Engineering Applications of Artificial Intelligence 163 (2026) 112812

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Research paper

Trustworthy distributed mirror learning for secure and private multi-agent coordination

Suhang Wei, Jinfang Jia, Xiang Feng *, Huiqun Yu

Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai, 200237, China

ARTICLE INFO

Dataset link: https://github.com/353055619/tdml

Keywords:
Multi-Agent Reinforcement Learning
Internet of Things
Malicious attacks
Zero-knowledge proofs
Split learning

ABSTRACT

Real-world deployment of Multi-Agent Reinforcement Learning (MARL) in Internet of Things (IoT) systems requires convergence, verifiability, and privacy to be jointly guaranteed, a capability absent in current approaches. While Multi-Agent Trust Region Learning (MATRL) ensures Nash equilibrium convergence, it risks data exposure and malicious attacks. To address this, we propose Trustworthy Distributed Mirror Learning (TDML), the first method to unify convergence, verifiability, and privacy in MARL. TDML theoretically breaks MATRL's centralized architecture into agent-local learning and inter-agent communication. This allows key data to be secured with advanced techniques without compromising the theoretical properties of trust-region learning. Specifically, TDML introduces three core innovations: (1) an information functional that unifies all communication behaviors in distributed MATRL and enables flexible integration of security mechanisms; (2) split advantage computation, which decouples raw inputs from global advantages via intermediate representations to protect local data privacy; and (3) a security scheme that ensures verifiable message exchange by attaching zero-knowledge proofs to inter-agent communications. We prove TDML converges to a Nash equilibrium while providing verifiability and privacy guarantees. More importantly, it constructs a mirror space for trustworthy MARL, where derivative algorithms inherit these theoretical guarantees. Experiments show TDML outperforms state-of-the-art methods, improving attack resilience by up to 76% (sign-flipping attack) and achieving a 90+% privacy reconstruction error, while reducing communication overhead by up to 99% compared to homomorphic encryption. TDML establishes a foundational framework for trustworthy MARL, from which derivative algorithms inherit core guarantees for secure, real-world IoT deployment.

1. Introduction

Next-generation Internet of Things (IoT) systems are increasingly characterized by their highly dynamic, distributed, and complex nature. Within these systems, intelligent agents must achieve efficient autonomous coordination. Multi-agent reinforcement learning (MARL) has emerged as a powerful approach for this critical need (Li et al., 2022). It empowers agents to autonomously learn optimal behaviors and interactions from their direct experience. However, despite its notable successes in simulation (Badia et al., 2020), its practical application in real-world IoT deployments still faces significant challenges, such as privacy preservation and vulnerability to malicious attacks (Standen et al., 2025).

These challenges reveal a fundamental trilemma in current MARL research, namely the difficulty of achieving convergence, verifiability, and privacy preservation simultaneously (Ma et al., 2023). Distributed frameworks, such as federated learning (FL) (Liu et al., 2024a), mitigate raw data exposure by aggregating local gradients. However, their reliance on homogeneous policy constraints often leads to sub-optimal policies or even policy collapse (Kuba et al., 2022). Cryptographic schemes like differential privacy (DP) offer privacy guarantees but typically impede policy gradient optimization due to noise injection (Yuan et al., 2024). Furthermore, these methods often lack verifiable mechanisms for joint policy updates, making them susceptible to malicious policy injection, such as backdoor attacks (Liu and Lai, 2023). While robust optimization techniques (Yu et al., 2024) enhance resilience through adversarial training, they ultimately fail to unify all three essential facets of the trilemma.

Compared to the above methods, Multi-Agent Trust Region Reinforcement Learning (MATRL) approach provides strong theoretical convergence guarantees for cooperative tasks (Kuba et al., 2022; Liu et al., 2024b; Xu et al., 2025). Central to this is the multi-agent advantage decomposition lemma, which enables sequential decomposition of team rewards into individual marginal contributions, without imposing structural constraints on the joint value function (unlike VDN Sunehag

* Corresponding author.
E-mail addresses: y20210087@mail.ecust.edu.cn (S. Wei), y10210104@mail.ecust.edu.cn (J. Jia), xfeng@ecust.edu.cn (X. Feng), yhq@ecust.edu.cn (H. Yu).

https://doi.org/10.1016/j.engappai.2025.112812
Received 16 July 2025; Received in revised form 21 September 2025; Accepted 14 October 2025
Available online 3 November 2025

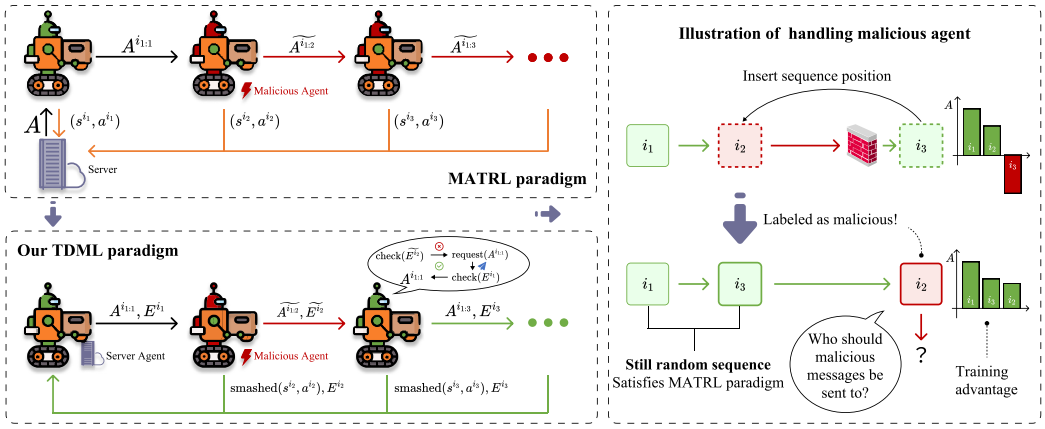

Fig. 1. Conceptual comparison of multi-agent learning paradigms. Let $i_{1:n} = (i_1, i_2, \ldots, i_n)$ denote the message pass sequence. In MATRL a malicious agent i_2 can mislead updates by injecting a crafted advantage $\widetilde{A^{i_{1:2}}}$; plaintext data (s^{i_m}, a^{i_m}) also risks privacy exposure. Our TDML paradigm computes advantages using a smashed representation smashed(s^{i_m}, a^{i_m}), decoupled from raw inputs. Messages are authenticated via Zero-Knowledge Proofs E^{i_m} to prevent malicious interference. The right side presents an example of malicious agent handling. When validation fails, the agent is marked and placed at the end of the update order, ensuring that its injected advantage values do not compromise the rest of the sequence.

et al., 2018 or QMIX Rashid et al., 2020) or requiring parameter sharing across agents, thereby naturally supporting heterogeneous policies. However, their reliance on an idealistic assumption of fully trustworthy agents and uncompromised communication is a critical limitation. This assumption rarely holds in real-world distributed environments. As illustrated in Fig. 1, adversaries can exploit system vulnerabilities to manipulate global policy optimization or even inject malicious policies (Liu and Lai, 2023). Moreover, the plaintext transmission of observation and action data (s^{i_m}, a^{i_m}) exposes sensitive information, which significantly limits MARL's applicability in privacy-sensitive scenarios.

The traditional MATRL paradigm faces two major practical limitations: (1) centralized advantage computation requires raw data sharing; and (2) the lack of verification mechanisms enables malicious agents to stealthily poison the system by injecting fabricated values. Its inherent assumption of global state observability further limits the theoretical flexibility to integrate diverse security mechanisms without compromising its core properties. Moreover, its limited compatibility with non-linear cryptographic schemes hinders its ability to achieve strong privacy protection and robustness against malicious attacks.

To address these challenges, we propose Trustworthy Distributed Mirror Learning (TDML), which, for the first time, unifies convergence, verifiability, and privacy preservation in MATRL. Our key insight is that the centralized architecture of MATRL can be theoretically decomposed into agent-local learning and inter-agent communication. This decomposition enables critical information (e.g., advantage A) to be secured independently. Specifically, we replace centralized advantage computation with a Split Advantage Computation method that uses intermediate representations smashed(s^{i_m}, a^{i_m}) to decouple raw data from updates. Furthermore, building on recent efficiency advances in zero-knowledge proofs (Sun et al., 2024), we introduce a Verifiable Privacy-Preserving (VP) scheme that attaches zero-knowledge evidence E^{i_m} to messages, enabling recipients to verify legitimacy without accessing private data. Malicious agents failing verification are moved to the end of the update sequence (Fig. 1), preventing contamination of other agents' updates. Specifically, this paper makes the following three contributions:

1. We extend the MATRL to a distributed architecture at a mathematical level and introduce an Information Functional to unify communication behaviors. This avoids the assumption of global state observability and provides theoretical and algorithmic space for embedding diverse security mechanisms.
2. We propose a Split Advantage Computation method using intermediate representations to decouple private data from policy updates, paired with a Verifiable Privacy-Preserving scheme based on zero-knowledge proofs for secure, verifiable communication.
3. We rigorously prove that TDML achieves optimal policy convergence, verifiable updates, and strong privacy preservation, thereby resolving the trilemma. Crucially, TDML establishes a mirror space framework in which any derived algorithm, such as TDPPO or TDA2C, automatically inherits these theoretical guarantees. Furthermore, TDML inherently supports heterogeneous policies by preserving the structural flexibility of MATRL, in contrast to federated or value-decomposition approaches that impose homogeneity constraints.
4. We implement and evaluate TDPPO and TDA2C across multiple mainstream MARL benchmarks, as shown in Fig. 2. Results confirm superior convergence, attack robustness, and privacy preservation with low communication overhead. We further deploy TDPPO in a real-world warehouse robot training service, where it demonstrates high efficiency and strong resilience against adversarial attacks.

2. Related work

Trustworthy Multi-Agent Reinforcement Learning (MARL) aims to jointly guarantee convergence, verifiability, and privacy in open, distributed IoT environments (Ma et al., 2023; Li et al., 2022). Most existing methods address only one or two of these goals. We categorize and analyze prior work below, highlighting how TDML overcomes their limitations.

2.1. Convergence-guaranteed MARL

Research in this category provides theoretical guarantees for convergence to a Nash Equilibrium (NE). Early cooperative MARL methods commonly adopted the Centralized Training with Decentralized Execution (CTDE) paradigm (Lowe et al., 2017). Its core principle involves learning a global value function to guide improvements in agents' joint policies, thereby theoretically guaranteeing policy convergence. Representative methods include VDN (Sunehag et al., 2018), QMIX (Rashid et al., 2020), and Q-DPP (Yang et al., 2020). However, these methods rely on strong structural assumptions, such as value function decomposability or monotonicity, which are rarely satisfied in complex real-world environments.

A significant breakthrough in recent years has been the extension of single-agent trust region learning theory to the multi-agent setting. Kuba et al. (2022) proposed the multi-agent advantage decomposition lemma, which rigorously proved that joint policies can converge to

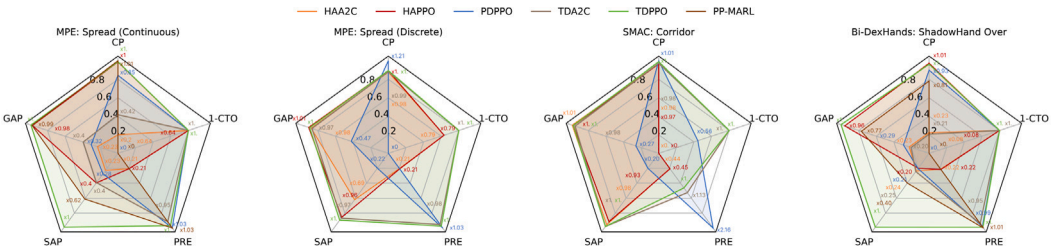

Fig. 2. Performance comparison of TDPPO/TDA2C (algorithms developed under the proposed TDML framework) with state-of-the-art trustworthy MARL methods across the Multi-Agent Particle Environment (MPE), StarCraft Multi-Agent Challenge (SMAC), and Bimanual Dexterous Hands Environment (Bi-DexHands) benchmarks. Evaluated metrics include Convergence Performance (CP↑), Gaussian Attack Performance (GAP↑), Sign Attack Performance (SAP↑), Privacy Reconstruction Error (PRE↑, higher values indicate greater difficulty for attackers to reconstruct private data), and Communication Traffic Overhead (1-CTO↑, where higher values indicate lower overhead). All metrics are normalized, and an upward arrow (↑) indicates that a higher value is preferred.

a NE without relying on restrictive value function structural assumptions. Building on this, Zhong et al. (2024) further introduced the HAML algorithm template, yielding practical algorithms like HAPPO and HAA2C. These not only inherit the convergence guarantees of trust regions but also support heterogeneous policies. Despite these theoretical advancements in MATRL, their practical deployment still faces severe challenges. They implicitly assume a fully trustworthy environment and lack defense mechanisms against malicious attacks, making them highly vulnerable in open IoT environments (Liu and Lai, 2023).

2.2. Verifiable MARL

This category focuses on ensuring the integrity and verifiability of the learning process. Traditional methods primarily rely on statistical robustness mechanisms. For instance, Yu et al.'s ADMAC (Yu et al., 2024) employs active defense mechanisms to detect and suppress unreliable communications based on statistical anomalies. While effective against certain attacks, these methods use statistical criteria for identifying malicious behavior, which can lead to misjudgments, impacting performance. Additionally, they often depend on shared observation data, thereby sacrificing privacy.

In recent years, Zero-Knowledge Proof (ZKP) has become a key method for building trustworthy distributed systems due to its cryptographic completeness in verifiability. Sasson et al. (2014) pioneered the use of efficient ZKP methods, specifically zk-SNARKs, in the Zerocash protocol for anonymous transaction verification on blockchains. Subsequently, Sun et al. (2024) further extended zero-knowledge verification capabilities to deep neural networks with the ZkDL framework, supporting proofs for non-linear operations like ReLU. This laid the foundation for verifying the computational integrity of policy and value functions in MARL. Driven by these advances, several studies have attempted to integrate ZKPs into multi-agent systems, such as combining them with blockchain for auditable decision-making (CHERIF et al., 2024) or using MARL to optimize UAV-assisted model quality verification (Hao et al., 2025).

However, none of these approaches embed ZKP into the communication protocol of MARL training to cryptographically verify the legitimacy of transmitted values. TDML addresses this critical gap. Our proposed VP scheme ensures that each agent, when communicating, attaches verifiable zero-knowledge evidence, confirming that the transmitted information was generated through legitimate computation. This approach not only avoids the misjudgment risks of statistical methods but also achieves verifiability of the MARL training process without transmitting any private data.

2.2.1. Privacy-preserving MARL

Privacy-preserving MARL aims to protect agents' sensitive observations and actions during training. Approaches fall into two main categories: federated learning and cryptography-based methods. Federated learning (FL) keeps data local and exchanges only model updates, constructing global policies via parameter aggregation. Representative examples include Liu et al.'s federated Q-learning (Liu et al., 2024a) and Lu et al.'s PDPPO (Lu et al., 2025). However, these methods commonly assume policy homogeneity, which can easily lead to sub-optimal joint policies (Kuba et al., 2022).

To build privacy mechanisms with mathematical guarantees, cryptographic methods such as secure multi-party computation (MPC), homomorphic encryption (HE), and differential privacy (DP) are being widely explored in MARL. Early attempts include the SecFloat framework (Mukherjee et al., 2023) based on MPC, which protects state and action privacy under a semi-honest model, or PE-VDN (Gohari et al., 2023), which combines additive secret sharing with differential noise to achieve (ϵ, δ)-level privacy protection while maintaining policy performance. However, these solutions are often difficult to deploy efficiently in resource-constrained IoT environments due to frequent protocol interactions and complex communication rounds. In contrast, PP-MARL (Yuan et al., 2024) combines homomorphic encryption with differential privacy and adopts a split-learning Critic to compute Q-values locally, avoiding raw data exposure. However, its reliance on homomorphic encryption increases communication overhead, and since it is built on MADDPG (Lowe et al., 2017), it lacks convergence guarantees.

Compared to existing MARL approaches, our proposed TDML presents the first integration of Split Learning (SL) and ZKP into MATRL. TDML splits sensitive raw information into multiple intermediate representations for transmission, achieving privacy protection effects similar to those of FL. But, unlike federated approaches such as PDPPO, TDML inherently supports heterogeneous policies. Our robust design shares conceptual similarities with ADMAC in detecting and mitigating unreliable messages. However, by leveraging ZKPs, TDML offers stronger guarantees in terms of computational verifiability and privacy protection. As summarized in Table 1, TDML demonstrates unique advantages in balancing the "convergence, verifiability, and privacy preservation" trilemma and supporting heterogeneous policies.

3. Preliminaries

3.1. Cooperative MARL problem formulation

Consider a cooperative MARL game $MG(\mathcal{N}, \boldsymbol{S}, \boldsymbol{A}, r, P, \gamma, \rho^0)$. Here, $\mathcal{N} = \{1, \ldots, n\}$ is the agent set; $\boldsymbol{S} = \prod_{i=1}^{n} \mathcal{S}^i$ and $\boldsymbol{A} = \prod_{i=1}^{n} \mathcal{A}^i$ are the joint state and action spaces, respectively, formed by Cartesian products of individual agent spaces. $r : \boldsymbol{S} \times \boldsymbol{A} \to \mathbb{R}$ is the joint reward function; $P : \boldsymbol{S} \times \boldsymbol{A} \times \boldsymbol{S} \to [0, 1]$ is the state transition kernel; $\gamma \in [0, 1)$ is the discount factor; and ρ^0 is the initial state distribution. We denote $\mathbf{s} \in \boldsymbol{S}$ and $\mathbf{a} \in \boldsymbol{A}$ as the joint state and action. Let Π^i be agent i's policy space, and $\boldsymbol{\Pi} \triangleq (\Pi^1, \ldots, \Pi^n)$ be the joint policy space.

The interaction process unfolds as follows: At time $t \in \mathbb{N}$, agents are in a joint state $\mathbf{s}_t = (s_t^1, \ldots, s_t^n)$. Each agent i independently selects action $a_t^i \sim \pi^i(\cdot|s_t^i)$ based on its local observation s_t^i, where $\pi^i \in \Pi^i$ is its policy.

Table 1
Comparison of various MARL methods based on privacy preservation, verifiability, convergence guarantees, and heterogeneous policy support. ✔/✗: supported/not supported. ▲: robustness without formal verifiability. **NE**: proven convergence to Nash Equilibrium. **Sub-optimal**: monotonic improvement only, no NE guarantee. –: no formal convergence analysis.

Method	Privacy	Verif.	Conv.	Het.
Convergence-Guaranteed				
VDN (Sunehag et al., 2018)	✗	✗	Sub-optimal	✗
QMIX (Rashid et al., 2020)	✗	✗	Sub-optimal	✗
Q-DPP (Yang et al., 2020)	✗	✗	Sub-optimal	✗
HAPPO/HAA2C (Zhong et al., 2024)	✗	✗	NE	✔
Verifiable				
ADMAC (Yu et al., 2024)	✗	▲	–	✗
Privacy-Preserving				
Fed Q-Learning (Liu et al., 2024a)	✔	✗	Sub-optimal	✗
PDPPO (Lu et al., 2025)	✔	✗	Sub-optimal	✗
PP-MARL (Yuan et al., 2024)	✔	✗	–	✔
SecFloat Mukherjee et al. (2023)	✔	✗	–	✗
PE-VDN (Gohari et al., 2023)	✔	✗	–	✗
TDML (Ours)	✔	✔	**NE**	✔

These individual actions form the joint action $\mathbf{a}_t = (a_t^1, \dots, a_t^n)$ according to the joint policy $\boldsymbol{\pi}(\cdot|\mathbf{s}_t) = \prod_{i=1}^n \pi^i(\cdot|s_t^i) \in \boldsymbol{\Pi}$. The environment then transitions to the next state $\mathbf{s}_{t+1} \sim P(\cdot|\mathbf{s}_t, \mathbf{a}_t)$ and provides a joint reward $r_t = r(\mathbf{s}_t, \mathbf{a}_t)$ to each agent. The joint policy $\boldsymbol{\pi}$, state transition function P, and initial state distribution ρ^0 collectively determine the marginal state distribution $\rho_t^{\boldsymbol{\pi}}$ and the discounted marginal state distribution $\rho_{\boldsymbol{\pi}} \triangleq \sum_{t=0}^{\infty} \gamma^t \rho_t^{\boldsymbol{\pi}}$. In this setting, the objective is to maximize the expected return of the joint policy (Zhong et al., 2024):

$$J(\boldsymbol{\pi}) \triangleq \mathbb{E}_{\mathbf{s}_{0:\infty} \sim \rho_{\boldsymbol{\pi}}^{0:\infty}, \mathbf{a}_{0:\infty} \sim \boldsymbol{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]. \tag{1}$$

The state-value function $V_{\boldsymbol{\pi}}(\mathbf{s})$, state–action value function $Q_{\boldsymbol{\pi}}(\mathbf{s}, \mathbf{a})$, and advantage function $A_{\boldsymbol{\pi}}(\mathbf{s}, \mathbf{a})$ are defined as:

$$\begin{aligned} V_{\boldsymbol{\pi}}(\mathbf{s}) &\triangleq \mathbb{E}_{\mathbf{a}_{0:\infty} \sim \boldsymbol{\pi}, \mathbf{s}_{1:\infty} \sim P} \left[\sum_{t=0}^{\infty} \gamma^t r_t \middle| \mathbf{s}_0 = \mathbf{s} \right], \\ Q_{\boldsymbol{\pi}}(\mathbf{s}, \boldsymbol{a}) &\triangleq \mathbb{E}_{\mathbf{s}_{1:\infty} \sim P, \boldsymbol{a}_{1:\infty} \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t r_t \middle| \mathbf{s}_0 = \mathbf{s}, \boldsymbol{a}_0 = \boldsymbol{a} \right], \\ A_{\boldsymbol{\pi}}(\mathbf{s}, \mathbf{a}) &\triangleq Q_{\boldsymbol{\pi}}(\mathbf{s}, \mathbf{a}) - V_{\boldsymbol{\pi}}(\mathbf{s}). \end{aligned} \tag{2}$$

In fully cooperative settings with globally shared rewards, agents' local observations s^i and actions a^i are the main privacy-sensitive data. To formalize sequential policy updates among agents, let $i_{1:m} = (i_1, \dots, i_m)$ denote an ordered subsequence of agent indices, with $-i_{1:m}$ representing its complement in $\mathcal{N}$. Moreover, let $i_{1:n} \in \mathrm{Sym}(n)$ represent a random permutation from the symmetric group of order n.

3.2. Cryptographic primitives

This section introduces TDML's cryptographic primitives. We first define global notations: λ denotes the security parameter, and *p.p.t.* stands for probabilistic polynomial time. We use $\hat{x}$ to denote authentic data x, and $\tilde{x}$ for fabricated data x. A function $\epsilon(\lambda)$ is negligible if, for any polynomial $\mathrm{poly}(\lambda)$, $\epsilon(\lambda) < 1/\mathrm{poly}(\lambda)$ as $\lambda \to \infty$.

3.2.1. Pseudo-Random function

A Pseudo-Random Function (PRF) generates random-looking outputs computationally indistinguishable from a truly random function without knowledge of its secret key. A PRF scheme PRF $\triangleq$ (Set, Eval) consists of two algorithms:

- $prf \leftarrow \mathbf{PRF.Set}(1^\lambda)$: Generate a secret key prf based on the security parameter 1^λ.
- $r \leftarrow \mathbf{PRF.Eval}(prf, x)$: Compute a pseudo-random value r given the secret key prf and input x.

3.2.2. Commitment

A Commitment Scheme allows an entity to commit to a value, keeping it hidden until a later reveal. A commitment scheme COM $\triangleq$ (Set, Cmt, Open) consists of three algorithms:

- $com \leftarrow \mathbf{COM.Set}(1^\lambda)$: Generate public commitment parameters com.
- $c \leftarrow \mathbf{COM.Cmt}(com, x, r)$: Compute a commitment c for message x, using parameters com and randomness r.
- $bool \leftarrow \mathbf{COM.Open}(com, c, x, r)$: Verify if commitment c corresponds to message x and randomness r under parameters com.

3.2.3. Signature

A Signature scheme ensures message authenticity and integrity. A signature scheme SIG $\triangleq$ (Set, Sign, Vry) consists of three algorithms:

- $(sk, pk) \leftarrow \mathbf{SIG.Set}(1^\lambda)$: Generate a private key sk and public key pk based on the security parameter 1^λ.
- $\sigma \leftarrow \mathbf{SIG.Sign}(sk, x)$: Produce a signature σ for message x using private key sk.
- $bool \leftarrow \mathbf{SIG.Vry}(pk, x, \sigma)$: Verify the validity of signature σ for message x using public key pk.

3.2.4. Zero-knowledge deep learning

zkDL implements zero-knowledge verification for deep neural network computations. A zkDL scheme zkDL $\triangleq$ (Set, Pro, Vry) consists of three algorithms:

- $zk \leftarrow \mathbf{zkDL.Set}(1^\lambda, \mathcal{G})$: Generate system parameters zk for neural network based on security parameter 1^λ and network's computational graph $\mathcal{G}$. $\mathcal{G}$ details its operational flow (*e.g.*, activation functions) and data dependencies, excluding specific model weights.
- $\mathcal{P} \leftarrow \mathbf{zkDL.Pro}(zk, \mu, (x, W))$: Construct a zero-knowledge proof $\mathcal{P}$ for public statement μ, given system parameters zk and witness (x, W), where (x, W) are input data and model weights, and μ is the output.
- $bool \leftarrow \mathbf{zkDL.Vry}(zk, \mu, \mathcal{P})$: Verify if $\mathcal{P}$ is a valid proof for statement μ using system parameters zk.

4. Trustworthy distributed mirror learning

To facilitate understanding of the notation used throughout this paper, we provide a summary of key symbols and their definitions in Table 2. In the following, we present a detailed description of our proposed method.

4.1. System architecture

The TDML architecture comprises three types of entities: Agents, a Bulletin Board, and Auditors, as illustrated in Fig. 3. All entities operate within a decentralized network with no pre-assumed trusted parties. Agents interact with the environment based on a MARL game MG and perform policy learning. The Bulletin Board, deployed as a distributed ledger on Ethereum, records cryptographic metadata of critical operations. Auditors, acting as trusted third parties, are responsible for asynchronously verifying the compliance of agents' local behaviors.

First, each agent m locally runs an Ethereum node EN_m connected to the Ethereum network, which supports the execution of the smart contract-based monitoring policy MP. MP is a self-executing program whose global state is maintained collectively by all network nodes through a Proof-of-Stake (PoS) consensus mechanism. In iteration k, the system operates according to the following procedure:

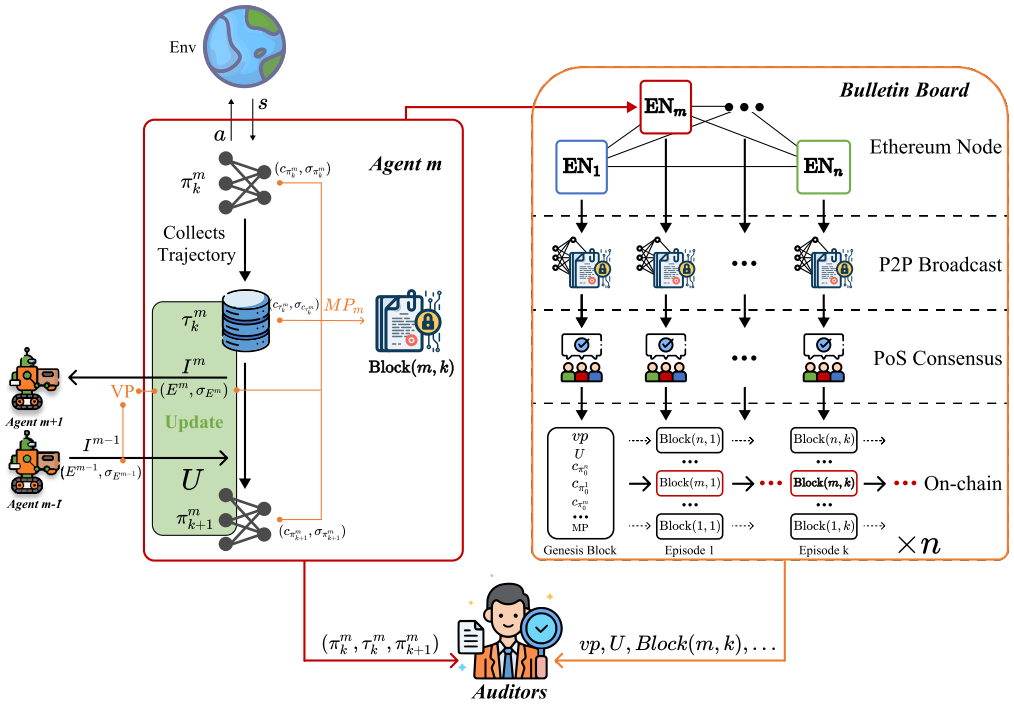

Fig. 3. TDML System Architecture. During system initialization, all agents register their cryptographic parameters and related metadata via Public Key Infrastructure (PKI) into the blockchain's Genesis Block. In iteration k, agent m performs local policy learning. It computes a new policy π_{k+1}^m using update rule U, based on its collected trajectory τ_k^m and received information I^m (which includes VP evidence and associated signatures). The monitoring policy MP_m then drives agent m to generate commitments for its policy and trajectory, along with corresponding zero-knowledge proofs and signatures. These are packaged into transactions and submitted to the agent's local Ethereum node EN_m. EN_m broadcasts the transactions over the peer-to-peer (P2P) network. Network nodes validate them, finalize inclusion through Proof-of-Stake (PoS) consensus, and record them on the blockchain. Any authorized party, such as a regulatory node, may act as an auditor by retrieving records from the bulletin board and issuing challenges to verify compliance.

Table 2
Key notation quick reference.

Symbol	Meaning
$\mathcal{N}$	Set of agents, $\mathcal{N} = \{1, \ldots, n\}$
$\boldsymbol{\pi}$	Joint policy, $\boldsymbol{\pi}(\mathbf{a}\|\mathbf{s}) = \prod_{i=1}^{n} \pi^i(a^i\|s^i)$
$\bar{\pi}$	Currently accepted policy
$\hat{\pi}$	Candidate update policy
I^{i_m}	Recursive information functional updated by agent i_m
$\mathcal{D}^{i_m}$	Distributed Heterogeneous Agent Drift Functional (DHADF)
$\mathcal{U}^{i_m}$	Distributed Neighborhood Operator (DNO)
$\mathcal{M}^{(\hat{\pi}^{i_m})}$	Distributed Heterogeneous Agent Mirror Operator (DHAMO)
$E = (c, \sigma_E, \mathcal{P})$	Evidence: commitment, signature, and zero-knowledge proof
$vp = (com, \{pk^i\}, zk)$	Global VP parameters (on-chain)
$loc^i = (prf^i, sk^i)$	Agent i's private parameters (local)
$\rho_{\boldsymbol{\pi}}^{\text{samp}}$	Sampling distribution for policy evaluation
$\mathbf{v}_t$	Sub-value vector input to parent value network
G_t^i	Gradient signal for agent i's sub-value network
$\bar{\boldsymbol{\phi}}$	Target critic network parameters
δ_t	Temporal Difference (TD) error for advantage estimation

Table 3
Attack Types and Robust Update Strategies. ✔ indicates self-tampering (modifying own data or policy), while ✗ indicates forging messages sent to other agents. In practical deployments, verification frequency can be dynamically adjusted, for example by incorporating probabilistic sampling, periodic verification, or trust scoring mechanisms, to balance security and efficiency.

Type	Self attack	Strategy
Single	✗	Move to the end of the update sequence
Single	✔	Repair in secure zone
Multiple	✗	Randomly delay one, repair others
Multiple	✔	All repaired in secure zone

1. Each agent m interacts with the environment based on its current policy π_k^m, collects trajectory data τ_k^m, and learns a new policy π_{k+1}^m locally. All computations occur locally, keeping raw data on-device for privacy.
2. During local updates, agents exchange messages I secured by the VP scheme. Each sender generates zero-knowledge evidence E and a signature σ_E. Recipients verify these before applying update rule U. If verification fails, the sender is flagged as malicious and the robust strategy in Table 3 is triggered.
3. To ensure accountability and non-repudiation, the monitoring policy MP_m, which acts as a local scheduler, drives agent m to generate cryptographic commitments and signatures for four items: the initial policy π_k^m, the trajectory τ_k^m, the updated policy π_{k+1}^m, and all zero-knowledge evidence-signature

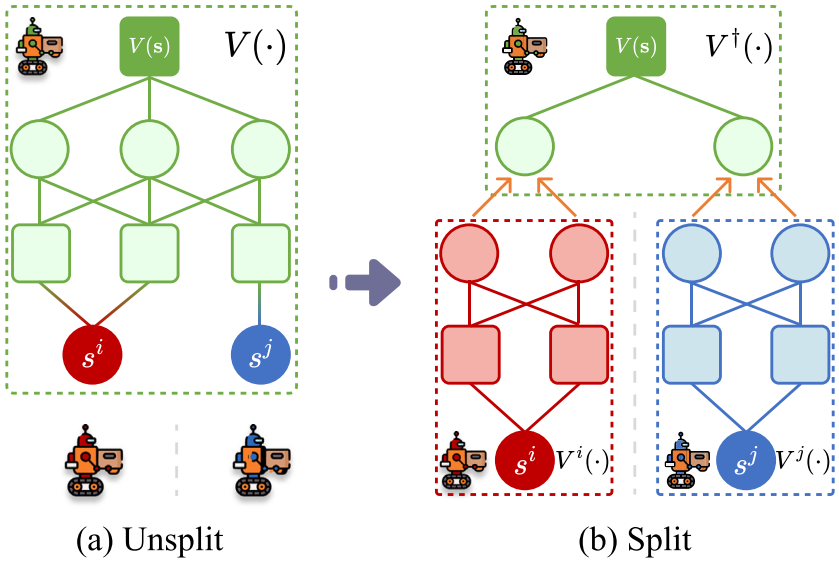

Fig. 4. Comparison between non-split and split computation of the global state value. (a) Conventional centralized value learning requires aggregating all agents' raw local states s^i to estimate $V(\mathbf{s})$, posing a privacy leakage risk. (b) In our method, each agent uploads only the compressed sub-value $V^i_{\phi^i}(s^i)$ generated by its local network; the global state value is then computed via aggregation, ensuring raw data never leaves the device and preventing privacy leakage.

pairs $\{(E, \sigma_E)\}$ sent in this round. These form Block$(m,k) \triangleq [(c_{\pi_k^m}, \sigma_{c_{\pi_k^m}}), (c_{\tau_k^m}, \sigma_{c_{\tau_k^m}}), \{(E,\sigma_E)\}, (c_{\pi_{k+1}^m}, \sigma_{c_{\pi_{k+1}^m}})]$. The block is broadcast to all network nodes via peer-to-peer (P2P) communication. After validating signatures, nodes package it into the current iteration's block, which is confirmed via PoS consensus and written to the blockchain.

Since the $c_{\pi_{k+1}^m}$ output in iteration k is the same value as the $c_{\pi_k^m}$ input in iteration $k+1$, commitments across iterations naturally form a linked chain for each agent. Although all Block(m,k) are recorded on the same Ethereum mainnet, the system generates n blocks in parallel per iteration (one per agent), thereby maintaining a logically independent behavioral trajectory chain for each agent m.

Whether agent m's policy update adheres to the public rule U is verified through off-chain asynchronous auditing. Specifically, an auditor retrieves agent m's on-chain record Block(m,k) for iteration k from the bulletin board and challenges the agent to provide the original data $(\pi_k^m, \tau_k^m, \pi_{k+1}^m)$. The auditor first uses SIG.Vry to verify that each signature σ was correctly generated using agent m's public key pk^m. Then, it uses COM.Open to verify the consistency between each commitment and the provided original data. Finally, the auditor applies the public update rule U to compute $\pi' = U(\pi_k^m, \tau_k^m)$ and checks whether $\pi' \stackrel{?}{=} \pi_{k+1}^m$. If any step fails, the agent is flagged as malicious.

The TDML framework's architectural design supports scalability from small deployments with a few agents to large-scale IoT systems with hundreds or thousands of agents. Its core lies in the fully decentralized P2P communication model, which eliminates bottlenecks associated with centralized communication. Concurrently, the sequential mechanism for agent policy updates ensures that the computational and communication overhead for each round strictly increases linearly with the total number of agents.

4.2. Verifiable privacy-preserving scheme

The VP scheme, as depicted in Fig. 3, is designed based on zero-knowledge proofs. It is defined as VP $\triangleq$ (Init, EGen, EVry), comprising three algorithms:

1. VP.Init$(1^\lambda, \{\mathcal{G}\})$: This algorithm generates global parameters vp and local parameters $\{loc\}$. The global parameters vp, including commitment scheme parameters com, public key set $\{pk\}$, and zkDL parameters zk, are written to the smart contract MP for network-wide access. The local parameters loc^m, comprising the PRF key prf^m and signing private key sk^m, are privately held by agent m.
2. VP.EGen$(vp, loc^m, I, (x, W))$: In iteration k, agent m generates evidence $E = (c, \mathcal{P})$ to declare I. The steps are as follows:
   (a) $r \leftarrow$ PRF.Eval$(prf^m, m \parallel k)$ to generate a deterministic random number.
   (b) $c \leftarrow$ COM.Cmt$(com, (x, W), r)$ to generate a commitment for the private data (x, W).
   (c) $\mathcal{P} \leftarrow$ zkDL.Pro$(zk, I, (x, W))$ to generate a zero-knowledge proof for information I.
   (d) $\sigma_E \leftarrow$ SIG.Sign$(sk^m, (c, \mathcal{P}))$, with agent m signing the evidence using its private key.
   (e) The tuple (E, σ_E) is submitted as a transaction to the local Ethereum node EN_m for on-chain recording, and separately transmitted to the receiving agent via the P2P network for immediate verification.
3. VP.EVry(vp, I, E, σ_E): The receiver verifies the validity of the evidence with the following steps:
   (a) Verify that commitment c is recorded on-chain by querying the bulletin board.
   (b) Verify the signature SIG.Vry$(pk^m, (c, \mathcal{P}), \sigma_E) \stackrel{?}{=}$ true.
   (c) Validate the zero-knowledge proof zkDL.Vry$(zk, I, \mathcal{P}) \stackrel{?}{=}$ true.

The VP scheme enables agents to locally generate and verify messages, ensuring the computational legitimacy of statement I while preserving the privacy of data x and model parameters W. This achieves verifiability and privacy protection for inter-agent communication.

4.3. Split advantage computation

Having discussed the design of the TDML method from a system architecture perspective, this section focuses on TDML's policy update algorithm. TDML is a trustworthy multi-agent distributed reinforcement learning method based on trust regions. In trust-region joint policy

optimization, estimating the value function $V(\mathbf{s})$ for the global state $\mathbf{s}$ is essential. As shown in Fig. 4(a), traditional methods require all agents to share their sensitive local states $\{s^i\}$ for centralized computation of $V(\mathbf{s})$, posing a significant risk of privacy leakage.

To address this, we propose a privacy-preserving Split Advantage Computation (SAC) method, which collaboratively estimates global value without sharing raw state data. As illustrated in Fig. 4(b), each agent $i \in \mathcal{N}$ is equipped with a sub-value network $V^i_{\phi^i}(\cdot)$. This network takes the agent's local state s^i as input and generates an intermediate representation $V^i_{\phi^i}(s^i)$. A designated advantage agent $i^\dagger \in \mathcal{N}$ then aggregates all $\mathbf{v} = \left(V^1_{\phi^1}(s^1), \ldots, V^n_{\phi^n}(s^n)\right)$, through a parent value network $V^\dagger_{\phi^\dagger}(\cdot)$ to compute the global state value $V(\mathbf{s}) = V^\dagger_{\phi^\dagger}(\mathbf{v})$. The advantage values can then be computed based on this global state value using Generalized Advantage Estimation (GAE) (Schulman et al., 2015):

$$\hat{A}_t = \sum_{l=0}^{\infty} (\gamma\beta)^l \delta_{t+l}, \tag{3}$$

where $\delta_t = r_t + \gamma V^\dagger_{\phi^\dagger}(\mathbf{v}_{t+1}) - V^\dagger_{\phi^\dagger}(\mathbf{v}_t)$, and $\beta \in [0,1]$ balances the trade-off between bias and variance. By mapping the raw state s^i to the intermediate representation $V^i_{\phi^i}(s^i)$ through a non-linear network, we eliminate its explicit association with the original input. This approach achieves data decoupling and privacy preservation during advantage computation.

Let $\boldsymbol{\phi} = (\phi^1, \ldots, \phi^n, \phi^\dagger)$ denote the set of critics parameters, and let $\bar{\phi}, \bar{\phi}^\dagger, \bar{\boldsymbol{\phi}}$ represent the corresponding target network parameters. The critic loss function is defined based on the Bellman error as follows:

$$\mathcal{L}^{\text{critic}}(\boldsymbol{\phi}) \triangleq \left(r_t + \gamma V_{\bar{\phi}^\dagger}(\bar{\mathbf{v}}_{t+1}) - V^\dagger_{\phi^\dagger}(\mathbf{v}_t)\right)^2, \tag{4}$$

where $\bar{\mathbf{v}}_{t+1} = \left(V^1_{\bar{\phi}^1}(s^1_{t+1}), \ldots, V^n_{\bar{\phi}^n}(s^n_{t+1})\right)$.

Let $\bar{\delta}_t = \left(r_t + \gamma V_{\bar{\phi}^\dagger}(\bar{\mathbf{v}}_{t+1}) - V^\dagger_{\phi^\dagger}(\mathbf{v}_t)\right)$. The parameters of the parent value network are updated according to the following rule:

$$\phi^\dagger \leftarrow \phi^\dagger - \alpha \cdot 2\bar{\delta}_t \frac{\partial V^\dagger_{\phi^\dagger}(\mathbf{v}_t)}{\partial \phi^\dagger}. \tag{5}$$

The sub-value networks receive gradient information $G^i_t = 2\bar{\delta}_t \frac{\partial V^\dagger_{\phi^\dagger}(\mathbf{v}_t)}{\partial V^i_{\phi^i}(s^i_t)}$ from the advantage agent $i^\dagger$, and use it to update their own parameters via:

$$\phi^i \leftarrow \phi^i - \alpha \cdot G^i_t \cdot \frac{\partial V^i_{\phi^i}(s^i_t)}{\partial \phi^i}. \tag{6}$$

We integrate the VP security scheme, introduced earlier, into this advantage computation process. This ensures the verifiability of all communication data (such as $V^i_{\phi^i}(s^i)$ and G^i_t). A detailed description of the SAC algorithm, which achieves both privacy preservation and verifiability, is provided in Algorithm 1.

Algorithm 1: Split Advantage Computation

Input: Stepsize α, discount factor γ, GAE lambda β, advantage agent $i^\dagger \in \mathcal{N}$, sub-value networks $\{\phi^i_k, \forall i \in \mathcal{N}\}$, parent value network $\phi^\dagger$, episode index k, security parameters $vp, \{loc\}$
Output: Updated sub-value networks $\{\phi^i_{k+1}, \forall i \in \mathcal{N}\}$, updated parent value network $\phi^\dagger_{k+1}$, advantage values $\hat{A}_t$, evidence $E^\dagger$
for each agent $i \in \mathcal{N}$ **do**
  Compute sub-values $\left(V^i_{\phi^i_k}(s^i_t), V^i_{\phi^i_k}(s^i_{t+1})\right)$
  Sign and prove: $(E^i, \sigma_{E^i}) \leftarrow$ VP.EGen $\left(vp,\ loc^i,\ \left(V^i_{\phi^i_k}(s^i_t), V^i_{\phi^i_k}(s^i_{t+1})\right),\ \left(s^i_t, s^i_{t+1}, \phi^i_k\right)\right)$
  if $i \neq i^\dagger$ **then**
    Transmit $\left(\left(V^i_{\phi^i_k}(s^i_t), V^i_{\phi^i_k}(s^i_{t+1})\right), E^i, \sigma_{E^i}\right)$ to advantage agent $i^\dagger$
  end
end
Advantage Agent $i^\dagger$ does:
for each received $\left(\left(V^i_{\phi^i_k}(s^i_t), V^i_{\phi^i_k}(s^i_{t+1})\right), E^i, \sigma_{E^i}\right)$ **do**
  if VP.EVry $\left(vp, \left(V^i_{\phi^i_k}(s^i_t), V^i_{\phi^i_k}(s^i_{t+1})\right), E^i, \sigma_{E^i}\right)$ fails: **then**
    Execute a robust strategy
  end
end
Estimate advantage value $\hat{A}_t$ using Eq. (3)
Sign and prove: $(E^\dagger_A, \sigma_{E^\dagger_A}) \leftarrow$ VP.EGen $\left(vp,\ loc^\dagger,\ \hat{A}_t,\ \left(\mathbf{v}_{t+1}, \mathbf{v}_t, \phi^\dagger\right)\right)$
Update parent value network $\phi^\dagger$ by Eq. (4) and Eq. (5)
Sign and prove: $(E^\dagger_{G^i}, \sigma_{E^\dagger_{G^i}}) \leftarrow$ VP.EGen $\left(vp,\ loc^\dagger,\ G^i_t,\ \left(\delta_t, \mathbf{v}_t, \phi^\dagger\right)\right)$
Transmit $\left(G^i_t, E^\dagger_{G^i}, \sigma_{E^\dagger_{G^i}}\right)$ to each agent $i \in \mathcal{N} \setminus \{i^\dagger\}$
for each agent $i \in \mathcal{N} \setminus \{i^\dagger\}$ **do**
  Receive $\left(G^i_t, E^\dagger_{G^i}, \sigma_{E^\dagger_{G^i}}\right)$ from advantage agent $i^\dagger$
  if VP.EVry$(vp, G^i_t, E^\dagger_{G^i}, \sigma_{E^\dagger_{G^i}})$ fails **then**
    Report error in advantage agent $i^\dagger$ and terminate
  end
  Update sub-critic networks by Eq. (6)
end

4.4. Trustworthy joint policy update

The TDML algorithm template achieves trustworthy joint policy updates through a three-phase process, as shown in Fig. 5. To rigorously formalize its behavior mathematically and support theoretical analysis, we introduce four key functionals and operators: the Information Functional (IF), Distributed Heterogeneous Agent Drift Functional (DHADF), Distributed Neighborhood Operator (DNO), and Distributed Heterogeneous Agent Mirror Operator (DHAMO). We define each formally below. First, we define necessary notations: let $\bar{\pi}$ denote the currently accepted policy, and $\hat{\pi}$ represent a candidate update policy. For any joint policy $\boldsymbol{\pi}$, we associate it with a positive and continuously state-dependent sampling distribution $\rho^{\text{samp}}_{\boldsymbol{\pi}}$ (Kuba et al., 2022), which guides policy evaluation and updates.

In the policy update phase of TDML, all messages transmitted between agents are uniformly modeled through an IF, which enables the compression of high-dimensional sensitive data and privacy protection.

Definition 1 (***Information Functional, IF***). Let $\mathcal{X}$, $\mathcal{W}$, and $\mathcal{I}$ be the data, network, and information spaces, respectively. These are finite sets (discrete spaces) or compact subsets of finite-dimensional Euclidean spaces, equipped with measure χ. For discrete spaces, $\chi(A) = |A|$; for continuous spaces, χ is the Lebesgue measure. We assume $\chi(\mathcal{I}) < \chi(\mathcal{X}) < \infty$ and $\chi(\mathcal{I}) < \chi(\mathcal{W}) < \infty$. The IF is defined as a mapping:

$$I : \mathcal{X} \times \mathcal{W} \times \mathcal{I} \to \mathcal{I},$$

where $I(x, W, I^{\text{prev}})$ is the non-injective result of joint input from data $x \in \mathcal{X}$, network parameters $W \in \mathcal{W}$, and received message $I^{\text{prev}} \in \mathcal{I}$. As a parameterized mapping, the IF compresses this joint input from the larger-measure $\mathcal{X}$, $\mathcal{I}$, and $\mathcal{W}$ spaces into the smaller-measure $\mathcal{I}$. This compression limits an attacker's success probability in inferring x from $I(x, W, I^{\text{prev}})$, manifesting as low discrete probability or near-zero probability density in continuous spaces due to the preimage set size.

Definition 2 (***Distributed Heterogeneous Agent Drift Functional, DHADF***).

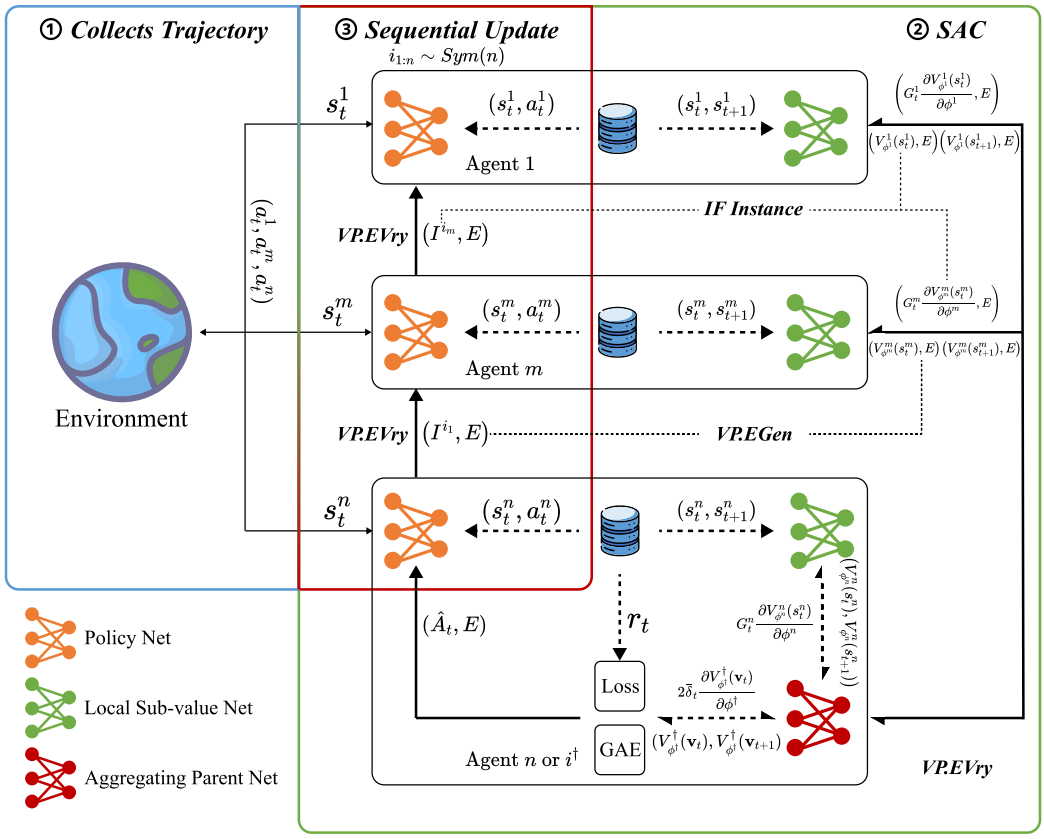

Fig. 5. TDML algorithm workflow. Each agent maintains a policy network and a local sub-value network. A designated advantage agent $i^\dagger$ additionally hosts an aggregating parent value network for global advantage computation. Secure inter-agent communication is indicated by solid arrows; dashed arrows denote internal data flow. For clarity, digital signatures are omitted — though all communications are signed in practice for non-repudiation. The algorithm operates in three phases: (1) Collects Trajectory: Agents $i \in \mathcal{N}$ interact with the environment under π_k^i, collecting $\{(s_t^i, a_t^i)\}$ locally. Raw data remains on-device for privacy. (2) SAC: Agents compute $V_{\phi^i}^i(s_t^i)$ and generate ZK evidence E via VP.EGen. Agent $i^\dagger$ aggregates verified inputs, computes $V(\mathbf{s})$ and $\hat{A}_t$ (via GAE), then broadcasts gradient signals G_t^i with evidence for sub-value network updates. (3) Sequential Update: A random permutation $i_{1:n}$ is sampled. Initial advantage $(\hat{A}_t, E_A^\dagger)$ is passed as I^{i_0} to i_1. Each agent i_m receives $(I^{i_{m-1}}, E^{i_{m-1}})$, verifies it via VP.EVry. If valid, updates π^{i_m} using DHAMO, then computes $I^{i_m} = \frac{\pi_{k+1}^{i_m}(a^{i_m}|s^{i_m})}{\pi_k^{i_m}(a^{i_m}|s^{i_m})} I^{i_{m-1}}$ with new evidence for next agent. If verification fails, sender is flagged as malicious and Table 3's robust strategy is triggered.

For an agent update sequence $i_{1:n} \in \mathrm{Sym}(n)$, the i_m-th agent in the sequence receives information $I^{i_{m-1}}$ containing projections of $\boldsymbol{\pi}, \bar{\boldsymbol{\pi}}^{i_{1:m-1}}$, and $\mathbf{s}$. The Distributed Heterogeneous Agent Drift Functional $\mathcal{D}^{i_m}$ for agent i_m is composed of a mapping:

$$\mathcal{D}^{i_m} : \mathcal{I} \times \Pi^{i_m} \times S^{i_m} \to \mathbb{R}_{\geq 0},$$

and satisfies the following conditions:

- Non-negativity: $\mathcal{D}^{i_m}_{I^{i_{m-1}}}(\hat{\pi}^{i_m}|s^{i_m}) \geq \mathcal{D}^{i_m}_{I^{i_{m-1}}}(\pi^{i_m}|s^{i_m}) = 0$
- Zero Gradient: Gateaux derivative of $\mathcal{D}^{i_m}_{I^{i_{m-1}}}(\hat{\pi}^{i_m}|s^{i_m})$ is zero in all directions at $\hat{\pi}^{i_m} = \pi^{i_m}$.

Intuitively, the drift $\mathcal{D}^{i_m}_{I^{i_{m-1}}}(\hat{\pi}^{i_m}|s^{i_m})$ measures the distance between policies $\hat{\pi}^{i_m}$ and π^{i_m} under the condition of information $I^{i_{m-1}}$. A typical instance of DHADF is the KL divergence $D^{i_m}_{I^{i_{m-1}}}(\hat{\pi}^{i_m}|s^{i_m}) = \mathrm{KL}\left(\hat{\pi}^{i_m}(\cdot|s^{i_m}) \parallel \pi^{i_m}(\cdot|s^{i_m})\right)$, whose non-negativity is guaranteed by KL divergence properties, and zero gradient naturally holds when $\hat{\pi}^{i_m} = \pi^{i_m}$.

Definition 3 (***Distributed Neighborhood Operator, DNO***)**.**

For an agent update sequence $i_{1:n} \in \mathrm{Sym}(n)$, the i_m-th agent in the sequence receives information $I^{i_{m-1}}$ containing projections of $\boldsymbol{\pi}$. The Distributed Neighborhood Operator $\mathcal{U}^{i_m}$ for agent i_m is composed of a mapping:

$$\mathcal{U}^{i_m} : \mathcal{I} \times \Pi^{i_m} \to \mathbb{P}(\Pi^{i_m})$$

where for all $\pi^{i_m} \in \Pi^{i_m}$, $\mathcal{U}^{i_m}_{I^{i_{m-1}}}(\pi^{i_m})$ contains a closed ball, *i.e.*, there exists a state-monotonically non-decreasing metric $\chi : \Pi^{i} \times \Pi^{i} \to \mathbb{R}$ such that there exists a radius $\delta^{i_m}_{I^{i_{m-1}}} > 0$ depending on $I^{i_{m-1}}$, satisfying $\chi(\pi^{i_m}, \bar{\pi}^{i_m}) \leq \delta^{i_m}_{I^{i_{m-1}}} \implies \bar{\pi}^{i_m} \in \mathcal{U}^{i_m}_{I^{i_{m-1}}}(\pi^{i_m})$.

Definition 4 (***Distributed Heterogeneous Agent Mirror Operator, DHAMO***)**.** For an agent update sequence $i_{1:n} \in \mathrm{Sym}(n)$, there exists an information chain where the information I^{i_m} at node i_m is recursively defined by the new-to-old policy ratio of agent i_m:

$$I^{i_m} = \frac{\bar{\pi}^{i_m}(a^{i_m}|s^{i_m})}{\pi^{i_m}(a^{i_m}|s^{i_m})} I^{i_{m-1}}$$

where the initial information is the advantage function $I^{i_0} = A_{\boldsymbol{\pi}}(\mathbf{s}, \mathbf{a})$. For agent i_m, given the received information $I^{i_{m-1}}$, the Distributed Heterogeneous Agent Mirror Operator $\mathcal{M}^{(\hat{\pi}^{i_m})}_{\mathcal{D}^{i_m}_{I^{i_{m-1}}}} I^{i_{m-1}}$ is defined as:

$$\left[\mathcal{M}^{(\hat{\pi}^{i_m})}_{\mathcal{D}^{i_m}_{I^{i_{m-1}}}} I^{i_{m-1}}\right](s^{i_m}) \triangleq \mathbb{E}_{a^{i_m} \sim \hat{\pi}^{i_m}}\left[\left(\frac{\hat{\pi}^{i_m}(a^{i_m} \mid s^{i_m})}{\pi^{i_m}(a^{i_m} \mid s^{i_m})} - 1\right) I^{i_{m-1}}\right] - \mathcal{D}^{i_m}_{I^{i_{m-1}}}\left(\hat{\pi}^{i_m} \middle| s^{i_m}\right).$$

Since DHADF is non-negative, any policy $\hat{\pi}^{i_m}$ that can improve DHAMO must increase agent i_m's multi-agent recursive advantage. Based on the above definitions, we propose the TDML algorithm template, detailed in Algorithm 2.

Algorithm Template 2: TDML

Input: Joint policy $\boldsymbol{\pi}_0 = (\pi_0^1, \dots, \pi_0^n)$, security parameter λ
Output: A limit-point joint policy $\boldsymbol{\pi}_\infty$
Initialize VP parameters: $vp, \{loc\} \leftarrow \text{VP.Init}(1^\lambda, \{\mathcal{G}\})$
for $k = 0, \dots, K-1$ **do**
  Draw a permutation $i_{1:n} \in \text{Sym}(n)$ of agents at random
  for advantage agent $i^\dagger$ **do**
    Compute the advantage $I^{i_0} = A_{\boldsymbol{\pi}_k}(\mathbf{s}, \mathbf{a})$ using a trustworthy distributed approach, *e.g.*, Algorithm 1
    Sign and Prove: $(E^\dagger, \sigma_{E^\dagger}) \leftarrow \text{VP.EGen}(vp, loc^\dagger, I^{i_0}, (\dots))$
    Transmit initial advantage tuple $(I^{i_0}, E^\dagger, \sigma_{E^\dagger})$ to agent i_1
  end
  for agent $m = 1:n$ **do**
    Receive tuple $\left(I^{i_{m-1}}, E^{i_{m-1}}, \sigma_{E^{i_{m-1}}}\right)$
    if $\text{VP.EVry}(vp, I^{i_{m-1}}, E^{i_{m-1}}, \sigma_{E^{i_{m-1}}})$ holds: **then**
      Update policy:

$$\pi_{k+1}^{i_m} = \underset{\pi^{i_m} \in \mathcal{U}^{i_m}_{I^{i_{m-1}}}(\pi_k^{i_m})}{\arg\max} \mathbb{E}_{s^{i_m} \sim \rho^{\text{samp}}_{\pi_k^{i_m}}} \left[\left[\begin{matrix} \mathcal{M}^{(\pi^{i_m})} \\ \mathcal{D}^{i_m}_{I^{i_{m-1}}} \end{matrix} I^{i_{m-1}} \right] (s^{i_m}) \right]$$

      Update Information $I^{i_m} = \frac{\pi_{k+1}^{i_m}(a^{i_m}|s^{i_m})}{\pi_k^{i_m}(a^{i_m}|s^{i_m})} I^{i_{m-1}}$
      Generate evidence and signature:

$$(E^{i_m}, \sigma_{E^{i_m}}) \leftarrow \text{Sign \& Prove}\left(vp, loc^{i_m}, I^{i_m}, \left(s^{i_m}, a^{i_m}, I^{i_{m-1}}, \pi_k^{i_m}, \pi_{k+1}^{i_m}\right)\right)$$

      Transmit $\left(I^{i_m}, E^{i_m}, \sigma_{E^{i_m}}\right)$ to agent i_{m+1}
    end
    else
      Execute a robust strategy
    end
  end
end

4.5. Practical algorithms derived from TDML

This section demonstrates how to derive practical MARL algorithms from the TDML template. By constructing appropriate DHADF, DNO, and sampling distribution $\rho_{\boldsymbol{\pi}}^{\text{samp}}$, we extend the classic single-agent reinforcement learning algorithms, A2C and PPO, into their trustworthy, distributed variants: TDA2C and TDPPO.

For a policy π_θ parameterized by θ, A2C updates parameters via policy gradients, with its objective function being:

$$J^{\text{A2C}}(\hat{\theta}) = \mathbb{E}_{s\sim\rho_{\pi_\theta}, a\sim\pi_\theta}\left[\mathbf{r}(\hat{\theta}) A_{\pi_\theta}(s,a)\right] \tag{7}$$

where $\mathbf{r}(\hat{\theta}) = \frac{\pi_{\hat{\theta}}(a|s)}{\pi_\theta(a|s)}$ is the new-to-old policy ratio.

PPO, on the other hand, enhances stability by introducing a trust region constraint, with its objective function's gradient being:

$$J^{\text{PPO}}(\hat{\theta}) = \mathbb{E}_{s\sim\rho_{\pi_\theta}, a\sim\pi_\theta}\left[\min\left(\mathbf{r}(\hat{\theta}) A_{\pi_\theta}(s,a), \text{clip}\left(\mathbf{r}(\hat{\theta}), 1\pm\epsilon\right) A_{\pi_\theta}(s,a)\right)\right] \tag{8}$$

Consider a multi-agent system of n agents, which update sequentially according to a permutation $i_{1:n} \in \text{Sym}(n)$. Under TDML, we define the following core components:

- DNO: $\mathcal{U}^{i_m} \triangleq \Pi^{i_m}$, unconstrained optimization within the policy space.
- Sampling Distribution $\rho_{\boldsymbol{\pi}_\theta}^{\text{samp}}$: On-policy trajectory data is used, consistent with A2C and PPO implementations.

For TDA2C, the DHADF is defined as $\mathcal{D}^{i_m} \triangleq 0$. Based on Algorithm Template 2, Eq. (6) is reformulated as:

$$\arg\max_{\hat{\theta}} \mathbb{E}_{s^{i_m}\sim\rho_{\pi_{\theta^{i_m}}^{i_m}}, a^{i_m}\sim\pi_{\hat{\theta}^{i_m}}^{i_m}}\left[\left(\mathbf{r}(\hat{\theta}^{i_m}) - 1\right) I^{i_{m-1}}\right]. \tag{9}$$

For TDPPO, the DHADF is defined as:

$$\mathcal{D}^{i_m}_{I^{i_{m-1}}} \triangleq \mathbb{E}_{a^{i_m}\sim\pi_{\theta^{i_m}}^{i_m}}\left[\text{ReLU}\left(\left[\mathbf{r}(\hat{\theta}) - \text{clip}\left(\mathbf{r}(\hat{\theta}), 1\pm\epsilon\right)\right]\cdot\mathbf{r}(\hat{\theta}) I^{i_m}\right)\right],$$

Based on Algorithm Template 2, Eq. (7) is reformulated as:

$$\begin{aligned}\max_{\hat{\theta}} \mathbb{E}_{s^{i_m}\sim\rho_{\pi_{\theta^{i_m}}^{i_m}}} \Bigg[& \mathbb{E}_{a^{i_m}\sim\pi_{\hat{\theta}^{i_m}}^{i_m}}\left[\left(\mathbf{r}(\hat{\theta}^{i_m}) - 1\right) I^{i_{m-1}}\right] \\ & - \mathbb{E}_{a^{i_m}\sim\pi_{\theta^{i_m}}^{i_m}}\left[\text{ReLU}\left(\left[\mathbf{r}(\hat{\theta}) - \text{clip}\left(\mathbf{r}(\hat{\theta}), 1\pm\epsilon\right)\right]\cdot\mathbf{r}(\hat{\theta}) I^{i_m}\right)\right] \Bigg]\end{aligned} \tag{10}$$

This can be proven to be equivalent to:

$$\max_{\hat{\theta}} \mathbb{E}_{s^{i_m}\sim\rho_{\pi_{\theta^{i_m}}^{i_m}}, a^{i_m}\sim\pi_{\theta^{i_m}}^{i_m}}\left[\min\left(\mathbf{r}(\hat{\theta}^{i_m})^2 I^{i_{m-1}}, \text{clip}\left(\mathbf{r}(\hat{\theta}^{i_m}), 1\pm\epsilon\right)\mathbf{r}(\hat{\theta}^{i_m}) I^{i_{m-1}}\right)\right]. \tag{11}$$

This form mirrors PPO's clipped objective by constraining updates within the trust region.

5. Trustworthy analysis

This section delves into the inherent trustworthy mechanisms of the TDML framework from three dimensions: convergence guarantees, verifiability defenses, and privacy preservation mechanisms.

5.1. Convergence

In MARL, policies must stably converge to a Nash Equilibrium (NE) to ensure predictable and reliable system performance. This convergence forms the theoretical foundation for building trustworthy agent collaboration systems. This is especially critical in industrial scenarios requiring long-term autonomous operation, such as smart grid dispatch and warehouse robot clusters. In these settings, algorithms must ensure policies remain stable and effective in complex, dynamic environments to prevent failures from policy oscillations or divergence at critical moments.

TDML inherits and extends the theoretical framework of MATRL, rigorously guaranteeing that the joint policy sequence converges to a NE even in distributed, heterogeneous, and adversarial environments. This convergence is cooperatively ensured by a three-fold mechanism:

- Distributed Neighborhood Operator (DNO): Provides hard limits on the policy update range, stabilizing policy updates.
- Distributed Heterogeneous Agent Drift Functional (DHADF): Offers a "policy potential gradient" to guide the update direction and control the degree of deviation.
- Distributed Heterogeneous Agent Mirror Operator (DHAMO): Transforms the multi-agent sequential advantage structure into an optimizable objective, ensuring that each update improves overall utility.

As illustrated in Fig. 6, this tripartite structure synergistically stabilizes the learning dynamics across distributed agents. We present its formal convergence guarantee in Theorem 1. The complete proof is provided in the Appendix A.

Theorem 1 (*Convergence*). *Let $\boldsymbol{\pi}_0 \in \boldsymbol{\Pi}$, $i_{1:n} \sim \text{Sym}(n)$, $m = 1, \dots, n$, and the sampling distribution $\rho_{\boldsymbol{\pi}}^{\text{samp}}$ continuously depend on $\boldsymbol{\pi}$. The joint policy sequence $(\boldsymbol{\pi}_k)_{k=0}^\infty$ is generated by the TDML algorithm induced by $\mathcal{D}^{i_m}_{I^{i_{m-1}}}, \mathcal{U}^{i_m}_{I^{i_{m-1}}}$, and $\rho_{\boldsymbol{\pi}}^{\text{samp}}$. Then, the joint policy output by this algorithm satisfies the following properties:*

- *Monotonic Improvement Property:* $J(\boldsymbol{\pi}_{k+1}) \geq J(\boldsymbol{\pi}_k)$.

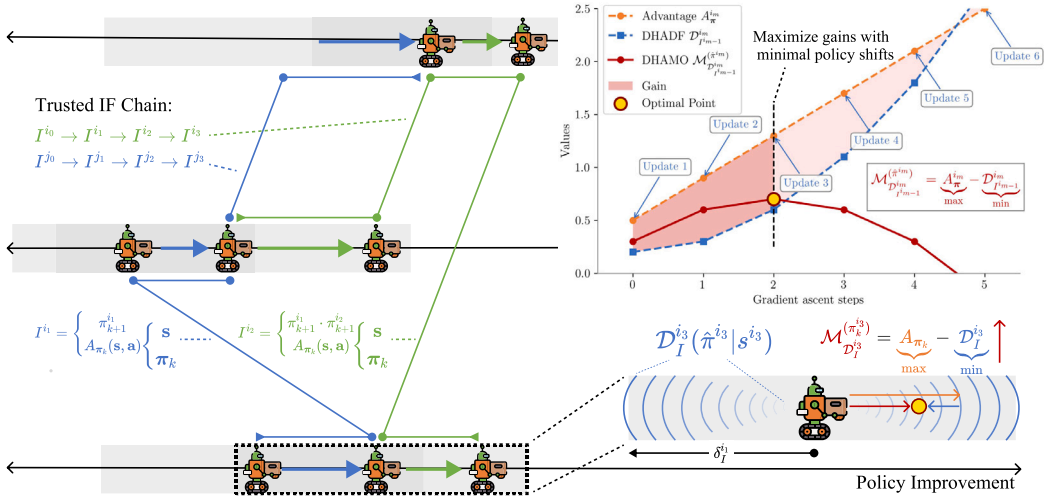

Fig. 6. TDML's Joint Policy Update Mechanism. Agents update their policies sequentially according to a random permutation $i_{1:n}$. Each agent i_m receives preceding information $I^{i_{m-1}}$ (containing projections of joint policies and states) and searches for an optimal update direction within a local neighborhood of its policy space. This neighborhood, defined by the DNO, constrains the update magnitude to prevent policy oscillations. The DHADF acts as a "potential field", guiding the policy to move smoothly within the trust region, thereby avoiding policy collapse due to excessively large update steps. The final optimization objective is defined by DHAMO, which integrates sequential advantages with a DHADF penalty term. This ensures stable policy updates by maximizing performance improvement while minimizing policy deviation. The line graph illustrates this trade-off: while larger update steps can lead to more significant advantage gains (orange curve), they also cause a sharp increase in the difference between old and new policies (blue curve), potentially leading to policy instability or collapse. The optimal update occurs within the "trust region" (red shaded area), where DHAMO is maximized under DHADF constraints, balancing performance and stability.

- *Value Function Convergence to Nash Value Function:* $\lim_{k\to\infty} V_{\boldsymbol{\pi}_k} = V^{NE}$.
- *Expected Return Convergence to Nash Return:* $\lim_{k\to\infty} J(\boldsymbol{\pi}_k) = J^{NE}$.
- *ω-limit set consists of Nash Equilibria.*

5.2. Verifiability

TDML's verifiability is collectively supported by the VP scheme, the auditor mechanism, and the bulletin board: the VP scheme ensures the cryptographic integrity of communication messages, the auditor verifies behavioral compliance through off-chain challenges, and the bulletin board (based on Ethereum smart contracts) provides tamper-proof operational records. Combined, these three elements achieve real-time verification, post-hoc auditing, and non-repudiation of actions. First, Theorem 2 formally characterizes the VP scheme's security against forgery. For a detailed proof, please refer to Appendix B.

Theorem 2 (*Verifiability*). *Let $\epsilon_{binding}(\lambda)$, $\epsilon_{euf\text{-}cma}(\lambda)$, and $\epsilon_{se}(\lambda)$ be negligible functions of the security parameter λ, representing the probabilities of breaking the commitment scheme's binding property, the signature scheme's the existential unforgeability under chosen-message attack, and the zkDL scheme's simulation extractability, respectively. Let $l(\lambda)$ be the bit length of the signature private key sk generated by SIG.Set(1^λ). Then, for any p.p.t. adversary $\widetilde{\mathcal{A}}$, the probability of successfully forging valid evidence $E = (c, \sigma, \mathcal{P})$ for a statement $\widetilde{I}$ is bounded by:*

$$\Pr[Forge^{VP}_{\widetilde{\mathcal{A}}}(\lambda) = 1] \leq \epsilon_{binding}(\lambda) + \epsilon_{euf\text{-}cma}(\lambda) + \epsilon_{se}(\lambda) + 2^{-l(\lambda)}.$$

This theorem demonstrates that for an attacker to forge verifiable communication evidence, they must simultaneously break the commitment's binding property, the existential unforgeability of the digital signature, the extractability of the zero-knowledge proof, and the secrecy of the private key. Since these security assumptions are computationally hard to break, the probability of successful forgery is negligible. In other words, any verified message can be cryptographically assured to have been legitimately generated by the sender based on authentic local data and in strict adherence to protocol rules.

By combining Theorem 2 with the auditing and bulletin board mechanisms, TDML can effectively defend against the following typical attacks (Zhou et al., 2024; Mo et al., 2024):

- Policy Poisoning and Backdoor Attacks: If a malicious node tampers with an advantage value $\widetilde{A}$, it must simultaneously forge the commitment c, signature σ, and zero-knowledge proof $\mathcal{P}$. According to Theorem 2, the success probability of this forgery is bounded by a negligible function.
- Adaptive Poisoning Attacks: Regardless of how an attacker modifies forged data, as long as its computation path is inconsistent with the local private input, the extractability guarantee of zkDL will prevent it from generating a valid $\mathcal{P}$.
- Collusion Attacks: Each agent's private key is independent, preventing them from forging others' signatures or proofs. Even if multiple nodes collude, it does not reduce the difficulty of forgery. The system relies on the robust update strategies in Table 3 for isolation and repair.
- Replay Attacks: Messages are bound to the iteration round k and sequence index m. The recipient verifies context consistency, and expired or out-of-order messages will be rejected.
- Sybil Attacks: These rely on the PKI registration mechanism during the initialization phase. Unregistered entities lack a legitimate private key sk, and the probability of forgery is bounded by $2^{-l(\lambda)}$.
- Bulletin Board Tampering: As a smart contract deployed on Ethereum, its historical state is protected by PoS consensus. Tampering would require controlling more than 2/3 of validators and incurring extremely high economic costs, making it practically infeasible.
- Malicious Auditor: An auditor only possesses "post-hoc challenge rights" and no "real-time intervention rights". Even if an auditor makes malicious accusations, these can be disproved through on-chain commitments and signatures, ensuring that accusations are falsifiable.

In critical IoT application scenarios, such as medical monitoring or industrial control, privacy protection should not come at the expense of

accountability. TDML, through its "verifiable non-repudiation" mechanism, preserves full behavioral auditability while hiding raw data. Although zero-knowledge proofs do not disclose input, a legitimate auditor can still issue an off-chain challenge, requesting the agent to provide raw data and digital signatures. This, combined with on-chain commitments from the bulletin board, allows verification that the agent's behavior strictly adhered to protocol rules. This design achieves a balance where "privacy does not hinder accountability, and confidentiality does not impede auditing". The system mechanistically prevents the misuse of privacy technology to evade responsibility, ensuring that technology is not exploited to avoid accountability.

5.3. Privacy protection

TDML achieves strong privacy protection through a synergistic three-tiered mechanism: structural, cryptographic, and storage. At the structural level, based on the split learning paradigm, agents only exchange low-dimensional intermediate representations (e.g., $V^i_{\phi^i}(s^i)$) generated by their local networks. These representations, as instances of the IF, are decoupled from the original state s^i, significantly increasing the difficulty for an attacker to reconstruct original inputs from communication data. At the cryptographic level, the VP scheme leverages zero-knowledge proofs to ensure that verifiers cannot obtain any additional information about private data (x, W) from the evidence $\mathcal{P}$. At the storage level, the bulletin board only records commitments c, whose cryptographic hiding property ensures that even if the data is public, an attacker cannot reconstruct the original content.

Regarding the VP scheme itself, its privacy can be formally characterized by Theorem 3. Full proof in Appendix B.

Theorem 3 (*Privacy*). *Let $\epsilon_{hinding}(\lambda)$ and $\epsilon_{zk}(\lambda)$ be negligible functions of the security parameter λ, representing the probabilities of breaking the commitment scheme's hiding property and the zkDL scheme's zero-knowledge property, respectively. Then, for any p.p.t. adversary $\widetilde{\mathcal{A}}$, the probability of successfully breaching the privacy of the VP scheme is bounded by:*

$$\Pr[\textit{privacy breach}|\widetilde{\mathcal{A}}] \leq \frac{1}{2} + \epsilon_{hiding}(\lambda) + \epsilon_{zk}(\lambda).$$

This upper bound indicates that even with powerful computational capabilities, an adversary's success rate in inferring raw data from VP communications is only slightly higher than random guessing, with its excess advantage strictly limited by two negligible functions. In other words, under VP protection, communication content is computationally indistinguishable from random noise to an adversary, thereby achieving strong privacy guarantees in a cryptographic sense.

6. Experiments

This section evaluates the TDML algorithm's performance across its key aspects: convergence, verifiability, and privacy preservation. To ensure methodological rigor, our experimental process follows the standard scientific workflow of planning, execution, analysis, and reporting (Risan et al., 2024b). We begin by comparing the overall performance of two TDML implementations (TDPPO and TDA2C) against prevalent methods on standard MARL benchmarks. We then conduct ablation studies to analyze the impact of split advantage computation on performance, communication overhead, and privacy preservation. Finally, we assess the computational and communication overhead of the security scheme VP across different model and data scales.

6.1. Experimental settings

6.1.1. Benchmarks

Our method is evaluated on MPE, SMAC, and Bi-DexHands MARL benchmarks, as shown in Fig. 7. These benchmarks cover four environments with both discrete and continuous action spaces:

- MPE (Multi-Agent Particle Environment): The Simple Spread task (Lowe et al., 2017) involves three agents cooperatively covering 2D target points while avoiding collisions. This benchmark includes both continuous and discrete action space variants.
- SMAC (StarCraft Multi-Agent Challenge): The corridor task (Samvelyan et al., 2019) features six Zealots combating 24 Zerglings. Agents collaborate through discrete actions (movement, attack) to test coordination and tactical stability.
- Bi-DexHands (Bimanual Dexterous Hands): In the ShadowHandOver task (Chen et al., 2023), two Shadow Hands (20 DoF each) cooperatively pass an object using high-dimensional continuous torque control.

6.1.2. Baselines

We selected the following methods as baselines for comparison:

- **HAA2C/HAPPO** (Zhong et al., 2024): SOTA heterogeneous multi-agent reinforcement learning methods.
- **PDPPO** (Lu et al., 2025): A privacy-preserving Multi-Agent PPO (Yu et al., 2022) variant under a federated learning framework.
- **PP-MARL** (Yuan et al., 2024): A representative cryptography-enhanced MARL method that integrates homomorphic encryption (HE) and differential privacy (DP) on top of the MADDPG (Lowe et al., 2017) framework, aiming to provide strong privacy guarantees.

6.1.3. Evaluation metrics

The evaluation metrics are:

- **Convergence Performance (CP)**: Average episode reward over training steps; Win Rate is used for SMAC.
- **Gaussian Attack Performance (GAP)** (He et al., 2023): Measures convergence under Gaussian noise attacks, where $\widetilde{I}^j_k \sim \mathcal{N}(I^j_k, \sigma)$ is injected into messages from a random agent $j \in \mathcal{N}$.
- **Sign Attack Performance (SAP)** (Fang and Chen, 2023): Evaluates robustness to sign-flipping attacks, with $\widetilde{I}^j_k = -I^j_k$ simulating malicious intent.
- **Privacy Reconstruction Error (PRE)**: Quantifies how well an adversary can reconstruct raw state/action data from transmitted intermediate information (*e.g.*, advantages, gradients). Specifically, during training, we record the intermediate information transmitted in each round and their corresponding raw state/action data. We then use a supervised learning model to reconstruct the original data from this intermediate information.
- **Communication Traffic Overhead (CTO)**: The average amount of data transmitted per episode.

The attack scenarios described above simulate two typical types of threats in real-world systems: Gaussian noise attacks reflect communication channel perturbations or mild data poisoning, while sign-flipping attacks simulate malicious nodes specifically tampering with the direction of policy updates. These two types cover a spectrum of threats from minor disturbances to high-intensity tampering, effectively testing the efficacy of the VP verification mechanism and robust update strategies under different attack strengths. More complex attack forms (e.g., collusion, replay) have been formally analyzed in Section 5 through cryptographic and protocol-level mechanisms. This experiment focuses on observable and quantifiable performance degradation metrics to ensure that evaluation results have clear comparative benchmarks and reproducibility.

6.1.4. Implementation details

The implementations of HAPPO and HAA2C are based on the open-source code provided by Zhong et al. (2024). For PDPPO, we adapted the algorithm implementation by Lu et al. (2025). Specifically, we

(a) MPE: Spread (continuous) (b) MPE: Spread (discrete) (c) SMAC: Corridor (d) Bi-Dexhands: ShadowHand Over

Fig. 7. Overview of the four environments across three benchmarks.

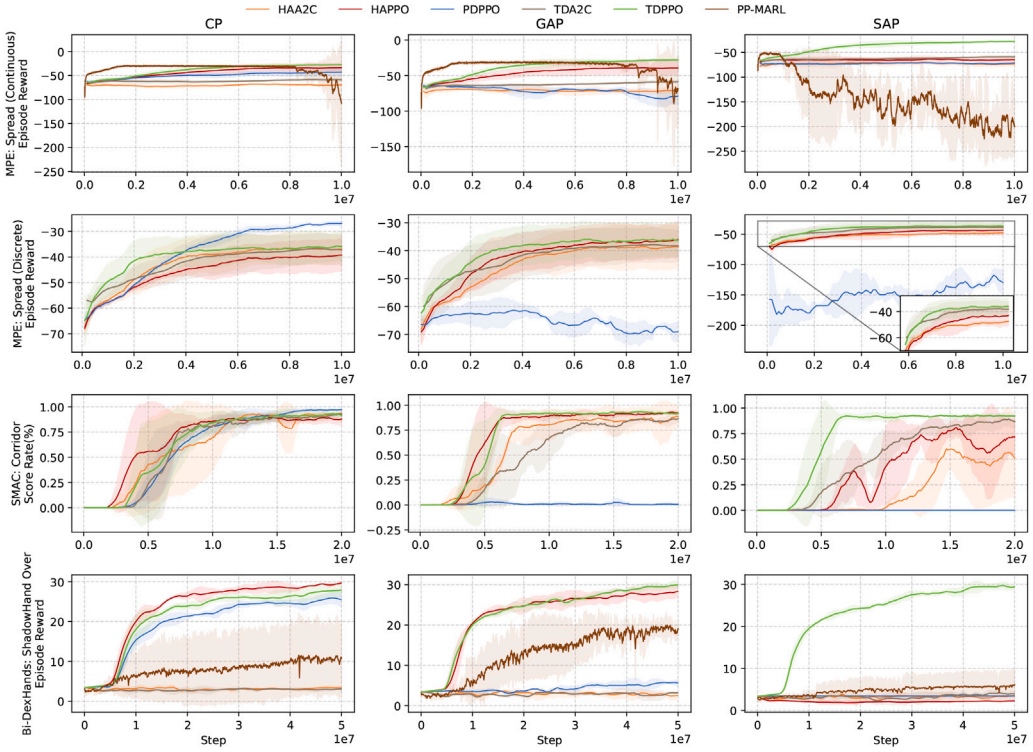

Fig. 8. Training performance curves for various algorithms. The x-axis represents training steps (episodes), and the y-axis represents the performance metric (Win Rate for SMAC Corridor, average episode return for other tasks). Each subplot corresponds to an attack scenario, from left to right: no attack, Gaussian noise attack, and sign-flipping attack. The compared algorithms include TDPPO, TDA2C, HAPPO, HAA2C, and PDPPO. PP-MARL is only applicable to continuous action space tasks and is thus not plotted in the Spread (Discrete) and SMAC Corridor environments. Shaded regions indicate the standard deviation of results across three random seeds.

removed the reward estimation module, which is designed for heterogeneous reward scenarios. For PP-MARL, we integrated homomorphic encryption (HE) and differential privacy (DP) into Zhong et al.'s MADDPG framework (Zhong et al., 2024). HE was implemented using the TenSEAL library (Benaissa et al., 2021) (CKKS scheme), and DP noise was calibrated according to the (ϵ, δ)-budget specified in the original paper (Yuan et al., 2024).

To ensure a fair comparison, all methods, including PDPPO, TDPPO, and TDA2C adopted the same default common hyperparameters as HAPPO and HAA2C. For PDPPO, the federated aggregation interval was set to 5. In TDPPO and TDA2C, the dimension of each agent's Critic intermediate representation $V^i_{\phi^i}(s^i)$ was set to 8 in the SMAC Corridor environment and 1 in other environments.

Regarding the VP scheme, we instantiated the Pseudo-Random Function scheme using HMAC and SHA-256 hash function (Bellare et al., 1996). The Pedersen commitment scheme (Pedersen, 1991) was employed for commitments, and the Boneh–Boyen signature scheme (Boneh et al., 2004) for signatures.

All experiments were conducted on two Linux servers, each equipped with an NVIDIA GeForce RTX 3090 GPU, an Intel Core i7-12700K CPU, and 64 GB of RAM. Each set of experiments was repeated with three different random seeds to ensure statistical significance. This practice follows established principles of experimental design, ensuring the statistical robustness of our results (Risan et al., 2024a). More detailed training configurations are listed in Appendix D.

6.2. Main results

The training curves for various algorithms across different benchmark tasks are shown in Fig. 8. Quantitative comparisons of Privacy Reconstruction Error (PRE) and average Communication Traffic Overhead (CTO) per episode are presented in Table 4. Overall, TDPPO consistently demonstrates competitive performance across most tasks, exhibiting superior stability, particularly in adversarial attack scenarios.

In attack-free environments, TDPPO demonstrates robust performance across most tasks. For instance, in the SMAC Corridor scenario, its final win rate is comparable to state-of-the-art methods like HAPPO and HAA2C. This indicates that TDML does not sacrifice the performance of original trust-region learning methods, aligning with the conclusion of Theorem 1.

Table 4
Comparison of privacy reconstruction error and communication traffic overhead.

Metric (↑ / ↓)	Environment	HAA2C	HAPPO	PDPPO	PP-MARL	TDA2C	TDPPO
PRE ↑	Spread (continuous)	0.01 ± 0.01	0.01 ± 0.00	1.0 ± 0.00	1.00 ± 0.00	0.90 ± 0.03	0.96 ± 0.01
	Spread (discrete)	0.01 ± 0.01	0.01 ± 0.00	1.0 ± 0.00	–	0.94 ± 0.02	0.97 ± 0.00
	Corridor	0.07 ± 0.00	0.08 ± 0.00	1.0 ± 0.00	–	0.45 ± 0.02	0.37 ± 0.01
	ShadowHandOver	0.03 ± 0.00	0.04 ± 0.02	0.98 ± 0.03	1.00 ± 0.00	0.92 ± 0.02	0.99 ± 0.00
CTO ↓	Spread (continuous)	56.25 ± 0.00	56.25 ± 0.00	94.42 ± 0.00	281488.67 ± 0.00	91.89 ± 0.00	91.89 ± 0.00
	Spread (discrete)	46.88 ± 0.00	46.88 ± 0.00	94.39 ± 0.00	–	91.89 ± 0.00	91.89 ± 0.00
	Corridor	1305.00 ± 0.00	1305.00 ± 0.00	618.77 ± 0.00	–	207.67 ± 0.00	207.67 ± 0.00
	ShadowHandOver	135.94 ± 0.00	135.94 ± 0.00	147.44 ± 0.00	91997.17 ± 0.00	395.38 ± 0.00	395.38 ± 0.00

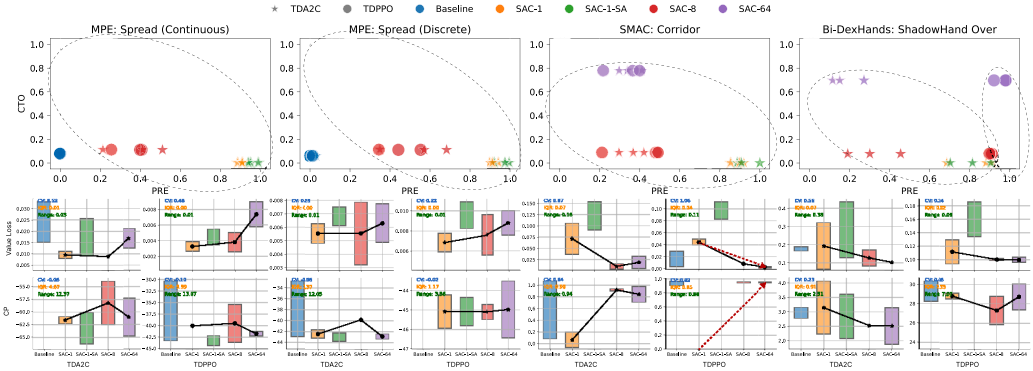

Fig. 9. Threefold impact of intermediate representation dimension on CP, CTO, and PRE. The top scatter plot illustrates the relationship between PRE and CTO across different representation dimensions. Bar charts below show the distribution of CP and Value Loss across random seeds. Value Loss is introduced to reflect the Critic network's estimation accuracy, with lower values indicating more stable value function learning. To assess stability, we also report Coefficient of Variation (CV), Range, and Interquartile Range (IQR), which quantify data dispersion and clustering—smaller values indicate less variability and stronger consistency.

Under light Gaussian noise interference, TDPPO maintains stable performance and achieves good results in the Spread (Continuous), Corridor, and ShadowHandOver tasks. Notably, in some environments, methods without explicit security guarantees, such as HAPPO, show a slight performance improvement. This can be attributed to the noise injection potentially enhancing exploration efficiency, allowing policies to escape local optima faster.

Under high-intensity sign-flipping attacks, TDPPO achieves the best performance across all four benchmark environments, highlighting its exceptional robustness against strong adversarial attacks. For example, in the ShadowHandOver task, TDPPO consistently maintains an average episode reward of approximately 30 points under attack, significantly outperforming other methods. This performance stems from the VP scheme's 100% accuracy in identifying malicious messages, which ensures that attack information is effectively isolated and triggers robust update strategies.

Table 4 presents the PRE and CTO. In terms of privacy preservation, TDPPO and TDA2C's PRE is slightly lower than PDPPO and PP-MARL, but still maintains a high level of 0.90–0.99. This demonstrates that SAC effectively blocks the path for attackers to infer raw state and action data from intermediate representations. In contrast, PDPPO and PP-MARL leverage cryptographic methods such as federated aggregation, homomorphic encryption, or differential privacy, inherently possessing stronger information-hiding capabilities. Nevertheless, TDML's 90%+ reconstruction error significantly increases the difficulty for attackers to recover original data in practice, notably outperforming HAPPO/HAA2C, which lack privacy mechanisms.

Regarding communication efficiency, TDPPO and TDA2C maintain a low CTO. This is primarily attributed to the data compression effect achieved by low-dimensional intermediate representations in SAC, effectively avoiding direct transmission of original high-dimensional observations or gradients. In comparison, PP-MARL's communication overhead is significantly higher, mainly due to the inherent ciphertext expansion effect of its homomorphic encryption mechanism. Meanwhile, the additional on-chain communication costs introduced by the VP scheme are relatively controllable and do not pose a significant burden on overall efficiency.

Furthermore, we observed that TDA2C, based on A2C, generally underperformed compared to the PPO-based TDPPO across various tasks, particularly in continuous action spaces. This is attributed to the fact that A2C lacks explicit policy update constraints, making it more prone to advantage estimation errors and training instability. These results suggest that the effectiveness of TDML-based methods is closely tied to the quality of their base algorithms. Therefore, building TDML upon well-founded and empirically successful algorithms like PPO is essential for developing high-performance and trustworthy multi-agent systems.

6.3. Ablation on split advantage computation

To evaluate the impact of SAC on algorithm convergence, privacy preservation, and communication efficiency, we compare different SAC variants of TDPPO and TDA2C against baseline methods utilizing centralized advantage computation.

In our SAC configurations, we set the dimension of the intermediate representation to three different values: 1, 8, and 64 (denoted as SAC-1, SAC-8, and SAC-64, respectively). This allows for a detailed analysis of how varying dimensions specifically influence performance, privacy preservation, and communication overhead. Additionally, we designed a sign-flipping attack (SA) scenario (denoted as SAC-1-SA), simulating a situation where a single agent outputs anomalous intermediate values. In this case, the anomalous output is replaced by the mean of the intermediate values from other legitimate agents.

Fig. 9 illustrates the threefold impact of different SAC settings on CP, CTO, and PRE. Firstly, SAC variants significantly outperform baselines in terms of PRE, demonstrating their effectiveness in decoupling raw data from intermediate information. Secondly, as the dimension of the intermediate representation decreases, communication overhead is proportionally reduced, which highlights the compression effect of intermediate representations. These two findings jointly characterize a

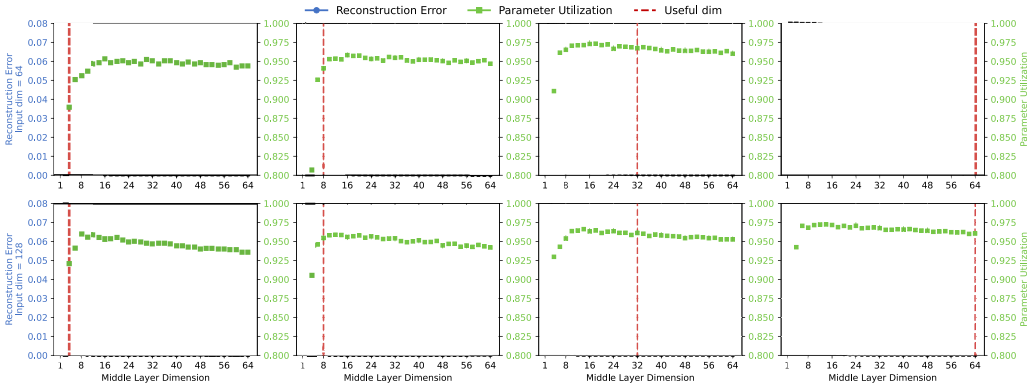

Fig. 10. Supervised regression experiment. The x-axis represents the intermediate layer dimension (1–64), the left y-axis denotes the Reconstruction Error (MSE), and the right y-axis indicates the Parameter Utilization Rate (defined as the proportion of non-zero weights). Experiments were conducted with two input dimensions (first row: 64, second row: 128) and four task-intrinsic dimensions (Useful Dim = 4, 8, 32, 64). For each group, a regression model was trained using 1000 samples.

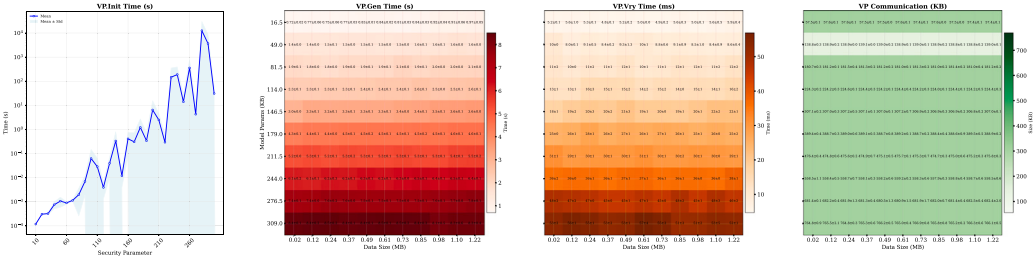

Fig. 11. Initialization time, evidence generation and verification times, and transmitted data size in bytes under $\lambda = 128$ across varying neural network sizes and data volumes.

key feature of the SAC module: we can enhance privacy and reduce communication volume by lowering the intermediate representation dimension, without significantly compromising performance. The CP bar chart further confirms that reducing the intermediate representation dimension generally does not significantly affect performance. However, it is crucial to note that in the SMAC environment, reducing the intermediate representation dimension leads to a noticeable increase in the Critic's Value loss and a decrease in CP. This intuitively critical phenomenon suggests that for complex environments, more information is required for accurate representation. The results suggest there is a compression threshold—reducing dimensions beyond this point hurts performance, while staying within it improves privacy and communication efficiency without loss of effectiveness.

To scientifically guide the selection of intermediate representation dimensions, we designed supervised regression experiments on synthetic data. Specifically, we generated regression tasks with input dimensions of 64 or 128 and a sample size of 1000. In these tasks, only the first $d \in \{4, 8, 32, 64\}$ dimensions were designated as "task-intrinsic dimensions" (i.e., dimensions that genuinely contribute to the output), while the remaining dimensions were noise. We trained a multi-layer perceptron, varying its intermediate representation dimension from 1 to 64, and recorded the reconstruction error (RE) upon convergence and the parameter utilization rate (defined as the proportion of weights with absolute values greater than 10^{-3}).

As shown in Fig. 10, experiments reveal two main patterns. First, reconstruction error drops sharply as the intermediate representation dimension grows from a very low value. Tasks with higher intrinsic dimension see a slower decline. Second, parameter utilization peaks at a certain dimension and then gradually falls. This peak occurs when dimensions are adequate but not excessive, as extra dimensions mostly fit noise. The turning point often appears near one-quarter of the input dimension, around $n/4$. This matches the heuristic $\lfloor 1 + n/4 \rfloor$ from pFedHN (Shamsian et al., 2021). That formula offers a practical starting point for choosing TDML's intermediate dimension. Further tuning on the validation set can then balance privacy, communication cost, and policy performance.

6.4. Evaluation of VP's computational and communication cost

To assess the computational and communication overhead of the VP security scheme, we measured its initialization time, evidence generation and verification times, and the size of transmitted information in bytes, across varying neural network sizes and data volumes.

The experimental results are shown in Fig. 11. It is noteworthy that the VP evidence generation time is significantly longer than the verification time, with the latter being on the millisecond scale and generally negligible for a single training round. The data volume does not significantly impact VP generation and verification times; this is primarily because the time consumption of the VP scheme is concentrated in compiling the model's computational circuit, which is closely related to the model size but largely independent of the data volume. Regarding communication overhead, the per-round cost is in the kilobyte (KB) range, which is entirely manageable. While the VP scheme's initialization time increases exponentially with the security bit length (security parameter λ), setting $\lambda = 128$ corresponds to 2^{128} brute-force operations, which is sufficient for the vast majority of application scenarios.

Table 5 further presents the Rollout time, training time, and model sizes for each environment, without considering the VP scheme's overhead. As observed, the scale of RL agent models is relatively small

Table 5
Comparison of computational and communication overhead across different algorithms and environments. The "Crypto Overhead" refers to the time spent on cryptographic operations (commitments, signatures, zkDL proofs). "Crypto Ratio" indicates the proportion of total training time consumed by cryptography. "Communication Cost" is the average data transmitted per episode.

Environment (Num Agents)	Algorithms	Rollout (s)	Update (s)	Crypto overhead (s)	Total (s)	Crypto ratio (%)	Communication cost (KB)
Spread (Continuous) (3)	HAPPO	1.87 ± 0.17	0.09 ± 0.05	0	0.96 ± 0.22	0.00%	56.25
	PP-MARL	0.68 ± 0.04	2.14 ± 0.09	0.68 ± 0.00	3.50 ± 0.13	19.43%	281488.67
	TDPPO	1.55 ± 0.19	0.17 ± 0.00	1.67 ± 0.08	3.39 ± 0.27	49.26%	91.89
Spread (Discrete) (3)	HAPPO	0.98 ± 0.61	0.53 ± 0.68	0	1.51 ± 1.29	0.00%	46.88
	PP-MARL	/	/	/	/	/	/
	TDPPO	0.51 ± 0.06	0.08 ± 0.05	1.68 ± 0.08	2.27 ± 0.19	74.00%	91.98
Corridor (6)	HAPPO	3.55 ± 0.07	0.18 ± 0.04	0	3.73 ± 0.11	0.00%	1305.00
	PP-MARL	/	/	/	/	/	/
	TDPPO	3.61 ± 0.07	0.22 ± 0.05	3.04 ± 0.16	6.87 ± 0.28	44.25%	207.67
ShadowHandOver (2)	HAPPO	2.17 ± 0.51	0.49 ± 0.39	0	2.66 ± 0.90	0.00%	135.94
	PP-MARL	1.72 ± 0.04	10.34 ± 3.65	0.18 ± 0.00	12.24 ± 3.69	1.47%	91997.17
	TDPPO	1.52 ± 0.41	0.51 ± 0.39	8.97 ± 0.43	11.00 ± 1.23	81.55%	395.38

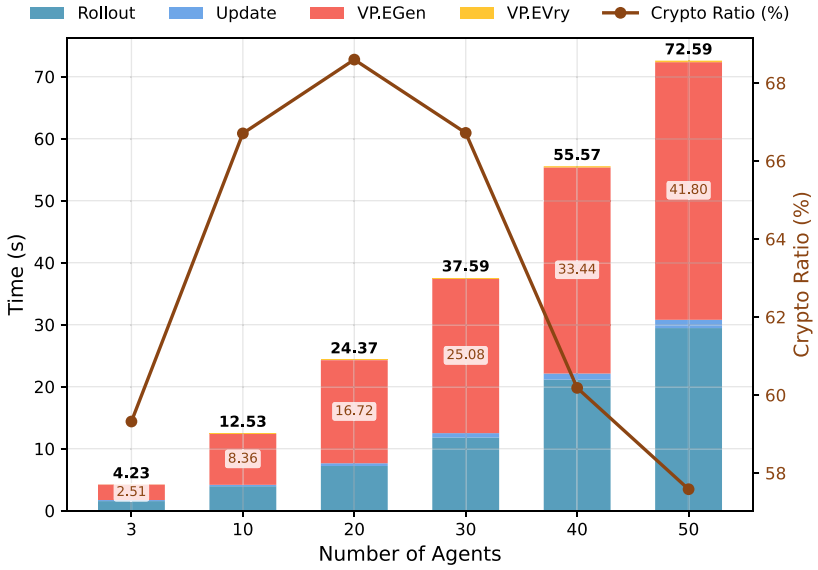

Fig. 12. Scalability analysis of the VP scheme with increasing number of agents. Experiments were conducted in the MPE Spread (Continuous) environment, scaling the number of agents from 3 to 50. The x-axis represents the number of agents. The left y-axis shows time overhead (seconds), and the right y-axis shows the cryptographic overhead ratio (Crypto Ratio, defined as the proportion of VP generation and verification time to total training time). The blue curve represents basic training time (Rollout + Update), the orange curve represents cryptographic overhead (VP.Gen + VP.Vry), and the green curve represents the cryptographic overhead ratio.

in most scenarios. Comparing this with Fig. 11, the VP.Gen time is approximately equivalent to the Rollout time. Overall, the overhead of the VP scheme is within a controllable range for current mainstream MARL applications. The current VP.Gen implementation requires recompiling the circuit for each evidence generation, leading to high time consumption. Future work can significantly reduce computational costs by minimizing the number of nodes requiring secure communication within the algorithm or by optimizing zero-knowledge proof algorithms.

To assess the VP scheme's scalability, we tested overhead as agent count rose from 3 to 50 in the MPE: Spread (Continuous) environment. Results in Fig. 12 show both training time and cryptographic cost grow linearly with agent number, confirming linear scalability. The crypto ratio, however, peaks near 20 agents at about 50 percent and then declines. This means that while absolute crypto cost rises, its relative burden shrinks in larger systems where training dominates. Practically, VP overhead may bottleneck small to medium setups under 20 agents, calling for hardware acceleration. In large-scale deployments like smart grids or vehicular networks, the relative overhead stays manageable, letting TDML remain real-time viable.

7. IoT application

Logistics warehouse management is central to the intelligent upgrade of global supply chains, directly shaping commerce's speed and cost. Within this context, automated order picking tasks are critical bottlenecks for enhancing warehouse throughput (Sodiya et al., 2024). This task can be modeled as a Lifelong Multi-Agent Path Finding (Lifelong MAPF) problem (Skrynnik et al., 2025). It involves continuously planning collision-free paths for robots on a grid, using known layouts and states, to move large volumes of goods efficiently. As an NP-hard problem, even suboptimal solutions are computationally heavy. Traditional centralized heuristics thus struggle with scalability and efficiency in large or dynamic warehouses.

To evaluate TDML's performance in a real-world IoT environment, we selected the digital twin topography of one of the largest warehouses in Bosnia and Herzegovina (Cogo et al., 2020) as our testing platform. As shown in Fig. 13, the warehouse layout is highly complex, simulating a challenging real-world operational environment. In this scenario, we deployed a group of mobile robots (i.e., agents) that continuously and autonomously transport assigned goods from inventory locations to designated packing or unloading stations. In such dynamic,

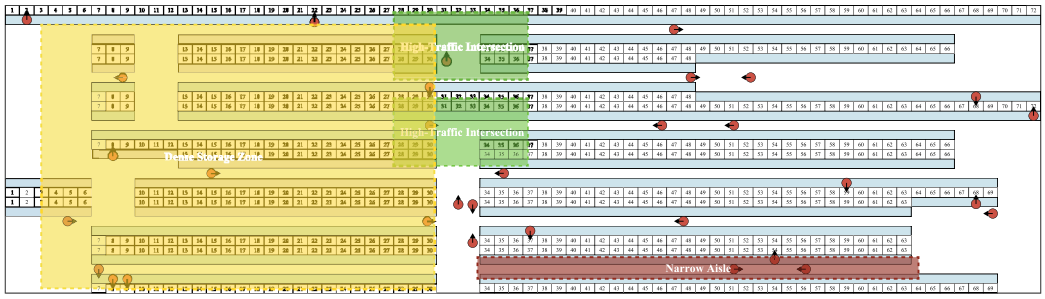

Fig. 13. 2D schematic representation of the warehouse layout. The diagram highlights representative High-Traffic Intersections (green), Narrow Aisles (yellow), and High-Density Storage Areas (maroon), which collectively form a highly challenging real-world operational environment.

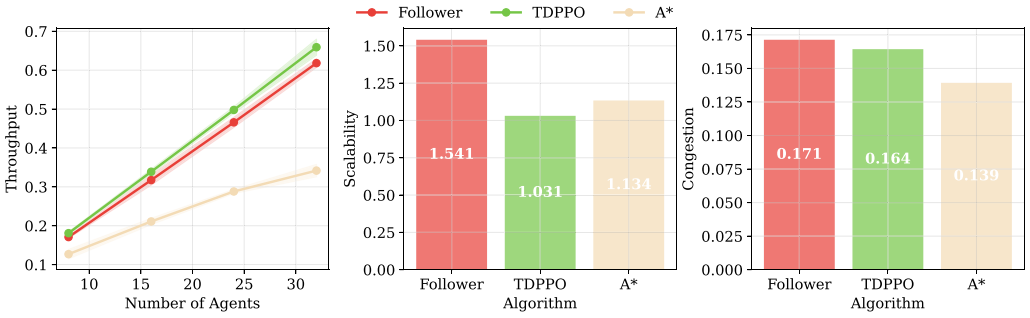

Fig. 14. Comparison results without attack.

high-density IoT environments, the system is highly susceptible to real-world threats, such as data poisoning attacks, where physically compromised sensors or malicious insiders tamper with robot data to cause collisions or deadlocks.

To evaluate performance rigorously, we adopt three core metrics from modern Lifelong MAPF research (Skrynnik et al., 2024). These serve as key performance indicators for our task, similar to critical success factors in other domains (Al-Zwainy and Al-Marsomi, 2023).

Throughput measures system efficiency as the average number of tasks completed per unit time by all agents. It is computed as

$$\text{Throughput} = \frac{\text{Total Goals Completed}}{\text{Episode Length}}. \tag{12}$$

Higher values indicate greater efficiency.

Scalability evaluates how runtime scales with agent count. It is defined as

$$\text{Scalability} = \min\left(1.0, \frac{1}{n}\sum_{i=1}^{n}\frac{\text{Scaled Runtime}_i}{\text{Scaled Runtime}_{i+1}}\right), \tag{13}$$

where $\text{Scaled Runtime} = \frac{\text{Runtime}}{\text{Number of Agents}}$ and n is the logarithm of the compared agent scales. Values closer to 1.0 indicate near-linear scaling behavior.

Congestion captures agent crowding and path conflict levels. It is calculated as

$$\text{Congestion} = \frac{\text{Map Density}}{\text{Average Agent Density}}, \tag{14}$$

with $\text{Map Density} = \frac{\text{Number of Agents}}{\text{Number of Traversable Cells on Map}}$, and average agent density taken from experimental logs. Lower congestion reflects more efficient path planning and reduced waiting due to conflicts.

We deployed the TDPPO algorithm onto each warehouse picking robot to perform distributed path planning tasks, aiming to validate its value in real-world IoT environments. Experiments were conducted with agent scales of 8, 16, 32, and 64 robots to systematically explore performance across low to high density operational states.

We first evaluated TDPPO against the SOTA reinforcement learning-based MAPF method, Follower (Skrynnik et al., 2024), and the classical A* algorithm, under no-attack conditions. As shown in Fig. 14, TDPPO consistently outperforms baseline methods across various agent scales. Thanks to its distributed architecture, TDPPO achieves a scalability score close to 1, indicating nearly constant runtime as the number of agents increases. The low congestion metric further suggests a more uniform agent distribution, resulting in reduced traffic bottlenecks.

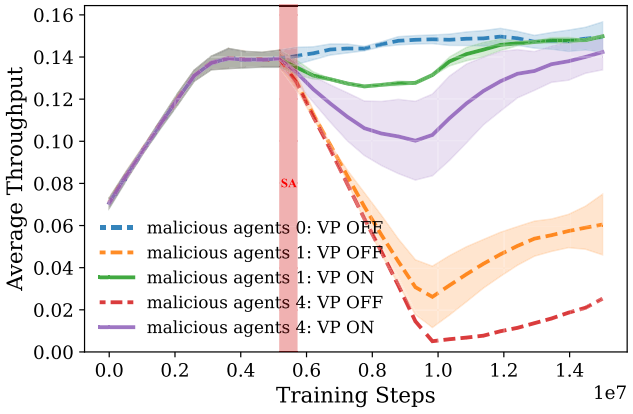

Fig. 15. TDPPO's training performance under various attack scenarios.

Furthermore, we conducted attack experiments in a 32-agent scenario to evaluate the system's robustness under high-intensity threats. At 5e6 training steps, we continuously injected a comprehensive data poisoning attack for 5 episodes: attackers controlled 1, 4, or 8 agents and applied a sign-flipping attack to all their outbound communications, simulating the most extreme malicious behavior. Test results, presented in Fig. 15, show that TDPPO with the VP scheme enabled exhibits exceptional stability. Benefiting from the real-time verification capability of zero-knowledge proofs, the system could 100% identify and isolate malicious messages, regardless of whether the attack originated from a single agent or multiple colluding agents. Subsequently, the system triggered a robust update strategy (as shown in Table 3, by

moving malicious agents to the end of the update sequence), effectively mitigating the attack's impact. While this defense mechanism may incur a slight performance sacrifice (information loss) due to replacing malicious values with the mean of outputs from other legitimate agents, its core value lies in its ability to completely block the propagation of malicious policies, fundamentally preventing catastrophic consequences such as backdoor attacks, thereby ensuring the system's basic usability and security in adversarial environments.

The TDML framework, with distributed learning, verifiable communication, and privacy-preserving computation as its core design principles, is well-suited for large-scale industrial IoT scenarios requiring high security. Its successful deployment in warehouse robot clusters has validated its effectiveness and scalability in high-density dynamic environments. This framework can be readily transferred to smart grid applications, coordinating thousands of distributed energy units by exchanging zero-knowledge verified power prediction information to achieve global supply–demand balance while protecting users' raw electricity consumption data. In vehicular networks, autonomous vehicles can use TDML to share verified abstract road condition information without exposing precise locations or raw sensor data, balancing driving safety and privacy. Smart manufacturing production lines can also benefit from this framework, with different workstation devices coordinating task progress through verifiable communication, preventing trade secret leaks and malicious command injection. In summary, TDML constructs a trustworthy mirror space, offering a general multi-agent collaborative learning method for open, adversarial IoT environments.

8. Discussion

This paper introduces TDML, a novel method that unifies convergence, verifiability, and privacy preservation in MARL. TDML provides a trustworthy training paradigm for autonomous agent collaboration in open IoT environments. Our key contribution lies in the decentralized reformulation of the centralized MARL framework, which enables private and verifiable multi-agent learning through SL and ZKP. We validate the effectiveness of TDML through extensive experiments on multiple MARL benchmark tasks, as well as in a real-world application involving warehouse picking robots in an IoT setting.

TDML inherits the theoretical advantages of the MATRL approach (Liu et al., 2024b) while guaranteeing convergence to a Nash equilibrium. This is a crucial property often lacking in other prominent MARL methods, such as QMIX (Rashid et al., 2020) or MAPPO (Yu et al., 2022) (as summarized in Table 1). Furthermore, the superior performance of TDML's derived algorithms, TDPPO and TDA2C, under various attack scenarios (Fig. 8) directly challenges the vulnerabilities highlighted by Liu and Lai (2023). Their work indicated that MARL algorithms lacking security mechanisms are highly susceptible to poisoning attacks. This enhanced robustness is primarily attributed to our proposed VP scheme, which plays a pivotal role in ensuring the authenticity, integrity, and verifiability of communicated data. Regarding privacy, our SAC method achieves PRE comparable to federated learning approaches like PDPPO (Lu et al., 2025) (Table 4), while also reducing communication overhead. Unlike PDPPO, which is constrained by homogeneous policies, TDML inherently supports heterogeneous policies.

An unexpected but meaningful finding emerged from our ablation studies on SAC. In simpler environments, reducing the intermediate representation dimension enhanced privacy and reduced communication overhead, with minimal impact on performance. However, in the more complex SMAC environment, such reduction led to a significant drop in CP (Fig. 9). This suggests the existence of an "information compression threshold". Below this threshold, further dimension reduction leads to a significant loss of critical information necessary for accurate value function estimation and effective policy learning.

Despite its advancements, this study acknowledges several limitations. First, the current implementation of VP.EGen incurs a computational overhead roughly equivalent to the Rollout time in certain environments (Table 5). Although acceptable in most MARL settings, this may pose challenges for resource-constrained edge devices. Second, the algorithm includes an excessive number of VP communication points, increasing both implementation complexity and runtime. Third, while our robust update strategies effectively dampen attack impacts, they cannot fully neutralize high-intensity coordinated attacks—especially those targeting the Critic network. Moreover, relying on a single advantage agent for global aggregation introduces a single point of failure: if compromised or under denial-of-service, training halts. Crucially, thanks to VP-secured communications, a malicious advantage agent cannot forge evidence or corrupt peer policies; its worst effect is a pause, not systemic collapse. Still, this dependency remains a structural limitation for system availability.

Future work will focus on reducing communication points—not just for efficiency, but to fundamentally limit exposure to adversarial attacks by minimizing opportunities for interference. Four directions will be prioritized: (1) Optimizing zero-knowledge proofs via efficient zk-SNARKs or aggregate proofs to cut evidence generation cost. (2) Trimming unnecessary VP verification steps to boost efficiency and shrink the attack surface. (3) Introducing decentralized auditing with incentive-driven "fraud proof challenges", allowing any node to flag suspicious behavior while preserving regulators' role as privileged challengers—reducing reliance on centralized trust without sacrificing compliance. (4) Eliminating single points of failure by replacing static advantage agents with dynamic rotation or consensus-based election, enabling decentralized fault tolerance for advantage computation while preserving aggregation efficiency and strengthening resilience in adversarial settings.

CRediT authorship contribution statement

Suhang Wei: Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Jinfang Jia:** Writing – review & editing, Methodology, Conceptualization. **Xiang Feng:** Supervision, Project administration, Funding acquisition. **Huiqun Yu:** Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 62276097), Key Program of National Natural Science Foundation of China (No. 62136003), National Key Research and Development Program of China (No. 2020YFB1711700).

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.engappai.2025.112812.

Data availability

All data and source code used in this study are publicly available at: https://github.com/353055619/tdml.

References

Al-Zwainy, F., Al-Marsomi, M., 2023. Structural equation modeling of critical success factors in the programs of development regional. J. Proj. Manag. 8 (2), 119–132.

Badia, A.P., Piot, B., Kapturowski, S., Sprechmann, P., Vitvitskyi, A., Guo, Z.D., Blundell, C., 2020. Agent57: Outperforming the atari human benchmark. In: International Conference on Machine Learning. PMLR, pp. 507–517.

Bellare, M., Canetti, R., Krawczyk, H., 1996. Keying hash functions for message authentication. In: Advances in Cryptology—CRYPTO'96: 16th Annual International Cryptology Conference Santa Barbara, California, USA August 18–22, 1996 Proceedings 16. Springer, pp. 1–15.

Benaissa, A., Retiat, B., Cebere, B., Belfedhal, A.E., 2021. TenSEAL: A library for encrypted tensor operations using homomorphic encryption. In: ICLR 2021 Workshop on Distributed and Private Machine Learning. DPML 2021.

Boneh, D., Lynn, B., Shacham, H., 2004. Short signatures from the weil pairing. J. Cryptology 17, 297–319.

Chen, Y., Geng, Y., Zhong, F., Ji, J., Jiang, J., Lu, Z., Dong, H., Yang, Y., 2023. Bi-dexhands: Towards human-level bimanual dexterous manipulation. IEEE Trans. Pattern Anal. Mach. Intell. 46 (5), 2804–2818.

CHERIF, A.N., YOUSSFI, M., EN-NAIMANI, Z., TADLAOUI, A., SOULAMI, M., BOUATTANE, O., 2024. CQRS and blockchain with zero-knowledge proofs for secure multi-agent decision-making. Int. J. Adv. Comput. Sci. Appl. 15 (11).

Cogo, E., Žunić, E., Beširević, A., Delalić, S., Hodžić, K., 2020. Position based visualization of real world warehouse data in a smart warehouse management system. In: 2020 19th International Symposium INFOTEH-JAHORINA. INFOTEH, IEEE, pp. 1–6.

Fang, P., Chen, J., 2023. On the vulnerability of backdoor defenses for federated learning. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 37, pp. 11800–11808.

Gohari, P., Hale, M., Topcu, U., 2023. Privacy-engineered value decomposition networks for cooperative multi-agent reinforcement learning. In: 2023 62nd IEEE Conference on Decision and Control. CDC, IEEE, pp. 8038–8044.

Hao, M., Shang, C., Wang, S., Jiang, W., Nie, J., 2025. Uav-assisted zero knowledge model proof for generative ai: A multi-agent deep reinforcement learning approach. IEEE Internet Things J..

He, S., Han, S., Su, S., Han, S., Zou, S., Miao, F., 2023. Robust multi-agent reinforcement learning with adversarial state uncertainties. Trans. Mach. Learn. Res. https://openreview.net/forum?id=CqTkapZ6H9, 2023.

Kuba, J., Chen, R., Wen, M., Wen, Y., Sun, F., Wang, J., Yang, Y., 2022. Trust region policy optimisation in multi-agent reinforcement learning. In: ICLR 2022-10th International Conference on Learning Representations. The International Conference on Learning Representations (ICLR), p. 1046.

Li, T., Zhu, K., Luong, N.C., Niyato, D., Wu, Q., Zhang, Y., Chen, B., 2022. Applications of multi-agent reinforcement learning in future internet: A comprehensive survey. IEEE Commun. Surv. Tutor. 24 (2), 1240–1279.

Liu, Z., Garg, N., Ratnarajah, T., 2024a. Multi-agent federated Q-learning algorithms for wireless edge caching. IEEE Trans. Veh. Technol..

Liu, X., Guo, D., Zhang, X., Liu, H., 2024b. Heterogeneous embodied multi-agent collaboration. IEEE Robot. Autom. Lett..

Liu, G., Lai, L., 2023. Efficient adversarial attacks on online multi-agent reinforcement learning. Adv. Neural Inf. Process. Syst. 36, 24401–24433.

Lowe, R., Wu, Y.I., Tamar, A., Harb, J., Abbeel, P., Mordatch, I., 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. Adv. Neural Inf. Process. Syst. 30.

Lu, S., Cai, Y., Liu, Z., Lian, Y., Chen, L., Wang, H., 2025. A preference-based multi-agent federated reinforcement learning algorithm framework for trustworthy interactive urban autonomous driving. IEEE Trans. Intell. Transp. Syst..

Ma, C., Li, J., Wei, K., Liu, B., Ding, M., Yuan, L., Han, Z., Poor, H.V., 2023. Trusted ai in multiagent systems: An overview of privacy and security for distributed learning. Proc. IEEE 111 (9), 1097–1132.

Mo, K., Ye, P., Ren, X., Wang, S., Li, W., Li, J., 2024. Security and privacy issues in deep reinforcement learning: Threats and countermeasures. ACM Comput. Surv. 56 (6), 1–39.

Mukherjee, A., Kumar, P., Yang, B., Chandran, N., Gupta, D., 2023. Privacy preserving multi-agent reinforcement learning in supply chains. arXiv preprint arXiv:2312.05686.

Pedersen, T.P., 1991. Non-interactive and information-theoretic secure verifiable secret sharing. In: Annual International Cryptology Conference. Springer, pp. 129–140.

Rashid, T., Samvelyan, M., De Witt, C.S., Farquhar, G., Foerster, J., Whiteson, S., 2020. Monotonic value function factorisation for deep multi-agent reinforcement learning. J. Mach. Learn. Res. 21 (178), 1–51.

Risan, H.K., Al-Azzawi, A.A., Al-Zwainy, F.M., 2024a. Statistical Analysis for Civil Engineers: Mathematical Theory and Applied Experiment Design. Elsevier.

Risan, H.K., Serhan, F.M., Al-Azzawi, A.A., 2024b. Management of a typical experiment in engineering and science. AIP Conf. Proc. 2864 (1), 050001.

Samvelyan, M., Rashid, T., Schroeder de Witt, C., Farquhar, G., Nardelli, N., Rudner, T.G., Hung, C.-M., Torr, P.H., Foerster, J., Whiteson, S., 2019. The StarCraft multi-agent challenge. In: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems. pp. 2186–2188.

Sasson, E.B., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M., 2014. Zerocash: Decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy. IEEE, pp. 459–474.

Schulman, J., Moritz, P., Levine, S., Jordan, M., Abbeel, P., 2015. High-dimensional continuous control using generalized advantage estimation. arXiv preprint arXiv:1506.02438.

Shamsian, A., Navon, A., Fetaya, E., Chechik, G., 2021. Personalized federated learning using hypernetworks. In: International Conference on Machine Learning. PMLR, pp. 9489–9502.

Skrynnik, A., Andreychuk, A., Borzilov, A., Chernyavskiy, A., Yakovlev, K., Panov, A., 2025. POGEMA: A benchmark platform for cooperative multi-agent pathfinding. In: The Thirteenth International Conference on Learning Representations. pp. 5576–5606.

Skrynnik, A., Andreychuk, A., Nesterova, M., Yakovlev, K., Panov, A., 2024. Learn to follow: Decentralized lifelong multi-agent pathfinding via planning and learning. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 38, pp. 17541–17549.

Sodiya, E.O., Umoga, U.J., Amoo, O.O., Atadoga, A., 2024. AI-driven warehouse automation: A comprehensive review of systems. GSC Adv. Res. Rev. 18 (2), 272–282.

Standen, M., Kim, J., Szabo, C., 2025. Adversarial machine learning attacks and defences in multi-agent reinforcement learning. ACM Comput. Surv. 57 (5), 1–35.

Sun, H., Bai, T., Li, J., Zhang, H., 2024. Zkdl: Efficient zero-knowledge proofs of deep learning training. IEEE Trans. Inf. Forensics Secur..

Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W.M., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J.Z., Tuyls, K., et al., 2018. Value-decomposition networks for cooperative multi-agent learning based on team reward. In: Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems. pp. 2085–2087.

Xu, H., Xuan, J., Zhang, G., Lu, J., 2025. Twin Trust Region policy optimization. IEEE Trans. Syst. Man Cybern.: Syst..

Yang, Y., Wen, Y., Wang, J., Chen, L., Shao, K., Mguni, D., Zhang, W., 2020. Multi-agent determinantal q-learning. In: International Conference on Machine Learning. PMLR, pp. 10757–10766.

Yu, L., Qiu, Y., Yao, Q., Shen, Y., Zhang, X., Wang, J., 2024. Robust communicative multi-agent reinforcement learning with active defense. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 38, pp. 17575–17582.

Yu, C., Velu, A., Vinitsky, E., Gao, J., Wang, Y., Bayen, A., Wu, Y., 2022. The surprising effectiveness of ppo in cooperative multi-agent games. Adv. Neural Inf. Process. Syst. 35, 24611–24624.

Yuan, T., Chung, H.-M., Fu, X., 2024. PP-MARL: Efficient privacy-preserving multi-agent reinforcement learning for cooperative intelligence in communications. IEEE Netw. 38 (5), 196–203.

Zhong, Y., Kuba, J.G., Feng, X., Hu, S., Ji, J., Yang, Y., 2024. Heterogeneous-agent reinforcement learning. J. Mach. Learn. Res. 25 (32), 1–67.

Zhou, Z., Liu, G., Tang, Y., 2024. Multiagent reinforcement learning: Methods, trustworthiness, applications in intelligent vehicles, and challenges. IEEE Trans. Intell. Veh..