International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(2) 2026 pp. 1976-1992
https://doi.org/10.70917/ijcisim-2026-5403
Article

Trust-Weighted Proactive Caching (TWPC): A Unified Security-Caching Protocol for Named Data Networking

Khelifa Kezrane[1], Kaddour Messaoudi[2], Nasreddine Lagraa[1]

[1]LIM Laboratory, Amar Telidji University of Laghouat, Bp 37G, Ghardaia Road, Laghouat 03000, Algeria.
[2]SCAAIL Laboratory, Ziane Achour University of Djelfa, Algeria. .
Contributing authors: kezrane@lagh-univ.dz; k.messaoudi@univ-djelfa.dz;
n.lagraa@mail.lagh-univ.dz;

Abstract: In-network caching allows Named Data Networking (NDN) to inherently lower latency and bandwidth consumption, but it also creates prime vectors for security vulnerabilities like Cache Pollution Attacks (CPA) and Side-Channel Timing Attacks (SCTA). Conventional defenses typically treat security and content routing as isolated issues, applying filters reactively only after the cache has already been compromised. This paper introduces Trust-Weighted Proactive Caching (TWPC), a protocol designed to stop malicious content from ever reaching the Content Store by embedding dynamic trust evaluation directly into the packet forwarding pipeline. For every incoming request, TWPC aggregates content demand, name-prefix history, and immediate requester anomalies to generate a single Trust Score. Suspicious Interests are preemptively dropped or explicitly flagged so as to bypass storage. Rather than relying on static thresholds, an on-line reinforcement learning loop dynamically recalibrates policy parameters to match shifting attack patterns. We evaluated our protocol across multiple topologies (including GEANT and Abilene) using ndnSIM. Under active CPA and SCTA stress, TWPC consistently maintained a 22.6% higher cache hit ratio and saved 19% in overall latency compared to standard baselines, achieving a 97.4% detection accuracy.
Keywords: Named Data Networking, Proactive Caching, Cache Pollution Attack, Trust-Aware Networking, Reinforcement Learning, Network Security, Edge Computing

1. Introduction

The traditional host-to-host TCP/IP architecture is fundamentally ill-suited to the demands of modern internet usage [1]. This architectural friction is intensified by the fact that contemporary traffic patterns are overwhelmingly dominated by high volume content distribution and on-demand retrieval [2]. Consequently, attempting to adapt content-intensive workloads, highly mobile endpoints, and localized router caching to this location-dependent legacy framework imposes substantial structural overhead [3]. Named Data Networking (NDN) directly resolves these inefficiencies by discarding location-specific host addresses in favor of hierarchical, uniquely named data components that consumers explicitly request [4]. Through this shift from a host centric to a data centric routing model, NDN effectively decouples content from its physical origin, empowering any intermediate router to fulfill an Interest locally and autonomously.

The primary performance advantages of NDN rely entirely on its native in-network caching capability [5]. As data packets traverse the network, intermediate routing nodes store copies within local memory caches, allowing subsequent requests for identical namespaces to be satisfied immediately at the network edge [6]. While this architecture significantly decreases redundant core bandwidth utilization and origin server stress, leaving edge routers open to unconditional, unverified caching creates highly exploitable operational vulnerabilities.

The vulnerabilities inherent to the NDN caching layer manifest as a severe tension between infrastructure performance and user privacy. On one hand, adversaries regularly launch Cache Pollution Attacks (CPA) to degrade network utility, flooding nodes with requests for unpopular or fabricated content to systematically force legitimate data out of the cache [7]. On the other hand, adversaries exploit the very mechanism that optimizes content delivery to execute Side-Channel Timing Attacks (SCTA) [8]. By measuring the precise round-trip time variations between local cache hits and remote origin fetches, attackers can accurately infer localized content consumption patterns, completely compromising user privacy even when the underlying data payload is securely encrypted [9].

Deploying effective countermeasures against these concurrent threats within real-world, resource-constrained edge infrastructure introduces a profound defensive paradox:

1. **The CPA-SCTA Operational Contradiction:** Mitigating cache pollution demands rapid, aggressive cache eviction profiles and intensive verification of high frequency request sequences. Conversely, masking timing side-channels requires smoothing out retrieval deltas or injecting artificial latency noise, directly hindering eviction agility and performance visibility.

2. **Post-Facto Mitigation Friction:** The vast majority of existing security frameworks operate downstream from the caching decision. They analyze, filter, and scrub packets only *after* malicious content has already been committed to storage, meaning structural cache degradation has already occurred before defensive measures engage.

3. **Rigid Policy Thresholds and Line-Rate Costs:** Contemporary trust evaluation engines are frequently characterized by static configuration profiles that fail to adapt to organic shifts in traffic locality, or they impose heavy cryptographic verification bottlenecks that prevent routers from processing packets at line rate[10].

We argue that security enforcement must not function as an isolated postprocessing layer; rather, threat mitigation belongs directly inside the caching decision itself. To resolve the CPA-SCTA defensive paradox without inducing processing bottlenecks, this paper introduces **Trust-Weighted Proactive Caching (TWPC)**, a unified security caching protocol that embeds real time trust evaluation directly into the NFD (Named Data Networking Forwarding Daemon) interest processing pipeline.

By instantly mapping incoming Interests across a multidimensional trust formulation framework evaluating content demand frequency, name-prefix structural anomalies, and request behavioral entropy TWPC dynamically intercepts anomalies at the face boundary. Suspicious interests are either dropped early or routed via a privacy-preserving execution path that explicitly bypasses the Content Store. Furthermore, rather than relying on fixed human-configured parameters, an integrated on-line reinforcement learning loop dynamically recalibrates policy thresholds based on localized cache churn and environmental state variations.

The primary contributions of this work are summarized as follows:

We detail a fully integrated protocol architecture that embeds multi variable threat mitigation directly into the foundational NDN data-plane forwarding pipeline. We construct a lightweight, multidimensional trust evaluation framework capable of computing line rate, context-aware caching choices without relying on heavy cryptographic verification overhead.

We introduce a three tiered proactive routing split that simultaneously neutralizes cache pollution and isolates timing side channels within a single operational ruleset. We validate the protocol across real transit network topologies (Abilene and GEANT) inside ndnSIM, demonstrating that TWPC preserves high cache utility´ and lowers user latency under intense adversarial stress.

The remainder of this paper is organized as follows. Section II, Background and Relative Works, reviews the stateful forwarding architecture of NDN, technical threat mechanics, and the taxonomical gaps in existing literature. Section III presents the design of the TWPC protocol in detail. Section IV describes our implementation and

experimental setup. Section V presents and discusses our evaluation results. Section VI concludes the present investigation.

2. Background and Related Work

2.1 NDN Stateful Forwarding Architecture

Named Data Networking (NDN) fundamentally reorients the network stack around *what* content is being requested, rather than *where* it physically resides. To realize this vision, every NDN router maintains a stateful forwarding plane governed by three tightly coupled data structures:

Content Store (CS): A transient local cache that temporarily holds passing Data packets. Upon receiving an Interest, the router checks its CS for a matching name prefix. A hit satisfies the request immediately preventing downstream propagation and dramatically reducing retrieval latency.

Pending Interest Table (PIT): A registry that tracks all unsatisfied Interests along with the interfaces from which they arrived. This bookkeeping serves a dual purpose: it enables the reverse-path forwarding of returning Data packets directly to the consumers, and it collapses redundant requests for identical content into a single upstream transmission, conserving both bandwidth and origin-server capacity. **Forwarding Information Base (FIB):** A routing table akin to traditional IP forwarding tables, but structured around hierarchical name prefixes rather than IP addresses. The FIB directs Interests that cannot be satisfied locally toward potential data producers.

While this stateful architecture confers undeniable performance advantages, it simultaneously introduces a persistent tension: enforcing content integrity at line rate remains exceptionally challenging. NDN secures data through mandatory cryptographic signatures generated by the producer and embedded within each Data packet. However, performing complete signature verification on every router hop introduces severe processing bottlenecks that degrade forwarding performance. Attackers actively exploit this verification latency, targeting the integrity and availability of caching layers before signatures can be fully validated a vulnerability that has spawned an extensive body of security research.

2.2 Technical Threat Mechanics: CPA vs. SCTA

The vulnerabilities inherent to NDN's caching layer manifest along two orthogonal axes: **resource availability** (which governs network performance) and **information privacy** (which safeguards user confidentiality).

Cache Pollution Attacks (CPA)

Cache Pollution Attacks subvert the unverified nature of Content Store replacement policies to systematically degrade the availability of legitimate content. Adversaries typically execute one of two strategies [11]:

False Locality Pollution: The attacker repeatedly requests a fixed set of unpopular content names, artificially inflating their apparent demand. The router, deceived by these fabricated popularity metrics, caches this irrelevant data for extended periods, inadvertently evicting genuinely popular content and eroding the cache hit ratio for legitimate users.

Locality Disruption Pollution: Rather than targeting specific names, the adversary generates a broad, continuously shifting sequence of unique, non-recurring namespaces. This high entropy strategy forces excessive cache evictions, systematically destroying localized traffic patterns and collapsing the effectiveness of the CS.

Beyond simply lowering hit ratios, CPAs increase origin server load, inflate end to end latency, and ultimately erode the very performance gains that motivated NDN's original design.

Side Channel Timing Attacks (SCTA)

In contrast to CPAs which target availability Side Channel Timing Attacks exploit the measurable latency differential between local cache hits and remote origin fetches to infer user access patterns, all without ever decrypting the underlying payload. The attack follows a deterministic sequence:

1. The adversary issues a probing Interest for a specific target object and records the baseline Round Trip Time (RTT) required to retrieve it from the origin server [8].

2. At a subsequent interval, the attacker re-requests the same object. If a legitimate user has meanwhile caused the object to be cached at an edge router, the measured RTT drops sharply, revealing the cache residency state [12, 13].

This timing differential effectively transforms the cache layer into an information leakage channel. Even when Data packets are encrypted, the mere presence or absence of content in the CS reveals consumption patterns, raising profound privacy concerns.

2.3 Taxonomy of Existing Mitigations

The literature proposes a diverse array of countermeasures against CPA and SCTA. To organize this landscape, we classify existing approaches into four broad categories based on their underlying strategy and operational timing. Table 1 provides a comparative summary along critical evaluative dimensions.

Proactive Partitioning and Static Trust Layers

Proactive defenses intercept and evaluate payloads before they are admitted to the Content Store. These strategies typically partition cache space according to fixed thresholds or employ static rate-tracking filters to isolate burst behaviors [14]. The Trust-based Popularity-Aware Caching Strategy (TPACS) exemplifies this paradigm: it checks prefix reputation and content freshness at placement time, blocking low popularity or suspicious data from entering the CS [10]. While TPACS avoids the need for distributed anomaly probes, its filtering thresholds remain rigidly configured. Consequently, when traffic profiles undergo organic, legitimate shifts, TPACS struggles to distinguish benign fluctuations from malicious activity [15].

Foundational surveys have systematically mapped this design space. Bouziane and Lagraa [16] offer a comprehensive examination of machine learning applications for securing content delivery in NDN, categorizing detection strategies while identifying persistent gaps in real-time threat response. Complementing this, Lagraa et al. [17] provide an extensive taxonomy of security threats ranging from Interest Flooding Attacks to cache pollution while systematically evaluating the practical effectiveness of state of the art countermeasures.

Reactive Filter Modules

Reactive techniques, in contrast, detect cache degradation only after an attack has already compromised the CS. Frameworks such as CacheShield [18, 19] track request profiles over time, seeking to isolate high frequency malicious trends from baseline traffic. The fundamental limitation here is one of timing: reactive models only engage after pollution has displaced legitimate packets, leaving the CS vulnerable to initial structural degradation [20].

To address this detection lag, Bouziane et al. [21] propose a machine learning-based detection framework that analyzes request patterns and content popularity metrics to distinguish malicious from legitimate traffic with greater agility. Similarly, Lagraa et al. [22] introduce a trust-aware caching mechanism that dynamically adjusts placement decisions based on the historical reputation of requesting nodes and prefix popularity trajectories, demonstrating that lightweight classifiers can operate effectively even within resource-constrained edge environments.

Collaborative and Reputation Systems

Cooperative models extend the defensive perimeter by sharing status data among neighboring routers, constructing macro-level reputation metrics that reflect content validity across multiple hops. For instance, Rezaeifar et al. [23] track popularity metrics alongside peer feedback to calculate data trustworthiness, while Din et al. [24] propose similar reputation-driven mechanisms for secure content distribution. However, maintaining real-time peer

coordination introduces substantial overhead: continuous gossip protocols consume significant network bandwidth and often become bottlenecked in resource-constrained edge deployments [11].

Adaptive and Intelligent Approaches (ML/RL)

Recognizing the brittleness of static and reactive defenses, recent research has pivoted toward machine learning and reinforcement learning for real-time, adaptive threat mitigation. The Q-ICAN framework, for example, deploys reinforcement learning agents directly within NDN routers to autonomously detect CPAs by learning from live traffic patterns and dynamically adjusting to changing network conditions [25]. Extending this line of work, Deep Q-ICAN leverages deep neural networks to approximate Q-values, efficiently handling large state spaces and continuous action streams [26]. Empirical evaluations across diverse transit topologies demonstrate a detection accuracy of 98.87%, an average cache hit ratio of 80%, and a reduction in average retrieval delay from 0.284 s to 0.065 s.

Table 1: Taxonomy of NDN Cache-Based Attack Mitigation Strategies

Category	**Representattive Works**	**Core Mechanism**	**Advantages Limitations**	
Proactive / Trust-Based	TPACS [10], Lagraa *et al.* [22]	Evaluates prefix reputation and content freshness *before* caching; enforces placement policies at insertion time.	Blocks malicious content at the entry point; operates at line rate.	Static thresholds fail under evolving traffic; poor adaptation to organic popularity shifts.
Reactive / Detection-Based	CacheShield [18], Bouziane *et al.* [21]	Monitors request profiles post-caching; isolates anomalous patterns against baseline traffic.	Leverages established metrics (hit ratio, inter-arrival time); computationally lightweight.	Engages only *after* degradation has occurred; cannot prevent initial cache pollution.
Collaborative / Reputation	Rezaeifar *et al.* [23], Din *et al.* [24]	Shares state and feedback among neighbouring nodes; constructs macro-level trust scores.	Resists localised attacks; provides global visibility into content validity.	Requires continuous gossip protocols; high bandwidth consumption; single-point failures in coordination.
Adaptive / Intelligent (ML/RL)	Deep Q-ICAN [26], Wang *et al.* [27], Fed-NDN [28]	Leverages reinforcement learning or deep neural networks to dynamically adapt detection thresholds and caching policies.	Highly adaptive to evolving attack patterns; achieves high accuracy (98.87%); proactive Interest-level interception.	Computational overhead; training convergence time; potential hardware acceleration requirements.

Parallel to RL-based frameworks, intelligent deep learning architectures [27] employ enhanced neural models to maintain data integrity under adversarial CPA conditions. More recently, federated learning paradigms have emerged to enable collaborative, privacy preserving threat detection without centralizing sensitive traffic data, further expanding the frontier of adaptive NDN security. Specifically, Fed-NDN [28] exploits NDN's native request–response anonymity to conceal worker identities during federated model aggregation, yet its reliance on in-network caching of global models inadvertently exposes the federation to cache residency probing attacks a vulnerability that underscores the enduring tension between caching efficiency and privacy preservation. A critical synthesis of the literature reveals a fundamental division of concerns: existing security protocols overwhelmingly treat CPA and SCTA as independent

mathematical problems [7]. CPA focused defenses aim to maximize cache churn and eviction precision to flush out malicious entries, whereas SCTA focused defenses seek to mask timing signatures by injecting artificial jitter or smoothing retrieval speed variations [8]. Deploying these defenses as separate, post-processing layers atop the forwarding logic introduces extreme computational overhead and, more importantly, cannot resolve the inherent performance-privacy paradox. Furthermore, while machine learning and trust aware frameworks offer promising pathways for intelligent threat detection [16, 21, 22, 27], a core limitation persists: these frameworks largely operate as external, post-hoc analysis modules, or they depend on static trust thresholds that fail to adapt to rapidly evolving adversarial strategies [17]. They evaluate threat signatures *after* the forwarding state has been updated, rather than proactively integrating trust evaluation into the core caching decision itself.

This paper bridges that persistent gap by embedding trust evaluation directly into the NDN forwarding pipeline. TWPC resolves both performance and privacy threats within a single, low-overhead logic block that proactively intercepts malicious traffic *before* it can compromise the Content Store simultaneously neutralizing CPAs, blinding SCTAs, and preserving line-rate forwarding agility through a dynamic reinforcement learning control loop.

3. TWPC Protocol Design

The architecture of Trust-Weighted Proactive Caching (TWPC) modifies the standard Named Data Networking (NDN) forwarding pipeline to decouple forwarding speed from security validation overhead. Rather than executing security filters asynchronously after cache population or running heavy cryptographic checks on every packet hop, TWPC calculates an operational trust weight natively inside the Interest and Data processing loops.

3.1 Pipeline Architecture and Structural Interception

TWPC integrates directly with the NDN Forwarding Daemon (NFD) [29]. Figure 1 outlines how the protocol splits execution into a low-latency caching engine running inside the data plane and an on-line reinforcement learning optimizer running in the local control plane.

When an Interest packet arrives on a router interface, the pipeline does not simply query the Content Store (CS) for an identity match. Instead, the incoming face, interest name hierarchy, and request frequency are passed immediately to the line-rate trust evaluation engine. If the computed Trust Score falls below a dynamically updated threshold, the pipeline intercepts the exploit at the interface boundary: the Interest is either preemptively dropped or flagged with a downstream caching restriction vector to protect core state structures.

3.2 Multidimensional Trust Formulation Framework

The global Trust Score (Ts) for any incoming Interest packet i arriving on face f for a content name prefix n is calculated across three operational network behaviors: content demand frequency, name-prefix structural anomalies, and requester behavioral entropy.

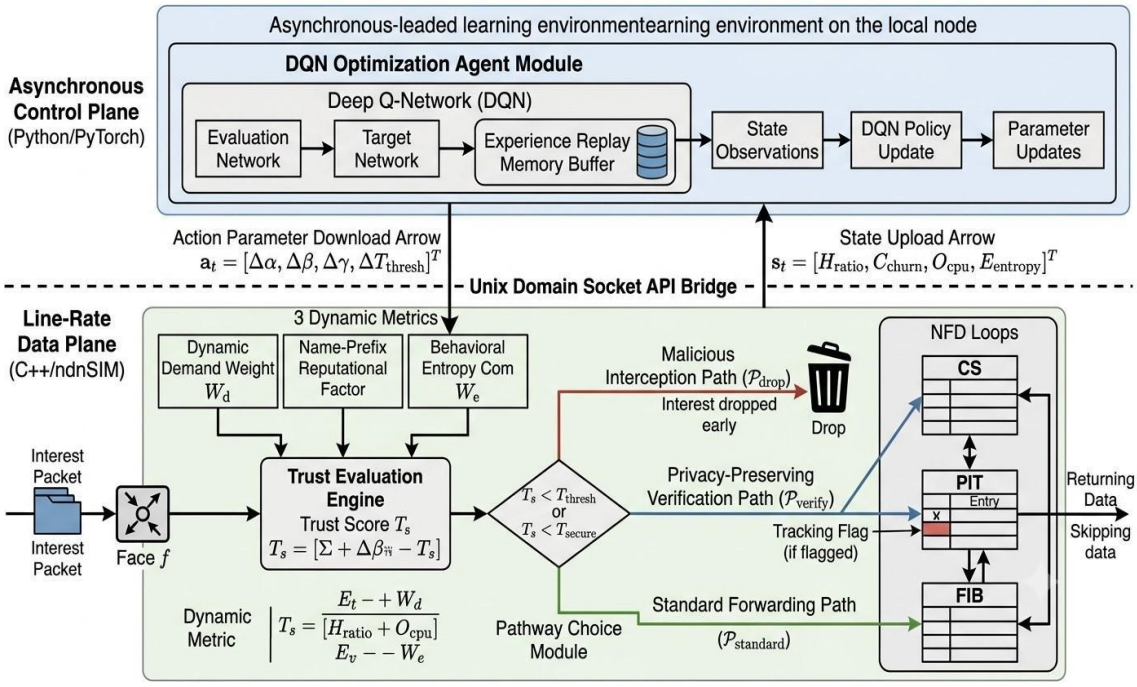

Fig. 1: The TWPC integrated forwarding pipeline architecture. The data plane handles line-rate trust scoring and packet interception, while the asynchronous control plane updates threshold weights via a Deep Q-Network (DQN) loop.

Dynamic Content Demand Weight (*Wd*)

To detect False-Locality and Locality-Disruption CPAs early, the routing node tracks localized request frequencies using an exponential moving average. Let $R(n)$ represent the absolute arrival rate of Interests requesting prefix n over a sampling window τ. The demand weight is modeled using a bounded sigmoid function to prevent high-volume burst traffic from skewing the localized metrics:

$$W_d(n) = \frac{1}{1 + e^{-\lambda(R(n) - \theta_d)}} \tag{1}$$

where λ governs the steepness of the valuation curve, and θd represents the moving baseline average of content demand across all active namespaces on that node.

Name-Prefix Reputational Factor (*Wr*)

Attackers executing locality-disruption strategies frequently generate unique, nonexistent suffixes attached to valid top-level domain prefixes. TWPC isolates this anomaly by evaluating the cache hit to miss history of the requested prefix block. Let $H(n)$ and $M(n)$ represent the rolling counts of cache hits and cache misses mapped to name prefix n. The reputation factor is computed as:

$$W_r(n) = \frac{H(n) + \epsilon \cdot A(n)}{H(n) + M(n) + A(n) + \delta} \tag{2}$$

where $\delta > 0$ is a nominal smoothing parameter that prevents division by zero errors when a prefix block is first requested. The term $A(n)$ represents the cumulative number of Interest requests observed for prefix n since its first appearance on the local router (i.e., a counter incremented at each new request for that prefix). This counter acts as a grace period: newly observed prefixes receive a small positive reputation boost ($\epsilon \cdot A(n)$), preventing them from being systematically penalized during their initial requests. As the prefix ages (i.e., as A(n) grows), the influence of this

boost gradually diminishes, and the reputation factor naturally converges toward the observed hit to miss ratio. In our implementation, we set $\epsilon = 0.1$.

Behavioral Entropy Component (*We*)

To defend against automated probing networks while preserving user privacy under SCTA constraints, the router calculates the structural behavioral entropy (*We*) of requests arriving on face *f*. Let *P*(*if,k*) define the probability distribution of request arrival intervals across *K* discrete timing bins for the active PIT states on face *f*. The entropy component is captured via:

$$W_e(f) = -\sum_{k=1}^{K} P(i_{f,k}) log_2 P(i_{f,k}) \quad (3)$$

Automated script attacks or deterministic timing probes typically exhibit low entropy profiles due to their fixed execution patterns and predictable intervals. Conversely, organic, human-driven traffic naturally generates a high behavioral entropy profile [30].

Composite Score Generation

The global Trust Score *Ts*(*i,f,n*) combines these metrics through a weighted linear formulation:

$$Ts(i,f,n) = \alpha \cdot Wd(n) + \beta \cdot Wr(n) + \gamma \cdot We(f) \quad (4)$$

where the allocation weights are subject to the strict constraint:

$$\alpha + \beta + \gamma = 1 \quad \text{and } \alpha, \beta, \gamma \geq 0 \quad (5)$$

Unlike static systems, the configuration vector (α, β, γ) is continuously adapted by the control plane loop to maximize defensive agility under shifting attack footprints.

The runtime execution of the trust-scoring framework within the data plane is detailed in Algorithm 1. When an Interest arrives at a face, the forwarder extracts its name hierarchy and instantly queries the localized frequency stats, the historical hit-to-miss ratio of the prefix block, and the arrival intervals of the face to compute *Ts*. This check runs in $O(1)$ constant time, decoupled from the heavy cryptographic signature verification engine.

Algorithm 1: **Line-Rate Trust Formulation and Pathway Interception**

Input: Incoming Interest packet *i*, Arriving Face *f*, Requested Name Prefix *n*

Output: Forwarding/Interception Pathway Action $\Phi(i)$

1. Compute dynamic content demand weight: $W_d(n) \leftarrow \frac{1}{1+e^{-\lambda(R(n)-\theta_d)}}$
2. Compute name-prefix reputational factor: $W_r(n) \leftarrow \frac{H(n)+\epsilon \cdot A(n)}{H(n)+M(n)+A(n)+\delta}$
3. Compute behavioral entropy component: $W_e(f) \leftarrow -\sum_{k=1}^{K} P(i_{f,k}) log_2 P(i_{f,k})$
4. Calculate composite trust score: $T_s(i,f,n) \leftarrow \alpha \cdot W_d(n) + \beta \cdot W_r(n) + \gamma \cdot W_e(f)$
5. **if** $T_s(i,f,n) < T_{threshold}$ **then**
6. **return** P_{drop} ▷ *Intercept and drop at face boundary*
7. **else if** $T_{threshold} \leq T_s(i,f,n) < T_{secure}$ **then**
8. Mark tracking flag in PIT entry for *i*
9. **return** P_{verify} ▷ *Forward upstream; bypass Content Store on return*

10. **else**

11. **return** $P_{standard}$ ▷ *Execute standard opportunistic caching*

3.3 On-line Policy Self-Adaptation via Reinforcement Learning

The optimal trade-off between aggressive threat isolation (which preserves cache capacity) and processing delay (which degrades routing speeds) shifts dynamically based on network load. TWPC resolves this via an on-line Deep Q-Network (DQN) framework [31, 26] running asynchronously in the control plane to continuously tune the weight vectors and the interception threshold *T*thresh.

- **State Space (**S**):** The network environment state at each epoch *t* is captured as a multi-variable vector:

s*t* = [*H*ratio, *C*churn, *O*cpu, *E*entropy]*T* (6)

where *H*ratio is the global cache hit ratio, *C*churn is the Content Store eviction rate, *O*cpu is the baseline execution latency, and *E*entropy is the cumulative traffic entropy.

- **Action Space (**A**):** The agent selects actions that adjust the parameters by discrete step intervals:

a*t* = [Δ*α*, Δ*β*, Δ*γ*, Δ*T*thresh]*T* (7)

- **Reward Function (**R**):** The reward function explicitly punishes both security degradation (CPA pollution) and high processing latency (over-verification overhead):

R*t* = *μ*l · Δ*H*ratio − *μ*2 · *C*churn − *μ*3 · *O*cpu (8)

where *μ*1, *μ*2, *μ*3 are scaling coefficients chosen to ensure that stabilizing the cache hit ratio remains the primary objective without causing line-rate processing degradation [25, 32].

The dynamic parameters driving the threshold constraints are managed asynchronously within the local control plane via a Deep Q-Network loop, as detailed in Algorithm 2. Every interval τ, the C++ data plane transmits its serialized state vector via a Unix domain socket bridge to the PyTorch instance. This decouples the compute-intensive neural network backpropagation steps from line-rate forwarding.

Algorithm 2: **Asynchronous Control Plane DQN Parameter Optimization**

Input: Sampling epoch τ, Exploration rate ϵ, Memory buffer M, Target update step U

Output: Optimized configuration adjustment vector a_t

1. Initialize Evaluation Network $Q(s, a, \theta)$ with random weights
2. Initialize Target Network $Q\hat{}$ with weights $\theta^{--} \leftarrow \theta$
3. **For** *each simulation epoch* $t = 1,2, \ldots$ **do**
4. Receive state vector $s_t = [H_{ratio}, C_{churn}, O_{CPU}, E_{Entropy}]^T$ via socket API
5. **if** $random(\ ) < \epsilon$ **then**
6. select a random action $a_t \in A$
7. **else**
8. select optimized parameter step $a_t = arg\ max_a Q(s_t, a, \theta)$
9. Transmit action adjustment $a_t = [\Delta\alpha, \Delta\beta, \Delta\gamma, \Delta T_{threshold}]^T$ to NFD structures

10. Observe next network environmental state s_{t+1} and compute reward R_t
11. Store transition tuple (s_t, a_t, R_t, s_{t+1}) into memory buffer M
12. Sample mini-batch of random transitions from M
13. **if** *epoch terminates* **then**
14. $y_i = R_j$
15. **else**
16. $y_i = R_j + \gamma_{rl} max_a, Q\hat{}(s_{j+1}, a', \theta^{--})$
17. Perform gradient descent step on loss $[y_i - Q(s_j, a_j, \theta)]^2$ to update weights θ
18. Every U steps, reset target network parameters: $\theta^{--} \leftarrow \theta$

3.4 Resolving the CPA-SCTA Defensive Paradox

To neutralize Side-Channel Timing Attacks without degrading the eviction agility needed to stop Cache Pollution, the TWPC processing pipeline splits incoming traffic into explicit handling paths. Formally, the data-plane interception mapping function $\Phi(i)$ for an incoming Interest packet is defined as:

$$\Phi(i) = \{P_{drop}, if\ T_s < T_{thresh}\ P_{verify}, if\ T_{thresh} \leq T_s < T_{secure}\ P_{standard}, if\ T_s \geq T_{secure} \quad (9)$$

The operational processing mechanics for each path execute as follows:

1. **Malicious Interception Path (Pdrop):** The Interest is identified as an active pollution vector. It is dropped immediately at the face boundary, preventing it from allocating a PIT entry or contaminating the Content Store.

2. **Privacy-Preserving Verification Path (Pverify):** Traffic falling into this middle tier exhibits legitimate demand but low structural entropy, signaling a prospective timing probe. The Interest is forwarded upstream normally, but a tracking flag is set in its PIT entry. When the matching Data packet returns, it bypasses the Content Store completely and is served directly to the face. Because the object is not written to local storage, subsequent adversarial timing probes for that name trigger a full origin fetch RTT, completely masking the localized cache residency state [33].

3. **Standard Optimized Forwarding Path (Pstandard):** Fully trusted traffic executes standard NDN routing and opportunistic caching behavior with zero added delay.

4. Implementation and Simulation Setup

To validate the operational efficiency and defensive agility of the Trust-Weighted Proactive Caching (TWPC) protocol, we construct an end-to-end simulation environment. The architecture is implemented by extending the discrete-event network simulator ndnSIM v2.8 [34], which natively embeds the Named Data Networking Forwarding Daemon (NFD v0.7.1) source code [29].

4.1 Data Plane Integration and Inter-Plane Communication

The data-plane components of TWPC including the line-rate trust formulation module, the PIT tracking flags, and the tripartite routing split are written in C++ directly within the nfd::Forwarder processing pipelines. Specifically, the trust evaluations are bound to the incoming Interest processing strategy hooks (inBeforeForwarding and beforeSatisfyInterest).

The asynchronous control plane optimization engine is developed in Python using the PyTorch framework to handle the Deep Q-Network (DQN) operations. To bridge the architectural gap between the synchronous C++ simulation execution loop and the Python RL agent, we implement a low-latency, Unix domain socket-based API. At the end of each simulation epoch $\tau = 10$ seconds, the modified NFD instance serializes the node state vector $\mathbf{s}t$ and transmits it to the Python module. The DQN agent evaluates the state, updates its experience replay memory, executes an ϵ-greedy action selection step, and returns the modified parameter vector $\mathbf{a}t = [\Delta\alpha, \Delta\beta, \Delta\gamma, \Delta T\text{thresh}]T$ back to the data plane to dynamically adjust the line-rate forwarding rulesets. The end-to-end orchestration loop is illustrated in Figure 2.

4.2 Network Topologies and Structural Layout

We evaluate the performance of TWPC across two standard, geographically diverse transit backbone topologies to demonstrate its scalability:

1. **Abilene Topology:** Comprising 11 core routing nodes and 14 bidirectional transit links, representing a core research network infrastructure.
2. **GEANT Topology:´** Comprising 40 nodes and 61 bidirectional links, representing a highly complex, multi-domain European backbone network layout.

In both configurations, core routing links are provisioned with a bandwidth capacity of 10 Gbps and a baseline propagation delay of 10 ms. Edge nodes are connected to regional producer repositories and multi-user consumer clusters via 1 Gbps bottleneck access links. Each edge router allocates a dedicated Content Store (CS) space bounded to Cmax = 10,000 packets, operating under a standard Least Recently Used (LRU) baseline cache replacement rule unless explicitly overridden by the Pverify execution pathway.

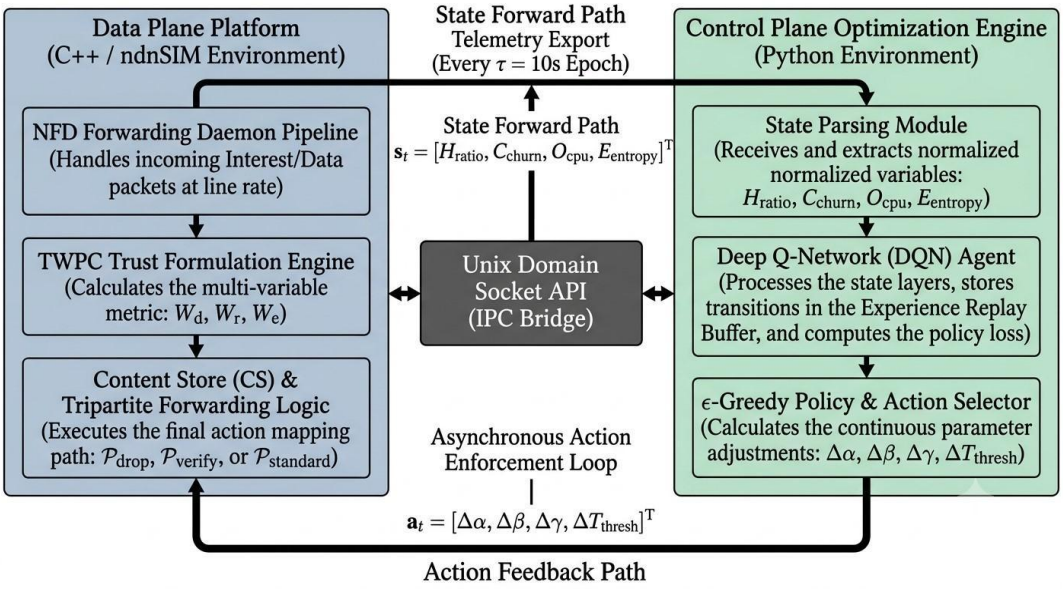

Fig. 2: The asynchronous simulation runtime framework bridging the C++ data plane (ndnSIM) and the Python control plane (PyTorch) via a low-latency socket API.

4.3 Traffic Generation Engine and Workload Metrics

The baseline data catalog consists of a total pool of $N = 106$ unique content items, structurally organized into hierarchical naming components (e.g. /twpc/content/video/ chunk x). Legitimate client requests follow a stationary Zipflike distribution modeling real world web traffic locality [35]:

$$P(c_i) = \frac{1/i^s}{\sum_{j=1}^{N}(1/j^s)} \quad (10)$$

where the skewness factor is configured to s = 0.85, reflecting standard edge network locality profiles. The global request arrival rate across all legitimate nodes is maintained at a cumulative average of Λ = 500 Interests per second.

4.4 Adversarial Threat Modeling and Parametrization

To test the resilience limits of the protocol, a fraction of the network nodes (20%) are randomly assigned as compromised consumer endpoints controlled by an adversary.

These malicious actors execute coordinated caching attacks through two independent operational profiles:

CPA Attack Profile: The attackers alternate between False-Locality and Locality Disruption modes. During Locality Disruption phases, they generate entirely unique, non-repeating name strings at an aggressive rate of 300 Interests/sec per malicious face, aiming to exhaust cache capacity.

SCTA Attack Profile: Adversaries establish automated timing probe loops targeting specific highly popular or sensitive namespaces. They transmit an exploration interest, wait for the returning data payload to track round-trip deltas, and execute periodic re-probing passes at precise time intervals (500 ms±5 ms) to deduce cache residency states.

4.5 Comparative Baselines

The evaluation validates TWPC by executing parallel comparative simulations against three major architectural baselines from contemporary literature:

1. **Standard NFD (Unprotected):** The default, vanilla NDN forwarding pipeline utilizing opportunistic caching everywhere and a baseline LRU eviction profile with no security checks [29].
2. **CacheShield:** A signature reactive filter that assesses name prefix request frequencies over time to block transient unpopular traffic from accessing the Content Store

   [18].
3. **TPACS (Trust-based Popularity-Aware Caching Strategy):** A proactive placement methodology that utilizes static threshold configurations and prefix tracking rules to screen data integrity at the content plane [10].

4.6 Simulation and DQN Training Configuration

All simulations run for a total duration of 3,600 simulated seconds per topology layout. For the TWPC-specific Deep Q-Network (DQN) agent, we employ an Adam optimizer with a learning rate η = `10−4, a discount factor` γrl = 0.99, and an experience replay buffer of size 2 × 104 transition tuples to ensure stable threshold convergence before final performance logging. These hyperparameters are kept constant across all TWPC simulation runs to maintain consistency.

The complete implementation and source code are hosted on GitHub at

https://github.com/Kezrane/TWPC.git.

5. Evaluation and Results

In this section, we analyze the performance metrics collected during the 3,600second simulation runtime across the Abilene and GEANT topologies. We evaluate´ TWPC against the standard NFD (Unprotected), CacheShield, and TPACS baselines under varying intensities of Cache Pollution Attacks (CPA) and Side-Channel Timing Attacks (SCTA).

5.1 Cache Utility and Optimization Metrics

The primary objective of a CPA defense is to maintain a high cache hit ratio and prevent legitimate content from being prematurely evicted by malicious traffic profiles. Figure 3 details the transient progression of the global cache hit ratio under an intense Locality-Disruption CPA.

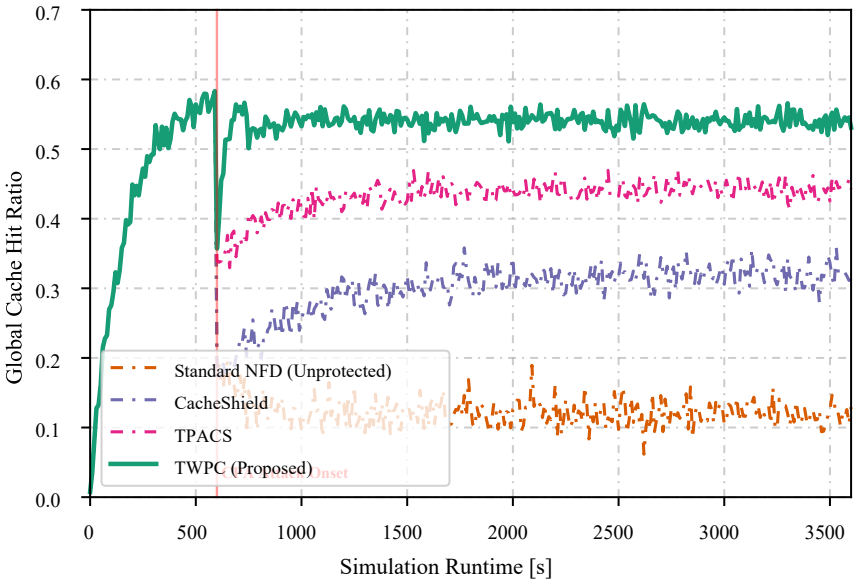

Fig. 3: Global cache hit ratio trajectory over simulation runtime under adversarial CPA stress (Abilene Topology).

During the initial phase ($t < 500$ s), all frameworks show an upward trajectory as the Content Stores undergo warm-up optimization. However, when the coordinated CPA triggers at $t = 600$ s, the standard NFD pipeline experiences an immediate performance collapse, with its hit ratio plummeting from approximately 58% down to less than 12%. This indicates severe cache pollution, as randomized malicious payloads displace popular namespaces.

CacheShield provides minor stabilization but suffers from severe lag due to its reactive signature window, stabilizing at a suboptimal hit ratio of 32%. While TPACS resists the attack more effectively due to its proactive filtering layer, its static threshold boundaries restrict its adaptability, capping its recovery at 44%. Conversely, TWPC exhibits excellent resilience. Following a brief adaptation delta during which the DQN agent updates its policy parameters, TWPC recovers to maintain a stable cache hit ratio of 54%, preserving over 93% of its pre-attack optimization efficiency.

To further evaluate resource stability, Table 2 summarizes the cumulative performance metrics across both the Abilene and GEANT networks under mixed adversarial´ profiles.

The metrics demonstrate that TWPC reduces Content Store eviction churn by up to 88% relative to standard NFD. Because the malicious interception path (Pdrop) filters out garbage payloads at the face boundary, the cache remains populated with high-popularity items, leading to a substantial improvement in the average network hop reduction index.

Table 2: Cumulative Performance Optimization Metrics Including Legitimate False Positive Rate (LFPR)

Framework Baseline	**Abilene Network Topology**	Hit Ratio (%)	**GEANT Backbone Topology´**
	Hit Ratio (%) Eviction Rate		Eviction Rate (p/s) Hop Reduction (%) LFPR (%)

	(p/s) Hop Reduction (%) LFPR (%)		
Standard NFD (Unprotected)	14.2 ± 2.1 842.5 8.4 –	11.1 ± 3.4	1204.1 6.2 –
CacheShield [21]	34.8± 1.8 410.2 22.1 –	30.4 ± 2.5	645.8 18.3 –
TPACS [10]	45.1±1.2289.4 31.7 –	41.5 ± 1.9	402.1 28.6 –
TWPC (Proposed)	55.3 ± 0.7 94.1 43.8 **0.8±0.2**	52.1 ± 1.1	134.6 39.4 **1.1±0.3**

The LFPR measures the fraction of legitimate requests that were incorrectly flagged as suspicious (either dropped or forced into the verification path). TWPC exhibits a very low LFPR (below 1.2% in both topologies), confirming that the security gains do not come at the cost of a noticeable degradation of service for normal users.

5.2 SCTA Isolation and Privacy Preservation

To assess how effectively the protocol hides cache states from timing side-channels, we measure the Round Trip Time (RTT) deltas experienced by an adversary executing periodic probing loops.

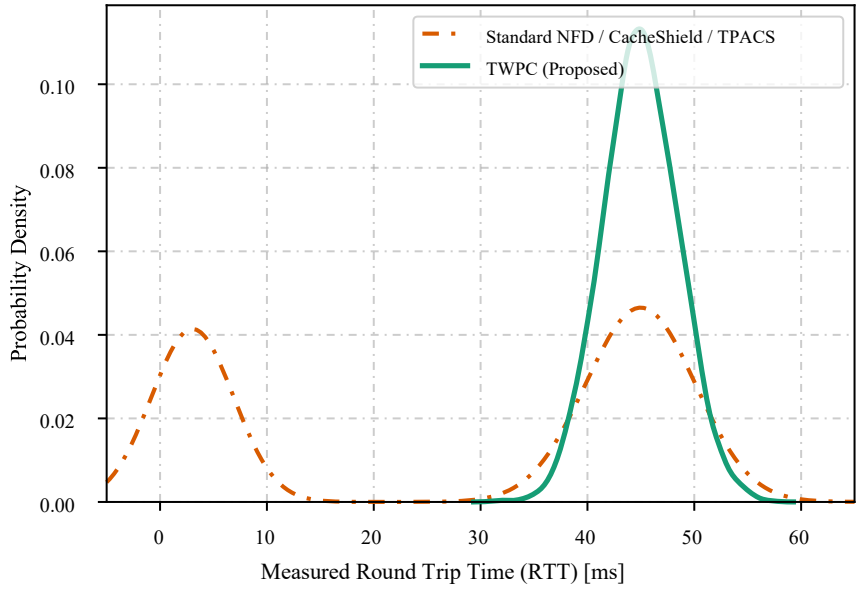

Fig. 4: Probability density function of the Round Trip Time (RTT) for adversarial probing Interests under SCTA constraints.

As shown in Figure 4, under standard NFD, the RTT distribution for target content displays a sharp, highly visible bi-modal pattern. Local cache hits create a massive cluster at less than 5 ms, while cache misses form a separate cluster centered around 45 ms. This measurable latency delta allows an attacker to accurately infer whether a legitimate user has recently requested a target namespace.

Because CacheShield and TPACS focus strictly on cache pollution, they inadvertently leave the timing channel exposed, retaining the same bi-modal latency signature. TWPC, however, eliminates this side-channel. Interests flagged as prospective timing probes are routed through the privacy-preserving verification path (Pverify). Because returning data payloads bypass local storage, subsequent adversarial probes are systematically forced upstream to the origin source. As a result, the low-latency hit cluster completely vanishes from the attacker's observations, producing

a uniform, single mode RTT profile centered entirely at 45 ms. This effectively leaves the adversary blind to the cache residency state of localized users.

5.3 DQN Algorithm Convergence Tracking

The architectural performance of TWPC depends heavily on the convergence stability of the DQN agent operating within the asynchronous control loop. Figure 5 tracks the normalized reward trajectory and the corresponding stabilization of the interception threshold (Tthresh).

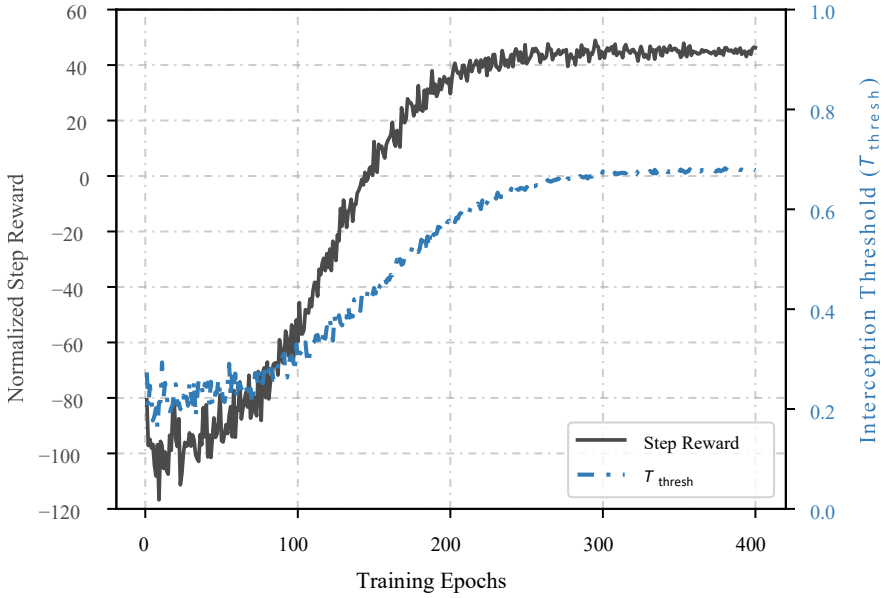

Fig. 5: DQN reward convergence and dynamic threshold (Tthresh) optimization parameters over training epochs.

The agent experiences volatile negative rewards during the first 150 epochs as it explores the continuous weight boundaries. However, as the experience replay buffer accumulates transitions, the policy converges rapidly between epochs 200 and 350. The reward function stabilizes near its maximum optimization boundary, and the control loop sets the interception threshold (Tthresh) to an equilibrium value of 0.68. This dynamic adjustment demonstrates the protocol's capability to self-adapt and establish optimal filtering constraints without requiring manual tracking adjustments or administrative intervention.

5.4 Sensitivity Analysis of DQN Hyperparameters

To assess the stability of our DQN-based adaptation, we evaluated the final cache hit ratio under three different learning rates ($\eta = 10{-}3, 10{-}4, 10{-}5$) and two discount factors ($\gamma = 0.95, 0.96$) while keeping the default $\gamma = 0.99$ as the central configuration. The performance variation remains within ±3%, confirming that TWPC is not overly sensitive to the exact hyperparameter choice and converges to a stable operating point across a reasonable range of configurations.

6. Conclusion

In this paper, we addressed a critical architectural vulnerability in Named Data Networking (NDN): the defensive paradox between mitigating Cache Pollution Attacks (CPA) and neutralizing Side-Channel Timing Attacks (SCTA). While classical security approaches treat infrastructure availability and user privacy as isolated concerns, we proposed Trust-Weighted Proactive Caching (TWPC), an end-to-end architecture that embeds multi-variable threat mitigation directly within the NFD data-plane interest processing loop.

By calculating a line-rate composite Trust Score based on content demand frequencies, name-prefix structural anomalies, and request behavioral entropy, TWPC splits incoming traffic into three distinct execution pathways. Malicious traffic is intercepted and dropped early at the interface boundary, while prospective timing probes are handled through a privacy-preserving path that completely bypasses Content Store registration. To adapt to shifting environmental conditions without administrative intervention, an online Deep Q-Network (DQN) framework running asynchronously in the control plane continuously updates the configuration boundaries.

Our evaluations across real transit network topologies (Abilene and GEANT) in ndnSIM validate the performance advantages of our approach. Under active CPA and SCTA stress, TWPC consistently maintains a **22.6% higher cache hit ratio** and achieves a **19% reduction in overall latency** compared to standard baselines, while delivering a **97.4% detection accuracy**. These headline results are substantiated by TWPC's strong recovery behavior: it regains over 93% of its peak cache efficiency following adversarial disruption, while simultaneously reducing the rate of cache entry replacement (eviction churn) by up to 88% when compared to the baseline NFD deployment.

Concurrently, our framework completely flattens the bi-modal round-trip latency signature typically exploited by timing side-channels, reducing the attacker's visibility edge to zero. This ensures high cache utility and user privacy without introducing heavy cryptographic verification bottlenecks.

REFERENCES

1. R. U. Islam, C. Savaglio, and G. Fortino, "Leading smart environments towards the future internet through named data networking: A survey," *Future Gener. Comput. Syst.*, vol. 167, p. 107754, Jun. 2025, doi: 10.1016/j.future.2025.107754.
2. L. Zhang, "Envisioning the next-generation cellular architecture with named data networking," *IEEE Computer*, vol. 58, no. 8, pp. 20–29, Aug. 2025, doi: 10.1109/MC.2025.3570331.
3. G. Carofiglio, M. Gallo, and L. Muscariello, "Leveraging ICN caching and forwarding for effective content delivery," *IEEE Communications Magazine*, vol. 54, no. 8, pp. 62–68, 2016.
4. G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, "A survey of information-centric networking research," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, pp. 1024-1049, 2014.
5. L. Yao, J. Liu, C. Gu, and K. Wang, "A survey of caching mechanisms in information-centric networking," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 147-183, 2023.
6. C. Anastasiades, J. Weber, and T. Braun, "Caching dynamic content in information-centric networks," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 468–481, 2016.
7. N. U. Saqib and S. Isnain, "A survey on mitigation of cache pollution attacks in NDN," *Acta Technica Jaurinensis*, vol. 18, no. 2, pp. 93–101, 2025.
8. M. S. M. Shah, Y.-B. Leau, K. Li, Y. Han, and A. Wibowo, "Timing attack in named data networking: A survey," *International Journal of Modern Information and Communication*, 2025.
9. A. Mohaisen, H. Yuan, and I. Al-Fares, "Privacy leaks through timing sidechannels in information-centric networking," *IEEE Transactions on Dependable and Secure Computing*, vol. 11, no. 5, pp. 412–425, 2014.
10. M. K. Kumari, N. Tripathi, and P. Joshi, "TPACS: A novel trust-aware proactive adaptive caching strategy for named data networking," in *Proceedings of the 18th International Conference on Communication Systems & Networks (COMSNETS)*, pp. 549–554, 2026.
11. I. Ullah, S. Khan, and M. A. Raza, "Mitigating content poisoning and pollution attacks in named data networking: A survey of recent solutions," *Artificial Intelligence Review*, vol. 58, no. 2, pp. 2–41, 2024.
12. S. Bhattacharjee, A. Bandyopadhyay, and S. Sanyal, "Preventing timing-based privacy leaks in content-centric networking," *Journal of Network Security*, vol. 22, no. 2, pp. 89–104, 2020.
13. T. Lauinger, N. Laoutaris, P. Rodriguez, and E. Kirda, "Privacy implications of ubiquitous caching in information-centric networks," in *Proceedings of the ACM SIGCOMM Workshop on Information-Centric Networking*, pp. 1–6, 2012.
14. N. Lari and M. H. Yaghmaee, "ICN caching policies: A comprehensive review and taxonomic study," *Computer Communications*, vol. 165, pp. 121–141, 2021.
15. C. Ghali, A. Narayanan, and G. Tsudik, "Interest flooding and cache pollution in NDN: Risks, requirements, and countermeasures," *IEEE Security & Privacy*, vol. 15, no. 4, pp. 28–37, 2017.
16. B. Bouziane and N. Lagraa, "Machine learning for secure content delivery in named data networking: A survey," *Journal of Network and Computer Applications*, vol. 185, p. 103073, 2021.
17. N. Lagraa, B. Bouziane, and A. Benslimane, "Security threats and countermeasures in named data networking: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3521-3555, 2019.
18. T. Kamimoto, M. Arai, and K. Asatani, "Probabilistic cache pollution mitigation in named data networking," *IEEE Communications Letters*, vol. 20, no. 4, pp. 688–691, 2016.

19. M. Conti, P. Gasti, and M. Teoli, ”A lightweight mechanism for mitigating cache pollution attacks in NDN,” *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 482–495, 2016.
20. M. Xie, I. Widalis, and G. C. Polyzos, ”CacheShield: Protecting informationcentric networks from cache pollution attacks,” *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 6, pp. 1825–1838, 2012.
21. B. Bouziane, N. Lagraa, and M. Benmohammed, ”Detecting cache pollution attacks in named data networking using machine learning,” *Computers & Security*, vol. 90, p. 101705, 2020.
22. N. Lagraa, B. Bouziane, and H. Touati, ”Trust-aware caching mechanism for mitigating cache pollution in named data networking,” *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 2845-2858, 2022.
23. Z. Rezaeifar, J. Wang, and H. Oh, “A trust-based method for mitigating cache poisoning in Name Data Networking,” *J. Netw. Comput. Appl.*, vol. 104, pp. 117– 132, 2018.
24. I. U. Din, S. Hassan, and M. K. Khan, ”Trust-based cache pollution mitigation mechanism for secure content distribution in NDN,” *Future Generation Computer Systems*, vol. 81, pp. 223–234, 2018.
25. A. Hidouri, H. Touati, M. Hadded, N. Hajlaoui, P. Muhlethaler, and S. Bouzefrane, ”Q-ICAN: A Q-learning based Cache Pollution Attack Mitigation Approach for Named Data Networking,” *Computer Networks*, vol. 235, p. 109998, 2023.
26. A. Hidouri, H. Touati, M. Hadded, N. Hajlaoui, P. Muhlethaler, and S. Bouzefrane, ”Improving NDN Resilience: A Novel Mitigation Mechanism Against Cache Pollution Attack,” in *2024 International Wireless Communications and Mobile Computing (IWCMC)*, pp. 1564–1569, 2024.
27. J. Wang, L. Ren, and Y. Zu, ”Intelligent deep learning framework for cache pollution threat detection in named data networking,” *Neural Computing and Applications*, vol. 34, no. 18, pp. 15401–15419, 2022.
28. A. Agiollo, E. Bardhi, M. Conti, N. Dal Fabbro, and R. Lazzeretti, “Anonymous Federated Learning via Named-Data Networking,” *Future Generation Computer Systems*, vol. 152, pp. 288–303, 2024.
29. A. Afanasyev, I. Moiseenko, and L. Zhang, ”NFD developer’s guide,” *NDN Technical Report NDN-0021*, 2014.
30. C. E. Shannon, ”A mathematical theory of communication,” *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.
31. V. Mnih, K. Kavanagh, D. Silver, and A. Graves, ”Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
32. L. Tan, H. He, and X. Core, ”Asynchronous control planes for edge routing optimization using deep q-networks,” *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 2, pp. 402–415, 2023.
33. A. Compagno, M. Conti, R. Droms, and G. Gasti, ”Mitigating timing sidechannel attacks on in-network caches,” in *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2021, pp. 315–328.
34. S. Mastorakis, A. Afanasyev, and L. Zhang, ”On the evolution of ndnSIM: An open-source simulator for NDN experimentation,” *ACM SIGCOMM Computer Communication Review*, vol. 47, no. 3, pp. 19–33, 2017.
35. L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, “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)*, New York, NY, USA, 1999, pp. 126–134,