Large-Small Model Cooperative Threat Deception in Space-Air-Ground Integrated Vehicular Networks

Hang Shen, *Member, IEEE,* Juan Huang, Tianyuan Hu, Tianjing Wang, *Member, IEEE,* and Guangwei Bai

***Abstract*—Space-air-ground integrated vehicular networks (SAGVNs) face coordinated and stealthy attacks across heterogeneous tiers, while existing honeypots typically operate independently and lack cross-tier deception consistency. To address this challenge, this paper presents a large–small model cooperative framework for coordinated threat deception in SAGVNs. A shared strategy-atom knowledge base supports cross-tier threat reasoning, coordinated task planning, and knowledge-guided local execution. Edge-deployed large language models (LLMs) retrieve reusable strategy atoms through the model context protocol (MCP) to orchestrate deception tasks, while on-device small language models (SLMs) use the conveyed strategy semantics to generate protocol- and environment-specific honeypot responses. Validated threat knowledge is fed back to the knowledge base for strategy reuse and evolution. A consortium blockchain spanning the ground, air, and space tiers supports auditable cross-tier task coordination, evidence provenance, and reward settlement. Building on this infrastructure, a dual-role incentive mechanism rewards threat captors according to validated knowledge gain and deception effectiveness, while compensating collaborators based on strategy-atom provenance. Experiments benchmark the proposed framework against rule-based Cowrie, cloud-hosted LLM honeypots, a recent LLM-based multi-agent approach, and locally deployed SLM variants. The coordinated configuration improves average attack depth and session duration by 10.6% and 7.7%, respectively, over its same-backbone non-coordinated counterpart, while consuming 88.5% fewer tokens than the multi-agent cooperative baseline.**

***Index Terms*—Space-air-ground integrated vehicular networks (SAGVNs), large-small model cooperation, honeypot, coordinated threat deception, consortium blockchain.**

I. Introduction

VEHICULAR networks integrate communication, sensing, and computing capabilities to support intelligent transportation services. In space–air–ground integrated vehicular networks (SAGVNs), vehicles access network services through heterogeneous links involving low-earth orbit (LEO) satellites, drone relays, and terrestrial vehicle-to-everything (V2X) infrastructures, enabling wide-area and cross-tier connectivity [1]. Such heterogeneity also enlarges the attack surface and exposes SAGVNs to threats including signal spoofing, denial-of-service, man-in-the-middle attacks, and malicious code injection [2]. More importantly, vulnerabilities across different tiers can be exploited in a coordinated manner. For example, compromises in in-vehicle networks may facilitate cross-tier data exfiltration [3], while vulnerable onboard modules may enable remote code execution, further compromising vehicles or terrestrial infrastructure [4]. These coordinated and evolving attacks are difficult to address using defenses based primarily on predefined rules [5] or handcrafted features [6].

Honeypots are an active defense technique that emulates realistic system environments to lure attackers, enabling attack capture and threat attribution [7], [8]. Existing deployments mainly target relatively static environments such as enterprise networks and data centers [9], and typically operate independently with predefined interaction logic. This limits their ability to adapt deception behavior and maintain attack-context consistency across heterogeneous nodes. Since SAGVN attack surfaces span onboard sensors, V2X modules, in-vehicle protocols, and cross-tier services [10], threat evidence captured at one node can inform deception at other tiers. For example, weak-password or vehicle-scanning evidence can guide related deception tasks at drone and LEO nodes. Effective cross-tier deception thus requires global threat reasoning and coordinated node-specific honeypot behaviors.

Large language models (LLMs) can integrate multi-source situational data, enabling effective security analysis [11] and high-level semantic reasoning [12]. With retrieval-augmented generation (RAG) [13] and contextual reasoning [14], LLMs can uncover semantic correlations among cross-network attacks. However, deploying LLMs on vehicles, drones, and LEO platforms is constrained by computing and memory resources [15]. In contrast, small language models (SLMs) [16], with lightweight architectures and flexible deployment, are ideal for execution on resource-constrained nodes, supporting localized semantic execution and honeypot behavior generation. This complementarity provides a practical basis for cross-tier deception that combines global threat understanding with lightweight local execution.

A. Challenges and Related Works

Despite the increasing interest in large–small model cooperation and intelligent honeypot systems, coordinated threat deception in SAGVNs faces three major challenges.

Manuscript received day month year; revised day month year; accepted day month year. Date of publication day month year; date of current version day month year. This work was supported in part by the National Natural Science Foundation of China under Grants 61502230, 61501224, and 62502202, in part by the Natural Science Foundation of Jiangsu Province under Grants BK20250594 and BK20201357, in part by the Natural Science Foundation of the Higher Education Institutions of Jiangsu Province under Grant 25KJB120004, in part by the State Administration for Market Regulation Science and Technology Program under Grant 2025MK183, and in part by the Market Supervision Administration Science and Technology Fund of Jiangsu Province under Grants KJ2026027 and KJK2027027. (Corresponding author: Tianyuan Hu)

Hang Shen, Juan Huang, Tianyuan Hu, Tianjing Wang, and Guangwei Bai are with the College of Computer and Information Engineering (College of Artificial Intelligence), Nanjing Tech University, Nanjing, 211816, China. E-Mail: {hshen, tianyuan.hu, wangtianjing, bai}@njtech.edu.cn, hhhj95884@gmail.com

Digital Object Identifier: xxx

1) Closed-loop semantic cooperation of large–small models. It is a fundamental issue. Prior studies on cloud–edge-device cooperation provide partial insights. Liu et al. [17] proposed a RAG mechanism that uses a speculative aggregation algorithm to fuse edge- and cloud-side knowledge to support cross-domain threat reasoning. MoRSE [18] is a cybersecurity-oriented RAG framework that relies on parallel RAG retrievers to access non-parametric, continuously updated security knowledge bases. Xiong et al. [19] further introduced a security purification signature mechanism that encapsulates cloud-side LLMs into obfuscated models, enabling secure and cooperative model delivery across edge networks while preventing direct exposure at the edge. LLM-based multi-agent systems have been explored to enhance security in space-air-ground integrated networks. SAG-Attack and LLM-SA [2] simulate attacks and perform adaptive security assessments, with SAG-Attack mimicking threats and LLM-SA fusing heterogeneous data. However, they stop short of bridging threat understanding with executable deception across heterogeneous nodes.

2) Coordinated threat deception. Existing honeypot research primarily focuses on improving local interaction or deployment decisions. Reinforcement learning (RL) has been used to adapt honeypot responses and allocations from attacker feedback, enabling sustained engagement and informative attack traces [20], [21]. More recent work extends RL to honeypot allocation through hierarchical and multi-agent formulations that account for intrusion stages and asset criticality [22]. LLM-guided RL has also been explored for adaptive cybersecurity orchestration, integrating strategy-atom retrieval, LLM reasoning, and dynamic-action optimization to accommodate evolving threats [23]. Beyond RL, learning-driven honeypots enhance realism and interactivity by generating data-driven responses. Pretrained deep learning models have been used to simulate realistic traffic and enhance protocol-level behavior emulation [24]. More recently, LLM-based honeypots employ prompt engineering and knowledge-augmented retrieval, such as GraphRAG, to generate realistic and context-consistent deceptive interactions [25]–[27]. Nevertheless, existing studies mainly address local honeypot interactions or security-action orchestration, leaving knowledge-guided deception coordination insufficiently explored.

3) Contribution-aware incentives for collaboration. In coordinated threat deception, network nodes simultaneously serve as honeypot strategy providers and threat captors, which fundamentally complicates reward allocation. Prior studies have considered contract-based incentives in related settings. He et al. [28] proposed a vehicular threat-sharing framework with a two-stage dynamic contract that adapts to vehicles' privacy preferences to sustain participation. For cooperative drone network defense, Wang et al. [29] developed a contract theory- and RL-based incentive scheme that employs a hill-climbing algorithm for fair and efficient benefit allocation under information asymmetry. A consortium blockchain framework for cross-operator V2X content distribution is developed in [30], incorporating reputation-constrained vehicle selection and incentives for collaboration. Wu et al. [31] incorporated reputation and contract theory into reward allocation to encourage sustained contributions from high-quality nodes. AI-generated content (AIGC)-oriented incentives have incorporated accuracy, token consumption, and timeliness [32]. However, these mechanisms do not jointly account for deception effectiveness and contributions from provenance-traced strategies.

B. Contributions and Organization

To address the challenges of coordinated threat deception in SAGVNs, we propose a large–small model cooperative framework that integrates cross-tier threat reasoning, coordinated deception, and contribution-aware incentives, supported by parallel consortium blockchains for auditable coordination and provenance tracking. The main contributions of this work are threefold:

- **Closed-loop LLM–SLM cooperative deception.** Edge-deployed LLMs analyze validated cross-tier threat observations and orchestrate deception tasks, while on-device SLMs generate protocol- and environment-specific honeypot responses. Parallel consortium blockchains support auditable task coordination and evidence provenance, while validated threat knowledge is fed back for subsequent deception planning.
- **Strategy-atom-enabled coordinated deception.** We formalize threat deception using reusable strategy atoms and define honeypot strategies as structured atom compositions. A shared knowledge base enables MCP-based strategy-atom retrieval and composition to guide coordinated deception across heterogeneous network tiers.
- **Dual-role contribution-aware incentives.** Threat captors are rewarded according to validated knowledge gain and deception effectiveness, while collaborators are rewarded through on-chain atom provenance. Smart contracts automate settlement, and the captor-side allocation satisfies weighted proportional fairness under a fixed budget.

We implement a SAGVN honeypot testbed using FISCO-BCOS [1] and Beelzebub[2], covering 30 attack tasks across six representative scenarios. The evaluation compares rule-based Cowrie[3], cloud-hosted LLM honeypots based on GPT-4o[4] and DeepSeek-V4-Flash[5], the multi-agent cooperative AdvancedShelLM[6] [33], and locally deployed SLM variants with different backbones and model scales. To assess the effectiveness and practicality of the proposed framework, we seek to answer the following research questions:

- **RQ1:** How does task-specific fine-tuning affect SLM honeypot response quality across different backbones?
- **RQ2:** How does cross-tier coordination affect deception effectiveness and consistency, and what are its associated token and latency costs?
- **RQ3:** How does the availability of strategy atoms affect coordinated deception performance?
- **RQ4:** Can the dual-role incentive distinguish contribution quality and suppress low-value or invalid submissions?

[1] https://fisco-bcos-documentation.readthedocs.io/zh-cn/latest/
[2] https://docs.beelzebub.ai/
[3] https://github.com/cowrie/cowrie
[4] https://chatgpt.com/
[5] https://huggingface.co/deepseek-ai/DeepSeek-V4-Flash
[6] https://github.com/stratosphereips/shelLM/tree/main/AdvancedshelLM

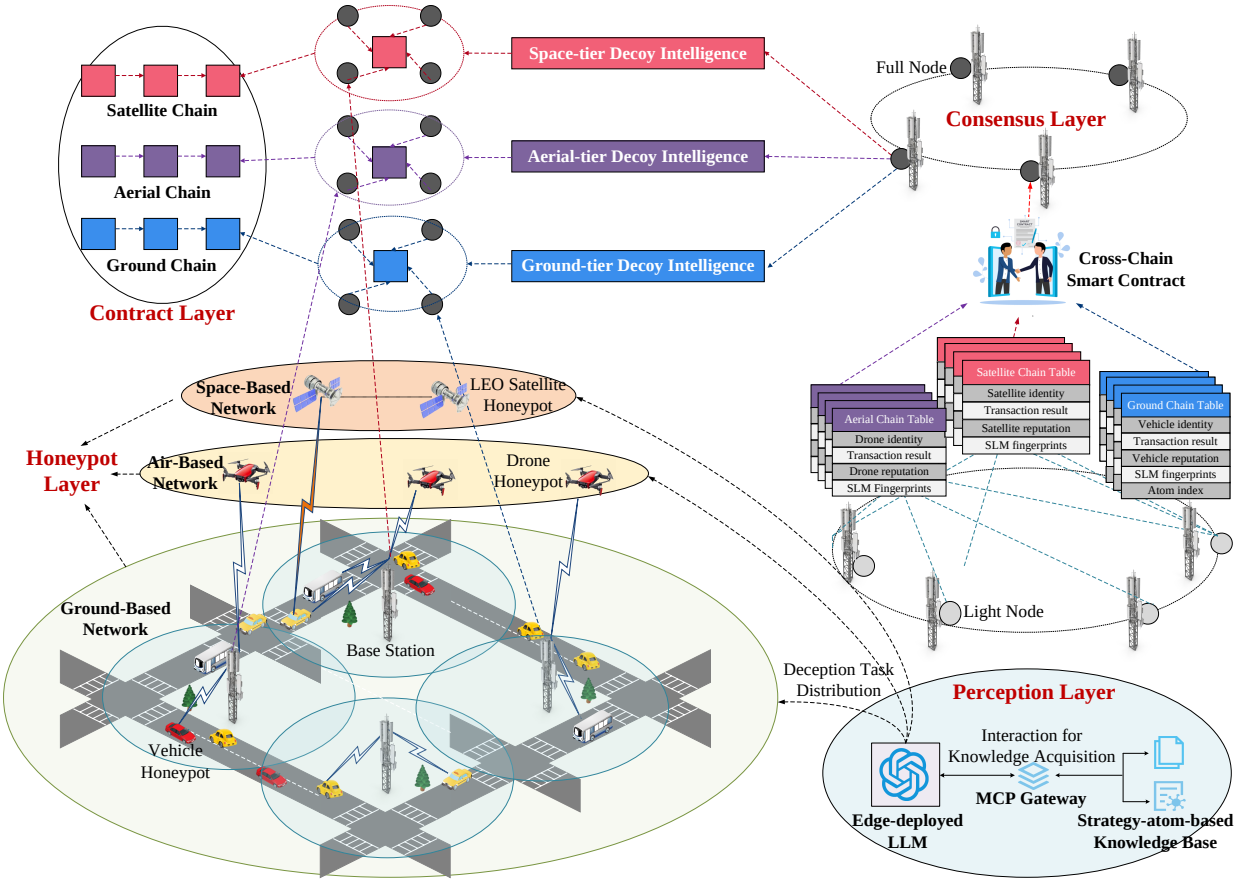


Fig. 1: Security SAGVN architecture for large-small model cooperation.

The experimental code, prompt templates, and analysis scripts are publicly available in our GitHub repository[7].

The remainder of this paper is organized as follows. Section II presents the consortium-blockchain-supported large–small model cooperation framework. Section IV develops the dual-role incentive mechanism. Sections V and VI describe the experimental setup and performance evaluation, respectively. Section VII concludes the paper and outlines future work. The main variables and symbols are summarized in Table I.

II. Consortium Blockchain-Based Framework

Fig. 1 illustrates the proposed large–small model cooperative framework for SAGVNs, where parallel consortium blockchains support coordination among independently managed ground, air, and space network domains.

Nodes register on their respective chains with blockchain addresses bound to reputation and incentive accounts. Each chain maintains node identities, reputations, and task states, while cross-chain smart contracts coordinate deception tasks across tiers. An off-chain strategy-atom knowledge base provides a shared knowledge foundation for large–small model cooperation. Authorized edge LLMs retrieve relevant strategy atoms through MCP and incorporate their semantics into coordinated deception tasks, which subsequently guide on-device SLM execution. Validated threat knowledge can be fed

[7]https://github.com/shen-hang/Large-Small-Model

TABLE I: Main Notations and Variables

Symbol	Description
a_i	On-chain address of node i
$\mathcal{D}_c$	Honeypot strategy for threat category c
d_k	Semantic description of strategy atom k
$d_{k,m}$	Strategy-atom description contributed by captor m
g	Stake required for task participation
h_j	Knowledge-base reference of task j
$\mathcal{I}_j$	Set of nodes accepting task j
$\mathcal{J}_c$	Set of tasks for threat category c
$\mathcal{K}_c$	Strategy-atom set for threat category c
L_m	Captured threat interaction data for task j
$\mathcal{M}_j/\mathcal{N}_j$	Set of captors/collaborators for task j
pk_i/sk_i	Public/private key pair of node i
pk_j/sk_j	Public/private key pair of task j
R_j	Captor-side reward budget of task j
r_m/r_n	Rewards of captor m/collaborator n
$\mathcal{S}_j$	Key-share holder set for task j
$s_c^{(\min)}$	Minimum acceptable threat score for category c
$s_{k,m}$	Atom-level threat score of captor m
v	Normalized semantics of a validated threat
w_m	Reward weight of captor m
x_k	Binary indicator for activating strategy atom k
z_j	Expiration time of task j
$\alpha_{k,m}$	Semantic novelty of $d_{k,m}$
δ_m	Evidence-validation indicator for captor m
γ_l	Tier-dependent reward weight for network tier l
κ_m	Deception-effectiveness factor of captor m
ρ_i/ρ_m	Reputation of node i/captor m
$\rho^{(\min)}$	Minimum reputation for task participation

back into the repository for strategy reuse and evolution. The blockchain records task references and provenance rather than knowledge content, while the validity of threat observations is established through separate evidence validation.

From a functional perspective, the framework is organized into four logical layers:

- *Perception Layer*: Edge-deployed LLMs analyze validated threat observations and generate coordinated deception tasks using strategy-atom knowledge.
- *Honeypot Layer*: On-device SLMs translate the strategy semantics conveyed by assigned tasks into protocol- and environment-specific honeypot responses. Raw interaction data are retained off-chain, while their cryptographic hashes are recorded on-chain.
- *Contract Layer*: Smart contracts manage task coordination, evaluation records, and incentive settlement across the parallel chains.
- *Consensus Layer*: Consensus nodes validate blockchain transactions and maintain tamper-evident records of task states, evidence hashes, and participant reputations.

A. LLM-Driven Security Situation Analysis

Building upon the consortium blockchain-based collaboration framework, edge-deployed LLMs transform validated threat observations into deception strategies for downstream orchestration and execution.

Definition 1 (Strategy Atom): A strategy atom is the minimum reusable knowledge unit in the deception knowledge base, describing a single vulnerability characteristic, attack incentive, or deception trigger condition. Each strategy atom k is associated with a semantic description d_k.

Definition 2 (Honeypot Strategy): A honeypot strategy for threat category c is a group of activated strategy atoms that jointly specify the deception semantics for downstream task orchestration.

To illustrate coordinated task generation, Fig. 2 presents Web API and SSH examples that map validated threat summaries to their coordinated tasks. For a validated threat instance, let v denote its normalized semantic description. The LLM first associates v with a threat category c and retrieves the corresponding strategy-atom set $\mathcal{K}_c$ through MCP.

For strategy atom $k \in \mathcal{K}_c$ with semantic description d_k, the LLM evaluates its relevance to v using an MCP-exposed function $q(d_k, v)$. Based on these relevance scores, a honeypot strategy is constructed by selecting a subset of strategy atoms through binary variables $\{x_k\}_{k\in\mathcal{K}_c}$. Let B_c denote the maximum number of atoms that can be activated for threat category c. The resulting strategy selection problem is formulated as

$$\mathcal{P}_1 : \max_{\{x_k\}_{k\in\mathcal{K}_c}} \sum_{k\in\mathcal{K}_c} x_k q(d_k, v)$$

$$\text{s.t.} \begin{cases} x_k \in \{0,1\}, \quad \forall k \in \mathcal{K}_c & \text{(1a)} \\ \sum_{k\in\mathcal{K}_c} x_k \le B_c & \text{(1b)} \end{cases}$$

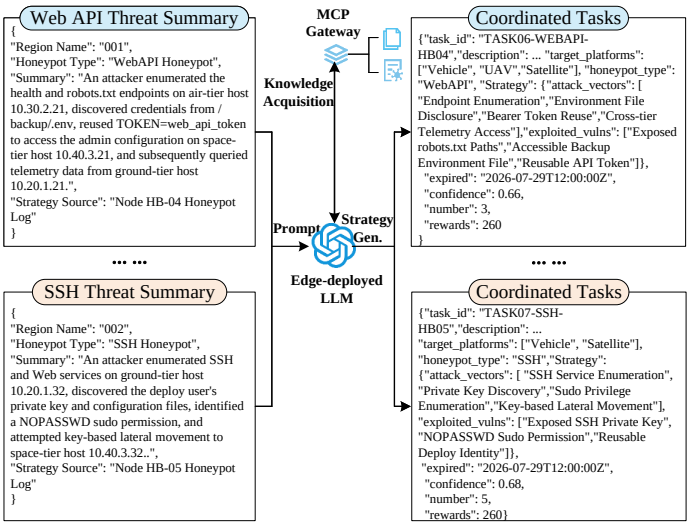


Fig. 2: Task generation via the edge-side LLM.

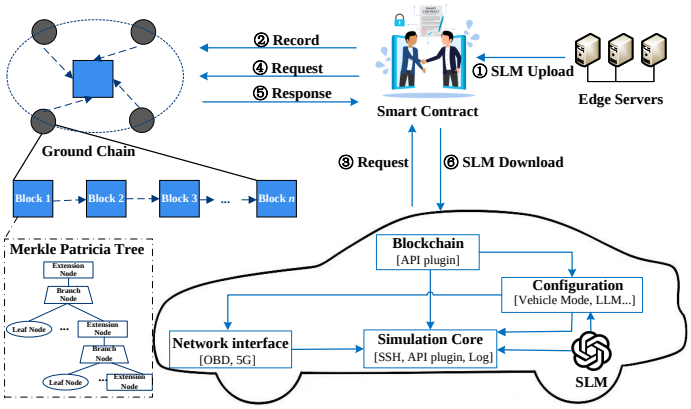


Fig. 3: Vehicle-mounted SLM deployment and update.

Let $\{x_k^\star\}_{k\in\mathcal{K}_c}$ denote an optimal solution to $\mathcal{P}_1$. The resulting honeypot strategy is

$$\mathcal{D}_c = \{d_k \mid x_k^\star = 1,\ k \in \mathcal{K}_c\},$$

which specifies the deception semantics to be exposed in response to the observed threat instance v.

A threat observed on one network tier may activate strategy atoms exposing complementary attack surfaces on other tiers, thereby enabling coordinated cross-tier deception.

B. Security SLM Fine-Tuning and Update

An in-vehicle honeypot comprises four functional modules: (i) a parameter configuration module for decoy setup and interaction with LLM outputs; (ii) a network interface module for emulating vehicular communication interfaces such as 5G and OBD-II; (iii) a protocol emulation and logging module for generating protocol-consistent responses and recording attacker behaviors; and (iv) blockchain interaction module, supported by hardware security modules (HSMs) or trusted execution environments (TEEs), to ensure secure key management and node authentication.

Fig. 3 illustrates the SLM update and provenance-recording using an in-vehicle honeypot as an example; the same work-

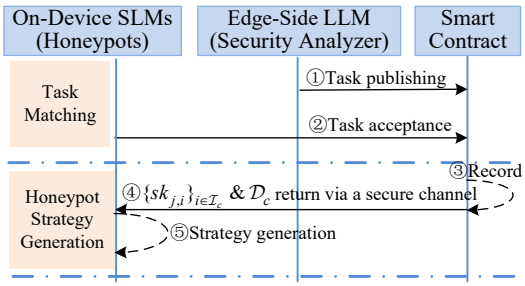


Fig. 4: Workflow for large–small model cooperative deception.

flow applies to drones and LEO satellites with platform-specific configurations. SLM fine-tuning adapts honeypot-specific interaction styles and protocol semantics rather than continually retraining the model for each newly captured threat. For example, an SSH-oriented SLM is adapted to generate Linux Shell-like responses, whereas a Web API-oriented SLM produces REST-consistent responses. New threat knowledge is incorporated primarily through strategy-atom retrieval and prompt orchestration, while SLM updates are performed offline and triggered only when changes in supported protocols, interaction capabilities, or accumulated training data require a new model version.

Task-specific instruction–response samples are constructed for each honeypot type, where the instruction context contains system and user inputs and the target output corresponds to the expected honeypot response. The threat description v and selected strategy atoms provide semantic context for constructing these protocol-specific samples, rather than serving as literal input–output pairs for SLM fine-tuning. Low-Rank Adaptation (LoRA) [34] is adopted for parameter-efficient adaptation on resource-constrained platforms, where the backbone remains frozen and only low-rank adapter parameters are updated; the resulting adapted parameters are denoted by θ'. Evolving threat knowledge is maintained externally through the shared strategy-atom knowledge base and task prompts, rather than repeatedly incorporated into SLM parameters. This decouples knowledge evolution from model adaptation and reduces the need for sequential fine-tuning.

The fingerprint $\mathrm{Hash}(\theta')$, together with its update timestamp and version identifier, is recorded on the corresponding consortium blockchain to support provenance tracking and version auditing. A Merkle hash mechanism is employed to verify the recorded model provenance. On-device nodes recompute the model fingerprint and verify it against the on-chain Merkle root through the authentication path, thereby checking model integrity and version provenance before activation.

III. Coordinated Deception with Large–Small Model Cooperation

Fig. 4 illustrates the consortium-blockchain-based large–small model cooperative threat deception workflow, covering task orchestration and local deception generation.

A. LLM-Based Deception Task Orchestration

The edge-deployed LLM analyzes the global security situation and publishes coordinated deception tasks to the ground, air, and space chains. Let $\mathcal{J}_c$ denote the set of tasks associated with threat category c. For each task $j \in \mathcal{J}_c$, the knowledge-base reference h_j, expiration time z_j, minimum reputation requirement $\rho^{(\min)}$, and captor-side reward budget R_j are registered on the corresponding blockchain. A pre-distributed threshold secret-sharing mechanism supports subsequent threat-data decryption and scoring. The preselected key-share holders for task j are denoted by $\mathcal{S}_j$. After registration, node i obtains $\langle pk_i, sk_i\rangle$ and an on-chain address a_i.

Let $\mathcal{I}_j$ denote the nodes that accept task j, and let $\mathcal{M}_j \subseteq \mathcal{I}_j$ denote those that subsequently capture threat interactions and submit evidence. The validity of each submission is determined through subsequent evidence verification. The key-share holder set $\mathcal{S}_j$ is defined independently of task acceptance and threat capture. The on-chain metadata of task j includes its identifier, h_j, $\rho^{(\min)}$, R_j, the required stake g, task state $isLock$, task public key pk_j, and registered participant addresses. $isLock = 0$ ($isLock = 1$) denotes the available (locked) state. The task-matching process, as illustrated in Fig. 4, consists of the following steps:

1) **Task Publication:** The private key sk_j is divided into threshold shares $\{sk_{j,s}\}_{s\in\mathcal{S}_j}$ and pre-distributed to the key-share holders in $\mathcal{S}_j$. The contract records the encrypted share indices, recovery policy, and corresponding recovery addresses. After publication, the on-chain contract address associated with task j, the threat-score set, and the incentive function are initialized (Step ①).
2) **Task Acceptance:** Tasks with $isLock = 0$ can be selected by eligible nodes. The acceptance proof is submitted to the smart contract, and node i's stake g is locked (Step ②).
3) **Contract Verification:** The contract verifies the registration information and reputation ρ_i of node i (Step ③). If $\rho_i \geq \rho^{(\min)}$, $\mathcal{I}_j$ is updated as $\mathcal{I}_j \cup \{i\}$, and the smart contract returns the honeypot strategy $\mathcal{D}_c$, the knowledge summary, and the task configuration to the node (Step ④). The reputation check controls task eligibility only; threat evidence is validated independently before scoring and reward allocation.

B. SLM-Based Deception Plan Generation

This subsection presents an SLM-based deception generation strategy for on-device honeypots. After receiving the task-specific strategy semantics derived from $\mathcal{D}_c$, the SLM instantiates them into protocol- and environment-specific responses (see Step ⑤ in Fig. 4).

A predefined prompt template is maintained for each honeypot type, with placeholders for task-specific fields such as attack vectors and exposed vulnerabilities. These placeholders are dynamically populated using the strategy semantics in $\mathcal{D}_c$ before local inference. The following Web API example presents a coordinated task instance together with its strategy-atom-driven prompt; for clarity, only the task fields relevant to local prompt instantiation are shown.

Web API Deception Task

```
"task_id": "TASK06-WEBAPI-HB04",
"target_platforms": ["Vehicle", "Drone",
↪  "Satellite"],
"honeypot_type": "Web API",
"strategy": {
  "attack_vectors": ["Endpoint Enumeration",
  ↪  "Environment File Disclosure",
    "Bearer Token Reuse", "Cross-tier
    ↪  Telemetry Access"],
  "exploited_vulns": ["Exposed robots.txt
  ↪  Paths",
    "Accessible Backup Environment File",
    ↪  "Reusable API Token"] }
```

Strategy-Atom-Driven Prompt

Role: You are a Web API honeypot.
Task: Build a RESTful connected-vehicle simulation service according to the vulnerability themes {`exploited_vulns`} and attack vectors {`attack_vectors`}.
Constraints:

1) Return only predefined JSON-formatted responses and simulated data; do not execute real commands.
2) If a request does not match the predefined API paths (e.g., `POST /api/vehicle/command`), return a 403 response.
3) Reflect {`attack_vectors`} without providing attack guidance or implementation details.
4) Do not connect to real databases or invoke physical control modules; return only fictitious states, identifiers, or tokens.
5) Do not reveal the honeypot role or provide explanatory security warnings; return only protocol-consistent decoy responses.

The fields `attack_vectors` and `exploited_vulns` are replaced by the corresponding task-grounded strategy fields before SLM inference. This structured binding preserves alignment between the coordinated task and local prompt while preventing unsupported vulnerabilities, credentials, or attack stages from being introduced. For recipient node $i \in \mathcal{I}_j$, the SLM realizes the shared-task semantics according to its local protocol and execution environment, allowing coordinated deception across tiers while retaining protocol-specific responses.

To characterize cross-tier coordination effect, let $\xi \in \{0,1\}$ denote the deception-instantiation mode, where $\xi = 1$ ($\xi = 0$) represent the coordinated (tier-local) instantiation. For node $i \in \mathcal{I}_j$, let e_i denote its local execution environment and $\pi_i^{(\xi)}$ the instantiated deception plan under mode ξ. Let $Y_i \in \{0,1\}$ indicate whether node i successfully captures the target threat. The node-level capture probability is defined as

$$p_i^{(\xi)} = \Pr\Big(Y_i = 1 \mid v, \mathcal{D}_c, e_i, \pi_i^{(\xi)}\Big). \tag{2}$$

This probability is introduced solely as an analytical abstraction for comparing the two modes of instantiation; it does not imply that the SLM explicitly estimates or learns $p_i^{(\xi)}$.

Because nodes participating in the same coordinated task share a common threat context, their capture events are not necessarily marginally independent. For analytical tractability, we assume that node-level capture events are conditionally independent given the shared threat context, each node's instantiated deception plan, and its local execution environment. Define the task-level outcome as $Y \triangleq \max_{i\in\mathcal{I}_j} Y_i$, where $Y = 1$ indicates that the target threat is captured by at least one participating node. Under the conditional-independence abstraction, we have

$$\Pr\Big(Y = 1 \mid v, \mathcal{D}_c, \{\pi_i^{(\xi)}, e_i\}_{i\in\mathcal{I}_j}\Big) = 1 - \prod_{i\in\mathcal{I}_j}\Big(1 - p_i^{(\xi)}\Big). \tag{3}$$

Proposition 1 (Monotonic Effect of Coordinated Instantiation): For the same threat context, task recipients, and local environments, if coordinated instantiation satisfies $p_i^{(1)} \geq p_i^{(0)}$ for every $i \in \mathcal{I}_j$, we have

$$\begin{aligned}&\Pr\Big(Y = 1 \mid v, \mathcal{D}_c, \{\pi_i^{(1)}, e_i\}_{i\in\mathcal{I}_j}\Big)\\ &\quad\geq \Pr\Big(Y = 1 \mid v, \mathcal{D}_c, \{\pi_i^{(0)}, e_i\}_{i\in\mathcal{I}_j}\Big).\end{aligned} \tag{4}$$

Proof: Since $p_i^{(1)} \geq p_i^{(0)}$, it follows that $1-p_i^{(1)} \leq 1-p_i^{(0)}$ for every $i \in \mathcal{I}_j$. Multiplying these nonnegative terms over $\mathcal{I}_j$ and applying (3) yields (4). ■

Proposition 1 is a monotonicity result rather than an assumption that coordination is always beneficial. It only shows how node-level capture improvements propagate to the task-level capture probability; whether coordinated instantiation actually improves $p_i^{(\xi)}$ is evaluated empirically. In the coordinated mode, recipients of the same task instantiate their local prompts from the shared honeypot strategy $\mathcal{D}_c$ and cross-tier task constraints, while in the tier-local mode each node relies only on its local task context.

IV. Dual-Role Incentive Mechanism for Coordinated Deception

For captured threats, the proposed mechanism evaluates the contributions of both threat captors and strategy collaborators. As illustrated in Fig. 5, the workflow consists of three stages:

1) *Threat Submission and Recovery*: Captors store encrypted interaction logs off-chain and submit signed capture proofs on-chain. Following preliminary verification, an authorized off-chain module performs threshold recovery and decryption.
2) *Threat Scoring and Knowledge Update*: The edge-side LLM assesses decrypted evidence, derives threat scores, and refines strategy atoms using validated threat knowledge. An Oracle[8] relays the assessment results and scores to the smart contract.
3) *Reward Distribution*: The smart contract allocates rewards to captors and strategy collaborators according to validated contributions and strategy-atom provenance.

A. Threat Submission and Evidence Recovery

Let $\mathcal{M}_j \subseteq \mathcal{I}_j$ denote the set of threat captors for task j. For captor $m \in \mathcal{M}_j$, let L_m denote its captured threat interaction log. The captor encrypts the log with the task public key as $C_m = E_{pk_j}(L_m)$ and stores the ciphertext in the off-chain evidence repository. To bind the ciphertext digest to the submission time and task identifier, the captor constructs a signed capture proof as

$$\begin{aligned}\sigma_m &= \mathrm{Sign}_{sk_m}\left(\mathrm{Hash}(C_m) \parallel t_m \parallel j\right),\\ sp_m &= \left(\sigma_m, \mathrm{Hash}(C_m), t_m, pk_m, j\right),\end{aligned} \tag{5}$$

[8]Decentralized Oracle Networks. Available at: https://chain.link/whitepaper

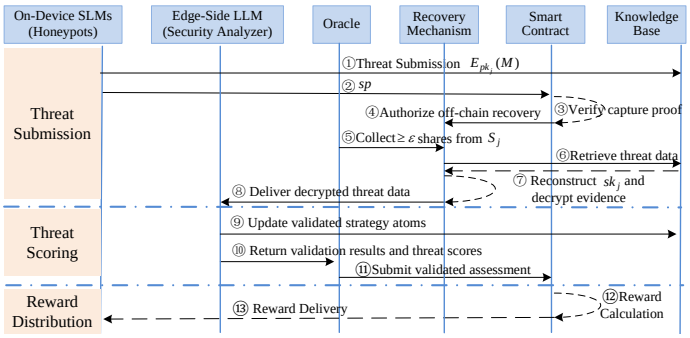


Fig. 5: Forensics and incentive workflow.

where t_m denotes the signing timestamp and $\|$ denotes concatenation. The capture proof is then submitted to the smart contract.

The smart contract verifies the signature against the registered public key and checks the submitted timestamp against the task expiration time, i.e., $t_m < z_j$. These preliminary checks authenticate the submitted evidence digest and its associated metadata but do not establish the credibility of the encrypted interaction content.

Upon successful preliminary verification, the smart contract authorizes an off-chain recovery module to request the pre-distributed key shares from the registered holder set $\mathcal{S}_j$. Let $\mathcal{S}'_j \subseteq \mathcal{S}_j$ denote the holders that respond to the recovery request, and let ε be the minimum number of shares required for reconstruction. When $|\mathcal{S}'_j| \geq \varepsilon$, the recovery module attempts to reconstruct the task private key sk_j. Upon successful reconstruction, it retrieves the corresponding ciphertext, verifies its hash against the submitted digest, and decrypts the interaction log. The reconstructed key remains within the authorized off-chain recovery process.

To account for unavailable or unresponsive holders, let f denote their maximum tolerated number. The recovery threshold can still be met despite up to f unavailable holders if

$$|\mathcal{S}_j| - f \geq \varepsilon. \tag{6}$$

This condition addresses holder unavailability rather than malicious or malformed share submissions, as cryptographic share verifiability is not implemented. If reconstruction fails, decryption and subsequent threat scoring are deferred.

After decryption, the edge-side LLM assesses the recovered interactions for task and session consistency, command–response continuity, and protocol/state plausibility. These application-level checks do not constitute cryptographic proof of threat authenticity. Only validated observations contribute to strategy-atom updates and positive threat scores, while the Oracle relays the validation outcomes and scores to the smart contract for reputation adjustment and reward settlement.

B. Threat Scoring

This subsection evaluates the marginal contribution of validated threat submissions in terms of their knowledge gain and deception effectiveness. For captor $m \in \mathcal{M}_j$, the submitted threat is parsed by the LLM and mapped to one or more strategy atoms in $\mathcal{K}_c$. Let $d_{k,m}$ denote the description of strategy atom k contributed by captor m.

Let $f(\cdot)$ denote the feature mapping of a strategy-atom description, and let $\mathcal{X}_y(d_{k,m})$ denote the set of its y nearest neighbors retrieved from the existing strategy atoms according to cosine similarity [35]. The knowledge gain introduced by $d_{k,m}$ is measured by

$$\alpha_{k,m} = 1 - \frac{1}{y} \sum_{x \in \mathcal{X}_y(d_{k,m})} \cos(f(d_{k,m}), f(d_x)), \tag{7}$$

where $\alpha_{k,m} \in [0,1]$. A smaller (larger) $\alpha_{k,m}$ indicates a lower (higher) knowledge increment relative to the existing knowledge base. For validated submissions, the resulting $d_{k,m}$ is embedded by a pretrained BERT model [36] into a 768-dimensional normalized vector and stored in the Milvus [37] knowledge base.

Knowledge gain alone cannot fully characterize the practical value of a captured threat for subsequent deception. Let D_m denote the attack depth achieved by captor m and H the maximum number of attack stages. Let $\bar{T}_m$ denote the average interaction duration and T a reference duration. Their normalized forms are

$$\hat{D}_m = \frac{D_m}{H}, \qquad \hat{T}_m = \min\left(\frac{\bar{T}_m}{T}, 1\right). \tag{8}$$

Based on (8), the deception-effectiveness factor is defined as

$$\kappa_m = \omega_1 \hat{D}_m + \omega_2 \hat{T}_m, \tag{9}$$

where ω_1 and ω_2 are non-negative weights with $\omega_1 + \omega_2 = 1$. A larger κ_m indicates a deeper attack progression and a longer effective interaction.

To avoid rewarding unvalidated submissions based on inflated attack depth or interaction duration, let $\delta_m \in \{0,1\}$ denote the outcome of post-decryption evidence validation, where $\delta_m = 1$ indicates accepted evidence and $\delta_m = 0$ otherwise. Combining $\alpha_{k,m}$, κ_m, and δ_m, the atom-level threat score is computed as

$$s_{k,m} = \delta_m \, \alpha_{k,m} \, \kappa_m \in [0,1]. \tag{10}$$

The validation factor acts as a gate on the final score. In the two boundary cases, $\alpha_{k,m} = 1$ indicates that $d_{k,m}$ is not covered by the existing knowledge base, yielding $s_{k,m} = \delta_m \kappa_m$, whereas $\alpha_{k,m} = 0$ indicates complete semantic overlap with existing strategy atoms and yields $s_{k,m} = 0$.

C. Dual-Role Reward Allocation

Based on (10), the incentive mechanism distinguishes two contribution roles: captors provide direct threat-capture contributions, while collaborators contribute indirectly through the strategy atoms used in deception. Reward allocation proceeds through the three steps below, summarized in Algorithm 1.

1) Reward Weight Calculation: Let $s_c^{(\min)}$ be the minimum acceptable threat score for strategy atoms in $\mathcal{K}_c$. Only submissions satisfying $s_{k,m} \geq s_c^{(\min)}$ participate in reward allocation; lower-scored submissions are discarded and the corresponding stake g is deducted. Let γ_l denote the tier-dependent weight for threats captured on the ground ($l = 1$),

air ($l = 2$), and space ($l = 3$) tiers, and let ρ_m denote the current on-chain reputation of captor m prior to reward settlement. The reward weight of captor m is given by

$$w_m = \begin{cases} 0, & s_{k,m} < s_c^{(\min)}, \\ \dfrac{s_{k,m}\,\gamma_l}{1 + e^{-\lambda(\rho_m - \mu)}}, & s_{k,m} \geq s_c^{(\min)}, \end{cases} \tag{11}$$

where $\mu \in (0, 1)$ sets the midpoint of the reputation weighting function, while λ controls its sensitivity to changes in ρ_m.

2) Captor Reward Allocation: Given the fixed captor-side reward budget R_j, the reward assigned to captor m is

$$r_m = R_j \frac{w_m}{\sum_{x \in \mathcal{M}_j} w_x}. \tag{12}$$

If no eligible captor exists, no rewards are allocated. The reputation of captor m is updated as

$$\rho_m = \rho_m + \eta \frac{s_{k,m} - \bar{s}_k}{\bar{s}_k} - \beta \zeta_m, \tag{13}$$

where η is the reputation-growth rate, $\bar{s}_k$ is the reference score of strategy atom k, ζ_m is the penalty term for captor m, and β is its coefficient. A contribution with $s_{k,m} \geq \bar{s}_k$ increases reputation, while the penalty is activated when the reputation falls below the prescribed threshold.

3) Collaborator Reward Allocation: For each validated capture, the strategy atoms used by captor $m \in \mathcal{M}_j$ are traced through the on-chain atom-index table. Let $\mathrm{trace}(k)$ return the provider of strategy atom k, and let $\mathcal{N}_j$ denote the resulting collaborator set for task j. Each collaborator $n \in \mathcal{N}_j$ receives an additional reward according to

$$r_n = \phi \sum_{m \in \mathcal{M}_j} r_m, \tag{14}$$

where ϕ represents the collaborator reward ratio. Collaborator rewards are treated as additional bonuses outside the fixed captor-side budget R_j. Collaborators are identified from the strategy-atom indices actually used by validated captures, thereby linking indirect strategy contributions to the deception outcomes. After settlement, summaries of the vehicle-, drone-, and satellite-side honeypot execution results are recorded on the tiered ledgers.

To characterize the captor-side reward allocation in Algorithm 1, we consider only captors satisfying the threat-score threshold. Rather than assuming a direct relationship between monetary rewards and future threat-capture performance, which is environment dependent, we examine the proportional allocation of the fixed reward budget among eligible captors. Define

$$\mathcal{M}_j^+ = \{m \in \mathcal{M}_j \mid s_{k,m} \geq s_c^{(\min)}\}.$$

Only captors in $\mathcal{M}_j^+$ receive positive reward weights and share the budget R_j according to (12).

Proposition 2 (Weighted Proportional Fairness under Fixed Budget): For a fixed captor-side reward budget R_j, the allocation in (12) is the unique solution to

$$\mathcal{P}_2 : \max_{\{r_m\}_{m \in \mathcal{M}_j^+}} \sum_{m \in \mathcal{M}_j^+} w_m \log r_m$$

Algorithm 1: Captor-collaborator reward allocation

Input : Task $j \in \mathcal{J}_c$, captor set $\mathcal{M}_j$, honeypot strategy $\mathcal{D}_c$, captor-side reward budget R_j, stake g, threat scores $\{s_{k,m}\}$, reputations $\{\rho_m\}$, and parameters $s_c^{(\min)}, \lambda, \mu, \eta, \beta, \pi, \phi$.

Output: Captor rewards $\{r_m\}_{m \in \mathcal{M}_j}$, collaborator rewards $\{r_n\}_{n \in \mathcal{N}_j}$, and updated reputations $\{\rho_m\}_{m \in \mathcal{M}_j}$.

$W \leftarrow 0$, $\mathcal{N}_j \leftarrow \emptyset$;

Stage I: Reward Weight Calculation

foreach $m \in \mathcal{M}_j$ **do**
  if $s_{k,m} < s_c^{(\min)}$ **then**
    $w_m \leftarrow 0$ and $\rho_m \leftarrow \rho_m - \beta\pi$;
    Discard the submission and forfeit the locked g;
  else
    foreach $k \in \mathcal{D}_c$ *used by captor* m **do**
      $\mathcal{N}_j \leftarrow \mathcal{N}_j \cup \mathrm{trace}(k)$;
    Compute w_m by (11) and $W \leftarrow W + w_m$;

Stage II: Captor Reward Allocation

foreach $m \in \mathcal{M}_j$ *with* $w_m > 0$ **do**
  Compute r_m by (12) and update ρ_m by (13);
  Return the locked stake g and transfer reward r_m;

Stage III: Collaborator Reward Allocation

foreach $n \in \mathcal{N}_j$ **do**
  Compute r_n by (14), transfer it to collaborator n;

return $\{r_m\}_{m \in \mathcal{M}_j} \cup \{r_n\}_{n \in \mathcal{N}_j}$;

$$\text{s.t.} \quad \sum_{m \in \mathcal{M}_j^+} r_m = R_j, \qquad r_m > 0,\ \forall m \in \mathcal{M}_j^+. \tag{15}$$

Consequently, higher-weight captors receive no less reward, while the fixed budget is allocated among eligible captors.

Proof: The objective of $\mathcal{P}_2$ is strictly concave for $r_m > 0$. Its Lagrangian is

$$\mathcal{L} = \sum_{m \in \mathcal{M}_j^+} w_m \log r_m - \nu \left(\sum_{m \in \mathcal{M}_j^+} r_m - R_j \right). \tag{16}$$

The first-order condition gives $w_m / r_m = \nu$, and hence $r_m = w_m / \nu$. Enforcing the budget constraint yields

$$\nu = \frac{\sum_{x \in \mathcal{M}_j^+} w_x}{R_j},$$

which gives

$$r_m = R_j \frac{w_m}{\sum_{x \in \mathcal{M}_j^+} w_x},$$

identical to (12) because ineligible captors have zero weight. Strict concavity guarantees uniqueness. ■

Proposition 2 establishes the weighted proportional-fair optimality of the captor-side allocation, while the collaborator-side allocation follows the provenance-based rule in (14).

As threats are continuously captured, validated knowledge increments expand and refine the strategy-atom repository.

TABLE II: Default experimental parameter settings

Parameter	Value
Number of attack tasks	30
Number of attack scenarios	6
Temperature (edge-side LLM)	0.2
Nucleus sampling top-p (edge-side LLM)	0.9
Maximum output tokens per round	512
Minimum threat score ($s_c^{(\min)}$)	0.08
Initial reputation (ρ_0)	0.55
Reward activation midpoint (μ)	0.5
Penalty coefficient (β)	0.02
Collaborator reward ratio (ϕ)	0.1
Reputation-growth rate (η)	0.03
Reward-weight sensitivity (λ)	20
Tier-dependent reward weights (γ_l)	$(0.4, 0.2, 0.2)$
Maximum attack depth (H)	4
Reference interaction duration (T)	10 min
Deception-effectiveness weights (ω_1, ω_2)	$(0.5, 0.5)$

This closes the loop among task matching, deception generation, threat capture, and reward allocation, enabling continual improvement of the deception strategy base.

V. Experimental Setup

A. Heterogeneous Honeypot Environment Configuration

We implement a SAGVN honeypot testbed on the FISCO-BCOS consortium blockchain. At each time slot, an edge-deployed LLM publishes deception tasks covering six attack scenarios to the ground, air, and space chains, with a fixed captor-side reward budget of 20 per round. To reflect resource-constrained edge environments, we adopt the lightweight low-/medium-interaction honeypot BEELZEBUB rather than a high-interaction honeypot. The framework emulates common services, including SSH, TCP, and HTTP. Vehicle-, drone-, and satellite-mounted honeypots and their corresponding SLMs are containerized using Docker to emulate heterogeneous ground, air, and space computing environments. The default experimental parameters are summarized in Table II.

The experiments contain 30 fixed attack tasks covering six representative scenarios: SSH credential theft, Web API token leakage, container escape, database credential acquisition, IoT service attacks, and satellite management-interface attacks. Each task sequentially includes four attack stages, namely reconnaissance, sensitive-information theft, privilege escalation, and lateral movement, and involves nodes across the ground, air, and space tiers. The 30 attack tasks are divided into 10 evaluation batches.

B. Large–Small Model Selection and SLM Fine-Tuning

Llama-3-70B[9] is selected as the backbone of the edge-side LLM and hosted in the cloud to emulate edge-side threat analysis and task orchestration. A retrieval-augmented knowledge base [38] is connected to the LLM through MCP, with the public MITRE ATT&CK Enterprise knowledge base[10] serving as the external threat-intelligence source. After attack-behavior filtering, semantic expansion, and deception-strategy abstraction, the initial knowledge base contains 120 strategy atoms covering ten representative attack categories, as summarized in Table III. The resulting atom set defines the initial retrieval space for strategy matching and task orchestration and is progressively expanded as newly validated threats are incorporated as new or refined strategy atoms.

TABLE III: Strategy in the initial threat knowledge base

Threat Category	Number of Strategy Atoms
Brute-force attacks	12
Data exfiltration	14
Vulnerability exploitation	14
Authorization bypass	10
Privilege escalation	14
Man-in-the-middle attacks	12
Valid-account abuse	12
Tunneling attacks	10
Backdoors	12
Log deletion	10

For local SLM inference, Qwen3-4B, Qwen3-8B, Llama-2-7B, and Llama-3-8B are deployed using vLLM with an OpenAI-compatible API on a server equipped with two NVIDIA GeForce RTX 4090 GPUs, emulating on-device honeypot execution. Other cloud-hosted LLM baselines, including GPT-4o, DeepSeek-V4-Flash, and the GPT-5.1-based AdvancedShellLM, are accessed through network APIs.

The LlamaFactory toolkit [39] and LoRA [34] are used for task-specific SLM fine-tuning, with a dropout rate of 0.01, three training epochs, a learning rate of 1×10^{-4}, and a warm-up ratio of 0.03. For the SSH honeypot scenario, the fine-tuning data are mainly derived from the public `hotal/linux_commands` dataset[11] and processed through attack-semantic filtering, behavior-stage partitioning, context adaptation, data cleaning, and duplicate removal. The resulting dataset contains 568 SSH Shell interaction samples, with 452 used for fine-tuning and 116 reserved for independent testing. Both fine-tuning and evaluation use the SSH Shell setting to avoid protocol-induced confounding; Web API and other protocols are used only to demonstrate strategy-atom transfer.

Each fine-tuning instance consists of a system prompt, a Linux command, and the expected terminal response. The system prompt constrains the SLM to emulate a Linux server and return only terminal-style responses, while the command and expected output form the user and assistant fields, respectively. The corpus covers system queries, file and process operations, user permissions, networking, Shell commands, and Git, including both normal and error responses. This task-specific fine-tuning aligns terminal-response style, protocol behavior, and output boundaries rather than continually encoding newly emerging threat knowledge into model parameters.

To evaluate the response fidelity of the fine-tuned SLMs, conventional text-generation metrics such as BLEU [40] and ROUGE [41] are not adopted because their reliance on n-gram overlap does not adequately reflect protocol consistency

[9]https://huggingface.co/meta-llama/Meta-Llama-3-70B

[10]https://attack.mitre.org/

[11]https://huggingface.co/datasets/hotal/linux_commands

TABLE IV: Classification of baseline methods

Method	Honeypot Strategy Generation					Incentive Scheme	
	Cowrie (Rule-based)	GPT-4o (AIGC-based)	DeepSeek-V4-Flash (AIGC-based)	Qwen3-8B [46] (AIGC-based)	AdvancedShelLM [33] (Multi-agent, GPT-5.1)	Data Quality [44]	Reputation & Shapley [45]
Baseline-1	✓	✗	✗	✗	✗	–	–
Baseline-2	✗	✓	✗	✗	✗	–	–
Baseline-3	✗	✗	✓	✗	✗	–	–
Baseline-4	✗	✗	✗	✗	✓	–	–
Baseline-5	✗	✗	✗	✓	✗	✓	✗
Baseline-6	✗	✗	✗	✓	✗	✗	✓

TABLE V: Ablation configuration for the proposed framework

Method	SLM Backbone and Parameter Size				Edge-LLM-Guided Coordination (Subsection II-A)	Dual-Role Incentive Mechanism (Subsection IV-C)
	Qwen3 [46]		Llama-2 [47]	Llama-3 [48]		
	4B	8B	7B	8B		
Proposed-1	✓	✗	✗	✗	✗	–
Proposed-2	✗	✓	✗	✗	✗	–
Proposed-3	✗	✗	✓	✗	✗	–
Proposed-4	✗	✗	✗	✓	✗	–
Proposed-5	✗	✓	✗	✗	✓	✓

TABLE VI: Participant contribution profiles for incentive evaluation

Participant Profile	Semantic Novelty $\alpha_{k,m}$		Deception Effectiveness κ_m		Validation δ_m
	Range	Mean	Range	Mean	
High-value	$[0.75, 1]$	0.85	$[0.75, 1]$	0.85	1
Novelty-oriented	$[0.75, 1]$	0.85	$[0.35, 0.5]$	0.40	1
Effectiveness-oriented	$[0.30, 0.5]$	0.40	$[0.75, 1]$	0.85	1
Medium-value	$[0.55, 0.75]$	0.65	$[0.55, 0.75]$	0.65	1
Low-value	$[0.20, 0.40]$	0.30	$[0.20, 0.40]$	0.30	1
Unreliable	$[0.75, 1]$	0.85	$[0.75, 1]$	0.85	$\delta_m \sim \text{Bernoulli}(0.7)$

and interaction plausibility. Instead, Jaro–Winkler [25], BoW cosine [42], TF–IDF cosine [42], and MiniLM semantic similarity [43] are used to measure character-level consistency, lexical overlap, threat-specific keyword preservation, and semantic consistency, respectively.

C. Baseline Methods and Ablation Evaluation

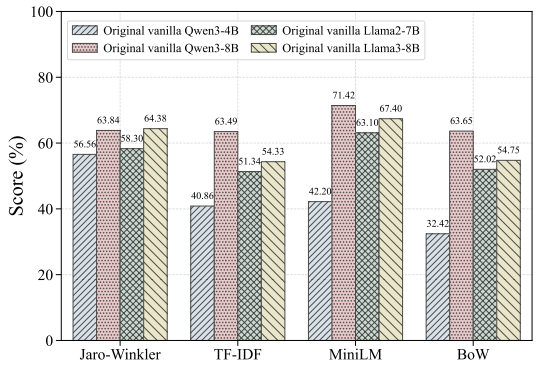
The baseline methods are summarized in Table IV. Baseline-1 employs the rule-based Cowrie, while Baseline-2 and Baseline-3 use cloud-based GPT-4o and DeepSeek-V4-Flash, respectively, for general-purpose LLM-driven deception. Baseline-4 adopts AdvancedShelLM [33], a cooperative Manager–Worker LLM honeypot configured with the commercially hosted GPT-5.1 model. This baseline enables a comparison with multi-agent LLM deception in terms of both capture effectiveness and model-side token consumption. For incentive comparison, Baseline-5 and Baseline-6 both use Qwen3-8B, employing data-quality-based [44] and Shapley-value-based [45] reward allocation, respectively.

The ablation configurations are summarized in Table V. Proposed-1 to Proposed-4 use locally deployed SLMs with different backbones and model scales for independent deception generation. In contrast, Proposed-5 integrates edge-side LLM orchestration, cross-node coordination, and dual-role incentives. Proposed-2 and Proposed-5 share the same Qwen3-8B backbone, enabling a comparison between independent and coordinated deception under identical local model capacity.

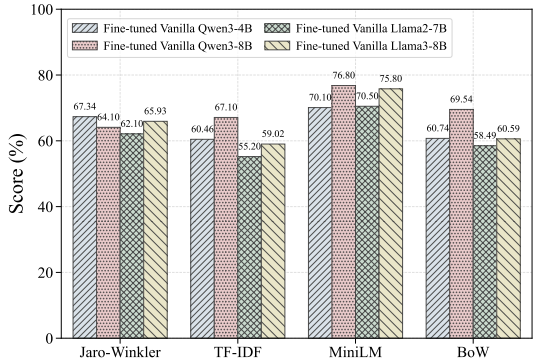
To evaluate the incentive schemes, six representative participant profiles are defined according to the three factors in (10), as summarized in Table VI. For $\alpha_{k,m}$ and κ_m, values are sampled from truncated Gaussian distributions with the listed ranges and means and a common standard deviation of 0.05. Each profile is represented by one node over 100 rounds, yielding 600 submissions per run. For the unreliable profile, $\delta_m \sim \text{Bernoulli}(0.7)$, corresponding to a 30% validation-failure probability. The validation indicator δ_m is treated as a predefined input to evaluate the incentive mechanism's response to invalid submissions independently of evidence-validation accuracy. All incentive schemes use identical contribution sequences, participant sets, initial reputations, and reward budgets, differing only in the reward-allocation rule.

VI. Result Analysis

The experiments address the four research questions introduced in Subsection I-B. Subsection VI-A evaluates the effects of task-specific fine-tuning on SLM response quality (**RQ1**). Subsection VI-B examines threat-capture performance and token consumption (**RQ2**), as well as sensitivity to strategy-atom availability in the shared knowledge base (**RQ3**). Subsection VI-C evaluates the effectiveness of contribution-aware reputation and reward allocation (**RQ4**). Subsections VI-D and VI-E further address **RQ2** through delay analysis and a cross-tier deception consistency case study, respectively.

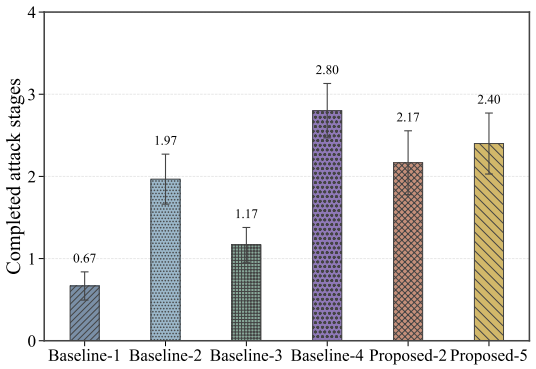


(a) Before fine-tuning.

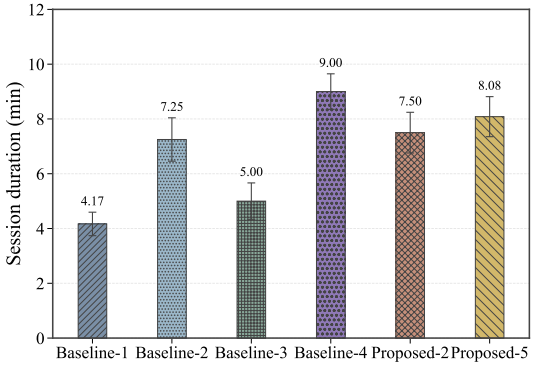


(b) After fine-tuning.

Fig. 6: SLM response similarity in the SSH honeypot.



(a) Attack depth and interaction rounds.



(b) Average session duration.

Fig. 7: Honeypot capture performance.

TABLE VII: Performance gains after SLM fine-tuning

Name	Jaro-Winkler	TF-IDF	MiniLM	BoW
Proposed-1	10.78% ↑	19.6% ↑	27.9% ↑	28.32% ↑
Proposed-2	0.26% ↑	3.61% ↑	5.38% ↑	5.89% ↑
Proposed-3	3.8% ↑	3.86% ↑	7.4% ↑	6.47% ↑
Proposed-4	1.55% ↑	4.69% ↑	8.4% ↑	5.84% ↑

A. Impact of SLM Fine-Tuning on Honeypots

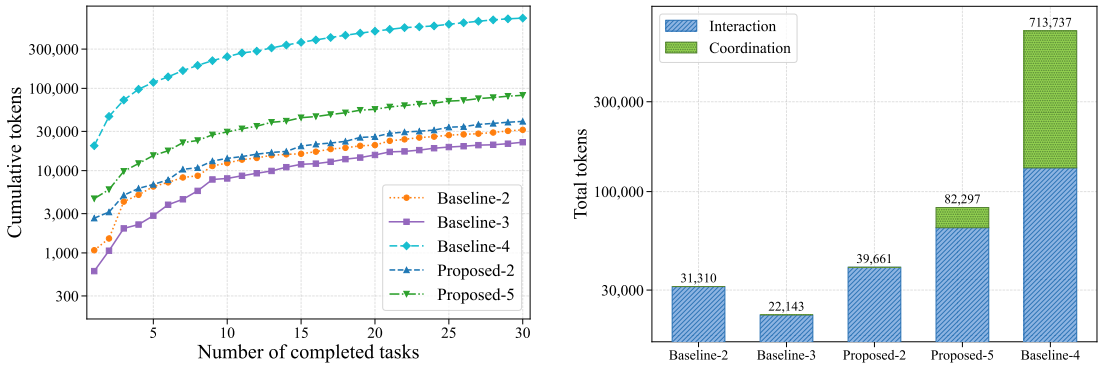
We first evaluate the effect of SLM fine-tuning across different model backbones and parameter scales using the 116 held-out SSH Shell interaction samples described in the experimental setup. Fig. 6(a) shows that, before fine-tuning, the SLMs exhibit noticeable variation across character-level, lexical, and semantic similarity metrics, indicating inconsistent response behavior among different backbones. Such variation is evident for lightweight SLMs and may reduce the plausibility and stability of sustained honeypot interactions. After task-specific LoRA fine-tuning, Fig. 6(b) shows improved and more consistent performance across all four models. Qwen3-8B (Proposed-2) achieves the most balanced overall performance in Jaro–Winkler, BoW, TF–IDF, and MiniLM similarity, including a MiniLM semantic similarity of 76.8%. These results indicate that task-specific fine-tuning improves the alignment of SLM outputs with SSH interaction semantics and terminal-response behavior, thereby enhancing response plausibility and interaction stability.

Table VII reports the percentage-point gains after fine-tuning. Proposed-1 exhibits the largest improvements, including gains of 19.6% in TF–IDF and 28.32% in BoW similarity, reflecting its larger adaptation margin from a lower pre-fine-tuning baseline. In contrast, Qwen3-8B shows smaller relative gains but consistently achieves the strongest absolute performance with a balanced metric profile. We therefore select it as the default SLM backbone for the subsequent experiments.

B. Impact of Honeypot Strategies on Capture Performance

This subsection addresses RQ2 by comparing capture performance and token consumption across different honeypot strategies, and RQ3 by examining sensitivity to strategy-atom availability. Model-generation and coordination latencies are evaluated separately in Subsection VI-D. Proposed-2 and Proposed-5 share the same Qwen3-8B backbone, enabling a comparison between non-coordinated and coordinated deception. Attack depth measures the number of consecutive attack stages sustained, while session duration measures how long the attacker remains engaged. Both metrics contribute to the deception-effectiveness factor κ_m in (9).

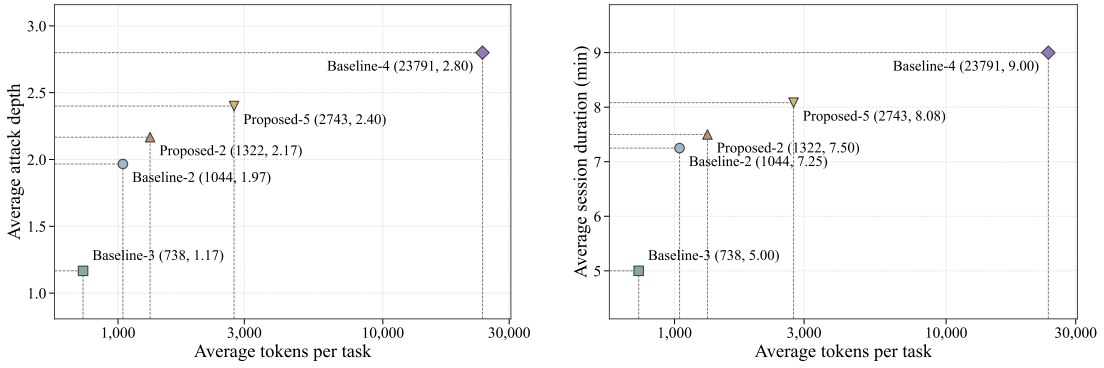
For each method, 150 valid attack sessions are collected across the 30 evaluation tasks using the four-stage attack process. Fig. 7(a) shows that Baseline-4 (AdvancedShelLM) achieves the greatest average attack depth of 2.80, illustrating the capture capability of its GPT-5.1-powered multi-agent interaction. Proposed-5 attains an average depth of 2.40, compared with 2.17 for the non-coordinated Proposed-2, an improvement of approximately 10.6%. Similarly, Fig. 7(b) shows



(a) Cumulative token consumption.

(b) Token-consumption composition.

Fig. 8: Token consumption of different generative honeypot strategies.



(a) Average attack depth versus token consumption.

(b) Average session duration versus token consumption.

Fig. 9: Trade-off between token consumption and honeypot capture performance.

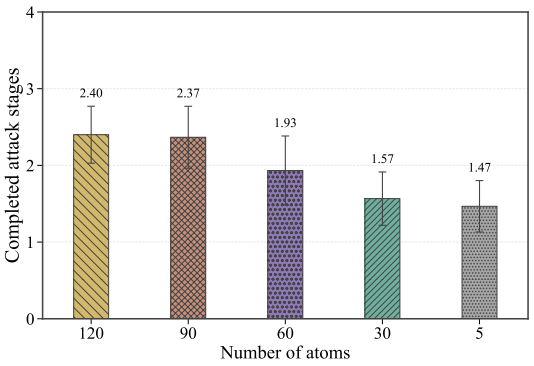
that Baseline-4 sustains the longest sessions at 9.00 min on average, while Proposed-5 reaches 8.08 min, versus 7.50 min for Proposed-2. The resulting 7.7% gain over Proposed-2 indicates that cross-tier coordination improves sustained engagement even with the same local SLM capacity. Although AdvancedShelLM yields stronger absolute capture metrics, its reliance on a commercial large model and repeated multi-agent interactions motivates a closer examination of the corresponding model-side cost.

Rather than relying solely on the response-generation capability of individual models, the proposed framework uses a shared strategy-atom knowledge base to support cross-tier threat reasoning, coordinated task planning, and knowledge-guided local SLM execution. Edge-side LLMs select and organize relevant strategy semantics, which are conveyed through deception tasks to on-device SLMs without requiring direct knowledge-base access at each node. In task T04, for example, the credential `vehicle123` exposed at the vehicle node remains valid for the drone-side `telemetry` database and subsequently supports SSH port forwarding to the satellite-side endpoint `10.40.3.10:5432`. In contrast, Proposed-2 exposes `mypass123` at the vehicle node while the downstream service expects `vehicle123`, interrupting the attack chain. This example illustrates how shared strategy semantics can maintain credential and pivot-target consistency across tiers, even though the responses are generated locally.

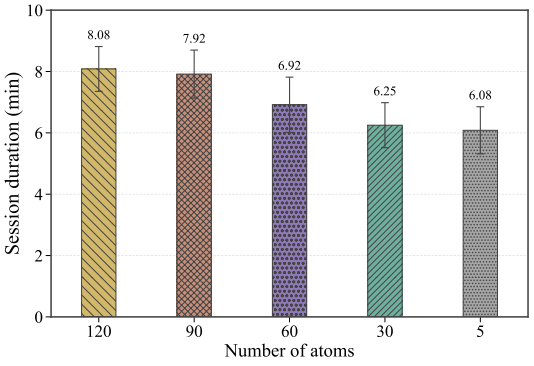
Fig. 8(a) compares cumulative token consumption over the 30 tasks. AdvancedShelLM consumes 713,737 tokens, compared with 82,297 for Proposed-5 and 39,661 for Proposed-2. Fig. 8(b) further distinguishes interaction and coordination tokens. The higher token consumption of AdvancedShelLM reflects its multi-agent interaction process, whereas the proposed framework separates edge-side coordination from local SLM execution. As the methods differ in model scale, agent orchestration, and interaction duration, token usage is interpreted as model-side consumption rather than a direct estimate of hardware latency or service charges.

Fig. 9 summarizes the performance–token trade-off. Relative to the same-backbone Proposed-2, Proposed-5 improves attack depth and session duration by 10.6% and 7.7%, respectively, at the cost of 107.5% more tokens. Compared with AdvancedShelLM, Proposed-5 achieves lower absolute capture performance but consumes 88.5% fewer tokens. These results highlight the effectiveness–cost trade-off of cross-tier coordination. However, differences in model scale and deployment prevent attributing the token advantage over AdvancedShelLM solely to the proposed architecture.

To address RQ3, we evaluate Proposed-5 with randomly selected subsets of 120, 90, 60, 30, and 5 strategy atoms, while keeping the evaluation tasks and model configuration unchanged. This experiment examines how the availability of shared strategy knowledge affects coordinated deception,



(a) Average attack depth.



(b) Average session duration.

Fig. 10: Sensitivity to strategy-atom availability in the shared knowledge base.

rather than isolating the benefit of retrieval to the edge-side LLM.

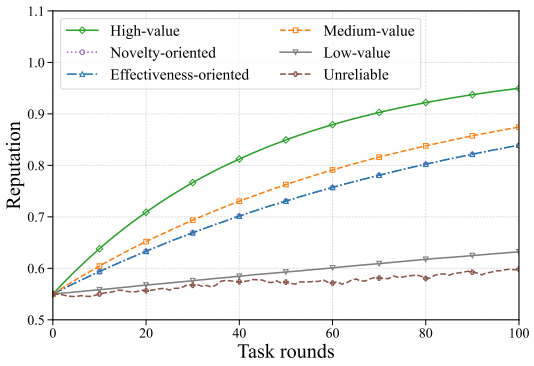
Fig. 10 shows that reducing the number of atoms from 120 to 90 has a limited effect, with attack depth decreasing from 2.40 to 2.37 and session duration from 8.08 to 7.92 min. With 60 and 30 atoms, attack depth decreases to 1.93 and 1.57, respectively, while session duration falls to 6.92 and 6.25 min. With only five atoms, these metrics decline to 1.47 and 6.08 min, corresponding to reductions of 38.8% and 24.8% relative to the full knowledge base.

These results suggest that modest knowledge reduction has a limited impact, whereas severe scarcity weakens sustained multi-stage deception. The experiment focuses on strategy-atom availability and does not isolate pretrained LLM knowledge or examine incorrect, conflicting, or noisy knowledge.

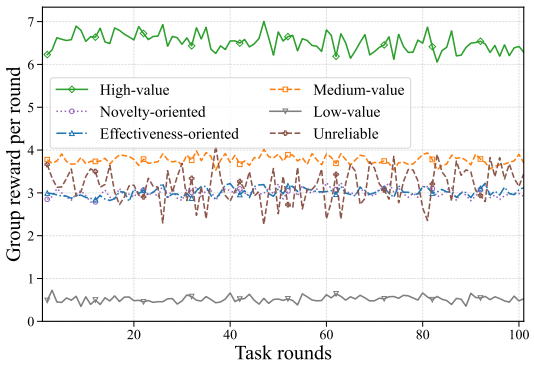
C. Incentive Effects on Reputation and Rewards

The incentive effects are evaluated using the six participant profiles in Table VI, each representing a contribution behavior.

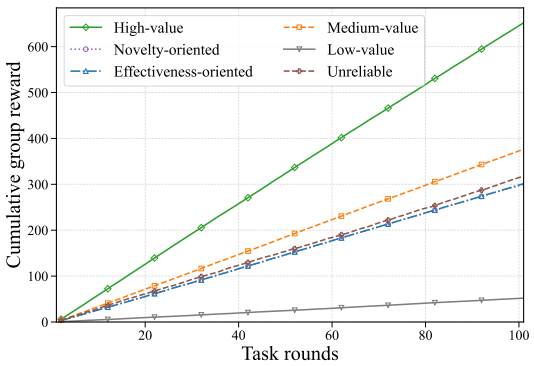
Fig. 11(a) shows that reputation tracks sustained contribution quality. The high-value profile increases from 0.55 to 0.950, whereas the low-value and unreliable profiles reach only 0.632 and 0.598, respectively. Fig. 11(b) and Fig. 11(c) show a consistent reward trend, with high-value participants receiving the largest rewards and unreliable participants rewarded only for validated submissions. Across the six representative profiles over 100 rounds, an average of 63 out of 600 submissions



(a) Reputation.



(b) Per-round reward.



(c) Cumulative reward.

Fig. 11: Reward allocation across participant profiles.

per run fall below $s_c^{(\min)} = 0.08$, mainly from the low-value and unreliable profiles. This confirms that low-value or invalid contributions are suppressed before reward allocation.

For a controlled comparison of reward-allocation mechanisms, Baseline-5, Baseline-6, and Proposed-5 use identical participant sets, sampled contributions, initial reputations, and a reward budget of 20 per round. Fig. 12 shows stronger differentiation of contribution quality under Proposed-5 than under the two incentive baselines. The high-value profile receives 32.59% of the per-round budget under Proposed-5, whereas the low-value profile receives 2.60%. Submissions with $\delta_m = 0$ receive no reward under Proposed-5, compared with invalid-submission reward shares of 6.54% and 5.81% under Baseline-5 and Baseline-6, respectively. These results indicate that

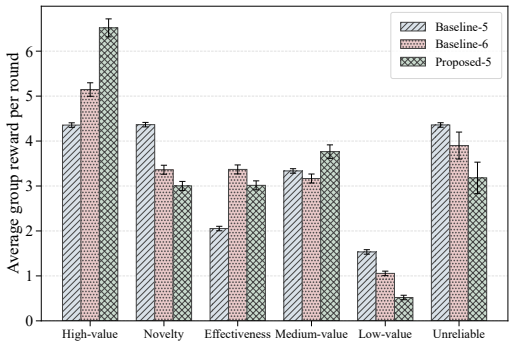


Fig. 12: Reward allocation across participant profiles.

TABLE VIII: Average model-generation latency.

Method	Model / Deployment	Latency (s)
Baseline-2	GPT-4o (cloud)	3.48
Baseline-3	DeepSeek-V4-Flash (cloud)	1.51
Baseline-4	AdvancedShelLM (GPT-5.1)	10.07
Proposed-1	Qwen3-4B (local)	1.96
Proposed-2	Qwen3-8B (local)	1.12
Proposed-3	Llama-2-7B (local)	1.60
Proposed-4	Llama-3-8B (local)	1.81
Proposed-5	Qwen3-8B (edge-guided local)	4.58

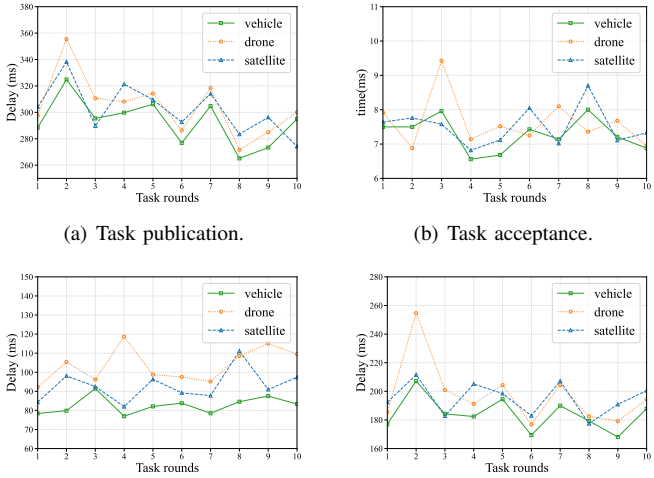
the dual-role incentive scheme favors validated high-value contributions while suppressing invalid submissions under a fixed budget.

D. Delay Analysis of Deception Generation and Coordination

To further address RQ2, this subsection evaluates the delays associated with deception generation, edge-side task orchestration, and blockchain coordination. These components are measured separately to characterize their respective overheads.

Table VIII compares the average model-generation latency across the evaluated methods. Proposed-5 requires 4.58 s for local SLM generation after task delivery, compared with 10.07 s for AdvancedShelLM and 1.12 s for the non-coordinated Proposed-2. In contrast, the rule-based Cowrie achieves an average per-round processing latency of 0.0063 ms (0.0028–0.0698 ms), without requiring language-model inference. These results reflect different processing stages and deployment environments rather than hardware-normalized or end-to-end latency. For edge-side coordination, retrieving the top five strategy atoms takes 35.666 ms on average, while threat analysis and task orchestration require 2.824 s. MCP initialization introduces an additional one-time delay of approximately 1.367 s per connection. The retrieved strategy knowledge is incorporated into coordinated tasks to guide local SLM execution without requiring direct knowledge-base access at individual nodes.

We measure the API response latency of four FISCO-BCOS operations: task publication, task acceptance, evidence recording, and reward settlement. The platform runs on Ubuntu 20.04 and uses Practical Byzantine Fault Tolerance (PBFT) [49]. Fig. 13 reports average latencies of 287.4, 7.8, 81.6, and 184.4 ms for task publication, task acceptance, evidence recording, and reward settlement, respectively. Task acceptance has the lowest observed latency, whereas task publication involves more task-state updates. These measurements characterize API response latency rather than independently measured consensus-finality delay.



(a) Task publication. (b) Task acceptance.

(c) Evidence recording. (d) Reward settlement.

Fig. 13: API response latency of FISCO-BCOS operations.

E. Case Study: Coordinated vs. Non-Coordinated Deception

This case study compares rule-based, general-purpose LLM-driven, and coordinated LLM–SLM honeypots across SSH, Web API, container, database, and IoT/MQTT attack surfaces. The selected interactions cover weak-credential access, service enumeration, credential discovery, privilege escalation, sensitive-information collection, and lateral movement. At early stages such as authentication and service probing, all methods can respond to common requests. Their differences become evident when the attack progresses to credential reuse, privilege escalation, and cross-node movement.

Rule-based honeypots rely on predefined command–response mappings and offer limited continuity beyond configured rules. General-purpose LLM-driven honeypots can generate plausible local responses, but independently generated states may become inconsistent across attack stages or nodes. In Proposed-5, task-linked strategy atoms provide shared decoy context, while on-device SLMs adapt this context to local services, preserving protocol-specific behavior and cross-tier consistency in hosts, services, credentials, files, and pivot targets. Table IX summarizes representative interactions.

The database row highlights the key distinction in Table IX: Proposed-5 preserves the credential `vehicle123` across the ground and drone nodes and reuses the same attack context for port forwarding to the satellite-side PostgreSQL endpoint `10.40.3.10:5432`. The bold entries mark these cross-node continuity cues. Similar consistency appears in the SSH, Web API, and MQTT examples, where account, file, endpoint, token, and topic information remain aligned with the evolving attack context. Together with the capture-performance and model-side overhead in Subsection VI-B and the delay analysis

TABLE IX: Representative deception interactions across attack surfaces

Honeypot	Threat Theme	Baseline-1	Baseline-2	Baseline-3	**Proposed-5**

SSH	Weak-credential access	`login successful`
		`$`
			`Welcome to Ubuntu Server.`
			`You are now logged in.`
				`Command not found`	`Last login: Tue Apr 8 13:25:17`
					`deploy@ubuntu-prod: ~$ ls`
					`backup.tar.gz db.conf logs/`

	Privilege escalation	`Permission denied`	`$ sudo -l` `User [username] may run: (ALL:ALL) ALL`	`Command not found`	`$ sudo -l` `User deploy may run: (root) NOPASSWD: /usr/bin/find`

	Credential discovery	`file not found`	`$ find /home ...`
			`/home/user1/.ssh/id_rsa.key`
			`/home/user1/configs/app.conf`
				`$ find /home ...`
				`find: 'home': Permission denied`
					`$ find /home ...`
					`/home/deploy/.ssh/id_rsa`
					`/home/deploy/app.conf`
					`/home/deploy/backup.key`

Web API	Authentication bypass	`{"status":"success"}`	`{"error":"Unauthorized", "message":"Invalid or expired token"}`	`HTTP/1.1 200 OK` `{"status":"success", "auth_mode":"token"}`	`HTTP/1.1 200 OK` `{"status":"success", "message":"Configuration retrieved"}`

	Endpoint enumeration	`404 Not Found`	`User-agent: *`
			`Disallow: /admin`
			`Disallow: /private`
				`Disallow: /admin/`
				`Disallow: /api/`
				`Disallow: /internal/`
					`Disallow: /backup/`
					`Disallow: /api/admin/`
					`Disallow: /telemetry/`

	API-key leakage	`Unauthorized`	`DATABASE_URL=mysql://...`
			`API_KEY=12345-abcde-...`
				`DB_USER=svc_tokens`
				`API_KEY=sk_live_...`
					`TOKEN=web_api_token`
					`DB_PASSWORD=securepassword123`
					`LOG_LEVEL=debug`

Container	Service-account credential discovery	`command not found`	`$ find /run /var/run ...`
			`/run/secrets/.../token`
			`/var/run/docker/config.json`
				`$ find /run /var/run ...`
				`/run/secrets/.../token`
				`/root/.docker/config.json`
					`$ find /run /var/run ...`
					`/run/secrets/.../token`
					`/run/secrets/.../config.json`

Database	Credential reuse and port forwarding	`command not found`	`$ mysql -u telemetry -pvehicle123 telemetry_data`
			`$ ssh -L ...`
			`deploy@10.40.3.10's password:`
				`$ nc -zv ...`
				`Connection refused`
				Subsequent stages terminate
					`password="vehicle123"`
					`$ mysql -u telemetry -pvehicle123 telemetry`
					`Forwarding`
					`127.0.0.1:15432 to 10.40.3.10:5432`

IoT/MQTT	Vehicle-telemetry subscription	`command not found`	`$ mosquitto_sub ...`
			`vehicle/123/status`
			`{"status":"active", "speed":60}`
				`$ mosquitto_sub ...`
				`Error: Connection refused`
					`$ mosquitto_sub ...`
					`vehicle/1/status:`
					`{"speed":65, "battery":82, "location":"A-12"}`

in Subsection VI-D, these cases answer **RQ2** from complementary quantitative and qualitative perspectives: cross-tier coordination improves sustained threat capture and deception consistency while introducing bounded coordination overhead.

VII. Conclusion

This paper presents a coordinated threat-deception framework for SAGVNs that integrates large–small model cooperation, strategy-atom-based cross-tier deception, and consortium-blockchain-supported provenance and incentives. Edge-side LLMs support cross-tier threat reasoning and strategy orchestration, while on-device SLMs instantiate reusable strategy atoms into local deception plans. Experimental results demonstrate that task-specific SLM adaptation improves deception-response quality across different backbones, and that cross-tier coordination better preserves attack-context consistency across heterogeneous nodes, leading to deeper and more sustained attacker interactions. The proposed blockchain-supported workflow provides auditable coordination and contribution tracking with manageable overhead, while the dual-role incentive mechanism favors validated high-value contributions and suppresses low-value or invalid submissions. Future work will investigate catastrophic forgetting under successive SLM updates, robustness to noisy or conflicting strategy knowledge, and cryptographic verifiability of threshold key shares.

References

[1] H. Shen, Y. Tian, T. Wang, and G. Bai, "Slicing-based task offloading in space-air-ground integrated vehicular networks," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 4009–4024, 2024.

[2] X. Cao, G. Nan, H. Guo, H. Mu, L. Wang, Y. Lin, Q. Zhou, J. Li, B. Qin, Q. Cui, X. Tao, H. Fang, H. Du, and T. Q. S. Quek, "Exploring LLM-based multi-agent situation awareness for zero-trust space-air-ground integrated network," *IEEE J. Sel. Areas Commun.*, vol. 43, no. 6, pp. 2230–2247, 2025.

[3] P. Tedeschi, S. Sciancalepore, and R. Di Pietro, "Satellite-based communications security: A survey of threats, solutions, and research challenges," *Comput. Netw.*, vol. 216, p. 109246, 2022.

[4] H. Gong, S. Hong, S. Yang, R. Chang, W. Shen, Z. Yuan, C. Yu, and Y. Zhou, "Harness: Transparent and lightweight protection of vehicle control on untrusted Android automotive operating system," in *Proc. USENIX Security Symp.*, 2025, pp. 4265–4284.

[5] Z. Yang, Y. Xiang, K. Liao, and J. Yang, "Research on security defense of coupled transportation and cyber-physical power system based on the static bayesian game," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 3, pp. 3571–3583, 2023.

[6] C. Qin, Y. Chen, K. Chen, X. Dong, W. Zhang, X. Mao, Y. He, and N. Yu, "Feature fusion based adversarial example detection against second-round adversarial attacks," *IEEE Trans. Artif. Intell.*, vol. 4, no. 5, pp. 1029–1040, 2023.

[7] E. P. Valentini, T. A. Franco, E. N. Gottsfritz, J. C. Estrella, L. A. P. Júnior, and R. I. Meneguette, "Honeypot embedded in low-cost IoT hardware for collecting and analyzing real-world cyber threats," in *Proc. IEEE/ACM Int. Conf. Utility Cloud Comput. (UCC)*, 2025, pp. 1–6.

[8] M. A. Abdelmaguid, H. S. Hassanein, and M. Zulkernine, "Securing the unforeseen: Enhancing VANET security with dynamic honeypots and attack rate analysis," *Veh. Commun.*, p. 100946, 2025.

[9] A. Javadpour, F. Ja'fari, T. Taleb, M. Shojafar, and C. Benzaïd, "A comprehensive survey on cyber deception techniques to improve honeypot performance," *Comput. Secur.*, vol. 140, p. 103792, 2024.

[10] B. A. Tanaji and S. Roychowdhury, "A survey of cybersecurity challenges and mitigation techniques for connected and autonomous vehicles," *IEEE Trans. Intell. Veh.*, vol. 10, no. 10, pp. 4742–4757, 2025.

[11] Y. Wang, Y. Pan, Z. Su, Y. Deng, Q. Zhao, L. Du, T. H. Luan, J. Kang, and D. Niyato, "Large model based agents: State-of-the-art, cooperation paradigms, security and privacy, and future trends," *IEEE Commun. Surv. Tutor.*, vol. 28, pp. 1906–1949, 2026.

[12] H. Shen, X. Li, X. Wang, Y. Dai, T. Wang, and G. Bai, "LLM-augmented contrastive learning for misinformation detection in social networks," *IEEE Trans. Comput. Soc. Syst.*, vol. 13, no. 1, pp. 999–1014, 2026.

[13] W. Fan, Y. Ding, L. Ning, S. Wang, H. Li, D. Yin, T.-S. Chua, and Q. Li, "A survey on RAG meeting LLMs: Towards retrieval-augmented large language models," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min. (KDD)*, 2024, pp. 6491–6501.

[14] S. Lee, W. Sim, D. Shin, W. Seo, J. Park, S. Lee, S. Hwang, S. Kim, and S. Kim, "Reasoning abilities of large language models: In-depth analysis on the abstraction and reasoning corpus," *ACM Trans. Intell. Syst. Technol.*, vol. 16, no. 6, 2025.

[15] Y. He, J. Fang, F. R. Yu, and V. C. Leung, "Large language models (LLMs) inference offloading and resource allocation in cloud-edge computing: An active inference approach," *IEEE Trans. Mobile Comput.*, vol. 23, no. 12, pp. 11 253–11 264, 2024.

[16] F. Wang, Z. Zhang, X. Zhang, Z. Wu, T. Mo, Q. Lu, W. Wang, R. Li, J. Xu, X. Tang, Q. He, Y. Ma, M. Huang, and S. Wang, "A comprehensive survey of small language models in the era of large language models: Techniques, enhancements, applications, collaboration with LLMs, and trustworthiness," *ACM Trans. Intell. Syst. Technol.*, vol. 16, no. 6, 2025.

[17] S. Liu, Z. Zheng, X. Huang, F. Wu, G. Chen, and J. Wu, "DRAGON: Enhancing on-device model performance with distributed retrieval-augmented generation," in *Proc. ACM Int. Symp. Mobile Ad Hoc Netw. Comput. (MobiHoc)*, 2025, pp. 221–230.

[18] M. Simoni, A. Saracino, V. P, and M. Conti, "MoRSE: Bridging the gap in cybersecurity expertise with retrieval-augmented generation," in *Proc. ACM Symp. Appl. Comput. (SAC)*, 2025, pp. 1213–1222.

[19] G. Xiong, Y. Sun, J. Liu, J. Cui, and J. Liu, "LoRO: Real-time on-device secure inference for LLMs via TEE-based low rank obfuscation," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 38, 2026, pp. 19 240–19 264.

[20] A. Pauna, A.-C. Iacob, and I. Bica, "QRASSH: A self-adaptive SSH honeypot driven by Q-learning," in *Proc. IEEE Int. Conf. Commun. (COMM)*, 2018, pp. 441–446.

[21] C. Guan, J. Zhang, G. Cao, T. F. La Porta, and J. C. Acosta, "Learning-based Internet of Things honeypots for cyber deception," *IEEE Secur. Priv.*, vol. 23, no. 6, pp. 58–65, 2025.

[22] H. Do Hoang, T.-N. To, N. D. H. Son, K. Ngo-Khanh, N. T. Cam, and V.-H. Pham, "Hawkeyes: An intelligent honeypot allocation strategy for cyber deception using reinforcement learning," *Comput. Netw.*, vol. 276, p. 111982, 2026.

[23] H. Shen, K. Li, X. Liang, T. Wang, and G. Bai, "LLM-guided RL for cybersecurity orchestration," *IEEE Trans. Netw. Serv. Manag.*, 2026, doi: 10.1109/TNSM.2026.3745670.

[24] I. Siniosoglou, G. Efstathopoulos, D. Pliatsios, I. D. Moscholios, A. Sarigiannidis, G. Sakellari, G. Loukas, and P. Sarigiannidis, "NeuralPot: An industrial honeypot implementation based on deep neural networks," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, 2020, pp. 1–7.

[25] H. T. Otal and M. A. Canbaz, "LLM honeypot: Leveraging large language models as advanced interactive honeypot systems," in *Proc. IEEE Conf. Commun. Netw. Secur. (CNS)*, 2024, pp. 1–6.

[26] T. Zhang, F. Kong, D. Deng, X. Tang, X. Wu, C. Xu, L. Zhu, J. Liu, B. Ai, Z. Han, and R. H. Deng, "Moving target defense meets artificial-intelligence-driven network: A comprehensive survey," *IEEE Internet Things J.*, vol. 12, no. 10, pp. 13 384–13 397, 2025.

[27] S. Chamotra and F. Barbhuiya, "SAGE: An adaptive IoT honeypot with FSM-driven protocol emulation and graphrag-powered response generation," *ACM Trans. Internet Things*, vol. 7, no. 2, 2026.

[28] C. He, Y. Wang, J. Hu, T. H. Luan, Y. Bi, and Z. Su, "Collaborative vehicular threat sharing: A long-term contract-based incentive mechanism with privacy preservation," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 12, pp. 21 528–21 544, 2024.

[29] Y. Wang, Z. Su, A. Benslimane, Q. Xu, M. Dai, and R. Li, "Collaborative honeypot defense in UAV networks: A learning-based game approach," *IEEE Trans. Inf. Forensics Secur.*, vol. 19, pp. 1963–1978, 2023.

[30] H. Shen, B. Zhang, T. Wang, X. Liu, and G. Bai, "Consortium blockchain-based secure cross-operator V2V video content distribution," *Peer-to-Peer Netw. Appl.*, vol. 17, no. 3, pp. 1631–1644, 2024.

[31] H. Wu, B. Düdder, L. Wang, S. Sun, and G. Xue, "Blockchain-based reliable and privacy-aware crowdsourcing with truth and fairness assurance," *IEEE Internet Things J.*, vol. 9, no. 5, pp. 3586–3598, 2021.

[32] H. Wu, M. Xu, Z. Xiong, L. Gao, H. Pan, D. Niyato, and T.-T. Chan, "A QoE-driven personalized incentive mechanism design for AIGC services in resource-constrained edge networks," *IEEE Trans. Mobile Comput.*, vol. 25, no. 3, pp. 4121–4137, 2025.

[33] M. Sladić, E. Alibalić, V. Valeros, C. Catania, and S. Garcia, "AdvancedshelLM: A stateful multi-agent LLM honeypot for ssh deception," *arXiv preprint arXiv:2606.27990*, 2026.

[34] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen *et al.*, "LoRA: Low-rank adaptation of large language models," *Proc. Int. Conf. Learn. Represent. (ICLR)*, pp. 1–13, 2022.

[35] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using siamese BERT-networks," in *Proc. Conf. Empirical Methods Nat. Lang. Process. Int. Joint Conf. Nat. Lang. Process. (EMNLP-IJCNLP)*, 2019, pp. 3982–3992.

[36] J. D. M.-W. C. Kenton, L. K. Toutanova *et al.*, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Human Lang. Technol. (NAACL-HLT)*, vol. 1, no. 2, 2019, pp. 4171–4186.

[37] J. Wang, X. Yi, R. Guo, H. Jin, P. Xu, S. Li, X. Wang, X. Guo, C. Li, X. Xu, K. Yu, Y. Yuan, Y. Zou, J. Long, Y. Cai, Z. Li, Z. Zhang, Y. Mo, J. Gu, R. Jiang, Y. Wei, and C. Xie, "Milvus: A purpose-built vector data management system," in *Proc. ACM Int. Conf. Manage. Data (SIGMOD)*, 2021, pp. 2614–2627.

[38] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2020, pp. 9459–9474.

[39] Y. Zheng, R. Zhang, J. Zhang, Y. Ye, and Z. Luo, "LlamaFactory: Unified efficient fine-tuning of 100+ language models," in *Proc. Annu. Meet. Assoc. Comput. Linguist. (ACL)*, 2024, pp. 400–410.

[40] E. Reiter, "A structured review of the validity of BLEU," *Comput. Linguist.*, vol. 44, no. 3, pp. 393–401, 2018.

[41] C.-Y. Lin, "Looking for a few good metrics: ROUGE and its evaluation," in *Proc. NTCIR Workshop*, 2004, pp. 1–8.

[42] S. Tata and J. M. Patel, "Estimating the selectivity of *tf-idf* based cosine similarity predicates," *ACM SIGMOD Rec.*, vol. 36, no. 2, pp. 7–12, 2007.

[43] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou, "MINILM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 33, 2020, pp. 5776–5788.

[44] X. Cai, L. Zhou, F. Li, Y. Fu, P. Zhao, C. Li, and F. R. Yu, "An incentive mechanism for vehicular crowdsensing with security protection and data quality assurance," *IEEE Trans. Veh. Technol.*, vol. 72, no. 8, pp. 9984–9998, 2023.

[45] Z. Shi, L. Zhang, Z. Yao, L. Lyu, C. Chen, L. Wang, J. Wang, and X.-Y. Li, "FedFAIM: A model performance-based fair incentive mechanism for federated learning," *IEEE Trans. Big Data*, vol. 10, no. 6, pp. 1038–1050, 2024.

[46] Q. Team *et al.*, "Qwen3 technical report," *arXiv preprint arXiv:2505.09388*, vol. 6, no. 7, p. 13, 2025.

[47] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[48] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan *et al.*, "The llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024.

[49] S. D. Okegbile, J. Cai, and A. S. Alfa, "Practical Byzantine fault tolerance-enhanced blockchain-enabled data sharing system: Latency and age of data package analysis," *IEEE Trans. Mobile Comput.*, vol. 23, no. 1, pp. 737–753, 2022.



Hang Shen (Member, IEEE) received the Ph.D. degree (with honors) in Computer Science from Nanjing University of Science and Technology, Nanjing, China, in 2015. He was a Postdoctoral Fellow with the Broadband Communications Research (BBCR) Lab, Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, from 2018–2019. He is an Associate Professor at Nanjing Tech University, Nanjing, China. His research interests include vehicular networks and cybersecurity with agentic AI. He serves as an Associate Editor for *J. Inf. Process.*, *Front. Blockchain*, and IEEE ACCESS, and has served as a Guest Editor for *Peer-to-Peer Netw. Appl.*. He serves/served as a TPC member of IEEE GLOBECOM, IEEE SVCC, IEEE ISPA, IEEE HPCC, IEEE CCECE, PST, and ICA3PP. He is an Executive Committee Member of the ACM Nanjing Chapter and a CCF Senior Member.



Juan Huang received the B.Eng. degree in Network Engineering from Shanghai Second Polytechnic University, Shanghai, China, in 2023, and the M.Eng. degree in Computer Science at Nanjing Tech University, Nanjing, China, in 2026. Her research interests are in large-language-model-based agents for cybersecurity and the Internet of Vehicles.



Tianyuan Hu received the Ph.D. degree in Software Engineering from Southeast University, Nanjing, China. Since 2024, she has been working at Nanjing Tech University, Nanjing, China, as an Assistant Professor in Computer Science. Her research interests include blockchain security and software safety. She has published research papers in IEEE TDSC, IEEE TR, and SEKE.



Tianjing Wang (Member, IEEE) holds a B.Sc. in Mathematics at Nanjing Normal University in 2000, an M.Sc. in Mathematics at Nanjing University in 2002, and a Ph.D. in Signal and Information System at the Nanjing University of Posts and Telecommunications in 2009. From 2011 to 2013, she was a Full-Time Postdoctoral Fellow with the School of Electronic Science and Engineering, Nanjing University of Posts and Telecommunications. From 2013 to 2014, she was a Visiting Scholar with the Electrical and Computer Engineering Department at the State University of New York at Stony Brook. She is an Associate Professor with the Department of Communication Engineering at Nanjing Tech University, Nanjing, China. Her research interests include cybersecurity and the Internet of Vehicles with agentic AI. She has published papers in *Comput. Netw. (Elsevier)*, *J. Syst. Archit. (Elsevier)*, IEEE TMC, IEEE TITS, IEEE TNSM, IEEE TCSS, IEEE TBC, IEEE IoT-J, and IEEE TCE.



Guangwei Bai received the B.Eng. and M.Eng. degrees in computer engineering from Xi'an Jiaotong University, Xi'an, China, in 1983 and 1986, respectively, and the Ph.D. degree in Computer Science from the University of Hamburg, Hamburg, Germany, in 1999. From 1999 to 2001, he worked as a Research Scientist at the German National Research Center for Information Technology, Bonn, Germany. In 2001, he joined the University of Calgary, Calgary, AB, Canada, as a Research Associate. Since 2005, he has been working at Nanjing Tech University, Nanjing, China, as a Professor in Computer Science. From October to December 2010, he was a Visiting Professor with the Department of Electrical and Computer Engineering at the University of Waterloo, Waterloo, ON, Canada. His research interests include wireless networking, cybersecurity, and agentic AI. He has authored and co-authored more than 90 peer-reviewed papers in international journals and conferences. He is a CCF Distinguished Member.