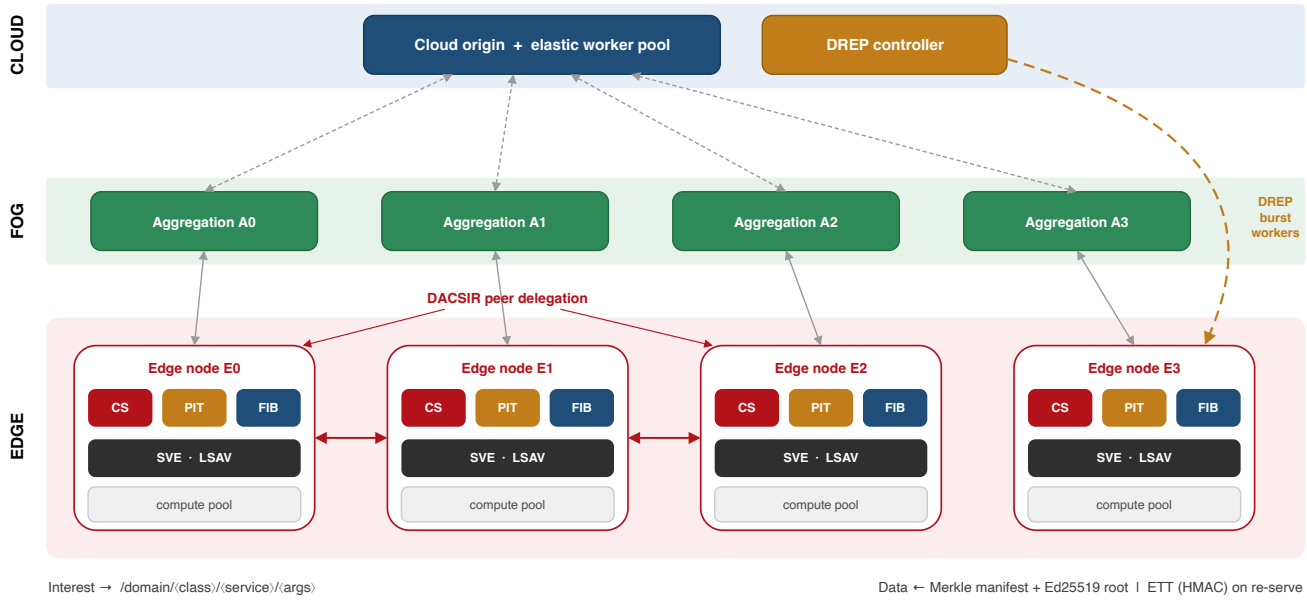

Figure 1. Reference architecture of SLED-NDN. Each edge forwarder couples the CS, PIT and FIB datapath with a Security Verification Engine running LSAV and with a compute pool. The SVE and the compute pool share the node's cores, which is the coupling formalised in Eq. (4). DACSIR may delegate a named function to a sibling edge inside the trust domain, to the fog tier, or to the cloud. DREP dedicates cloud burst workers to an overloaded edge and reclaims them when demand subsides.

a Security Verification Engine.

Names are hierarchical and carry the service class explicitly:

$$/\langle \text{domain} \rangle / \langle \text{class} \rangle / \langle \text{service} \rangle / \langle \text{args} \rangle / \langle \text{ver} \rangle. \qquad (1)$$

Placing the class in the second component of Eq. (1) allows a forwarder to obtain the corresponding deadline budget and utility weight through longest-prefix matching, without additional control-plane signalling or session state. Let $\mathcal{K}$ be the set of classes; each $k \in \mathcal{K}$ is associated with a relative deadline D_k, a utility weight v_k, and a nominal compute demand ω_k.

3.2 Demand model

Interests arrive as a Poisson process with rate λ, distributed uniformly across the edge nodes. Names are drawn from a catalogue $\mathcal{N}$ of $M = 5,000$ objects with Zipf popularity of exponent α [36], so name n of rank r_n is requested with probability $p_n \propto r_n^{-\alpha}$. A fraction ϕ of names correspond to named functions requiring ω_n cycles; the remainder are static named content. Object n has size s_n bytes and therefore occupies $P_n = \lceil s_n / \mathrm{MTU} \rceil$ Data packets, where MTU denotes the payload capacity of one Data packet. A request r for name n inherits the class of the name and hence the deadline D_r and utility weight v_r. For such a request, let $P_r = P_n$ and $\omega_r = \omega_n$.

3.3 Latency decomposition and the security-resource coupling

We build on the standard delay decomposition used in computation-offloading models [21, 45], extended here with Content Store and verification terms, so that the delay of request r is decomposed as

$$T_r = T_r^{\mathrm{tx}} + T_r^{\mathrm{prop}} + T_r^{\mathrm{q}} + T_r^{\mathrm{cs}} + T_r^{\mathrm{cmp}} + \tau_{\mathrm{sec}}(r), \qquad (2)$$

where the terms represent transmission, propagation, queueing at links and compute stations, Content Store lookup, computation, and cryptographic verification, respectively. For a link ℓ with capacity R_ℓ (bit/s) carrying s bytes,

$$T_\ell^{\mathrm{tx}} = \frac{8s}{R_\ell}, \qquad T_\ell^{\mathrm{q}} = \frac{\rho_\ell \, T_\ell^{\mathrm{tx}}}{2(1-\rho_\ell)}, \qquad (3)$$

where ρ_ℓ is the measured link utilisation and T_ℓ^{q} is the Pollaczek-Khinchine mean waiting time of an M/D/1 station [46]. Node i has m_i cores, each with nominal rate f_i cycles per second.

The key modelling assumption concerns the SVE. Let ρ_i^{sve} denote the fraction of node i's total core capacity consumed by verification. Because verification executes on the same cores as named functions, the effective rate available to computation is

$$\tilde{f}_i = f_i \, (1 - \rho_i^{\mathrm{sve}}) \, , \qquad (4)$$

and the expected compute queueing delay at node i holding a backlog of Q_i cycles is $T_i^{\mathrm{q}} = Q_i/(m_i \tilde{f}_i)$. Equation (4) formalises the argument in Section 1: increasing verification cost does not merely add a constant to Eq. (2); it reduces the capacity available to application execution. Consequently, treating τ_{sec} only as an additive delay understates the effect of verification on a constrained node under load.

3.4 Cryptographic cost model

Let t_{sig}, t_{ver}, t_{mac} and t_{h} denote the measured latencies of signature generation, signature verification, message authentication, and a 64-byte hash compression, respectively, each scaled to the tier in question. Let $t_{\mathrm{h}}^{\mathrm{leaf}}$ denote the cost of hashing one packet payload. For a legacy per-packet scheme with N_{ver} on-path verification points,

$$\tau_{\mathrm{sec}}^{\mathrm{legacy}}(r) = P_r \cdot N_{\mathrm{ver}} \cdot t_{\mathrm{ver}}. \tag{5}$$

Under Lsav, a producer accumulates B Data packets, builds a Merkle tree over their content digests, and signs only the root. A verifier checks one root signature per window, validates each packet against the root along a proof path of $\lceil \log_2 B \rceil$ inner hashes, and mints an Edge Trust Token for the object, giving

$$\tau_{\mathrm{sec}}^{\mathrm{miss}}(r) = P_r\Big(\tfrac{1}{B} t_{\mathrm{ver}} + \lceil \log_2 B \rceil t_{\mathrm{h}} + t_{\mathrm{mac}}\Big), \tag{6}$$

$$\tau_{\mathrm{sec}}^{\mathrm{hit}}(r) = P_r\, t_{\mathrm{mac}}, \tag{7}$$

$$\tau_{\mathrm{sec}}^{\mathrm{sign}}(r) = P_r\Big(\tfrac{1}{B} t_{\mathrm{sig}} + t_{\mathrm{h}}^{\mathrm{leaf}}\Big). \tag{8}$$

The first term of Eq. (6) decreases with B, whereas the Merkle-path term grows with $\lceil \log_2 B \rceil$ and the token term is constant, giving diminishing returns as B increases. For a deadline-critical request with $D_r \le D^{\mathrm{def}}$, the verify-deferred path replaces Eq. (6) on the critical path with

$$\tau_{\mathrm{sec}}^{\mathrm{def}}(r) = P_r\big(t_{\mathrm{mac}} + \epsilon\, \tau_{\mathrm{sec}}^{\mathrm{miss}}(r)/P_r\big), \tag{9}$$

where ϵ is the audit sampling rate. The full verification is performed off the critical path for every object, while a sampled fraction is also verified on the critical path. Section 4.5 discusses the security implications and limitations of this mechanism.

3.5 Utility model

Deadline attainment is scored softly, so a request missing its deadline by a small margin is not treated identically to one missing it by an order of magnitude:

$$\sigma_r = \begin{cases} 1, & T_r \le D_r, \\ \exp\!\left(-\kappa \dfrac{T_r - D_r}{D_r}\right), & T_r > D_r, \end{cases} \tag{10}$$

where κ controls the decay rate. Let $\mathcal{R}$ be the set of admitted requests, partitioned into $\mathcal{R}_E$ served by the edge or fog tier, including Content Store hits, and $\mathcal{R}_C$ served by the cloud. Let C_E and C_C denote total edge and cloud core-seconds, $\Theta = \sum_r \tau_{\mathrm{sec}}(r)$ the total on-path verification time, $\bar{M}$ the mean Content Store occupancy, W the WAN bytes carried, and β^{ws} the burst worker-seconds lent by the cloud.

Then

$$U_{\mathrm{edge}} = \tfrac{1}{|\mathcal{R}|}\Big(\textstyle\sum_{r\in\mathcal{R}_E} v_r\sigma_r - c_e C_E - c_s\Theta - c_m\bar{M}|\mathcal{R}| - c_b\beta^{\mathrm{ws}}\Big), \tag{11}$$

$$U_{\mathrm{cloud}} = \tfrac{1}{|\mathcal{R}|}\Big(\textstyle\sum_{r\in\mathcal{R}_C} v_r\sigma_r - c_c C_C - c_w W + p_b\beta^{\mathrm{ws}}\Big), \tag{12}$$

$$\mathrm{ONU} = \mu\, U_{\mathrm{edge}} + (1-\mu)\, U_{\mathrm{cloud}}, \tag{13}$$

where c_e, c_c, c_s, c_m, c_w and c_b price edge compute, cloud compute, on-path verification, cache occupancy, WAN transport and borrowed burst capacity, p_b is the price the cloud earns for lending, and μ weights the two tiers.

Because of the normalisation by $|\mathcal{R}|$, ONU is a utility per request; the results in Section 6 report it scaled to 10^3 requests. The term $c_s\Theta$ in Eq. (11) is central to the formulation: it prices verification against delivered value in the same units, allowing a trust model to be assessed as an economic trade-off rather than only as a latency cost. Because c_s is a policy parameter, Section 6 reports how the ranking varies with it.

The two headline reliability metrics, the deadline satisfaction ratio (DSR) and the average service delay (ASD), follow directly:

$$\mathrm{DSR} = \frac{|\{r : T_r \le D_r\}|}{|\mathcal{R}|}, \qquad \mathrm{ASD} = \frac{1}{|\mathcal{R}|}\sum_r T_r. \tag{14}$$

Both are reported alongside the overall network utility (ONU) of Eq. (13).

3.6 Resource provisioning dynamics

The cloud holds a contracted slice of m_c cores, of which at most $\lfloor \gamma m_c \rfloor$ may be dedicated as burst workers. Let $\beta_j(t)$ denote the workers held by edge j and $u_j(t)$ its measured utilisation over control epoch T_c. Drep evolves

$$\beta_j(t{+}T_c) = \Big[\beta_j(t) + \zeta_p\big(u_j - \theta_{hi}\big)^{+} m_j - \zeta_r\big(\theta_{lo} - u_j\big)^{+}\beta_j(t)\Big]_0^{\beta_{\max}}, \tag{15}$$

subject to $\sum_j \beta_j(t) \le \gamma m_c$, where ζ_p and ζ_r are the provisioning and reclamation gains, θ_{hi} and θ_{lo} are the corresponding thresholds, $(x)^+ = \max(x, 0)$, and $\beta_{\max}$ is the per-edge cap. Reclamation is gated by N_h consecutive epochs below θ_{lo} to suppress oscillation. Workers dedicated to an edge are removed from the shared cloud pool, so lending incurs a real opportunity cost.

3.7 Optimisation problem

Let $x_{r,i} \in \{0, 1\}$ indicate that request r executes at node i, and let $\hat{T}_r(\mathbf{x})$ denote the predicted completion time under assignment $\mathbf{x}$. The placement problem is

$$\max_{\mathbf{x},\boldsymbol{\beta}} \quad \text{ONU} \tag{16}$$

$$\text{s.t.} \quad \sum_i x_{r,i} = 1 \qquad \forall r, \tag{17}$$

$$\sum_r x_{r,i}\omega_r \le m_i \tilde{f}_i \Delta \qquad \forall i, \tag{18}$$

$$\hat{T}_r(\mathbf{x}) \le D_r \qquad \forall r \in \mathcal{R}^{\text{crit}}, \tag{19}$$

$$\sum_j \beta_j \le \gamma m_c, \tag{20}$$

$$\rho_i^{\text{sve}} \le \rho^{\max} \qquad \forall i, \tag{21}$$

where Δ is the scheduling horizon and $\mathcal{R}^{\text{crit}}$ is the set of deadline-critical requests. Constraint (17) assigns each request to one execution site, Eq. (19) enforces the deadline for critical requests, and Eq. (20) limits the total lent capacity. The two constraints central to the proposed formulation are Eq. (18), which incorporates $\tilde{f}_i$ from Eq. (4) so that verification occupancy directly affects the available compute capacity, and Eq. (21), which bounds the fraction of a node's capacity consumed by verification. Problem (16) is a mixed-integer optimisation problem with a nonlinear objective through the queueing terms in $\hat{T}_r$ and is computationally intractable in the general case. Since requests must be placed online, Section 4 develops a greedy online relaxation that evaluates a bounded candidate set of at most $|\mathcal{P}(j)| + 4$ sites per Interest, i.e. at $O(|\mathcal{P}(j)|)$ cost.

4 Sled-Ndn Architecture and Algorithms

We designed three complementary mechanisms to address the competition between security verification and application execution at the edge: Dacsir accounts for verification cost in placement, Lsav reduces critical-path verification frequency, and Drep absorbs residual workload pressure through elastic cloud capacity.

4.1 Datapath and cache admission

An arriving Interest is first matched against the CS. A fresh hit is verified under Eq. (7) and returned. On a miss, the PIT is consulted, and an identical pending Interest is aggregated so that one execution satisfies all consumers requesting the result. Otherwise, the FIB is consulted and Dacsir selects an execution site. Returning Data is verified at the trust-domain ingress, admitted to the cache under the Deadline-Aware Popularity (DAP) policy, and delivered.

DAP scores a candidate victim n by

$$s(n) = w_p \hat{p}_n + w_d \,\mathrm{urg}(n) + w_c \,\widehat{\omega}_n, \tag{22}$$

combining an exponentially weighted popularity estimate $\hat{p}_n$, the urgency $\mathrm{urg}(n)$ of the class requesting n, and its normalised recomputation cost $\widehat{\omega}_n$, with weights w_p, w_d and w_c; the candidate with the lowest score is evicted first. Retaining objects that are expensive to recompute and requested under tight deadlines is appropriate in this setting, because in a computational NDN a cache miss incurs recomputation rather than a fetch.

4.2 Dacsir: deadline-aware cache-assisted secure routing

Dacsir, given as Algorithm 1, is the online relaxation of problem (16). For each Interest, it enumerates a bounded candidate set comprising local execution, sibling edges, the parent fog node, a dedicated burst lane when one is held, and the shared cloud; predicts each candidate's completion time including its verification term; and selects, among candidates meeting the deadline, the one with the lowest marginal utility cost using the same coefficients c_e, c_c, c_s, c_w and c_b as Eq. (11) and Eq. (12). Here $c_{(i)}$ denotes the compute price of the tier hosting node i. If none is feasible, it selects the fastest candidate. Using the same cost model for placement and evaluation avoids the confound in which a scheduler is tuned on one metric and evaluated on another.

4.3 Lsav: lightweight signature amortisation and validation

Lsav, given as Algorithm 2, reduces the number of critical-path verifications rather than their unit cost. A producer buffers up to B Data packets, builds a Merkle tree over their digests, and signs the root once with Ed25519; each packet carries its proof path of $\lceil \log_2 B \rceil$ nodes. At the trust-domain ingress, a verifier checks the root signature once per window, validates each packet by replaying its proof path, and mints an Edge Trust Token

$$\mathrm{Ett}(n) = \mathrm{HMAC}_{K_e}\big(n \,\|\, \mathrm{digest}(n) \,\|\, e \,\|\, \mathrm{exp}\big), \tag{23}$$

Algorithm 1: DACSIR: deadline-aware cache-assisted secure Interest routing

Input: Interest I for name n at edge j, arrival time t_0, current time t
Output: Execution site, or immediate Data

$(k, D, v) \leftarrow \textsc{ParseName}(n)$ `// from Eq. (1)`
if *CsLookup*(j, n) *is fresh* **then**
  $\tau \leftarrow \textsc{Sve}_j.\textsc{Verify}(\tau^{\text{hit}}_{\text{sec}})$ `// Eq. (7)`
  return $\textsc{Deliver}(n, \tau)$
end
if $n \in \text{PIT}_j$ **then**
  $\text{PIT}_j[n] \leftarrow \text{PIT}_j[n] \cup \{I\}$
  return AGGREGATED
end
$\text{PIT}_j[n] \leftarrow \{I\}$
$\text{slack} \leftarrow D - (t - t_0)$
$\mathcal{C} \leftarrow \{j\} \cup \mathcal{P}(j) \cup \{a(j)\} \cup \{\text{burst}_j \text{ if } \beta_j > 0\} \cup \{c\}$
forall $i \in \mathcal{C}$ **do**
  $\hat{q}_i \leftarrow Q_i/(m_i \tilde{f}_i)$ `// queueing under Sve-reduced capacity, Eq. (4)`
  $\hat{T}_i \leftarrow \text{rtt}(j, i) + \hat{q}_i + \omega_n/(m_i \tilde{f}_i) + \tau_{\text{sec}}(i)$
  $\hat{\Psi}_i \leftarrow c_{(i)}\, \omega_n/(m_i \tilde{f}_i) + c_s \tau_{\text{sec}}(i) + c_w s_n \mathbb{1}[i \in \{c, \text{burst}_j\}]$
end
$\mathcal{F} \leftarrow \{i \in \mathcal{C} : \hat{T}_i \le \text{slack}\}$
if $\mathcal{F} \neq \emptyset$ **then**
  $i^\star \leftarrow \arg\min_{i \in \mathcal{F}} \hat{\Psi}_i$ `// cheapest site that still meets the deadline`
else
  $i^\star \leftarrow \arg\min_{i \in \mathcal{C}} \hat{T}_i$ `// deadline unreachable: minimise lateness`
end
return $\textsc{Forward}(I, i^\star)$

Algorithm 2: LSAV: signature amortisation and edge validation

Producer side
buffer $\leftarrow$ Data packets $d_1, \ldots, d_B$
$\text{leaves} \leftarrow [\,\text{H}(d_i)\,]_{i=1}^{B}$; $R \leftarrow \textsc{MerkleRoot}(\text{leaves})$
$\Sigma \leftarrow \textsc{Ed25519Sign}_{sk_p}(R)$ `// one signature per` B `packets, Eq. (8)`
forall $i \in [1, B]$ **do**
  attach $\langle R, \Sigma, \textsc{Path}(i)\rangle$ to d_i
end

Verifier side at trust-domain ingress node j
Input: Data d with manifest $\langle R, \Sigma, \pi\rangle$, deadline D
if $D \le D^{\text{def}}$ ***and*** *deferred verification enabled* **then**
  $\tau \leftarrow t_{\text{mac}}$; $\textsc{EnqueueAudit}(d)$ `// full check off the critical path`
  if *Bernoulli*(ϵ) **then**
    $\tau \leftarrow \tau + \textsc{FullVerify}(d)$ `// sampled audit, Eq. (9)`
  end
else
  if $R \notin \textit{RootCache}_j$ **then**
    abort unless $\textsc{Ed25519Verify}_{pk_p}(R, \Sigma)$
    $\textsc{RootCache}_j \leftarrow \textsc{RootCache}_j \cup \{R\}$
  end
  abort unless $\textsc{ReplayPath}(\text{H}(d), \pi) = R$ `//` $\lceil \log_2 B \rceil$ `hashes`
  $\tau \leftarrow \frac{1}{B} t_{\text{ver}} + \lceil \log_2 B \rceil t_{\text{h}} + t_{\text{mac}}$
end
$\text{ETT}(d) \leftarrow \text{HMAC}_{K_e}(n \,\|\, \text{H}(d) \,\|\, e \,\|\, \text{exp})$ `// Eq. (23)`
$\textsc{CsInsert}(d, \text{ETT}(d))$; **return** τ

Re-serve from any CS inside the domain
abort unless $\textsc{HmacVerify}_{K_e}(\text{ETT}(d))$ `// Eq. (7)`

where K_e is a domain key rotated each epoch e and exp is the token expiry. Re-serving the object from any Content Store inside the domain then requires only one HMAC verification, which is the source of the asymmetry between Eq. (6) and Eq. (7).

The ETT provides a weaker property than producer authentication. Only the Ed25519 root signature establishes producer authenticity, and it is checked when the object first enters the domain; the token certifies only that a node holding K_e admitted the object after validation within the trust domain during epoch e. This is adequate when the intra-domain forwarders lie within the trust boundary, the standard assumption for an operator-managed edge deployment, and it is not appropriate across administrative boundaries. Sections 4.5 and 7.6 treat the consequences.

4.4 DREP: dynamic resource elasticity and provisioning

DREP, given as Algorithm 3, realises Eq. (15). Each control epoch it compares measured edge utilisation against θ_{hi} and θ_{lo}, dedicating cloud workers to an overloaded edge and returning idle ones after N_h consecutive quiet epochs. Because measured utilisation includes SVE occupancy, higher verification overhead can trigger provisioning through the same control loop, as intended by Eq. (4). Provisioned workers incur a cold start and are drawn from the same contracted slice that serves general offload,

Algorithm 3: DREP: dynamic resource elasticity and provisioning

Input: Epoch T_c, thresholds θ_{hi}, θ_{lo}, gains ζ_p, ζ_r, hysteresis N_h, lendable pool Γ

forall *control epoch* **do**
  forall $j \in \mathcal{E}$ **do**
    $u_j \leftarrow \text{BUSY}_j/(m_j T_c)$ `// includes SVE occupancy`
    if $u_j > \theta_{hi}$ ***and*** $\Gamma > 0$ **then**
      $g \leftarrow \min\left(\lceil \zeta_p(u_j - \theta_{hi})m_j \rceil, \Gamma, \beta_{\max} - \beta_j\right)$
      $\Gamma \leftarrow \Gamma - g$; $m_c \leftarrow m_c - g$; $\beta_j \leftarrow \beta_j + g$
      SCHEDULEWARM(j, g, T_{cold}); $w_j \leftarrow 0$
    else if $u_j < \theta_{lo}$ ***and*** $\beta_j > 0$ **then**
      $w_j \leftarrow w_j + 1$
      if $w_j \geq N_h$ **then**
        $g \leftarrow \min\left(\lceil \zeta_r(\theta_{lo} - u_j)\beta_j \rceil, \beta_j\right)$
        $\beta_j \leftarrow \beta_j - g$; $\Gamma \leftarrow \Gamma + g$;
        $m_c \leftarrow m_c + g$; $w_j \leftarrow 0$
      end
    else
      $w_j \leftarrow 0$
    end
  end
end

so provisioning to one edge reduces shared cloud capacity.

4.5 Threat model and security analysis

Producers hold long-term Ed25519 keys bound to name prefixes by a trust schema [15]. Edge and fog nodes within one administrative domain share an epoch key K_e distributed out of band and rotated each epoch. The adversary may inject arbitrary Data, replay observed Data, request arbitrary names, and observe all traffic. The adversary may not compromise a forwarder inside the trust domain and may not recover K_e. This is an operator-managed trust-domain assumption, and it is the assumption on which every token-related property below rests.

Producer authenticity is established exclusively by the Ed25519 signature over the Merkle root. A packet is bound to that root through its proof path; consequently, *integrity* of an individual packet follows from the validity of the root signature and the collision resistance of the underlying hash function. *Cache poisoning* by an off-path adversary is prevented at ingress when deferred verification is not in use, because an object lacking a valid proof path to a correctly signed root is not admitted to the cache and therefore cannot receive a token; the deferred mode is treated separately below. *Replay* is bounded rather than completely prevented by two mechanisms: the freshness period governing Content Store retention and the token expiry field in Eq. (23). A replayed object that remains within the freshness window is therefore returned as the same authenticated object; the resulting risk concerns staleness rather than producer authenticity.

Because the ETT is a symmetric tag computed under K_e, compromise of a forwarder or of K_e reduces intra-domain re-validation to the security of the weakest key holder, while the root signature remains independently verifiable. Epoch rotation limits the exposure of such a compromise to one epoch. Across administrative boundaries, the token has no validity and the object must be re-validated, paying Eq. (6) again.

Deferred verification is optional and is disabled by default for any traffic whose actions are not recoverable. Where it is enabled for the critical class, an object may be served before its producer signature has been checked. The full check always occurs, off the critical path for every object and on the critical path with probability ϵ, so an adversary cannot predict which deliveries are audited; but a window exists in which an unverified object has been consumed, and the mechanism is therefore not equivalent to immediate verification. It suits actions that remain recoverable within that window and does not suit irreversible ones such as actuator commands, financial commitments or safety interlocks, which should disable deferral. We claim no formal security proof for any of these mechanisms, and in particular no bound on the adversary's advantage as a function of ϵ.

5 Evaluation Methodology

5.1 Simulator

The system was evaluated in a discrete-event simulator of approximately 1,400 lines that models the CS, PIT and FIB of every node, an explicit multi-core run queue per compute pool, per-link store-and-forward transmission, the SVE as formalised in Eq. (4), and the DREP controller. The simulator and supporting experimental code are available in our GitHub repository.[1] Compute and SVE workloads share the explicit multi-core run queue of each compute pool, so verification occupancy directly affects the service rate as modelled in Eq. (4). Link queueing uses the M/D/1

[1] https://github.com/mehbub160/SLED-NDN

waiting term of Eq. (3). Requests still in flight when the horizon expires are charged a lower-bound delay and scored as deadline misses rather than discarded, since discarding them would censor precisely the slowest traffic and make an overloaded configuration appear better than a lightly loaded one.

Each configuration point is the mean of 8 independent replications of 14 s of simulated time, with the first 15% discarded as transient. Error bars in the figures are Student-t 95% confidence half-widths.

Comparisons are paired: replication s uses the same seed for every architecture, so the arrival process, catalogue and topology are identical within a replication. Differences between architectures are therefore tested using paired statistics. This pairing reduces between-seed workload variance, which is particularly important near saturation. Figures report unpaired confidence intervals for readability, whereas comparative significance tests use the paired differences.

5.2 Cryptographic measurements

All host cryptographic costs are measured rather than assumed, in a batched harness on the host CPU. The benchmarks were run on an Intel i7 system with 16 GB RAM under Linux, using OpenSSL, for RSA-2048 (PKCS #1 v2.2 [47]), Ed25519 [31, 48] and HMAC-SHA256 [49]. Two biases are removed: timer overhead, by timing an inner batch and dividing, and the constant Python to OpenSSL foreign function interface crossing cost, an artefact of the harness language rather than of the primitive, measured at 1.08 μs per call and subtracted. Hash and message authentication code operations use thin C wrappers and receive no correction. Per-tier latencies are obtained by applying the scaling factors in Table 1; these factors are modelling estimates rather than measurements on the corresponding hardware. The BLAKE2s entry in Table 2 reports the cost for a 64-byte input; Merkle leaf nodes hash variable-length packet payloads and inner nodes hash pairs of 32-byte digests, so the per-packet τ_{sec} values derived from Eq. (6) use input-length-adjusted hash costs and may differ slightly from a naive product of the tabulated figure and $\lceil \log_2 B \rceil$.

5.3 Simulation parameters and workload

Table 1 lists the system and principal workload parameters used in the evaluation. The workload is synthetic, with Poisson arrivals over a Zipf catalogue, log-normal object sizes, and per-class compute demands. The service class is encoded in the name rather than assigned independently to each request, following Eq. (1).

Table 1. Simulation parameters. The cloud value denotes the contracted compute slice available to the service, which is dynamically managed by Drep.

Symbol	Parameter	Value
$\lvert\mathcal{E}\rvert$	Edge nodes	16
$\lvert\mathcal{A}\rvert$	Fog nodes	4
m_j, f_j	Edge cores, clock	4, 1.5 GHz
m_a, f_a	Fog cores, clock	6, 2.6 GHz
m_c, f_c	Contracted cloud cores, clock	24, 3.4 GHz
γ	Lendable fraction of cloud slice	0.5
M	Catalogue size	5,000
α	Zipf exponent	0.9
ϕ	Fraction of names that are functions	0.45
	CS capacity, edge / fog	2% / 8%
D_k	Deadlines by class	30 / 120 / 500 ms
v_k	Utility weights by class	3.0 / 1.5 / 1.0
ω_k	Median cycles by class	8 / 40 / 180 Mcycles
	Consumer to edge RTT	1 to 4 ms
	Edge to fog RTT	8 to 18 ms
	Fog to cloud RTT	36 to 70 ms
B	Amortisation window	32
ϵ	Audit sampling rate	0.05
D^{def}	Deferral eligibility threshold	60 ms
T_c	Drep control epoch	0.5 s
θ_{hi}, θ_{lo}	Drep thresholds	0.75, 0.35
ζ_p, ζ_r	Drep gains	0.60, 0.40
κ	Deadline-miss decay	3.0
c_e, c_c, c_s	Cost: edge, cloud, security	18, 12, 250
c_w, c_m, c_b	Cost: WAN, memory, burst	45, 0.08, 16
μ	Edge weight in ONU	0.6
	Crypto scaling, edge/fog/cloud/IoT	6.5 / 1.0 / 0.75 / 9.0

5.4 Baselines and ablations

Six systems are compared. *IP-Edge* is a host-centric stack with a TLS 1.3 style record layer [50], an application-layer result cache at the edge sized identically to the NDN edge Content Store, no Interest aggregation and no deadline-aware placement. *NDN* is vanilla Named Data Networking: CS, PIT, FIB, LRU (least recently used) replacement, per-packet RSA-2048 signatures verified at three on-path points, and no in-network computation, so named functions execute at the origin. *Edge-NDN-RSA* is the Sled-Ndn datapath with legacy NDN cryptography.

SLED-NDN-NoAmort is the Sled-Ndn datapath with Ed25519 but without Merkle amortisation, without the Ett and without deferral, so comparing it against Edge-NDN-RSA is intended to isolate the contribution of the primitive choice; this attribution holds only if the number of verification points is the same in

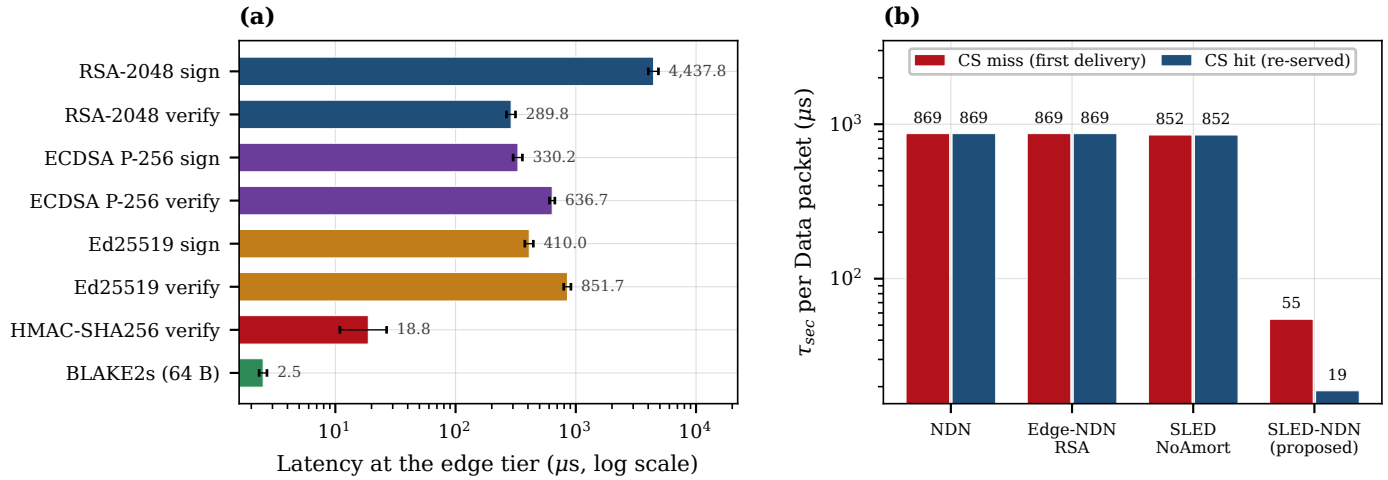

Figure 2. (a) Cryptographic costs measured on the host and scaled to the edge tier (logarithmic axis) and (b) the resulting per-packet verification latency for the evaluated architectures.

both configurations (see Section 6). *Edge-NDN-NoSec* performs no cryptographic work, provides no content authenticity, and is included solely as an upper bound on what any secure design could achieve. *SLED-NDN* is the full proposal.

Giving IP-Edge an application-layer cache is deliberate: without it the comparison would measure the presence of caching rather than the difference between in-network and endpoint caching, overstating the NDN advantage. The four edge-NDN variants share the same placement and datapath structure and differ only in their security configuration, making the resulting security ladder interpretable.

6 Results and Analysis

6.1 Cost of the cryptographic primitives

Table 2 summarises what each primitive costs per operation at the server, edge and IoT tiers, and Figure 2(a) plots the edge figures on a logarithmic axis. The server column is measured on the host CPU; the edge and IoT columns are scaled estimates (Table 1). These costs were measured because the argument of the paper turns on which operation lies on the recurring path, which cannot be settled analytically. At the edge tier, RSA-2048 signing costs 4,437.8 µs against 410.0 µs for Ed25519. Verification behaves differently: 289.8 µs for RSA-2048 against 851.7 µs for Ed25519, so Ed25519 is slower for this operation in the evaluated implementation. Composing the measurements into per-packet τ_{sec} through Eq. (5) to Eq. (7) gives 869.5 µs for legacy per-packet RSA (three verifications of 289.8 µs each) and 851.7 µs for per-packet Ed25519 (a single verification), a ratio of 1.02; this near-equality reflects three inexpensive RSA verifications costing about as much as one Ed25519 verification, not equal per-operation cost. SLED-NDN reduces the legacy figure to 54.6 µs on first delivery and 18.8 µs on re-serve, which are reductions of 15.9 × and 46.3 × relative to the legacy configuration, as Figure 2(b) depicts.

Table 2. Cryptographic primitive costs per operation across tiers. The Server column was measured on the host CPU; the Edge and IoT columns are derived from it with the scaling factors of Table 1 and are estimates, not measurements.

Primitive	Op.	Server (µs)	Edge (µs)	IoT (µs)
RSA-2048	sign	682.7	4,437.8	6,144.6
RSA-2048	verify	44.6	289.8	401.3
ECDSA P-256	sign	50.8	330.2	457.2
ECDSA P-256	verify	98.0	636.7	881.6
Ed25519	sign	63.1	410.0	567.7
Ed25519	verify	131.0	851.7	1,179.2
HMAC-SHA256	verify	2.9	18.8	26.0
BLAKE2s (64 B)	hash	0.4	2.5	3.5
AES-128-GCM	open	1.0	6.3	8.7
X25519	exchange	36.3	236.0	326.7

A Data packet is signed once but validated once per verification point the policy specifies, so substituting the curve improves the operation that does not recur (signing) while making the recurring one (verification) slower per operation. The size of the effect therefore depends on the validation policy as much as on the primitive; Section 7 returns to the mechanism.

6.2 Deadline satisfaction and service delay

Figure 3 shows the deadline satisfaction ratio and average service delay of Eq. (14) against offered load, and Table 3 summarises the nominal point. Load

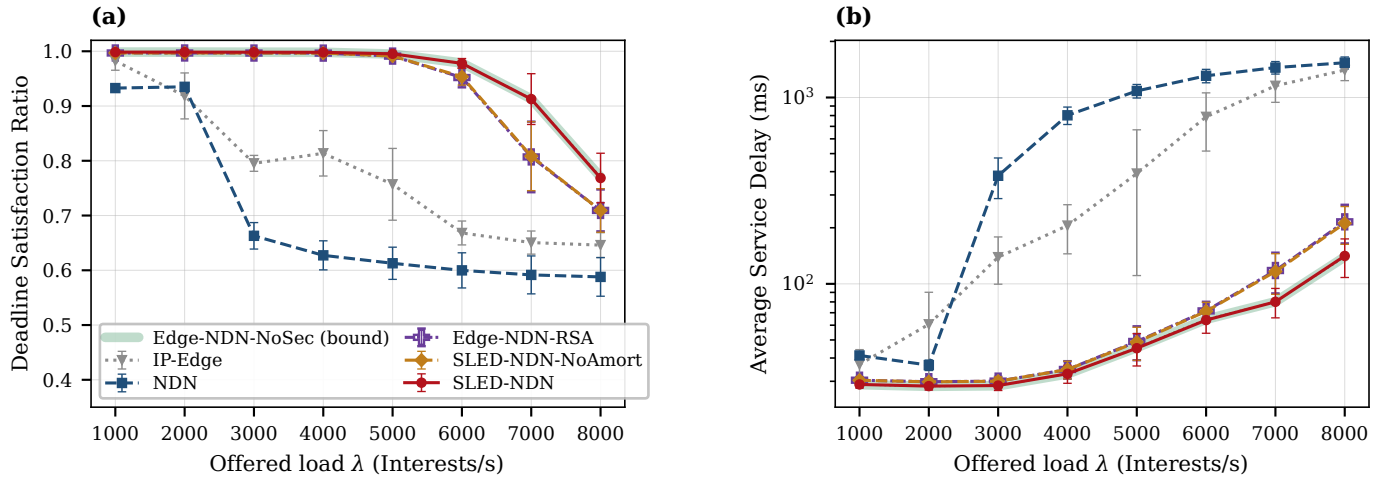

Figure 3. (a) Deadline satisfaction ratio and (b) average service delay against offered load, for all six systems. Error bars are 95% confidence half-widths over 8 seed-matched replications. Edge-NDN-NoSec is drawn as a wide translucent band because it coincides with SLED-NDN on both metrics; it is an insecure upper bound rather than a competing design.

Table 3. Headline comparison at the nominal operating point ($\lambda = 6{,}000$ Interests/s). Entries are means over independent replications; parentheses show the half-width of the 95% confidence interval. ONU is the overall network utility per 10^3 requests, and 99th pct. is the 99th percentile of service delay.

Architecture	DSR	ASD (ms)	99th pct. (ms)	τ_{sec} (μs)	CS hit ratio	ONU
IP-Edge	0.668 (0.0218)	787.5 (271.6)	4,424.9	51.4	0.276	417.4
NDN	0.600 (0.0321)	1,307.6 (108.2)	6,260.5	4,572.0	0.227	−233.2
Edge-NDN-RSA	0.952 (0.0179)	71.6 (8.5)	502.3	7,150.5	0.266	−392.1
Edge-NDN-NoSec†	0.978 (0.0080)	64.5 (9.4)	473.8	0.0	0.268	701.8
SLED-NDN-NoAmort	0.954 (0.0153)	71.2 (8.4)	502.1	2,191.5	0.267	352.8
SLED-NDN (proposed)	**0.978** (0.0088)	**63.9** (9.6)	474.3	258.4	0.269	**663.3**

†Provides no content authenticity whatsoever; included only as an upper bound on what any secure design can attain.

was swept because the coupling of Eq. (4) predicts that the architectures separate only as spare capacity disappears. At $\lambda = 6,000$ Interests per second, SLED-NDN attains DSR $= 0.978$, against 0.952 for Edge-NDN-RSA, 0.600 for vanilla NDN and 0.668 for IP-Edge. The paired difference between SLED-NDN and Edge-NDN-RSA is $+0.0259 \pm 0.0102$ with $p < 0.001$; the corresponding differences against vanilla NDN and IP-Edge are $+0.3781 \pm 0.0327$ and $+0.3098 \pm 0.0170$, respectively, both with $p < 0.001$. Mean service delay is 63.9 ms against 71.6 ms, 1,307.6 ms and 787.5 ms, which are reductions of 10.7 %, 95.1 % and 91.9 %.

The two families of baseline degrade for different reasons, and conflating them would misattribute the result. NDN and IP-Edge lose ground because of placement: vanilla NDN has no in-network computation, so every named function executes at the origin across a wide-area path, and IP-Edge has no Interest aggregation, so concurrent identical requests each consume a full execution. Edge-NDN-RSA and SLED-NDN-NoAmort share SLED-NDN's placement logic exactly and lose ground only because of verification: at high load their SVE occupancy contracts $\tilde{f}_i$ in Eq. (4) and starves the compute pool they depend on. Interpolating where each curve crosses a 0.95 service objective, SLED-NDN sustains $\lambda = 6,428$ Interests per second against 6,014 for Edge-NDN-RSA, an increase of 6.9 % in serviceable load attributable to the security construction alone.

Two qualifications are needed. The margin against Edge-NDN-RSA at the nominal point is modest in absolute terms because both systems still clear 95% there and the overhead is absorbed by slack; the separation widens as that slack disappears, and at $\lambda = 7,000$ the figures are 0.913 against 0.807, a gap of 10.6 points with non-overlapping unpaired intervals. Second, SLED-NDN's deadline attainment is not distinguishable from the insecure upper bound, at -0.0006 ± 0.0018 with $p = 0.488$. We read this as the intended outcome rather than an absence of effect: the target was to make content authenticity affordable, and at this load it costs nothing measurable in deadline terms.

Figure 4 disaggregates the nominal point by class,

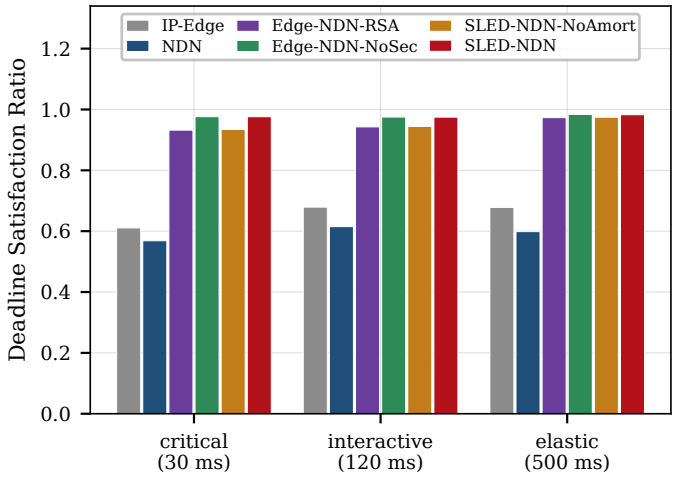

Figure 4. Per-class deadline satisfaction ratio at $\lambda = 6,000$. The critical class carries a 30 ms budget, against which a multi-millisecond per-request verification cost is a substantial fraction, so it is where verification overhead is least affordable.

isolating where the overhead binds. The critical class attains 0.976 under SLED-NDN against 0.932 under Edge-NDN-RSA, while the elastic class, whose 500 ms budget absorbs the same absolute overhead comfortably, separates far less. That the effect concentrates in the tightest class is what Eq. (2) predicts, and checks that the mechanism acts where the model says it should.

6.3 Network utility

Figure 5 reports overall network utility with its edge and cloud components, computed from Eq. (11) to Eq. (13). Utility was measured because deadline satisfaction alone cannot express whether the verification work was worth performing; the two are not redundant, since the soft attainment score of Eq. (10) credits a narrowly missed deadline with partial value whereas DSR counts it as a failure.

SLED-NDN achieves ONU $= 663.3$ per 10^3 requests, recovering 94.5 % of the insecure upper bound's 701.8. SLED-NDN-NoAmort reaches 352.8, so the amortisation structure adds 310.5 utility units over primitive substitution alone, with a paired difference of $+310.5 \pm 9.4$ and $p < 0.001$. Edge-NDN-RSA reaches -392.1, which is negative, a paired shortfall of $1,055.4 \pm 25.9$ utility units against SLED-NDN with $p < 0.001$. This ordering of steps also qualifies the deadline-based reading of Section 7: the step from Edge-NDN-RSA to SLED-NDN-NoAmort is nearly inert for DSR and ASD (0.952 to 0.954, and 71.6 ms to 71.2 ms) but is not inert for priced verification, where it raises ONU from -392.1 to 352.8, a larger gain than the subsequent amortisation step.

Under a per-packet RSA-2048 trust model at this load and this price of security, the service consumes more value in verification than it delivers in met deadlines, even though its raw deadline satisfaction of 0.952 appears acceptable. A latency-only evaluation would not surface that.

Because c_s is a policy parameter, Figure 6 sweeps it over multiples of its nominal value. The ordering of SLED-NDN above SLED-NDN-NoAmort above Edge-NDN-RSA is invariant across the range, as expected since the architectures differ monotonically in Θ; what varies is where the per-packet designs cross into negative utility. At $c_s = 0$ they move closer together and differ mainly through the second-order effect of SVE occupancy on $\tilde{f}_i$, confirming that the larger utility gap at the nominal setting is driven primarily by the priced verification term rather than by a modelling artefact.

Figure 7 shows verification cost and its share of the delay budget against load. The absolute cost per request is close to flat for each architecture, as Eq. (6) implies, while its share falls as queueing grows. The share is therefore a poor proxy for importance: verification matters most where it looks smallest, because by then it has already contracted the capacity that produced the queueing.

6.4 Caching, popularity skew and deadline tightness

Figure 8 sweeps Content Store capacity from 0.5 % to 16.0 % of the catalogue, over which the edge hit ratio rises from 0.191 to 0.399. Figure 9 sweeps the Zipf exponent from 0.5 to 1.3, raising the hit ratio from 0.046 to 0.725. These sweeps test whether caching, the mechanism NDN is usually credited with, changes the security trade-off. It does, in a direction easy to get backwards: both sweeps move the architectures the same way, but the gap between SLED-NDN and the per-packet designs widens as hit ratio rises. The reason is specific to computational NDN, in that a cache hit eliminates the computation but not the verification. Under a per-packet model every hit still costs a full public-key verification, so the service time of a hit is bounded below by τ_{sec}; under LSAV a hit costs one message authentication code, so nearly the full value of caching is realised.

Figure 10 scales all deadlines by a factor η, with $\eta = 1$ the nominal configuration. As η grows the architectures converge, because a sufficiently loose

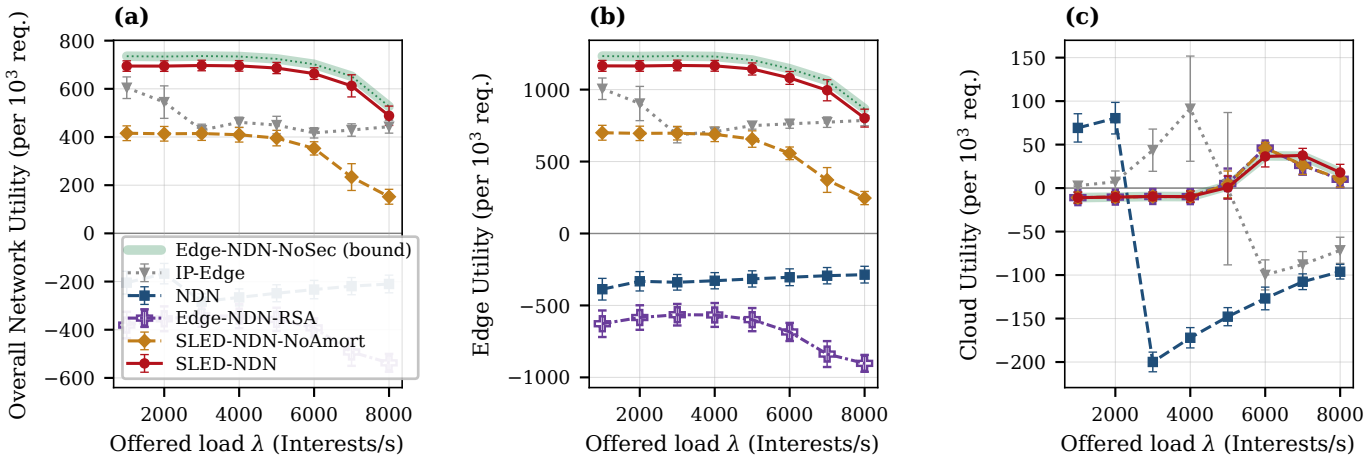

Figure 5. (a) Overall network utility, (b) edge utility and (c) cloud utility against offered load, reported per 10^3 requests. Per-packet RSA-2048 verification drives overall utility negative across the entire load range.

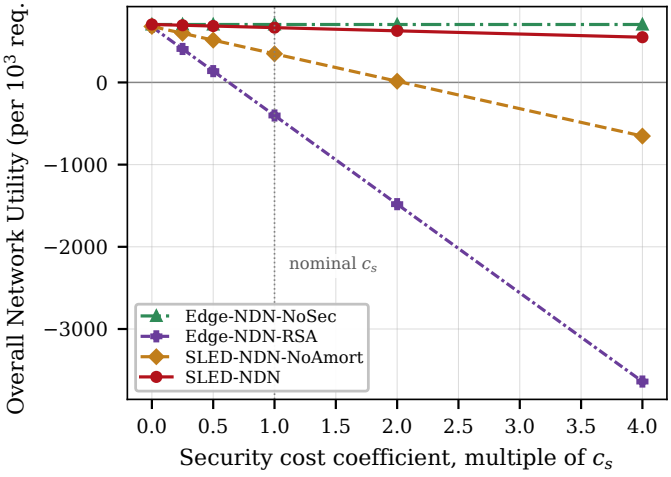

Figure 6. Overall network utility against the security cost coefficient c_s, expressed as a multiple of its nominal value. The ranking of the architectures is invariant; the crossing point into negative utility is not.

deadline absorbs any verification cost. As η falls below one they separate sharply. The verification budget therefore matters precisely in the regime that motivates edge computing in the first place.

6.5 Effect of the amortisation window

Figure 11 sweeps the amortisation window B, the parameter controlling how many critical-path verifications a delivered object requires. Panel (a) plots the analytical curve obtained by substituting the measured primitives into Eq. (6); panel (b) the simulated deadline satisfaction and utility. The analytical curve falls from 797.3 μs at $B = 1$ to 41.4 μs at $B = 128$, but flattens well before the end of the sweep. This follows directly from Eq. (6): the $\frac{1}{B}t_{\text{ver}}$ term is extinguished while the Merkle-path and token terms are not, so the curve approaches an irreducible floor. Simulated utility rises steeply to approximately $B = 32$ and then plateaus. Larger windows are not free in a deployment, because a producer must accumulate B packets before it can sign, which adds up to B packet intervals of publication latency and couples the security parameter to the producer's output rate. The knee near $B = 32$ is therefore a practical operating point for this configuration rather than an artefact of where the sweep was stopped.

6.6 Provisioning behaviour under a load surge

Figure 12 shows Drep under a flash-crowd profile peaking at 9,000 Interests per second, run because steady-state sweeps cannot show whether the controller of Eq. (15) reacts at the right time or oscillates. As edge utilisation crosses θ_{hi} the controller dedicates burst workers, exhausting the lendable pool of 12 at the surge peak; the shared pool contracts by the same amount, since capacity is moved rather than created. As demand subsides, hysteresis holds the workers for N_h epochs before reclamation returns them. Over the run there are 18 provisioning and 34 reclamation events, with 208 worker-seconds lent and 17 reclaimed. Reclamation events outnumber provisioning events because provisioning grants in larger blocks while reclamation is deliberately gradual, an asymmetry that suppresses oscillation at the cost of some retained idle capacity.

Panel (b) shows edge utilisation resting near zero away from the surge, which resembles idle hardware but is Dacsir behaving as specified. Off-peak, the fog tier is the cheapest feasible site for every class, since its higher clock means a named function consumes fewer core-seconds than at the edge and, unlike the cloud, it incurs no WAN transport cost. The edge is held

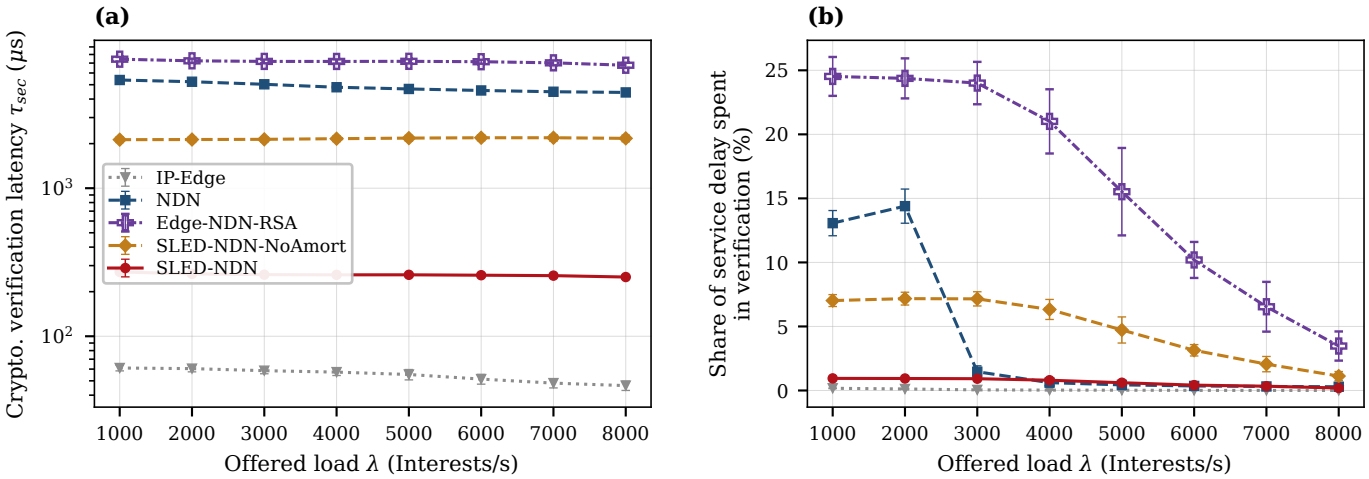

Figure 7. (a) Mean per-request verification latency and (b) the share of total service delay that verification consumes, against offered load. Edge-NDN-NoSec is omitted because $\tau_{\text{sec}} \equiv 0$ by construction.

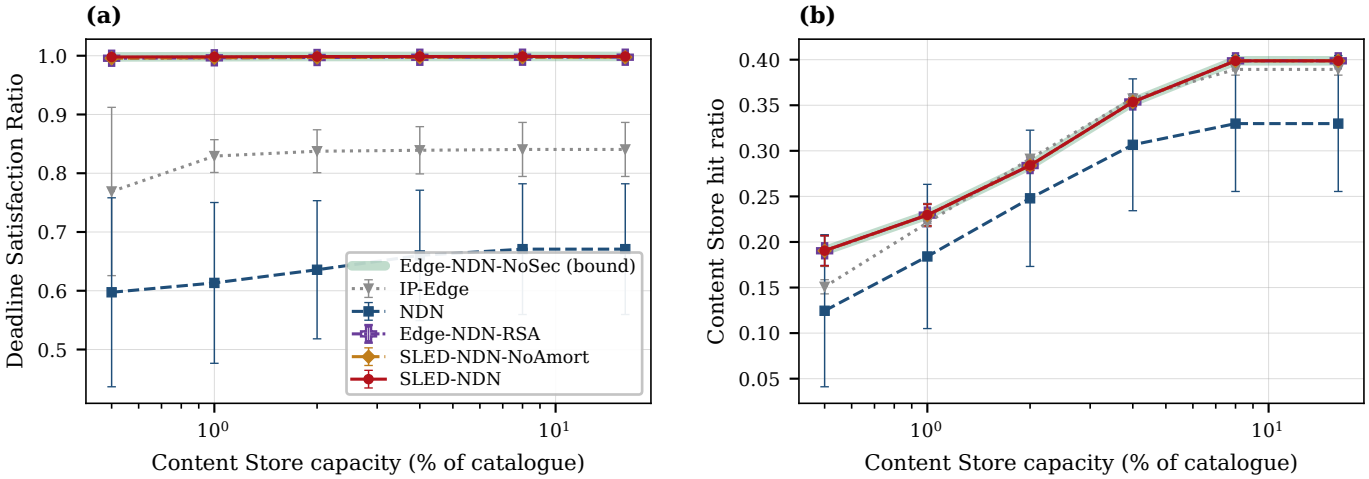

Figure 8. (a) Deadline satisfaction ratio and (b) Content Store hit ratio against CS capacity expressed as a fraction of the catalogue. The gap between SLED-NDN and the per-packet designs widens with hit ratio, because a cache hit removes computation but not verification.

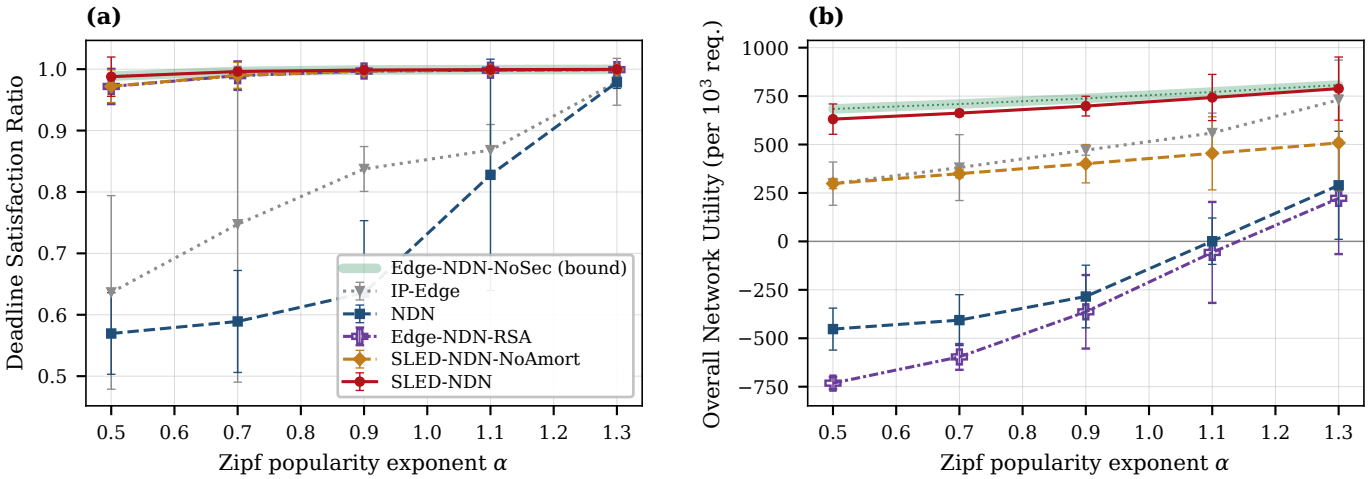

Figure 9. (a) Deadline satisfaction ratio and (b) overall network utility against the Zipf popularity exponent α. Higher skew raises the hit ratio and therefore amplifies the difference between per-packet verification and LSAV re-validation.

in reserve and recruited once the fog tier saturates, which is what a cost-minimising placement should do under the prices of Table 1. Edge share remains high throughout, at 93.1 % at the nominal load, because

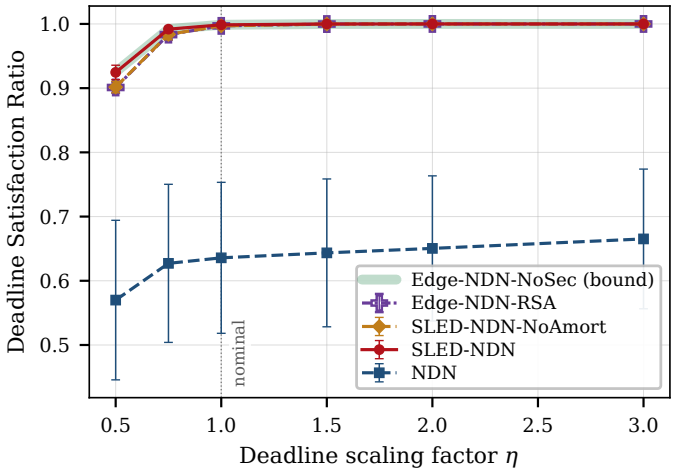

Figure 10. Deadline satisfaction ratio against a global deadline scaling factor η, where $\eta = 1$ is the nominal configuration of Table 1. Architectures converge under loose deadlines and separate under tight ones.

Content Store hits are served at the edge regardless of where the computation ran. An operator wanting the edge loaded earlier would lower c_e relative to c_c; the mechanism does not change, only the price it responds to.

6.7 Distribution of service delay

Figure 13 gives the empirical distribution of service delay at $\lambda = 6,000$ with the three class deadlines marked, examined because a mean cannot distinguish a uniform shift from a stretched tail and the two have different reliability consequences. SLED-NDN's 99th percentile is 474.3 ms against 502.3 ms for Edge-NDN-RSA. The shape matters more than the mean: a per-packet design's distribution is shifted by a roughly constant verification cost at low percentiles but is stretched at high percentiles, because SVE occupancy and compute queueing compound through Eq. (4) rather than adding. For a service with a reliability objective expressed as a tail percentile, which is the usual case, this compounding is the quantity that matters.

7 Discussion, Limitations, and Future Directions

7.1 Why security overhead moves the saturation point

An additive account of τ_{sec} predicts a delay curve displaced upward and a saturation point unchanged. The measured behaviour is different because Eq. (4) makes verification consume the capacity that fixes where saturation occurs, so the effective service rate falls as verification cost rises. This explains why SLED-NDN sustains 6.9 % more serviceable load than Edge-NDN-RSA despite sharing its placement logic exactly, and why the curves diverge primarily near saturation: below the knee, spare capacity absorbs the contraction; above it, that slack is exhausted. The implication plausibly extends beyond NDN. Any secure edge system in which the verifying node is also the executing node may inherit this coupling, and any evaluation charging verification as pure latency will understate its cost under load.

7.2 Why caching interacts differently with the two designs

One might expect caching to help the expensive designs most, since it removes the computation they can least afford. Under a per-packet model, every hit still incurs a full public-key verification cost, so the benefit of a hit is limited by the remaining verification overhead; under LSAV, a hit costs only one message authentication code, so nearly the full benefit of caching is realised. LSAV changes that limit by making the re-serve path cheaper: a hit costs 18.8 μs against 54.6 μs on first delivery (about 2.9 times less) and 46.3 $\times$ less than legacy per-packet verification. The ETT, conceptually the least elaborate component, plausibly contributes disproportionately, although the ablations do not isolate it from Merkle amortisation and deferral. The same reasoning applies to any content-reuse mechanism paired with per-object authentication.

7.3 Why the primitive substitution barely changes deadline satisfaction

The measurements in Figure 2(a) supply part of the mechanism. RSA-2048 verification with $e = 65537$ requires a small number of modular multiplications, whereas Ed25519 verification requires a double-scalar multiplication. In the evaluated implementation, RSA-2048 therefore has a lower verification cost than Ed25519 (289.8 against 851.7 μs at the edge tier) despite its higher signing cost. Since a Data packet is signed once but verified at each configured verification point, replacing the primitive alone leaves the per-packet verification time in the same range (869.5 against 851.7 μs), and the deadline metrics change little (Table 3), although the mean τ_{sec} and the priced utility do change. This finding is specific to the evaluated configurations, since the ratio depends on the library, exponent, platform and hardware acceleration. What generalises is the methodological point: per-operation microbenchmarks rank primitives, but only a system model identifies which operation lies on the recurring path.

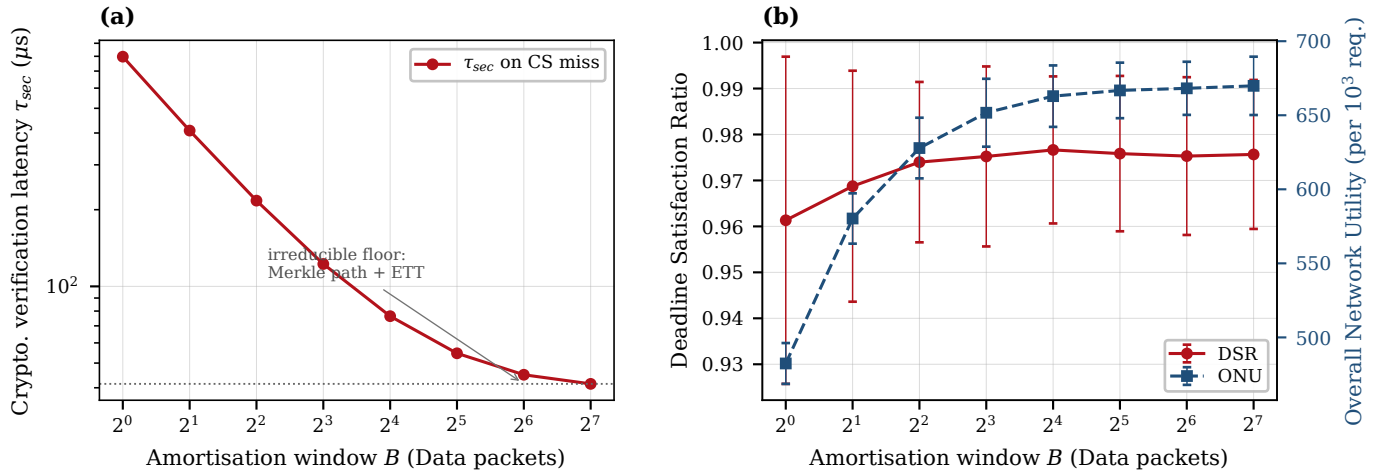

Figure 11. (a) Analytical per-packet τ_{sec} against the amortisation window B, obtained from the measured primitives through Eq. (6). (b) Simulated deadline satisfaction ratio and overall network utility against B, showing a knee near $B = 32$.

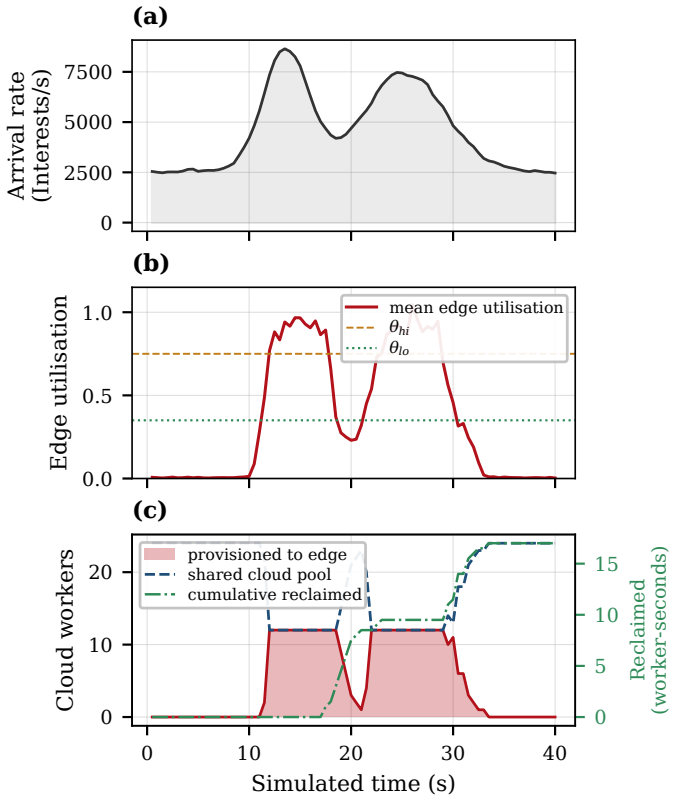

Figure 12. Drep under a flash crowd. (a) Offered arrival rate. (b) Mean edge utilisation against the provisioning and reclamation thresholds. (c) Cloud workers dedicated to the edge, the contracting shared pool, and cumulative reclaimed worker-seconds.

7.4 Why the amortisation window shows diminishing returns

Equation (6) contains one term that decays in B, one that grows logarithmically and one that is constant, so beyond the point where $\frac{1}{B}t_{\text{ver}}$ falls below the Merkle-path and token terms, further increases buy little while lengthening publication latency, since B packets must accumulate before a root can be signed. The knee is set by the ratio of signature cost to hash and token cost, all measured quantities, and moves if any of them changes.

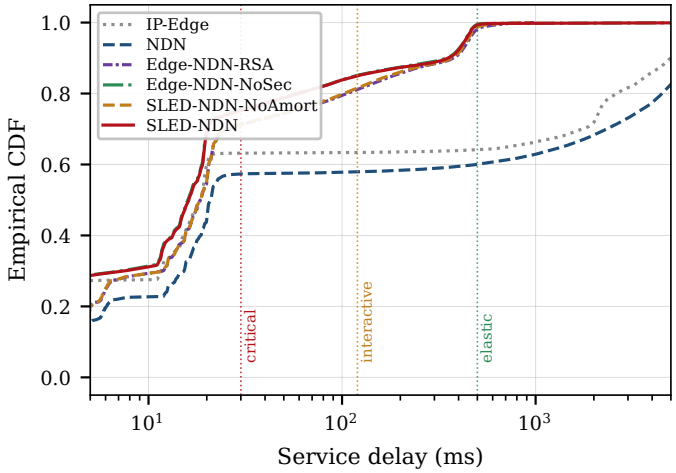

Figure 13. Empirical cumulative distribution of service delay at $\lambda = 6,000$. Dotted vertical lines mark the three class deadlines. Per-packet verification stretches the upper tail rather than shifting the whole distribution.

7.5 What Drep contributes and what it does not

Drep absorbs workload pressure that placement alone cannot, and because measured utilisation includes Sve occupancy, an expensive trust model triggers provisioning without a special case. It does not remove the coupling: burst capacity sits behind a wide-area path, so it serves the elastic and interactive classes and is largely unavailable to the critical class. Elasticity compensates at the margin; it does not substitute for reducing verification frequency.

7.6 Limitations

The boundary conditions of these results should be stated explicitly.

The system-level results are simulation results, not deployment measurements. Cryptographic primitive costs were measured on a host CPU, but the per-tier scaling factors in Table 1 are estimates, and measurement on actual edge hardware could shift the absolute values; the relative ordering is more robust, following from operation counts rather than scaling constants. No claim is made that the system is production ready.

Compute queueing and SVE occupancy are simulated explicitly through the shared run queue, whereas link queueing uses the M/D/1 mean of Eq. (3). This is adequate here because links never exceed roughly 15% utilisation in any reported configuration, but the model would understate delay in a link-limited deployment and captures only the mean rather than the full waiting-time distribution, so the tails in Figure 13 are likely to be understated with respect to link effects.

The security boundaries stated in Section 4.5 are limitations as well as assumptions. Token-based re-validation holds only inside one administrative domain sharing K_e; a federated deployment must re-validate at each boundary and pay Eq. (6) again, and no formal bound on the adversary's advantage as a function of ϵ is claimed.

The evaluation covers 16 edge nodes in a single region; behaviour at metropolitan scale, where FIB size and inter-domain routing become material, is not addressed. Node failures, mobility and adversarial load are not modelled: an Interest-flooding adversary [28, 29] would interact with the SVE in a way this model does not capture, and plausibly adversely, since forcing cache misses forces first-delivery verification. Demand is synthetic, which is standard in this literature and permits controlled sweeps, but real demand exhibits diurnal structure, correlated bursts and name locality that the model reproduces only in the surge profile of Figure 12. Finally, the cost coefficients in Eq. (11) and Eq. (12) are policy choices made to render the terms commensurable rather than derived from an operator's economics; Figure 6 shows the ranking is invariant to c_s, but utility magnitudes should be read as relative.

7.7 Future directions

Several extensions follow from the model developed here. Post-quantum security is the most pressing: module-lattice signatures [51] are considerably larger than Ed25519 signatures, so a per-packet signature adds a substantial number of bytes to every Data packet, and their verification cost on constrained tiers remains to be measured; amortisation may therefore move from an optimisation to a precondition. This framework is a natural setting for that analysis, and the Merkle construction composes directly with hash-based schemes [41, 52]. Mobility interacts with the Edge Trust Token, since a consumer moving between trust domains invalidates cached tokens and forces re-verification; token hand-off for vehicular and mobile settings [53] remains open. Learned or federated caching could replace the DAP heuristic of Eq. (22) with a policy predicting joint popularity and urgency across edge nodes without centralising the demand trace, subject to the security and privacy constraints that MEC deployments are known to face at the edge infrastructure layer [44].

Adversarial evaluation is needed against Interest flooding and cache poisoning, since forcing cache misses attacks the amortisation benefit directly. Testbed validation on real constrained hardware would replace the scaling factors of Table 1 with measurements, and formal analysis of the deferred path, bounding the adversary's advantage as a function of ϵ and the audit backlog, would place that mechanism on a firmer footing than the sampling argument given here. Extending the resource account to confidentiality rather than authenticity alone would require integrating encryption and key management operations into the same edge compute budget, building on the foundation established for efficient cryptographic data validation at resource-limited edge nodes [34].

8 Conclusion

Secure edge computing must account for the fact that verification and application execution compete for the same limited computational capacity. This work formalised this coupling by placing the Security Verification Engine inside the edge compute budget and developed SLED-NDN to incorporate verification cost into placement, utility optimisation, and verification frequency. Using cryptographic costs measured on a host CPU and scaled to each tier within a discrete-event simulation, the evaluation showed that, in the studied configurations, reducing verification frequency had a greater impact on deadline satisfaction and service delay than changing the signature primitive alone. At 6,000 Interests per

second, SLED-NDN achieved a deadline satisfaction ratio of 0.978 and recovered 94.5 % of the utility of the insecure upper bound, while per-packet RSA-2048 verification resulted in negative utility. These results indicate that security overhead should be treated as a first-class resource cost in reliable edge-cloud computing and jointly considered with workload placement and deadline requirements.

Data Availability Statement

The data that support the findings of this study are openly available. The discrete-event simulator, cryptographic benchmarking code, and experimental results are publicly accessible at https://github.com/mehbub160/SLED-NDN under an open-source licence.

Funding

This work was supported without any funding.

Conflicts of Interest

Mehbub Alam served as an Editorial Board Member of the *Journal of Reliable and Secure Computing* at the time of manuscript submission. To ensure the integrity of the peer-review process, Mehbub Alam was not involved in the editorial handling, peer review, or decision-making process for this manuscript, which was handled independently by another editor. The remaining authors declare no conflicts of interest.

AI Use Statement

The author declares that ChatGPT was used for language editing and grammar refinement of the manuscript. The authors have carefully reviewed, revised, and verified the AI-assisted output and take full responsibility for the content of the manuscript.

Ethical Approval and Consent to Participate

Not applicable.

References

[1] Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge computing: Vision and challenges. *IEEE Internet of Things Journal, 3*(5), 637-646. [CrossRef]

[2] Xiao, Y., Jia, Y., Liu, C., Cheng, X., Yu, J., & Lv, W. (2019). Edge computing security: State of the art and challenges. *Proceedings of the IEEE, 107*(8), 1608-1631. [CrossRef]

[3] Roman, R., Lopez, J., & Mambo, M. (2018). Mobile edge computing, Fog et al.: A survey and analysis of security threats and challenges. *Future Generation Computer Systems, 78*, 680-698. [CrossRef]

[4] Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M. F., Briggs, N. H., & Braynard, R. L. (2009, December). Networking named content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies* (pp. 1-12). [CrossRef]

[5] Zhang, L., Afanasyev, A., Burke, J., Jacobson, V., Claffy, K., Crowley, P., ... & Zhang, B. (2014). Named data networking. *ACM SIGCOMM Computer Communication Review, 44*(3), 66-73. [CrossRef]

[6] Yi, C., Afanasyev, A., Moiseenko, I., Wang, L., Zhang, B., & Zhang, L. (2013). A case for stateful forwarding plane. *Computer Communications, 36*(7), 779-791. [CrossRef]

[7] Sifalakis, M., Kohler, B., Scherb, C., & Tschudin, C. (2014). An information centric network for computing the distribution of computations. In *Proceedings of the 1st ACM Conference on Information-Centric Networking (ICN)* (pp. 137-146). [CrossRef]

[8] Krol, M., & Psaras, I. (2017, September). NFaaS: Named function as a service. In *Proceedings of the 4th ACM Conference on Information-Centric Networking* (pp. 134-144). [CrossRef]

[9] Krol, M., Habak, K., Oran, D., Kutscher, D., & Psaras, I. (2018, September). Rice: Remote method invocation in icn. In *Proceedings of the 5th ACM Conference on Information-Centric Networking* (pp. 1-11). [CrossRef]

[10] Krol, M., Mastorakis, S., Oran, D., & Kutscher, D. (2019, September). Compute first networking: Distributed computing meets ICN. In *Proceedings of the 6th ACM Conference on Information-Centric Networking* (pp. 67-77). [CrossRef]

[11] Scherb, C., Grewe, D., Wagner, M., & Tschudin, C. (2018, January). Resolution strategies for networking the IoT at the edge via named functions. In *2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)* (pp. 1-6). IEEE. [CrossRef]

[12] Amadeo, M., Ruggeri, G., Campolo, C., Molinaro, A., Loscri, V., & Calafate, C. T. (2019). Fog computing in IoT smart environments via named data networking: A study on service orchestration mechanisms. *Future internet, 11*(11), 222. [CrossRef]

[13] Kondo, D., Ansquer, T., Tanigawa, Y., & Tode, H. (2024). Resource breadcrumbs: Discovering edge computing resources over Named Data Networking. *IEEE Transactions on Network and Service Management, 21*(3), 3305-3316. [CrossRef]

[14] Zhang, Z., Yu, Y., Zhang, H., Newberry, E., Mastorakis, S., Li, Y., ... & Zhang, L. (2018). An overview of security support in named data networking. *IEEE Communications Magazine, 56*(11), 62-68. [CrossRef]

[15] Yu, Y., Afanasyev, A., Clark, D., Claffy, K. C., Jacobson, V., & Zhang, L. (2015, September). Schematizing trust

in named data networking. In *proceedings of the 2nd ACM Conference on Information-Centric Networking* (pp. 177-186). [CrossRef]

[16] Ahlgren, B., Dannewitz, C., Imbrenda, C., Kutscher, D., & Ohlman, B. (2012). A survey of information-centric networking. *IEEE Communications Magazine, 50*(7), 26-36. [CrossRef]

[17] Xylomenos, G., Ververidis, C. N., Siris, V. A., Fotiou, N., Tsilopoulos, C., Vasilakos, X., ... & Polyzos, G. C. (2013). A survey of information-centric networking research. *IEEE communications surveys & tutorials, 16*(2), 1024-1049. [CrossRef]

[18] Zhang, M., Luo, H., & Zhang, H. (2015). A survey of caching mechanisms in information-centric networking. *IEEE Communications Surveys & Tutorials, 17*(3), 1473-1499. [CrossRef]

[19] Javaheri, A., Bohlooli, A., & Jamshidi, K. (2024). Enhancing computation reuse efficiency in ICN-based edge computing by modifying content store table structure. *Computing, 106*(9), 2949-2969. [CrossRef]

[20] Mao, Y., You, C., Zhang, J., Huang, K., & Letaief, K. B. (2017). A survey on mobile edge computing: The communication perspective. *IEEE communications surveys & tutorials, 19*(4), 2322-2358. [CrossRef]

[21] Mach, P., & Becvar, Z. (2017). Mobile edge computing: A survey on architecture and computation offloading. *IEEE communications surveys & tutorials, 19*(3), 1628-1656. [CrossRef]

[22] Ndikumana, A., Tran, N. H., Ho, T. M., Han, Z., Saad, W., Niyato, D., & Hong, C. S. (2019). Joint communication, computation, caching, and control in big data multi-access edge computing. *IEEE Transactions on mobile Computing, 19*(6), 1359-1374. [CrossRef]

[23] Wang, C., Liang, C., Yu, F. R., Chen, Q., & Tang, L. (2017). Computation offloading and resource allocation in wireless cellular networks with mobile edge computing. *IEEE Transactions on Wireless Communications, 16*(8), 4924-4938. [CrossRef]

[24] Ren, J., Yu, G., He, Y., & Li, G. Y. (2019). Collaborative cloud and edge computing for latency minimization. *IEEE Transactions on Vehicular Technology, 68*(5), 5031-5044. [CrossRef]

[25] Liu, J. & Zhang, Q. (2018). Offloading schemes in mobile edge computing for ultra-reliable low latency communications. *IEEE Access, 6*, 12825-12837. [CrossRef]

[26] Tourani, R., Misra, S., Mick, T., & Panwar, G. (2018). Security, privacy, and access control in information-centric networking: A survey. *IEEE Communications Surveys & Tutorials, 20*(1), 566-600. [CrossRef]

[27] Nour, B., Khelifi, H., Hussain, R., Mastorakis, S., & Moungla, H. (2021). Access control mechanisms in named data networks: A comprehensive survey. *Acm computing Surveys (cSuR), 54*(3), 1-35. [CrossRef]

[28] Gasti, P., Tsudik, G., Uzun, E., & Zhang, L. (2013, July). DoS and DDoS in named data networking. In *2013 22nd International Conference on Computer Communication and Networks (ICCCN)* (pp. 1-7). IEEE. [CrossRef]

[29] Compagno, A., Conti, M., Gasti, P., & Tsudik, G. (2013, October). Poseidon: Mitigating interest flooding DDoS attacks in named data networking. In *38th annual IEEE conference on local computer networks* (pp. 630-638). IEEE. [CrossRef]

[30] Kumar, N., Singh, A. K., Aleem, A., & Srivastava, S. (2019). Security attacks in named data networking: A review and research directions. *Journal of Computer Science and Technology, 34*(6), 1319-1350. [CrossRef]

[31] Bernstein, D. J., Duif, N., Lange, T., Schwabe, P., & Yang, B.-Y. (2012). High-speed high-security signatures. *Journal of Cryptographic Engineering, 2*(2), 77-89. [CrossRef]

[32] Gündoğan, C., Amsüss, C., Schmidt, T. C., & Wählisch, M. (2020, June). IoT content object security with OSCORE and NDN: A first experimental comparison. In *2020 IFIP Networking Conference (Networking)* (pp. 19-27). IEEE.

[33] Wong, C. K., & Lam, S. S. (1998, October). Digital signatures for flows and multicasts. In *Proceedings Sixth International Conference on Network Protocols (Cat. No. 98TB100256)* (pp. 198-209). IEEE. [CrossRef]

[34] Xu, L., Yuan, X., Zhou, Z., Wang, C., & Xu, C. (2021). Towards efficient cryptographic data validation service in edge computing. *IEEE Transactions on Services Computing, 16*(1), 656-669. [CrossRef]

[35] Psaras, I., Chai, W. K., & Pavlou, G. (2012, August). Probabilistic in-network caching for information-centric networks. In *Proceedings of the second edition of the ICN workshop on Information-centric networking* (pp. 55-60). [CrossRef]

[36] Breslau, L., Cao, P., Fan, L., Phillips, G., & Shenker, S. (1999, March). Web caching and Zipf-like distributions: Evidence and implications. In *IEEE INFOCOM'99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No. 99CH36320)* (Vol. 1, pp. 126-134). IEEE. [CrossRef]

[37] Baccelli, E., Mehlis, C., Hahm, O., Schmidt, T. C., & Wahlisch, M. (2014, September). Information centric networking in the IoT: Experiments with NDN in the wild. In *Proceedings of the 1st ACM conference on information-centric networking* (pp. 77-86). [CrossRef]

[38] Amadeo, M., Campolo, C., Quevedo, J., Corujo, D., Molinaro, A., Iera, A., ... & Vasilakos, A. V. (2016). Information-centric networking for the internet of things: challenges and opportunities. *IEEE Network, 30*(2), 92-100. [CrossRef]

[39] Merkle, R. C. (1987, August). A digital signature based on a conventional encryption function. In *Conference*

on the theory and application of cryptographic techniques (pp. 369-378). Berlin, Heidelberg: Springer Berlin Heidelberg. [CrossRef]

[40] Laurie, B., Messeri, E., & Stradling, R. (2021). *Certificate transparency version 2.0*. RFC 9162, Internet Engineering Task Force. [CrossRef]

[41] National Institute of Standards and Technology (2024). Stateless hash-based digital signature standard. *FIPS 205, NIST*. [CrossRef]

[42] Mastorakis, S., Mtibaa, A., Lee, J., & Misra, S. (2020). Icedge: When edge computing meets information-centric networking. *IEEE Internet of Things Journal, 7*(5), 4203-4217. [CrossRef]

[43] Alghamdi, A., & Keshta, I. (2026). Blockchain consensus mechanisms and enhancement techniques for federated learning-based intrusion detection systems in IoT smart homes. *Journal of Reliable and Secure Computing, 2*(1), 1-26. [CrossRef]

[44] Ranaweera, P., Jurcut, A. D., & Liyanage, M. (2021). Survey on multi-access edge computing security and privacy. *IEEE Communications Surveys & Tutorials, 23*(2), 1078-1124. [CrossRef]

[45] Chen, X., Jiao, L., Li, W., & Fu, X. (2016). Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Transactions on Networking, 24*(5), 2795-2808. [CrossRef]

[46] Kleinrock, L. (1975). Queueing Systems, Volume 1: Theory. Wiley-Interscience, New York, NY, USA. https://dl.acm.org/doi/abs/10.5555/1096491

[47] Moriarty, K., Kaliski, B., Jonsson, J., & Rusch, A. (2016). *PKCS #1: RSA cryptography specifications version 2.2*. RFC 8017, Internet Engineering Task Force. [CrossRef]

[48] Josefsson, S. & Liusvaara, I. (2017). *Edwards-curve digital signature algorithm (EdDSA)*. RFC 8032, Internet Engineering Task Force. [CrossRef]

[49] Krawczyk, H., Bellare, M., & Canetti, R. (1997). *HMAC: Keyed-hashing for message authentication*. RFC 2104, Internet Engineering Task Force. [CrossRef]

[50] Rescorla, E. (2018). *The Transport Layer Security (TLS) protocol version 1.3*. RFC 8446, Internet Engineering Task Force. [CrossRef]

[51] National Institute of Standards and Technology (2024). Module-lattice-based digital signature standard. *FIPS 204, NIST*. [CrossRef]

[52] Bernstein, D. J. & Lange, T. (2017). Post-quantum cryptography. *Nature, 549*(7671), 188-194. [CrossRef]

[53] Khelifi, H., Luo, S., Nour, B., Moungla, H., Faheem, Y., Hussain, R., & Ksentini, A. (2020). Named data networking in vehicular ad hoc networks: State-of-the-art and challenges. *IEEE Communications Surveys & Tutorials, 22*(1), 320-351. [CrossRef]

Mehbub Alam is an Assistant Professor in the Department of Computer Science and Engineering at the Indian Institute of Information Technology Raichur, India, and also serves as the Executive Director of the AI Research and Innovation Arena at IIIT Raichur. He received the Ph.D. degree in Computer Science and Engineering from the Indian Institute of Information Technology Guwahati, focusing on efficient task offloading in fog and edge computing. He was previously a Postdoctoral Researcher at Khalifa University, Abu Dhabi, working on secure unmanned traffic management and drone authentication. His research interests include cybersecurity, UAV security, Internet of Things, fog and edge computing, and computation offloading. He is an IEEE member and has contributed to research in wireless and IoT networks, edge computing, and secure connected systems. (Email: mehbub@iiitr.ac.in)