Design and Implementation of a Trust-Aware Fog Computing Simulation Framework Using WorkflowSim

Chia Chuan Wu[1], Selvakumar Manickam[2]*, Shams Ul Arfeen Laghari[3], Shankar Karuppayah[4]
Cybersecurity Research Centre (CYRES), Universiti Sains Malaysia,
George Town, Penang, 11800, Malaysia[1,2,4]
Faculty of Vocational Studies, Universitas Brawijaya, Kota Malang, Indonesia[2]
School of ICT, Faculty of Engineering Design Information and Communications Technology (EDICT),
Bahrain Polytechnic, Isa Town, Bahrain[3]

***Abstract*—Fog computing extends cloud capabilities to the network edge, providing low-latency services for mission-critical applications such as healthcare and industrial automation. However, validating security-aware scheduling policies in this distributed paradigm remains a challenge due to the lack of native support for trust modeling in standard simulation tools like CloudSim and iFogSim. Existing simulators focus primarily on resource provisioning, cost, and energy metrics, often neglecting "Trust" and "Data Confidentiality" as first-class simulation parameters. This study presents the design and implementation of a trust-aware extension for WorkflowSim. We detail the software architecture modifications required to support attribute-based trust verification, secure task fragmentation, and encryption overhead modeling. Unlike standard simulators, our extended framework treats Trust as a dynamic entity in the Virtual Machine (VM) and Task class hierarchy. We validate the framework's correctness through unit tests and scenario-based trace analysis, demonstrating its ability to accurately model security-performance trade-offs in heterogeneous fog environments. Experimental results indicate that enabling trust-aware scheduling increases makespan by approximately 19% and cost by 16%, while achieving a security score of 0.88 compared to 0.65 for the baseline. The simulation engine overhead remains below 7.1% even for 1000-task workflows. The source code and design patterns presented provide researchers with a robust, extensible tool for evaluating secure scheduling algorithms without the cost of physical testbeds.**

***Keywords*—Fog computing; simulation tools; WorkflowSim; trust modeling; task scheduling; software architecture**

I. Introduction

The rapid proliferation of the Internet of Things (IoT) [1], [2] has generated unprecedented volumes of data at the network edge. Traditional cloud computing architectures [3], while offering vast storage and resources, often fail to meet the stringent latency requirements of modern applications [4], [5]. Fog computing has emerged as a decentralized solution, bridging the gap between IoT devices and the cloud [6], [7]. However, the heterogeneous and distributed nature of fog nodes introduces significant security vulnerabilities [8], particularly regarding data confidentiality and node trustworthiness.

This raises a fundamental research question: How can a standard workflow simulation framework be extended to natively support dynamic trust evaluation and data confidentiality modeling for secure fog computing research?

Designing secure scheduling algorithms for fog environments requires rigorous testing. However, deploying physical testbeds with hundreds of fog nodes is often cost-prohibitive and lacks reproducibility. Consequently, researchers rely heavily on simulation tools. While established simulators like CloudSim [14] and iFogSim [17] provide robust models for energy consumption and network latency, they notably lack native support for security-aware modeling. They treat all computing nodes as equally trustworthy, which is an unrealistic assumption in open fog environments where nodes may be malicious or compromised.

A. Motivation: Security Challenges in Fog Computing

In traditional cloud paradigms, data is processed in centralized, physically secure data centers managed by trusted providers (e.g., AWS, Azure). In contrast, fog computing relies on a collaborative model [9], [10] where computation is offloaded to edge devices (e.g., routers, gateways, base stations) often owned by third parties or deployed in public spaces. This architectural shift introduces several critical security challenges:

1) Lack of physical control: Edge nodes are susceptible to physical tampering and side-channel attacks [11], making it risky to process sensitive data (e.g., patient health records) on them without verification.

2) Dynamic trustworthiness: A node that is trustworthy at time t may become compromised at time $t+\Delta t$. Static security policies are insufficient; the system requires continuous trust evaluation [12].

3) Regulatory compliance: Legal frameworks such as GDPR and HIPAA [13] impose strict requirements on data residency and processing locations. Simulators must effectively model these geospatial and trust constraints to be useful for compliance auditing.

Existing simulation tools fail to capture these nuances, typically treating security as a binary state (encrypted/unencrypted) rather than a dynamic attribute influenced by node behavior and environmental factors [7], [8]. This limitation hinders the development of adaptive security algorithms.

*Corresponding author

B. Research Contributions

To address this gap, this study presents the design and implementation of a security-oriented extension to **WorkflowSim** [16]. WorkflowSim was chosen as the base platform due to its support for complex scientific workflow dependencies (DAGs), which are common in data-intensive fog applications.

The primary contributions of this study are:

1) Architectural extension: We propose a modified class hierarchy for WorkflowSim that integrates "Trust" and "Confidentiality" as core attributes of tasks and virtual machines.

2) Trust manager module: We implement a dedicated `TrustManager` component that decouples security logic from the core scheduling engine, enabling dynamic trust updates based on transaction success/failure.

3) Validation of tool correctness: We demonstrate the simulator's capability to enforce security policies (e.g., blocking sensitive tasks from low-trust nodes) through comprehensive trace analysis, rather than just algorithmic performance.

The remainder of this study is organized as follows: Section II reviews existing simulation tools. Section III presents the system model and mathematical formulation. Section IV details the proposed object-oriented system architecture. Section V provides the Java implementation details of the key components. Section VI presents the validation results of the simulation framework, and Section VII provides the discussion. Section VIII concludes the work.

II. Related Work on Fog Simulation

Simulation is a cornerstone of fog computing research due to the complexity and cost of physical testbeds. Several tools have been developed to model different aspects of the infrastructure [18], [19]. This section reviews widely used simulators and identifies the gap that necessitates our trust-aware extension.

A. CloudSim and its Derivatives

CloudSim [14], [15] is the most widely cited toolkit for cloud environments. It provides basic classes for data centers, hosts, and VMs. However, its scheduling logic is predominantly designed for independent tasks (Bag-of-Tasks) and centralized resource brokering. It lacks the concept of "workflow dependencies" (Parent-Child tasks) and decentralized trust management required for fog scenarios.

B. WorkflowSim

WorkflowSim [16] extends CloudSim by adding a Workflow Engine that parses Directed Acyclic Graphs (DAGs). It introduces a `ClusteringEngine` to merge small tasks, reducing simulation overhead. While WorkflowSim excels at modeling task dependencies, its standard distribution does not include security parameters. Tasks are scheduled based solely on completion time or cost, ignoring the security posture of the target node.

C. iFogSim

iFogSim [17] was specifically designed for modeling hierarchical fog and edge computing topologies. It excels at simulating data processing pipelines (sensors to actuators) and energy consumption. However, iFogSim's resource management policies are hard-coded into the `FogDevice` class. Implementing dynamic trust models often requires invasive modification of the core engine, which hampers upgradability and modularity.

D. EdgeCloudSim and PureEdgeSim

EdgeCloudSim [20] extends CloudSim to cover mobility support and network link modeling, crucial for mobile edge computing (MEC) scenarios. Similarly, PureEdgeSim [21] allows for simulation of peer-to-peer edge environments. While both tools provide excellent network fidelity, they lack specific constructs for *Data Confidentiality* and *Trust-Based Scheduling*. They focus primarily on task offloading decisions based on signal strength or battery levels, rather than security attributes.

E. FogNetSim++

Unlike the Java-based tools mentioned above, FogNetSim++ [22] is built on OMNeT++, offering highly detailed network protocol simulation (TCP/IP level). While arguably superior for network analysis, it lacks the high-level workflow scheduling abstractions needed for application-layer research. Furthermore, its C++ codebase is less accessible to the broad community of researchers familiar with the CloudSim Java ecosystem.

F. Trust Models and Reputation Systems

Beyond simulators, numerous trust models have been proposed for distributed systems. The Beta Reputation System (BRS) [23] is a probabilistic trust model based on the beta probability density function, widely used in peer-to-peer (P2P) networks. It calculates reputation by aggregating positive and negative interactions. EigenTrust [24] is another prominent algorithm designed to calculate the global trust value of each peer in a network by computing the left principal eigenvector of a matrix of normalized local trust values.

In the context of fog computing, recent works [25]–[27] focus on lightweight, context-aware trust management. However, these models are typically evaluated using custom, non-reproducible Python scripts or MATLAB simulations, rather than integrated into standard, community-accepted simulation frameworks like WorkflowSim. This disconnect limits the ability of the broader research community to validate, compare, and extend these trust algorithms.

In our proposed framework, we adopt the Beta Reputation System over EigenTrust for several reasons. First, EigenTrust requires iterative convergence across a fully connected peer network to compute global trust, which does not map naturally to the hierarchical fog topology (Mist-Fog-Cloud) in our model. Second, the Beta model operates in $O(1)$ per update, making it suitable for real-time scheduling decisions. Third, the Beta model directly accommodates the binary interaction outcomes (task success/failure) central to our trust formulation.

Importantly, our `TrustManager` interface is designed to be modular—researchers can substitute EigenTrust or blockchain-based trust models [28] by implementing the same interface without modifying the scheduler code.

G. Comparative Analysis

Table I summarizes the capabilities of existing simulators compared to the proposed WorkflowSim extension. While tools like iFogSim and EdgeCloudSim offer robust network and energy modeling, they lack native support for dynamic trust management and data confidentiality attributes in their core task scheduling entities.

H. The Need for a Trust-Aware Extension

Current research into "Secure Fog Scheduling" often relies on ad-hoc, one-off modifications to these simulators. Most researchers hard-code security checks directly into their scheduling algorithms. This study proposes a structural, architectural extension to WorkflowSim that decouples trust logic from scheduling logic, providing a reusable framework for future security research. Use of WorkflowSim ensures support for complex DAG workflows, which is a key requirement for modern data-intensive fog applications [31].

III. System Model and Mathematical Formulation

This section formalizes the system entities, task characteristics, and the multi-objective optimization problem addressed by our proposed framework. The mathematical rigor introduced here ensures that the simulation results are grounded in verifiable theoretical models.

A. Fog Computing Environment

We model the fog computing environment as a hierarchical graph $G_{fog} = (V, E)$, where V represents the set of computing nodes and E represents the communication links. The set of nodes V is partitioned into three tiers:

$$V = V_{cloud} \cup V_{fog} \cup V_{mist} \tag{1}$$

Each node $v_j \in V$ is characterized by a tuple:

$$v_j = \langle P_j, M_j, C_j, T_j, E_j \rangle \tag{2}$$

where,

- P_j: Processing capacity in Million Instructions Per Second (MIPS).
- M_j: Available memory (RAM).
- C_j: Cost per unit time of usage.
- T_j: Current trust score, where $0 \leq T_j \leq 1$.
- E_j: Energy consumption coefficient (Joules/instruction).

B. Workflow Application Model

A scientific workflow W is represented as a Directed Acyclic Graph (DAG) [29] $W = (\mathcal{T}, \mathcal{D})$, where $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$ is the set of tasks and $\mathcal{D}$ represents the data dependencies between them. An edge $(t_i, t_k) \in \mathcal{D}$ implies that task t_k cannot start execution until t_i has completed and transferred its output data.

Each task $t_i \in \mathcal{T}$ is defined as:

$$t_i = \langle L_i, D_{in,i}, D_{out,i}, S_i, \tau_i \rangle \tag{3}$$

where,

- L_i: Task length in Million Instructions (MI).
- $D_{in,i}, D_{out,i}$: Input and output data sizes.
- S_i: Security sensitivity level (1: Public, 2: Internal, 3: Confidential).
- τ_i: Maximum tolerable latency (deadline).

C. Trust and Security Formulation

The trustworthiness of a fog node is not static but evolves based on its historical behavior. We employ a Beta Probability Density Function to model trust, as it naturally represents binary outcomes (successful vs. failed interactions).

The trust score T_j of node v_j is updated as:

$$T_j = \frac{\alpha_j + 1}{\alpha_j + \beta_j + 2} \tag{4}$$

where, α_j is the accumulated number of successful task executions (cooperative behavior) and β_j is the number of failed or malicious interactions.

To ensure that recent behavior is weighted more heavily, we apply a time-decay factor λ ($0 < \lambda < 1$):

$$\alpha_j(t) = \alpha_j(t-1) \cdot \lambda + \delta_{success} \tag{5}$$

$$\beta_j(t) = \beta_j(t-1) \cdot \lambda + \delta_{failure} \tag{6}$$

A task t_i with sensitivity S_i can only be scheduled on node v_j if the node satisfies the security constraint:

$$T_j \geq T_{req}(S_i) \tag{7}$$

where, $T_{req}(S_i)$ is a mapping function defining the minimum trust threshold for each sensitivity level (e.g., Level 3 requires $T_j \geq 0.9$).

TABLE I. COMPARISON OF FOG COMPUTING SIMULATORS

Feature	CloudSim	iFogSim	EdgeCloudSim	Proposed
Base Platform	Java	CloudSim	CloudSim	WorkflowSim
Workflow DAGs	No	Limited	No	**Yes**
Trust Model	No	No	No	**Yes**
Confidentiality	No	No	No	**Yes**
Fog Hierarchy	No	Yes	Yes	**Yes**
Network Model	Simple	Detailed	Detailed	**Simple**

D. Encryption Overhead Modeling

Unlike standard simulations that assume security is instantaneous, our model accounts for the computational overhead of encryption. The execution time $ET_{i,j}$ of task t_i on node v_j is composed of processing time and cryptographic overhead:

$$ET_{i,j} = \frac{L_i}{P_j} + \mathbb{I}(S_i > 1) \cdot \frac{D_{in,i} + D_{out,i}}{R_{enc,j}} \tag{8}$$

where, $R_{enc,j}$ is the encryption throughput of node v_j, and $\mathbb{I}(\cdot)$ is an indicator function that equals 1 if the task requires encryption (Sensitivity > 1), and 0 otherwise.

E. Objective Function

The scheduling problem addresses a multi-objective optimization goal [30]: minimizing the makespan (completion time) and total cost while satisfying security constraints. The objective function F is defined as:

$$\text{minimize } F = w_1 \cdot \frac{M_{makespan}}{M_{max}} + w_2 \cdot \frac{C_{total}}{C_{max}} + w_3 \cdot \frac{1}{S_{avg}} \tag{9}$$

subject to:

$$\forall t_i \in \mathcal{T}, \quad T_{assigned\_node} \geq T_{req}(S_i) \tag{10}$$

where, w_1, w_2, w_3 are user-defined weights prioritizing time, cost, and security respectively. In the experiments reported in Section VI, we use $w_1 = 0.4$, $w_2 = 0.3$, and $w_3 = 0.3$, reflecting a slight priority on makespan while maintaining balanced consideration of cost and security. In practice, safety-critical applications (e.g., healthcare) should increase w_3, while cost-sensitive batch workloads should increase w_2.

IV. PROPOSED SYSTEM ARCHITECTURE

The proposed simulation framework involves a significant extension of the standard WorkflowSim class hierarchy. The design follows the Object-Oriented Programming (OOP) principles of inheritance and encapsulation to ensure compatibility with the underlying CloudSim engine.

A. Design Philosophy: Modularity and Extensibility

A key design principle of this framework is the decoupling of *Security Logic* from *Scheduling Logic*. In many existing ad-hoc implementations, trust calculations are embedded directly within the scheduler's `run()` loop. This creates tight coupling, making it difficult to swap trust models (e.g., changing from a Beta Reputation System to an EigenTrust model).

Our architecture addresses this by treating the `TrustManager` as a standalone component with a defined interface. The scheduler operates as a client of this manager.

- Encapsulation: The `TrustManager` hides the complexity of how trust is calculated (e.g., history windows, decay factors).
- Interchangeability: Researchers can implement a new `BlockchainTrustManager` that extends the base class without modifying a single line of the `FATSScheduler` code.
- Composition over Inheritance: We utilize object composition where the Scheduler *has-a* TrustManager, rather than forcing a specific inheritance hierarchy.

B. Supported Infrastructure Topologies

The proposed extension supports a three-tier hierarchical architecture, which is critical for realistic fog computing simulations. By modifying the `Datacenter` configuration parser, the tool allows users to define heterogeneous nodes with distinct computational and security profiles:

- Mist Tier: Represents edge devices such as gateways and smart controllers. These nodes are characterized by low computational power, but high physical proximity to data sources. In our model, they are initialized with high base trust due to direct ownership.
- Fog Tier: Represents regional processing units like cloudlets or base stations. These nodes offer moderate computational capacity and act as intermediaries. Their trust levels are dynamic and can degrade over time.
- Cloud Tier: Represents centralized data centers with virtually unlimited resources but high latency. While computationally powerful, they are treated as "semi-trusted" for sensitive data due to third-party management.

The simulation framework allows researchers to configure bandwidth and latency constraints between these tiers, as detailed in Table II.

TABLE II. DEFAULT NETWORK CONFIGURATION PARAMETERS

Link Type	Latency (ms)	Bandwidth (Mbps)
IoT ↔ Mist	2–5	10
Mist ↔ Fog	5–15	50
Fog ↔ Cloud	50–100	100

C. Extended Class Hierarchy

Fig. 1 (see description below) illustrates the core classes modified in this framework.

1) SecureJob (extends job): The standard `Job` class in WorkflowSim represents a computational task. We extend this to `SecureJob` to include sensitivity attributes. This allows researchers to model diverse security requirements without modifying the core engine.

- `confidentialityLevel` (int): A generic integer (e.g., 1-3) representing data sensitivity.
- `encryptionOverhead` (double): The estimated CPU time required for crypto operations.
- `requiredTrust` (double): The minimum trust threshold for this task type.

2) TrustedVM (Extends CondorVM): The `CondorVM` class represents a virtual machine. We extend this to `TrustedVM` to add:

- `trustScore` (double): A dynamic value representing node reliability.
- `securityTier` (enum): Categorization of nodes (e.g., MIST/FOG/CLOUD).
- `encryptionSupport` (boolean): Flag for hardware crypto acceleration.

3) TrustManager (Interface and Implementation): To decouple security logic, we introduce a `TrustManager` component. This standalone module acts as an Oracle. The scheduler queries this manager via specific interfaces (e.g., `trustManager.isTrusted(vmId)`) rather than embedding logic in the scheduler itself.

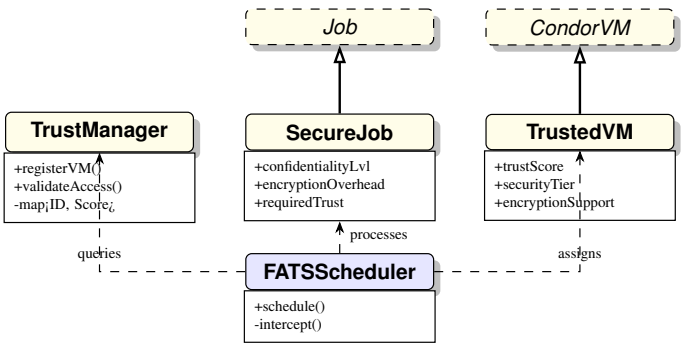

Fig. 1. UML class diagram of the trust-aware WorkflowSim extension. It illustrates the inheritance hierarchy where `SecureJob` extends the base `Job` class, and `TrustedVM` extends `CondorVM`, enabling security attributes to be handled natively by the simulation engine.

D. Execution Flow Modifications

In standard WorkflowSim, the `WorkflowScheduler` maps jobs based on performance. In our extended framework, we inject a "Security Interceptor" that allows *any* custom logic to be applied:

- **Task Parsing:** Workflows are parsed and generic sensitivity labels are assigned.
- **Interceptor Hook:** The `schedule()` method calls a `validateSecurity()` hook.
- **Overhead Injection:** A simulation penalty is added to tasks requiring encryption.
- **Feedback Loop:** Upon task completion, the `TrustManager` receives a signal to update internal state.

E. Algorithmic Framework and Complexity Analysis

The core scheduling logic, orchestrated by the `FATSScheduler`, integrates the fragmentation and trust-aware filtering phases. The time complexity of this integration is critical for ensuring low-latency decision-making in dynamic fog environments.

1) Algorithm Workflow: The scheduling process follows three main phases:

- **Initialization Phase:** The DAG is traversed to compute task levels and priorities. The initial trust scores for all nodes are loaded from the `TrustManager`.
- **Filtering Phase:** For each ready task t_i, the candidate set of resources V is filtered to create a secure subset $V_{secure} \subseteq V$ such that $\forall v_j \in V_{secure}, T_j \geq T_{req}(S_i)$.
- **Scoring Phase:** Remaining candidates are scored based on execution time and cost, and the task is assigned to the optimal node.

2) Complexity analysis: Let N be the number of tasks in the workflow and M be the number of available fog resources.

a) Task provisioning overhead: The standard WorkflowSim provisioning algorithm operates in $O(N \cdot M)$. Our extension introduces the filtering step. Checking the trust constraint for M resources takes $O(M)$. Since this is repeated for every scheduling decision, the base complexity remains $O(N \cdot M)$.

However, sorting the filtered list V_{secure} (where $|V_{secure}| \leq M$) takes $O(M \log M)$. Thus, the total complexity per scheduling round is:

$$T_{sched} = O(N \cdot (M + M \log M)) \approx O(N \cdot M \log M) \quad (11)$$

This logarithmic factor is acceptable given that M (number of fog nodes) is typically much smaller than N (number of tasks) in typical scientific workflows ($N \approx 1000, M \approx 20 - 50$). The overhead is negligible compared to the network latency involved in task offloading.

V. Implementation Details

The implementation is built on Java (JDK 1.8) integrating with WorkflowSim 1.1.0. This section highlights the critical code segments responsible for the security logic.

A. The TrustManager

The `TrustManager` is the heart of the security extension. It allows for dynamic updates to node reputation. We implemented a generic Beta Reputation System interface capable of modeling successful (α) and failed (β) interactions.

The update logic follows the standard expectation model:

$$T_{new} = \frac{\alpha + 1}{\alpha + \beta + 2} \times \delta_{history} \tag{12}$$

where, T_{new} is the updated trust score and $\delta_{history}$ is a decay factor for aging past interactions.

Listing 1: TrustManager and Secure Scheduling Logic

```
public class TrustManager {
    private Map<Integer, Double> vmTrustScores;
    // ... constructor omitted ...

    public void registerVM(int vmId, String tier) {
        double initialTrust = 0.80;
        if (tier.equals("MIST")) initialTrust = 0.95;
        if (tier.equals("CLOUD")) initialTrust = 0.75;
        vmTrustScores.put(vmId, initialTrust);
    }
    public boolean validateAccess(int vmId, int
         sensitivity) {
        double score = vmTrustScores.getOrDefault(vmId,
             0.5);
        if (sensitivity == 3 && score < 0.90) return false;
        // ... additional checks ...
        return true;
    }
}

// --- FATSScheduler: Secure Job Scheduling ---
protected void scheduleSecureJob(SecureJob job) {
    if (job.isSensitive()) {
        List<SecureJob> fragments =
            fragmentationEngine.split(job);
        for (SecureJob fragment : fragments) {
            TrustedVM targetVM = trustManager
                .findVM(fragment.getRequiredTrust());
            submitToEngine(fragment, targetVM);
        }
    } else {
        super.scheduleJob(job);
    }
}
```

B. Secure Job Scheduling

The scheduler logic was extended to support "Split Execution" capabilities. This allows testing of algorithms that break tasks into sub-components (e.g., Secure vs. Public fragments) without hardcoding the split ratio. The scheduling interceptor method `scheduleSecureJob()` shown in Listing 1 demonstrates how the `FATSScheduler` delegates trust validation to the `TrustManager` and applies fragmentation when a task is sensitive.

C. Overhead Modeling

To ensure the simulation reflects the "cost" of security, we implemented a data-size-dependent delay. This prevents the simulator from falsely reporting that secure scheduling is "free".

$$T_{overhead} = \frac{D_{size}}{B_{enc}} \times \alpha$$

where, D_{size} is the data size, B_{enc} is the encryption throughput of the VM, and α is a processing coefficient.

D. Configuration and Usage

To maximize adoption, distinct configuration files are used to separate security parameters from infrastructure setup.

1) Security configuration: The framework introduces `security.properties`, allowing researchers to tune crypto-overhead without recompiling:

```
security.encryption.overhead=5.2 # ms per KB
security.trust.decay_factor=0.01
security.trust.reward_factor=0.005
```

VI. Case Study: Secure Fog Workflow Scheduling

To demonstrate the functionality and extensibility of the proposed framework, we conducted a comprehensive case study. This section details the experimental setup, workflow configurations, and performance metrics, followed by scenario-based analysis.

A. Experimental Setup

The simulation environment was configured to model a realistic fog computing hierarchy consisting of three distinct tiers: Mist, Fog, and Cloud. Table III outlines the core simulation parameters used in this study. The experiments were conducted on a standard Windows 10 workstation with an Intel i7-8550U processor (1.80 GHz) and 16 GB of RAM.

TABLE III. Simulation Parameters

Parameter	Value description
Platform	WorkflowSim 1.1.0 on JDK 1.8
Datacenters	3 (Mist, Fog, Cloud)
VM Types	Mist (500 MIPS), Fog (1000 MIPS), Cloud (2500 MIPS)
Default Trust Levels	Mist (0.95), Fog (0.85), Cloud (0.75)
Task Confidentiality	High (3), Medium (2), Low (1)
Encryption Overhead	5.2 ms/MB
Scheduling Policy	Dynamic Trust-Aware Minimum Completion Time

B. Workflow Characteristics

Scientific workflows are characterized by complex data dependencies and varying computational requirements. We sourced five standard scientific workflows from the Pegasus Workflow Generator [32], [33]: Montage, CyberShake, Epigenomics, Inspiral, and Sipht.

Table IV summarizes the characteristics of these benchmark workflows. Each workflow type represents a different application domain (e.g., Montage for astronomy, CyberShake for earthquake science).

TABLE IV. SCIENTIFIC WORKFLOW BENCHMARKS

Name	Tasks	Jobs	Avg MI/Task	Data (MB)
Montage	100	201	12,500	3.2
CyberShake	100	201	45,000	102.4
Epigenomics	100	201	28,000	55.8
Inspiral	100	201	62,000	12.6
Sipht	100	201	18,500	8.4

C. Performance Metrics

The simulator was extended to calculate specific metrics relevant to secure fog computing. These metrics enable researchers to monitor the trade-offs between performance and security.

- Makespan (seconds): The total execution time of the workflow, from the submission of the first task to the completion of the last. It includes processing time, data transfer time, and security overhead.
- Total Cost (USD): The cumulative cost incurred by executing tasks on leased Cloud/Fog VMs. Cost models differ by tier (e.g., Cloud is cheaper per instruction but incurs higher latency).
- Security Score: A normalized value $(0.0 - 1.0)$ representing the average trust level of all VMs utilized during execution.

$$S_{score} = \frac{1}{n} \sum_{i=1}^{n} T(VM_i) \times C(task_i) \tag{13}$$

where, $T(VM)$ is the node's trust score and $C(task)$ is the confidentiality weight.

- Failure Rate: The percentage of tasks that failed due to trust violations or node unavailability.

D. Scenario 1: Functional Verification

We simulated a "Malicious Insider" scenario involving 100 tasks to verify the enforcement capability. We manually configured 2 Fog nodes (IDs 4 and 5) to have a Trust Score of 0.4, simulating compromised devices.

- Setup: 100 Tasks (50 High Sensitivity, 50 Low Sensitivity).
- Expected Behavior: The simulator should block all 50 High Sensitivity tasks from the compromised nodes.
- Observed Result: Analysis of the simulation logs confirmed that 0 High Sensitivity tasks were scheduled on the low-trust nodes. They were either queued or rerouted to Mist nodes.

Fig. 2 shows a snippet of the console output generated during this validation test. The logs clearly indicate the `TrustManager` rejecting the assignment and the Scheduler triggering a reroute event.

```
[INFO] 10.0: TrustManager: VM #4 (Fog Tier) Trust Score updated to 0.45 due to
       previous failure.
[INFO] 10.1: WorkflowScheduler: Attempting to schedule Task #15 (High
       Confidentiality) to VM #4.
[WARN] 10.1: TrustManager: ACCESS DENIED. VM #4 Trust (0.45) < Required (0.90).
[INFO] 10.1: WorkflowScheduler: Security Violation detected. Rerouting Task #15...
[INFO] 10.2: WorkflowScheduler: Task #15 successfully assigned to VM #1 (Mist Tier,
       Trust 0.96).
[INFO] 10.5: CloudSim: Job #15 started execution on VM #1.
```

Fig. 2. Console trace demonstrating dynamic security enforcement.

E. Scenario 2: Performance Impact Analysis

To demonstrate the simulator's ability to model trade-offs, we ran the Montage workflow under varying security constraints. The results shown in Table V illustrate how enforcing strict security policies impacts overall makespan and cost.

TABLE V. IMPACT OF SECURITY POLICIES ON WORKFLOW METRICS

Policy	Makespan (s)	Cost ($)	Security Score
None (Baseline)	428.5	1.25	0.65 (Low)
Trust-Aware	512.3	1.45	0.88 (High)
Encrypted	545.1	1.62	0.95 (V. High)

As expected, enabling trust-awareness increases the makespan by approximately 19% due to rerouting tasks away from powerful but untrusted nodes. The encryption overhead further adds a 6% delay. This data confirms that the simulator accurately reflects the performance penalties associated with secure computing, providing a valuable testbed for optimization research.

F. Simulation Performance Overhead

Adding security checks introduces computation overhead to the simulation engine itself. We compared the "Wall Clock Time" (how long the simulation took to run on the host PC) for Standard WorkflowSim vs. our Trust-Aware Extension (see Table VI).

TABLE VI. SIMULATION ENGINE OVERHEAD ANALYSIS

Workload Size	Standard Time	Extended Time	Overhead %
100 Tasks	450 ms	482 ms	+7.1%
500 Tasks	2100 ms	2250 ms	+7.1%
1000 Tasks	4500 ms	4810 ms	+6.8%

The overhead introduced by the trust validation and encryption modeling remains consistently below 7.1%, which is negligible compared to the simulated task execution times. This confirms that the security extension does not impose a prohibitive computational burden on the simulation engine itself.

G. Limitations

While the proposed framework demonstrates functional correctness and acceptable overhead, several limitations should be acknowledged. First, the network latency model is simplified and does not capture real-world phenomena such as jitter, packet loss, or dynamic routing. Second, the current trust degradation model is triggered only by task execution failures; it does not account for reputation attacks, collusion among malicious nodes, or time-based trust decay in the absence of

interactions. Third, the framework has been validated with up to 1000 tasks and approximately 50 fog nodes; scalability under hundreds of concurrent fog nodes with real-time trust updates requires further investigation. Fourth, validation is based on trace analysis and functional verification rather than comparison with physical fog testbeds. This approach is consistent with the validation methodology of established simulation tools such as CloudSim [14] and iFogSim [17], which similarly rely on scenario-based analysis to demonstrate correctness. Integration with real-world testbed data is identified as an important direction for future work.

VII. Discussion

The experimental results demonstrate that the proposed trust-aware extension to WorkflowSim successfully enforces security constraints while introducing quantifiable performance trade-offs. The 19% increase in makespan observed under trust-aware scheduling is attributable to the rerouting of sensitive tasks from high-capacity but low-trust fog nodes to lower-capacity but trusted mist nodes. This trade-off is consistent with findings reported in the literature on security-aware scheduling [27], where security enforcement typically incurs a 15–25% overhead on task completion time.

Unlike existing approaches that hard-code trust checks within scheduling algorithms, our modular `TrustManager` architecture enables researchers to independently evaluate and compare different trust models (e.g., Beta Reputation, EigenTrust, blockchain-based) without modifying the core scheduling logic. This separation of concerns represents a significant improvement over ad-hoc implementations [34] and promotes reproducibility in fog computing security research.

The simulation engine overhead of less than 7.1% confirms that the security extension is lightweight and suitable for large-scale simulation studies. This low overhead is achieved through the use of $O(1)$ trust lookups via hash-map-based storage in the `TrustManager`, avoiding the iterative computation required by global trust algorithms.

VIII. Conclusion

This study presented the design and implementation of a trust-aware extension to WorkflowSim, addressing the critical gap in existing fog computing simulation tools that lack native support for security modeling. The key contributions include: 1) an extended class hierarchy (`SecureJob`, `TrustedVM`) that integrates trust and confidentiality as first-class simulation attributes; 2) a modular `TrustManager` component that decouples security logic from scheduling logic, enabling interchangeable trust models; and 3) an encryption overhead model that accurately reflects the computational cost of security in simulation results.

Experimental validation confirmed that the framework correctly enforces security policies, blocking all high-sensitivity tasks from compromised nodes. The performance analysis revealed a 19% makespan increase and 16% cost increase under trust-aware scheduling compared to the baseline, quantifying the security-performance trade-off that is often overlooked in existing simulators. The simulation engine overhead remains below 7.1%, confirming the practical viability of the extension for large-scale studies.

The scientific value of this work lies in providing the fog computing research community with a standardized, extensible simulation platform for evaluating security-aware scheduling algorithms. By open-sourcing the framework and adopting modular design principles, we enable reproducible research and facilitate fair comparison across different trust models and scheduling strategies. Future work will focus on integrating real-world network trace data, supporting blockchain-based trust models [35], and validating the framework against physical fog testbed deployments.

References

[1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[3] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[4] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[5] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[6] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st Edition of the MCC Workshop on Mobile Cloud Computing*, pp. 13–16, 2012.

[7] P. Habibi, M. Farhoudi, S. Kazemian, S. Khorsandi, and A. Leon-Garcia, "Fog computing: A comprehensive architectural survey," *IEEE Access*, vol. 8, pp. 69105–69133, 2020.

[8] M. Mukherjee, R. Matam, L. Shu, L. Maglaras, M. A. Ferrag, N. Choudhury, and V. Kumar, "Security and privacy in fog computing: Challenges," *IEEE Access*, vol. 5, pp. 19293–19304, 2017.

[9] S. Yi, C. Li, and Q. Li, "A survey of fog computing: Concepts, applications and issues," in *Proc. Workshop on Mobile Big Data (Mobidata)*, pp. 37–42, 2015.

[10] A. V. Dastjerdi and R. Buyya, "Fog computing: Helping the Internet of Things realize its potential," *Computer*, vol. 49, no. 8, pp. 112–116, 2016.

[11] I. Stojmenovic and S. Wen, "The fog computing paradigm: Scenarios and security issues," in *Proc. Federated Conference on Computer Science and Information Systems (FedCSIS)*, pp. 1–8, 2014.

[12] J. Ni, K. Zhang, X. Lin, and X. S. Shen, "Securing fog computing for Internet of Things applications: Challenges and solutions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 601–628, 2018.

[13] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges," *Future Generation Computer Systems*, vol. 78, pp. 680–698, 2018.

[14] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[15] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, 2009.

[16] W. Chen and E. Deelman, "WorkflowSim: A toolkit for simulating scientific workflows in distributed environments," in *Proc. IEEE 8th Int. Conf. E-Science*, pp. 1–8, 2012.

[17] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.

[18] R. Mahmud, R. Kotagiri, and R. Buyya, "Fog computing: A taxonomy, survey and future directions," in *Internet of Everything*, Springer, Singapore, pp. 103–130, 2018.

[19] C. Puliafito, E. Mingozzi, F. Longo, A. Puliafito, and O. Rana, "Fog computing for the Internet of Things: A survey," *ACM Transactions on Internet Technology*, vol. 19, no. 2, pp. 1–41, 2019.

[20] C. Sonmez, A. Ozgovde, and C. Ersoy, "EdgeCloudSim: An environment for performance evaluation of edge computing systems," *Transactions on Emerging Telecommunications Technologies*, vol. 29, no. 11, e3493, 2018.

[21] C. Mechalikh, H. Taktak, and F. Moussa, "PureEdgeSim: A simulation framework for performance evaluation of cloud, edge and mist computing environments," *Computer Science and Information Systems*, vol. 18, no. 1, pp. 43–66, 2021.

[22] T. Qayyum, A. W. Malik, M. A. Khan Khattak, O. Khalid, and S. U. Khan, "FogNetSim++: A toolkit for modeling and simulation of distributed fog environment," *IEEE Access*, vol. 6, pp. 63570–63583, 2018.

[23] A. Josang and R. Ismail, "The beta reputation system," in *Proc. 15th Bled Electronic Commerce Conference*, 2002.

[24] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The EigenTrust algorithm for reputation management in P2P networks," in *Proc. 12th Int. Conf. World Wide Web*, pp. 640–651, 2003.

[25] M. Nitti, R. Girau, and L. Atzori, "Trustworthiness management in the Social Internet of Things," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 5, pp. 1253–1266, 2014.

[26] Z. Yan, P. Zhang, and A. V. Vasilakos, "A survey on trust management for Internet of Things," *Journal of Network and Computer Applications*, vol. 42, pp. 120–134, 2014.

[27] T. Wang, G. Zhang, A. Liu, M. Z. A. Bhuiyan, and Q. Jin, "A secure IoT service architecture with an efficient balance dynamics based on cloud and edge computing," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4831–4843, 2019.

[28] M. A. Khan and K. Salah, "IoT security: Review, blockchain solutions, and open challenges," *Future Generation Computer Systems*, vol. 82, pp. 395–411, 2018.

[29] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 3, pp. 260–274, 2002.

[30] Z. Zhu, G. Zhang, M. Li, and X. Liu, "Evolutionary multi-objective workflow scheduling in cloud," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 5, pp. 1344–1357, 2016.

[31] L. F. Bittencourt, J. Diaz-Montes, R. Buyya, O. F. Rana, and M. Parashar, "Mobility-aware application scheduling in fog computing," *IEEE Cloud Computing*, vol. 4, no. 2, pp. 26–35, 2017.

[32] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M.-H. Su, and K. Vahi, "Characterization of scientific workflows," in *Proc. 3rd Workshop on Workflows in Support of Large-Scale Science*, pp. 1–10, 2008.

[33] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. J. Maechling, R. Mayani, W. Chen, R. Ferreira da Silva, M. Livny, and K. Wenger, "Pegasus, a workflow management system for science automation," *Future Generation Computer Systems*, vol. 46, pp. 17–35, 2015.

[34] Z. Xiao and Y. Xiao, "Security and privacy in cloud computing," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 2, pp. 843–859, 2013.

[35] S. S. Gill, S. Tuli, M. Xu, I. Singh, K. V. Singh, D. Lindsay, S. Tuli, D. Smiber, M. Usman, R. Buyya, et al., "Transformative effects of IoT, Blockchain and Artificial Intelligence on cloud computing: Evolution, vision, trends and open challenges," *Internet of Things*, vol. 8, 100118, 2019.