UAV-Assisted Zero Knowledge Model Proof for Generative AI: A Multiagent Deep Reinforcement Learning Approach

Min Hao, Chen Shang, Siming Wang, Wenchao Jiang, *Member, IEEE*, and Jiangtian Nie

***Abstract*—As more users seek generative AI (GAI) models to enhance work efficiency, GAI and Model-as-a-Service will drive transformative changes and upgrades across all industries. However, when users utilize GAI models provided by the service provider, they cannot be certain that the model's quality matches the provider's claims. Considering the need to protect intellectual property, the service provider will not disclose model details for user verification. To this end, we take the Internet of Vehicles as research background, proposing a zero knowledge model proof architecture based on UAVs. We also introduce a multiagent reinforcement learning algorithm to optimize the verification process. In specific, we first propose a verification scheme for the key operations of generative adversarial networks based on noninteractive zero knowledge proof. The zero knowledge proof architecture ensures that model parameters cannot be stolen during the verification process. After that, we propose an Age of Verification (AoV) metric to ensure the timeliness and freshness of zero knowledge proof. We also construct a tradeoff optimization problem between the energy consumption of UAV as a verifier and the AoV of edge servers as service providers, and transform the problem based on Lyapunov optimization theory. Following that, we propose an enhanced multiagent proximal policy optimization algorithm to enable the collaborative verification of edge servers by multiple UAVs. The algorithm simulation results demonstrate that the reward value of our proposed algorithm is over 10% higher than that of the standard algorithm, with a faster and more stable overall convergence speed. Additionally, the zero knowledge proof performance test results indicate that the verification delay in our proposed architecture is less than 500 ms during the verification phase, meeting practical requirements.**

***Index Terms*—Generative AI (GAI), Lyapunov optimization, multiagent deep reinforcement learning, uncrewed aerial vehicle, zero knowledge proof.**

I. Introduction

With the evolution of 6G communication technology and advancements in big data, traditional artificial intelligence is transitioning toward generative AI (GAI) and broader generalization [1], [2], [3]. The large language generation model, ChatGPT, launched by OpenAI, has gained significant popularity [1], [4], [5]. Its latest version, ChatGPT-O1, exhibits human-like reasoning abilities and has demonstrated impressive performance across various tests. Behind these GAIs are neural network models trained on vast amounts of data and computational power, such as generative adversarial network (GAN), Diffusion, and Transformer models [6], [7]. However, these GAI models require a significant amount of advanced and expensive hardware for both training and inference, such as computing clusters composed of hundreds or thousands of NVIDIA A100 or H100 GPUs. This significantly raises the barrier for ordinary users to train and utilize GAI models. To meet the demands of large number of users, Model-as-a-Service (MaaS) platforms and service have emerged [8], [9]. Users can directly access pretrained GAI models through APIs or cloud platforms [10]. MaaS typically follows a pay-as-you-go pricing model, allowing users to pay based on their usage, thereby avoiding the high costs associated with computing resources and model development. Additionally, users can fine-tune general GAI models to better suit specific vertical domains, achieving more targeted and satisfactory results [11]. Although MaaS shows its significant advantages, the data security and privacy problems still remain. Ensuring the effective and secure of GAI models across various domains remains a critical and urgent area of research.

Taking MaaS in the field of smart transportation as an example, vehicles and other transportation equipment often lack sufficient computing power, requiring some AI generation tasks to be offloaded to roadside edge servers. The edge server is equipped with several neural network models pretrained by the model service provider [12]. Edge servers can use their powerful computing capabilities to provide personalized services to users. The advantage of this edge computing approach is that it reduces latency caused by communication processes [13]. However, it also introduces certain challenges. First, users are required to pay edge servers based on the complexity of the tasks, but they cannot verify whether the services provided are truly commensurate with the amount they have paid. For the edge servers, it is able to use models with fewer parameters to provide services for users, which reduces their energy consumption but decreases Quality of Experience (QoE) of users [14]. In addition, if users request the edge server to display the model parameters under the

Received 29 October 2024; revised 21 December 2024; accepted 4 January 2025. Date of publication 30 January 2025; date of current version 9 May 2025. This work was supported in part by the National Research Foundation, Singapore, and in part by the Infocomm Media Development Authority under its Future Communications Research and Development Programme. *(Corresponding author: Wenchao Jiang.)*

Min Hao, Chen Shang, Siming Wang, and Wenchao Jiang are with the Pillar of Information Systems Technology and Design, Singapore University of Technology and Design, Singapore 487372 (e-mail: minhao607@163.com; chenshang0420@gmail.com; simingwang30@163.com; wenchao_jiang@sutd.edu.sg).

Jiangtian Nie is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 638788 (e-mail: jnie001@e.ntu.edu.sg).

Digital Object Identifier 10.1109/JIOT.2025.3531914

pretext of verifying service quality, this creates a risk of the model being stolen. For service providers, these model parameters are valuable intellectual property and cannot be disclosed to the public freely. Moreover, the verification of services provided by edge servers is time-sensitive. This is especially important in the field of transportation, where users need timely verification of the models used by edge servers. If the quality of services from edge servers cannot be ensured promptly, it could adversely affect traffic safety. Lastly, while ensuring the timeliness of verification and protecting the data security of service providers, the energy consumption associated with verification can also impact users. Excessive energy consumption during model verification can negatively affect the user experience. For example, if electric vehicles use too much battery power for verification, their driving range will be significantly reduced [15].

To address the issues outlined above, we introduce a third-party verification device, the UAV, and propose a verification architecture based on zero knowledge proof combined with a verification optimization strategy leveraging a multiagent reinforcement learning algorithm. This approach aims to ensure secure, efficient, and timely model verification without compromising data security or user experience. In smart transportation scenarios, UAVs can quickly reach designated locations and flexibly perform verification. During communication, UAVs can stay close to users, thereby reducing the risk of communication interruptions. With their flexible and lightweight features, UAVs have been deployed in scenarios such as power grid inspection, cargo transportation, and emergency rescue operations [16], [17], [18], [19]. We selected UAVs as third-party verification devices to alleviate the computing burden on users and improve the timeliness of verification. This study focuses on the verification of the GAN model. The verification principle for the GAN model proposed in this study can be extended and applied to other GAI models as well.

We first employ noninteractive zero knowledge proof to address the issues of model verification and parameter protection. We convert the three most critical operations in the GAN model—transposed convolution, batch normalization (BatchNorm), and ReLU—into matrix operations, enabling the generation of verifiable zero knowledge evidence. During the verification of this evidence, the UAV can only confirm whether the operations were executed correctly, without accessing any additional model-related information, thereby preventing the leakage of model parameters. In the second place, we propose Age of Verification (AoV) to ensure the timeliness of verification. If the UAV verifies the edge server's GAI model every time, the verification efficiency would be extremely low. A more reasonable and practical approach is to set an AoV indicator for the edge server. When the AoV of the edge server exceeds a certain threshold, the UAV should prioritize verifying that device to ensure the edge server remains in an overall trustworthy state. In addition, we propose an enhanced multiagent proximal policy optimization (MAPPO) algorithm to balance the AoV of edge servers with the energy consumption of UAVs, ensuring efficient and timely verification while minimizing resource usage. The multiagent reinforcement learning algorithms enable these UAVs to cooperate more effectively, enhancing the overall verification process. The simulation results demonstrate that the enhanced MAPPO algorithm converges more quickly. In terms of reward comparison, the proposed algorithm outperforms conventional MAPPO by 10%. Furthermore, the performance test of zero knowledge proof also reflects the practicality of the proposed verification architecture.

The main contributions of this article are as follows.

1) We developed a zero knowledge proof architecture for the GAN model. Within this architecture, the UAV can verify critical operations of the GAN model without exposing the model's parameters.
2) We propose AoV as a new metric to ensure the timeliness of verification. Additionally, to balance AoV and energy consumption, we formulate an optimization problem based on Lyapunov theory.
3) We propose an enhanced multiagent reinforcement learning algorithm to solve the above optimization problem, enabling multiple UAVs to collaboratively verify the edge server.

The remainder of this article is organized as follows. Section II introduces the related work of AI model security. Section III presents an overview of the overall network architecture and the proposed zero knowledge proof architecture. Section IV outlines the process of establishing and analyzing the system model and the associated optimization problem. Section V introduces the transformation and algorithmic steps of optimization problems. The simulation setup and result analysis are presented in Section VI, while Section VII provides a summary of this entire article, highlighting key findings and conclusions.

II. Related Work

This section reviews existing work related to AI model security and data protection, highlighting the distinctive features of this study.

A. Applications of Zero Knowledge Proof

Zero knowledge proof have found numerous applications, including identity authentication, privacy protection, blockchain smart contracts, and verifying computational integrity. Fan et al. [20] and Lee et al. [21] applied zero knowledge proof to the integrity verification of convolutional neural network (CNN) models and conducted tests on CNN models of varying sizes. Ganescu and Passerat-Palmbach [22] applied zero knowledge proof to the NanoGPT model and examined the time required for proof generation. Wellington [23] employed zero knowledge proof to encrypt the content of a user's conversation with a large language model, allowing the user to receive computation results without revealing the plaintext. Lin et al. [24] utilized blockchain and zero knowledge proof to differentiate the semantic similarity between adversarial and real semantic data, verifying the authenticity of semantic data transformations. Tan et al. [25] proposed an efficient UAV certificate less group authentication mechanism to address the access authentication problem for UAV.

This scheme eliminates the need for edge IoT infrastructure, simplifying the authentication process while ensuring secure UAV access. Wang et al. [26] solves the problem of efficient, fair and privacy-preserving computation offloading in UAV applications through zero knowledge proof. Simulation results verify the effectiveness of the proposed algorithm in terms of offloading efficiency, cost savings and system overhead. However, the aforementioned research did not propose a verification architecture specifically for GAI models. Additionally, none of these studies addressed the optimization of verification efficiency and energy consumption, which are critical for practical deployment in resource-constrained environments.

B. AI Model Security Research

Zheng et al. [27] reviewed the trustworthiness issues in AI, focusing on advancements in intellectual property protection, privacy-preserving, federated learning, security verification, and AI security calibration. This comprehensive review highlights current approaches and challenges in ensuring the reliability and integrity of AI systems. Wang et al. [28] proposed a comprehensive credibility assessment scheme for large language models, designed primarily to identify effectively whether the output of these models violates relevant laws and regulations. To protect the privacy of GAI model users, Zhang et al. [29] proposed a customized privacy protection framework combined with a Homomorphic encryption scheme. This approach ensures the security of both user inputs and computation, providing a robust solution for privacy-preserving interactions with GAI models. Zhao et al. [30] proposed a text protection mechanism for large language models, enabling the model to decline generating responses when it receives protected text. This mechanism prevents the misuse of sensitive source text, reducing the risk of malicious exploitation. However, research on the trustworthiness and security of GAI models remains insufficient. When utilizing GAI models, it is essential not only to safeguard the privacy and data security of both users and the models but also to enhance the efficiency of security verification tailored to specific usage scenarios.

III. Scenario and Zero Knowledge Model Proof

This section begins with an overview of the network architecture, followed by a detailed explanation of the zero knowledge proof scheme applied to the GAN model, which serves as a representative example of GAI models.

A. Scenario Overview

Fig. 1 illustrates the network architecture, which primarily consists of cloud server, edge servers, and UAVs [31]. In the context of urban applications, we assume a base station is connected to a cloud server within a city area. Additionally, each road side unit (RSU) along the urban roads is connected to an edge server. Given the limited computing capabilities of end-user devices, when a GAI task is required, the user must offload the task to a nearby roadside edge server. The edge server leverages its stored model to deliver the appropriate content generation services to the users. The model deployed

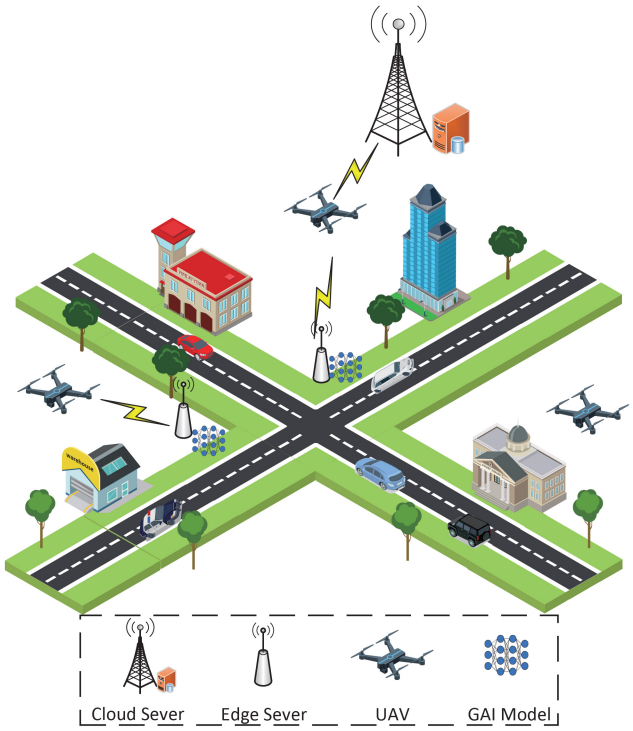

Fig. 1. Network architecture.

on the edge server is first trained by the cloud server and then prestored on the edge server, tailored to the specific needs of the region. However, users cannot verify whether the GAI model employed by the edge server meets their specific requirements. The edge server may opt for a simpler model to conserve its energy, but this compromises the quality of the generated content. To resolve the above challenges, we introduce UAVs to verify the model utilized by the edge server.

Assume there are M edge servers within the coverage area of a base station, forming a set $\mathcal{M} = \{1, 2, \ldots, m, \ldots, M\}$. The cloud server deploys multiple UAVs to verify the integrity of the model used by the edge server, ensuring that the edge server does not tamper with or simplify the model. The number of UAVs is N, forming a set $\mathcal{N} = \{1, 2, \ldots, n, \ldots, N\}$. The operation cycle of the verification system is divided into T equal time slots, with each time slot having a duration of τ. Each time slot is indexed by t and $t \in \mathcal{T} = \{0, 1, 2, \ldots, T\}$. At time slot t, the coordinates of UAV n are represented as $l_n(t) = \{x_n(t), y_n(t), H\}$, where H denotes the UAV's fixed flight altitude, which remains constant throughout this study. The coordinates of edge server m are denoted as $l_m = \{x_m, y_m, h\}$. Simultaneously, the coordinates of the cloud server are represented as $l_0 = \{x_0, y_0, h\}$. In comparison to the UAV, the height of both the edge server and the cloud server is negligible, so h is set to zero. At time slot t, the indicator binary variable $\varphi_{n,m}(t) \in \{0, 1\}$ is used to represent whether UAV n is verifying the model of edge server m. When $\varphi_{n,m}(t) = 1$, it indicates that at time slot t, the model of edge server m is verified by UAV n. Conversely, when $\varphi_{n,m}(t) = 0$, UAV n does not perform verification on the model of edge server m. In each time slot t, a UAV can verify at most one edge server, and each edge server can be verified by only

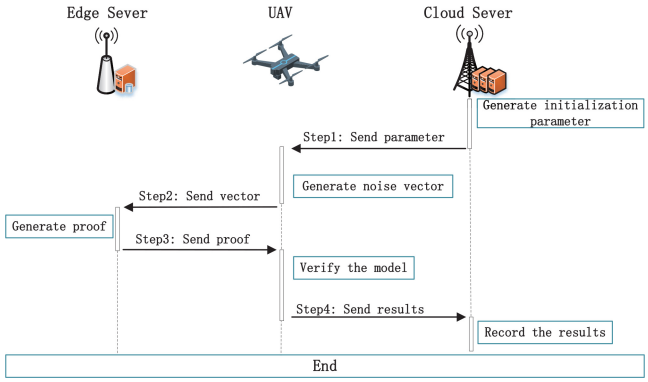

Fig. 2. Zero knowledge model proof process.

one UAV. The verification relationship between UAV and edge server can be expressed as

$$\sum_{m=1}^{M} \varphi_{n,m}(t) \leq 1 \quad \forall t \in \mathcal{T},\ n \in \mathcal{N} \tag{1}$$

$$\sum_{n=1}^{N} \varphi_{n,m}(t) \leq 1 \quad \forall t \in \mathcal{T}, m \in \mathcal{M}. \tag{2}$$

B. Zero Knowledge Proof

Zero knowledge proof primarily fall into two categories: 1) interactive and 2) noninteractive. Interactive proofs require multiple rounds of communication between the prover and verifier, while noninteractive proofs allow the prover to generate a single proof that the verifier can check independently. If the UAV uses an interactive zero knowledge proof with the edge server, it will need to hover for an extended period, leading to increased energy consumption. This prolonged hovering is due to the multiple communication rounds required in the interactive process [32].

Zero knowledge succinct noninteractive argument of knowledge (zk-SNARK) is a noninteractive zero knowledge proof protocol, and Groth-16 is the classic algorithm of this protocol. The Groth-16 implements efficient zero knowledge proof through elliptic curve and bi-linear peer-to-peer cryptography technology. The generated proof is small and the verification speed is fast, which is suitable for UAV verification scenarios.

1) Groth-16: The Groth-16 consists of three stages: 1) setup; 2) proof generation; and 3) verification [33]. In the setup stage, the system generates the necessary public parameters. During proof generation, the prover creates a proof using these parameters. Finally, in the verification stage, the verifier checks the proof's validity without learning any additional information. Corresponding to the scenario focused on in this study, the setup is completed by the cloud server, the prover is the edge server, and the verifier is the UAV. The entire verification process is shown in Fig. 2. In the setup phase, the cloud server extracts the matrix computation logic from the structure of the GAI model and converts it into an algebraic equation using Rank-1 constraint system (R1CS). Simultaneously, the cloud server generates two key parameters: 1) the evaluation key *EK* and 2) the verification key *VK*, based on the computation logic. These parameters are essential for the subsequent proof generation and verification stages. A constraint in each R1CS can be expressed as

$$\left(\sum_{i=1}^{n} A_i z_i\right) \times \left(\sum_{i=1}^{n} B_i z_i\right) = \left(\sum_{i=1}^{n} C_i z_i\right) \tag{3}$$

where A_i, B_i, and C_i are predefined coefficients. z_i is the variable. In a zero knowledge proof, the prover demonstrates that they possess a set of values z that satisfy all the R1CS constraints. By doing so, even complex computations can be transformed into a series of R1CS constraints, enabling the use of zero knowledge proof protocols like zk-SNARKs to verify the correctness of the computation without revealing the underlying values.

In the proof generation stage, the edge server represents the computational logic as a circuit and combines the public primary input *sta*, the witness of the circuit *w*, and the EK to generate the proof π

$$\pi \leftarrow \text{Prove}(EK, w, \text{sta}) \tag{4}$$

where proof π serves as evidence that the computation has been correctly performed without revealing the secret inputs.

In the verification stage, the UAV receives the proof π from the edge server and verifies it using the verification key *VK*

$$\text{True/False} \leftarrow \text{Verify}(VK, \pi, \text{sta}). \tag{5}$$

Specifically, the UAV's verification process includes confirming whether the edge server is using the parameters provided by the cloud server, checking the consistency of the edge server's R1CS variables, and verifying whether the edge server has correctly executed the corresponding GAI model.

2) Characteristics of Zero Knowledge Proof: Zero knowledge proof has several key characteristics that make it a powerful cryptographic tool. The first characteristic is completeness. If the statement being proved is indeed true and the prover provides the required proof correctly, the verifier will always accept the proof

$$\Pr\left[\text{Verify}(VK, \pi, \text{sta}) = \text{True}\right] = 1. \tag{6}$$

The second characteristic is soundness. If the statement being proved is false, no matter what kind of proof or evidence is provided, it will not pass the verifier's checks

$$\Pr\left[\text{Verify}\left(VK, \pi^*, \text{sta}\right) = \text{True}\right] = 0. \tag{7}$$

The final characteristic is zero knowledge. Regardless of whether the statement being proved is true or false, the verifier gains no additional information from the proof other than the fact that the statement is valid (or invalid).

These three characteristics—completeness, soundness, and zero knowledge ensure that the UAV can obtain accurate verification results when validating the edge server's model. At the same time, the model parameters remain secure and are not exposed during the verification process.

3) Zero Knowledge Proof for GAN: This study focuses on performing zero knowledge proof on the GAN model, but the proposed method can also be applied to other GAI models, such as VAEs and transformers, ensuring broad applicability across various GAI frameworks.

A GAN consists of two components: 1) a generator and 2) a discriminator. The generator's role is to create new data, while the discriminator's job is to differentiate between the generated data and real data. These two components engage in a competitive process, where the generator improves its ability to produce realistic data, and over time, it learns to generate high-quality fake data that the discriminator can no longer easily distinguish from real data. When training a GAN, the cloud server optimizes both the generator and the discriminator. Once trained, the optimized generator is sent to the edge server. During inference, the edge server uses the generator to create content for the user. Therefore, the UAV only needs to perform zero knowledge proof on the generator, as it is the key component used for generating content.

This work focuses on using conditional GAN (cGAN) [34], [35] as the subject for zero knowledge proof, as cGAN serves as a foundational model for other GAN variants. By proving cGAN, the method can potentially extend to other GAN-based models. The objective of the cGAN can be expressed as

$$L_{\text{cGAN}}(G, D) = \mathbb{E}_{x,y}\big[\log D(x, y)\big] + \mathbb{E}_{x,z}\big[\log(1 - D(x, G(x, z)))\big] \tag{8}$$

where x and z represent the image input and noise input, respectively, y represents the output result. G and D represent the generator and discriminator of cGAN, respectively. As noted in [34], the GAN model differs from a standard CNN. Its primary operations include transposed convolution, BatchNorm, and ReLU activation functions. Notably, GAN do not contain fully connected layers, which is a key distinction from traditional CNN architectures

When performing zero knowledge proof on the transposed convolution operation, the aim is to represent the up-sampling process using matrix multiplication, similar to how convolution operations are handled in CNN [20]. Let $\boldsymbol{A}$ represent the feature matrix extracted by the convolutional layer. The element in row i and column j of matrix $\boldsymbol{A}$ can be calculated by the following polynomial:

$$a_{ij} = x_{i,1}w_{1,j} + x_{i,2}w_{2,j} + \cdots + x_{i,\cdot}w_{\cdot,j} + b_i \tag{9}$$

where each multiplication of input $x_{i,\cdot}$ and weight $w_{\cdot,j}$, along with the final addition of bias b_i, can be extracted using the R1CS and converted into a zero knowledge proof. For the transposed convolution operation, depending on the settings of different parameters such as stride and padding, there are two ways to calculate each element a_{ij} of the output feature matrix $\boldsymbol{A}$

$$a_{ij} = \begin{cases} w \cdot x + b, & \text{none - overlapping} \\ \boldsymbol{WX} + b, & \text{overlapping} \end{cases} \tag{10}$$

when the positions corresponding to the transposed convolution kernels do not overlap, the output elements can be directly calculated by the dot product of the kernel weights w and the input value x. However, when the positions corresponding to the transposed convolution kernels overlap, the process becomes more complex. In this case, the weights and inputs can be combined into matrices $\boldsymbol{W}$ and $\boldsymbol{X}$, and the output elements can be computed through matrix multiplication. Regardless of whether the transposed convolution kernels overlap or not, the transposed convolution operation can be transformed into a series of polynomial multiplication or addition operations. These polynomial expressions capture the relationships between input values, kernel weights, and output elements. By converting the transposed convolution into these mathematical operations, the entire process can be encoded into a zero knowledge proof system.

In the GAN model, after the up-sampling operation via transposed convolution, the output is passed through BatchNorm. BatchNorm helps stabilize training by normalizing the output of each layer, ensuring that the distribution of activations remains consistent. To perform BatchNorm on the output of the transposed convolution, the first step is to calculate the mean

$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{ij} \tag{11}$$

where N represents the size of the mini-batch data, and x_{ij} denotes the value of the ith sample in the mini-batch on the jth feature dimension. Next, the variance for each feature dimension j is calculated as follows:

$$\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N} \big(x_{ij} - \mu_j\big)^2. \tag{12}$$

After obtaining the mean μ_j and variance σ_j^2, the normalization for each feature dimension j is applied using the following formula:

$$\hat{x}_{ij} = \frac{x_{ij} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}. \tag{13}$$

This normalization ensures that each feature dimension has a mean of 0 and a variance of 1 across the mini-batch. The key to BatchNorm is not only to normalize the data but also to introduce two learnable parameters: 1) the scaling parameter γ_j and 2) the offset parameter β_j. These parameters allow the model to adjust the normalized output, providing flexibility in the representation. After normalization, the output is transformed as follows:

$$y_{ij} = \gamma_j \hat{x}_{ij} + \beta_j. \tag{14}$$

When applying zero knowledge proof to BatchNorm, the primary objective is to verify whether the edge server has tampered with the parameters trained by the cloud server. Therefore, (14) is treated as a polynomial that will be verified by the UAV.

The most commonly used activation function in the GAN model is ReLU, and it is essential to verify the correct application of the ReLU operation during the zero knowledge proof process. The ReLU function is defined as

$$\text{ReLU}(x) = \max\ (0, x). \tag{15}$$

To convert the ReLU operation into a polynomial calculation, we first decompose the input matrix into a vector $\vec{x}$, and then construct a vector $\vec{o}$ of the same dimension as $\vec{x}$. The vector $\vec{o}$ contains only two possible elements: 0 and 1. Specifically,

$o_i = 1$, if the corresponding element in $\vec{x}$ is greater than 0. $o_i = 0$, if the corresponding element in $\vec{x}$ is less than or equal to 0. This binary vector $\vec{o}$ serves as an indicator for whether each element in $\vec{x}$ passes through the ReLU function unchanged or is set to 0. Thus, the ReLU operation can be expressed as

$$y_i = o_i \cdot x_i. \tag{16}$$

This binary representation enables the ReLU operation to be expressed in a form that can be easily verified in zero knowledge proof.

Zero knowledge proof systems are designed to prove the relationships between integer polynomials. However, in GAN models, the parameters are typically in floating-point format. To address this, this study draws inspiration from zero knowledge proof applied to CNN [20] and converts the original floating-point matrix into a fractional matrix for processing. In summary, this study primarily focuses on verifying the transposed convolution, BatchNorm, and ReLU operations of the GAN model, as these are the most critical modules within GAN. If these modules successfully pass the verification by the UAV, it confirms that the edge server has not tampered with the model. Additionally, the verification process ensures that the UAV cannot steal or gain access to the model parameters, maintaining both the integrity and confidentiality of the model during verification.

IV. System Model and Problem Formulation

This section begins by proposing a metric to quantify the degree of verification for edge servers. It then outlines the communication and computation model of the system, followed by a refinement of the mathematical model for the optimization problem.

A. Age of Verification

Given the limited number of UAVs, it is not feasible to verify the models of all edge servers during every time slot. Additionally, performing zero knowledge verification on each layer of the model would reduce verification efficiency and significantly increase the UAV's power consumption. To quantify the extent of edge server verification and enhance verification efficiency, we propose AoV indicator inspired by the concept of Age of Information (AoI). AoI is a metric used to measure the freshness of information, primarily employed to assess the real-time timeliness of information updates within a system. Similar to AoI, AoV is used to assess the timeliness of edge server verification by UAVs. The formula is given as

$$A_m(t+1) = \begin{cases} e^{-bc_m(t)} + a, & \varphi_{n,m}(t) = 1 \\ A_m(t) + 1, & \varphi_{n,m}(t) = 0 \end{cases} \tag{17}$$

where $c_m(t)$ represents the number of layers of the GAI model stored in edge server m that are verified by the UAV during time slot t. Constants a and b are used to adjust the rate of change in the AoV.

In the initial state, the AoV of the edge server is set to 0. Using a negative exponential function, the number of verification layers is inversely correlated with the AoV of the edge server. This implies that as the number of verification layers increases, the rate of AoV increment decreases, indicating an improvement in the timeliness of the verification process.

B. Communication Model

Throughout the zero knowledge proof process, the UAV communicates with both edge server and cloud server. In urban environments, obstructions, such as tall buildings and trees, result in both Line-of-Sight (LoS) and Non-LoS (NLoS) communication between UAVs and ground equipment. Therefore, the UAV communication model must account for both LoS and NLoS scenarios to accurately reflect real-world conditions.

At time slot t, the probability of a UAV communicating with edge server m under LoS conditions is expressed as follows:

$$\Lambda_{\mathrm{LoS}}(t) = \frac{1}{1 + \alpha \exp\left(-\beta\left(\theta_{n,m}(t) - \alpha\right)\right)} \tag{18}$$

where α and β are environment-dependent constants, which depend on the carrier frequency and the type of environment. The elevation angle, $\theta_{n,m}(t)$, between UAV n and edge server m can be calculated

$$\theta_{n,m}(t) = \frac{180}{\pi} \arcsin\left(\frac{H}{d_{n,m}(t)}\right) \tag{19}$$

where $d_{n,m}(t) = \sqrt{\|l_n(t) - l_m\|_2^2}$ is the distance between UAV and edge sever. Since there are only two LoS conditions for communication, the probability of NLoS communication is given as

$$\Lambda_{\mathrm{NLoS}}(t) = 1 - \Lambda_{\mathrm{LoS}}(t). \tag{20}$$

After determining the LoS conditional probability, the channel power gain between the UAV n and the edge server m can be expressed as

$$h_{n,m}(t) = \left(\Lambda_{\mathrm{LoS}}(t) + \mu \Lambda_{\mathrm{NLoS}}(t)\right) \gamma_0 d_{n,m}(t)^{-\tilde{\alpha}} \tag{21}$$

where μ represents the additional attenuation coefficient for the NLoS link. $\tilde{\alpha}$ is the path fading parameter. γ_0 is the channel gain per meter

$$\gamma_0 = \left(\frac{4\pi f_c}{c}\right)^{-2} \tag{22}$$

where f_c and c represent the carrier frequency and the speed of light, respectively. At time slot t, the data transmission rate between UAV n and edge server m can be expressed

$$R_{n,m}(t) = W \log_2\left(1 + \frac{P_n \left|h_{n,m}(t)\right|^2}{{\sigma_n}^2}\right) \tag{23}$$

where W is the wireless communication bandwidth and P_n represents the transmit power of UAV n. ${\sigma_n}^2$ is the UAV channel noise power. The delay for UAV n to send noise vectors or original images to the edge server m can be calculated as

$$\psi_{n,m}(t) = \frac{D_g}{R_{n,m}(t)} \tag{24}$$

where D_g represents the size of noise vectors or original image. The energy consumption for transmitting data from the UAV

to the edge server at time slot t is $e^g_{n,m}(t) = P_n\psi_{n,m}(t)$. $e^g_{n,m}(t)$ is the energy consumed in step 2 of Fig. 2.

Once the UAV completes the verification of the edge server, it must transmit the verification results along with the necessary verification records to the cloud server. This step ensures that the cloud server receives updated and accurate information regarding the integrity and status of the edge server's model. At time slot t, the elevation angle between the UAV and the cloud server is given as

$$\theta_{n,0}(t) = \frac{180}{\pi}\arcsin\left(\frac{H}{d_{n,0}(t)}\right). \tag{25}$$

Therefore, the LoS probability for communication between the UAV and the cloud server can be expressed as

$$\Gamma_{\text{LoS}}(t) = \frac{1}{1+\alpha\exp\left(-\beta\left(\theta_{n,0}(t)-\alpha\right)\right)} \tag{26}$$

and the probability of NLoS is $\Gamma_{\text{NLoS}}(t) = 1-\Gamma_{\text{LoS}}(t)$. Similar to (23), the data transmission rate between the UAV and the cloud server is

$$R_{n,0}(t) = W\log_2\left(1+\frac{P_n\left|h_{n,0}(t)\right|^2}{{\sigma_n}^2}\right) \tag{27}$$

where $h_{n,0}(t) = (\Gamma_{\text{LoS}}(t)+\mu\Gamma_{\text{NLoS}}(t))\gamma_0 d_{n,0}(t)^{-\tilde{\alpha}}$ represents the channel power gain between the UAV and cloud server. The wireless communication delay of UAV n transmitting the verification results to the cloud server is expressed as

$$\psi_{n,0}(t) = \frac{D_r}{R_{n,0}(t)} \tag{28}$$

where D_r is the size of verification results. Therefore, the energy consumption of UAV transmitting data to the cloud server is $e^r_{n,0}(t) = P_n\psi_{n,0}(t)$, which is the energy consumption of step 4 in Fig. 2.

C. Computation Model

At time slot t, after receiving the zero knowledge proof from the edge server, the UAV must verify the proof, which requires consuming energy to perform the necessary computational tasks.

Using the GAN model as a representative of GAI, the average data size of the zero knowledge proof for each layer in the GAN generator is denoted as D_{gan}. The computational workload of the UAV n for verifying the zero knowledge proof can be expressed as

$$L_n(t) = N_{\text{cpu}}D_{\text{gan}}c_m(t) \tag{29}$$

where N_{cpu} represents the number of integer operations required to verify a unit of data, as zero knowledge proof primarily involve integer-based computations. Additionally, B_{cpu} represents the number of CPU cycles per second, and N_{int} denotes the number of integer operations performed per CPU cycle. The computational energy consumption of UAV N is given as

$$e^c_n(t) = \frac{L_n(t)}{B_{\text{cpu}}N_{\text{int}}}\kappa\left(B_{\text{cpu}}\right)^3 = \frac{\kappa L_n(t)\left(B_{\text{cpu}}\right)^2}{N_{\text{int}}}. \tag{30}$$

The energy consumption expressed in (30) corresponds to the energy consumed by the UAV of step 3 in Fig. 2.

D. Problem Formulation

This work examines the tradeoff between the energy consumption of UAVs and the AoV of edge servers. Based on the UAV's communication and computation models, the total energy consumption of the UAV n at time slot t can be obtained as

$$E_n(t) = e^g_{n,m}(t) + e^r_{n,0}(t) + e^c_n(t). \tag{31}$$

Equation (31) does not account for the energy consumption of UAV flight, as it differs significantly from the energy used for executing zero knowledge proof. To ensure the flight safety of the UAV, this study restricts the total energy consumption during the entire verification cycle to meet the minimum energy requirements necessary for safe UAV operation.

To minimize the AoV of the edge server, the optimization problem in this article is defined as

$$\begin{aligned}
\text{P1}: \quad & \min_{l_n(t),\varphi_{n,m}(t),c_m(t)} \quad \lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\sum_{m=1}^{M}\lambda_m A_m(t) \\
\text{s.t.} \quad & C1: \lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}E_n(t) \le E_a \ \forall n\in\mathcal{N} \\
& C2: \sqrt{\|l_n(t)-l_{n'}(t)\|_2^2} \ge d_{\text{safe}} \ \forall n,n'\in N, n\ne n' \\
& C3: \sum_{m=1}^{M}\varphi_{n,m}(t) \le 1 \ \forall t\in\mathcal{T}, n\in\mathcal{N} \\
& C4: \sum_{n=1}^{N}\varphi_{n,m}(t) \le 1 \ \forall t\in\mathcal{T}, m\in\mathcal{M} \\
& C5: \varphi_{n,m}(t)\in\{0,1\} \ \forall t\in\mathcal{T}, n\in\mathcal{N}, m\in\mathcal{M} \\
& C6: c_m(t) \ge 1 \ \forall t\in\mathcal{T}, m\in\mathcal{M}
\end{aligned} \tag{32}$$

where λ_m indicates the importance of the edge server, $\lambda_m \ge 0$ and $\sum_{m=1}^{M}\lambda_m = 1$. By setting λ_m, the AoV of important edge servers can be made to increase more rapidly, compelling the UAV to conduct verification more frequently. The optimization objective in (32) is to minimize the total AoV of all edge servers throughout the entire verification cycle. The variables in this problem include the UAV's location and its verification strategy. The UAV's location impacts the energy consumption of communication, while its verification strategy influences the energy consumption required for computation. $C1$ is the energy consumption limit of the UAV. The total energy consumption in each time slot must remain below a preset threshold E_a. This constraint ensures the UAV has sufficient energy to safely return to its take-off and landing point. Constraint $C2$ specifies that a minimum safety distance d_{safe} must be maintained between any two UAVs at all times. Constraints $C3$ and $C4$ specify that during the time slot t, each UAV must be paired with exactly one edge server, and each edge server must be verified by exactly one UAV, ensuring a one-to-one correspondence. $C5$ and $C6$ impose certain limitations on the values of the UAV's verification strategy, ensuring it adheres to predefined parameters.

The complexity of the constraints prevents the above problem from being directly solved using a simple iterative algorithm. First, the UAV's verification strategy must adapt dynamically to the changing environment, and the verification strategies of multiple UAVs will influence each other. Second,

P1 is a complex long-term optimization challenge spanning multiple time slots, requiring the transformation of long-term constraints to ensure feasible solutions over time. Therefore, we reformulate P1 using Lyapunov optimization theory and propose a multiagent reinforcement learning algorithm to solve it efficiently.

V. Lyapunov Optimization-Based Deep Reinforcement Learning

In this section, we first transform the long-term energy consumption constraint into a deterministic problem through Lyapunov optimization. We then transform the optimization problem into a Markov decision process and solve it through a multiagent reinforcement learning algorithm.

A. Problem Transformation

To tackle the long-term energy consumption constraint $C1$ in P1, a virtual queue $q(t)$ is introduced to represent and manage the energy consumption over time. The calculation method for the virtual queue is

$$q(t+1) = \max\left[q(t) + E_n(t) - E_a, 0\right]. \tag{33}$$

When the value of the energy consumption virtual queue $q(t)$ is large, it indicates that the current energy consumption is likely approaching or exceeding the tolerable energy consumption threshold. To satisfy the long-term $C1$, the virtual queue should be stable. This stability is expressed as

$$\lim_{t\to\infty} \frac{\mathbb{E}[q(t)]}{t} = 0. \tag{34}$$

In order to achieve the queue stability shown in (34), the Lyapunov function is defined as

$$L(\Theta_t) = \frac{1}{2}q^2(t) \tag{35}$$

where $L(\Theta_t)$ can be used to evaluate the congestion level of the virtual energy consumption queue. When the UAV's position and verification strategy consistently push $L(\Theta_t)$ to a bounded value, the stability of the queue $q(t)$ can be ensured. This prevents the queue from growing indefinitely, thereby maintaining the system's energy consumption within acceptable limits. The Lyapunov drift function is defined as

$$\Delta(\Theta_t) \triangleq \left[L(\Theta_{t+1}) - L(\Theta_t)|\Theta_t\right] \tag{36}$$

where $\Delta(\Theta_t)$ represents the dynamics of the virtual queue $q(t)$ within a time slot in the Lyapunov function, capturing the changes in queue length over time and influencing the system's stability. Then, the queue stability can be incorporated into the objective of P1. The Lyapunov drift plus penalty function is obtained as

$$\Gamma = \Delta(\Theta_t) + \delta_1 \sum_{m=1}^{M} \lambda_m A_m(t) - \delta_2 \sum_{m=1}^{M} \left(A_m(t) - A_m(t-1)\right)^2 \tag{37}$$

where δ_1 and δ_2 are nonnegative parameters used to balance and adjust the relative weight between the AoV and energy consumption in the optimization process. $\sum_{m=1}^{M}(A_m(t) - A_m(t-1))^2$ is used to capture the effect of AoV changes on the optimization problem. Including this term encourages the UAV to verify edge servers with lower importance, helping to balance verification efforts across all servers. Based on the Lyapunov function, the optimization problem in (32) can be transformed into the following form:

$$\begin{aligned} \text{P2}: \quad & \min_{l_n(t),\varphi_{n,m}(t),c_m(t)} \quad \Gamma \\ & \text{s.t.} \quad C2 - C6. \end{aligned} \tag{38}$$

To minimize Γ, it is necessary to reduce the upper bound of $\Delta(\Theta_t)$. By minimizing this upper bound, the overall objective function is effectively optimized. According to the definition of $\Delta(\Theta_t)$, we can derive the following:

$$\begin{aligned} \Delta(\Theta_t) &\triangleq \left[L(\Theta_{t+1}) - L(\Theta_t)|\Theta_t\right] \\ &= \left[\frac{1}{2}q^2(t+1) - \frac{1}{2}q^2(t)|\Theta_t\right]. \end{aligned} \tag{39}$$

Let $\Psi_t = E_n(t) - E_a$, then we can further deduce the following:

$$\begin{aligned} q^2(t+1) &\le q^2(t) + (E_n(t) - E_a)^2 + 2q(t)(E_n(t) - E_a) \\ &= q^2(t) + \Psi_t^2 + 2q(t)\Psi_t \end{aligned} \tag{40}$$

and (40) can be used to deduce

$$\left[q^2(t+1) - q^2(t)|\Theta_t\right] \le \left[\Psi_t^2 + 2q(t)\Psi_t|\Theta_t\right]. \tag{41}$$

For function $\Psi_t = E_n(t) - E_a$, we can always find a fixed value $\Psi_{\max} > 0$, where $\Psi_{\max} = E_n^2(t)_{\max} + E_a^2$, that satisfies the following relationship:

$$\Delta(\Theta_t) \le \frac{1}{2}\Psi_{\max} + q(t)[\Psi_t|\Theta_t]. \tag{42}$$

Combining (42) with P2

$$\begin{aligned} \Gamma &= \Delta(\Theta_t) + \delta_1 \sum_{m=1}^{M} \lambda_m A_m(t) - \delta_2 \sum_{m=1}^{M} \left(A_m(t) - A_m(t-1)\right)^2 \\ &\le \frac{1}{2}\Psi_{\max} + q(t)\Psi_t + \delta_1 \sum_{m=1}^{M} \lambda_m A_m(t) \\ &\quad - \delta_2 \sum_{m=1}^{M} \left(A_m(t) - A_m(t-1)\right)^2. \end{aligned} \tag{43}$$

To minimize Γ, the terms containing variables in (43) are combined into a new optimization objective [36]

$$U = q(t)\Psi_t + \delta_1 \sum_{m=1}^{M} \lambda_m A_m(t) - \delta_2 \sum_{m=1}^{M} \left(A_m(t) - A_m(t-1)\right)^2. \tag{44}$$

Based on the above transformation, the original P1 can be transformed into P3

$$\begin{aligned} \text{P3}: \quad & \min_{l_n(t),\varphi_{n,m}(t),c_m(t)} \quad U \\ & \text{s.t.} \quad C2 - C6. \end{aligned} \tag{45}$$

P3 can be solved by iterating the $q(t)$ at each time slot t. To enable collaborative verification among multiple UAVs, we propose using an MAPPO algorithm to solve P3.

B. Multiagent Reinforcement Learning

1) Observation: Defining UAVs as intelligent agents. In the entire verification system, UAVs can directly observe the coordinates of all participants. Let $\mathcal{L}_{\mathcal{N}}(t) = \{l_1(t), l_2(t), \ldots, l_N(t)\}$ represent the coordinate set of all UAVs at time slot t, $\mathcal{L}_{\mathcal{M}} = \{l_1, l_2, \ldots, l_M\}$ represent the coordinate set of the edge servers, and $\mathcal{L}_0 = \{l_0, l_1, \ldots,\}$ represent the coordinate set of the cloud server. This study considers a scenario with only one cloud server, but the proposed solution is not restricted to a single cloud server and can accommodate multiple cloud servers. Additionally, the UAV can obtain the AoV set $\mathcal{A}_{\mathcal{M}}(t) = \{A_1(t), A_2(t), \ldots, A_M(t)\}$ for all edge servers at time slot t through interaction with the edge servers.

2) State: During the verification process, the locations of all UAVs, edge servers, cloud servers, and the AoV of all edge servers collectively form the overall system state at time slot t, which is expressed as

$$S_n(t) = \{\mathcal{L}_{\mathcal{N}}(t), \mathcal{L}_{\mathcal{M}}, \mathcal{L}_0, \mathcal{A}_{\mathcal{M}}(t)\}. \tag{46}$$

3) Action: The motion trajectory and verification strategy of each UAV define its action space. The policy function aims to optimize this action space to achieve higher reward. The action space of UAV n is defined as

$$A_n(t) = \left\{\Delta l_n(t), \varphi_{n,m}(t), c_m(t)\ \forall m \in \mathcal{M}\right\} \tag{47}$$

where $\Delta l_n(t) = \{\Delta x_n(t), \Delta y_n(t)\}$ represents the coordinate change of the UAV at time slot t. The horizontal and vertical coordinates of the UAV can be calculated as follows:

$$\begin{cases} x_n(t) = x_n(t-1) + \Delta x_n(t) \\ y_n(t) = y_n(t-1) + \Delta y_n(t). \end{cases} \tag{48}$$

4) Reward: The optimization goal of P3 is to minimize function U, which is equivalent to maximizing $-U$. Therefore, the reward function of UAV n is set as

$$R_n(t) = -U. \tag{49}$$

Based on (49), we enhance the reward function by normalizing it to better promote action space optimization and accelerate algorithm convergence. This normalization ensures a more consistent and effective learning process. The improved reward function is [37]

$$R_n(t) = \frac{R_n(t) - \bar{R}_n(t-1)}{\hat{R}_n(t-1) + \sigma} \tag{50}$$

where $\bar{R}_n(t-1)$ and $\hat{R}_n(t-1)$ denote the mean and standard deviation of $R_n(t-1)$. σ is a small constant to avoid the denominator being zero. $\bar{R}_n(t)$ and $\hat{R}_n(t)$ can be obtained by

$$\bar{R}_n(t) = \bar{R}_n(t-1) + \frac{R_n(t) - \bar{R}_n(t-1)}{t} \tag{51}$$

$$\hat{R}_n(t) = \sqrt{\frac{R_n(t-1) + \left[R_n(t) - \bar{R}_n(t-1)\right]\left[R_n(t) - \bar{R}_n(t)\right]}{t}}. \tag{52}$$

Algorithm 1: Training EMAPPO Algorithm in UAVs

```
Initialize parameters θ, φ, ε and δ;
Clear the collected history D_n;
for Each episode do
    Receive initial state S_n(0);
    for t = {1, 2, ..., T} do
        Each agent n obtains an action space
         a_n(t) = π̃_θ(a_n(t), o_n(t));
        UAV n performs action a_n(t) and obtains new
         observations o_n(t + 1);
        Calculate r_n(t) using Eq. (50);
        Store [o_n(t), a_n(t), r_n(t), o_n(t + 1)];
        for n = {1, 2, ..., N} do
            Estimate advantages A_n^θ based on the current
             value function V_φ;
            Update the parameters of the Actor network
             with θ = arg max_θ J_clip^θ(θ);
            Update the parameters of the Critic network
             through TD-Error;
        end
    end
end
```

5) EMAPPO: This study builds upon the MAPPO framework, transforming the problem using Lyapunov optimization and normalizing the reward function. The proposed algorithm is named enhanced-MAPPO (EMAPPO) and the pseudo code of its training process is shown in Algorithm 1. Each UAV is assigned an Actor network and a Critic network. The Actor network generates the UAV's action, while the Critic network evaluates the quality of the action produced by the Actor. The policy function for the Actor to select actions is defined as $\tilde{\pi}_\theta$, where θ is the parameter of the Actor network. The global UAV rewards are accumulated as

$$r_\theta(t) = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T} R(\tau_n)\tilde{\pi}_\theta(a_n(t)|s_n(t)) \tag{53}$$

where τ_n represents a record of the interaction between UAV n and the environment. As the UAV explores the environment more thoroughly, the influence of earlier rewards on the action space should diminish over time. To account for this, there is a discounted reward $R(\tau_n) = \sum_{t=1}^{T}\delta^t r(t)$ and δ is a discount constant. The optimization direction of the Actor network is to increase the value of (53), and the Actor network parameters are updated via gradients, which can be expressed as

$$\nabla\hat{r}_\theta = E_{\tau\sim\tilde{\pi}_\theta(\tau)}\left[A^\theta(s(t), a(t))\nabla\log\tilde{\pi}_\theta(a_n(t)|s_n(t)\right] \tag{54}$$

where A^θ is the advantage function and $A^\theta = \tilde{\pi}_\theta(s(t), a(t)) - V_\phi(s(t))$. $V_\phi(s(t))$ is the value function

$$V_\phi(s(t)) = \max\ \frac{1}{T}\sum_{\mathcal{D}_n}\sum_{t=0}^{T}\left(V_\phi(s(t)) - \hat{r}(t)\right)^2 \tag{55}$$

where $\mathcal{D}_n$ represents the recorded global Markov process.

For the On-Policy reinforcement learning algorithm, when the model parameters are updated, we have to sample training

TABLE I
SIMULATION PARAMETERS

Symbol	Value	Meaning
H	70 m	UAV flight altitude
$c_m(t)$	[1,7]	Number of layers to verify
α	10	Parameters for finding LoS/NLoS probability
β	0.6	Parameters for finding LoS/NLoS probability
γ_0	-30 db	Channel power gain
μ	0.2	Additional attenuation factor for NLoS links
$\tilde{\alpha}$	2	Path fading parameters
W	3 Mhz	Bandwidth
P_n	40 dBm	UAV transmission power
${\sigma_n}^2$	-110 dBm	Channel noise power of UAV
D_g	2 MB	Size of data packet sent by UAV
D_r	3 MB	Size of verification results
D_{gan}	40 MB	Average size of data for proof of neural networks
N_{cpu}	1000	Number of operations required to verify unit data
B_{cpu}	1 Ghz	CPU cycle
N_{int}	2	Number of integer operations per CPU cycle
κ	10^{-28}	CPU computing energy efficiency
d_{safe}	20 m	Safe distance between UAVs

TABLE II
GAN MODEL

Output size	Stride	Padding	Model layers
32×32	2	1	5
64×64	2	1	5
128×128	2	1	6

data again [38]. To reuse the sampled data, we adopt an Off-Policy strategy, and (54) can be rewritten as

$$\nabla \hat{r}_\theta = E_{\tau\sim\tilde{\pi}_\theta(\tau)}\left[\frac{\nabla\tilde{\pi}_\theta(a_n(t)|s_n(t))}{\tilde{\pi}_{\theta'}(a_n(t)|s_n(t))}A^{\theta'}(s(t),a(t))\right] \quad (56)$$

and the objective function is

$$J^{\theta'}(\theta) = E_{\tau\sim\tilde{\pi}_\theta(\tau)}\left[\frac{\tilde{\pi}_\theta(a(t)|s(t))}{\tilde{\pi}_{\theta'}(a(t)|s(t))}A^{\theta'}(s(t),a(t))\right]. \quad (57)$$

To minimize the difference between $\tilde{\pi}_\theta$ and $\tilde{\pi}_{\theta'}$, we added clip to the objective function

$$J^{\theta}_{\text{clip}}(\theta) = \sum_{(s(t),a(t))} \min\left[r_\theta(t)A^\theta, \text{clip}(r_\theta(t), 1-\varepsilon, 1+\varepsilon)A^\theta\right]. \quad (58)$$

In the algorithm iteration within the multiagent environment, each UAV uses its own policy $\tilde{\pi}_\theta(a(t), o(t))$ to generate actions based on the observed environmental state $o_n(t)$. The actions of each UAV are subsequently evaluated by the global Critic network, which provides feedback to improve their performance.

VI. PERFORMANCE EVALUATION

A. Simulation Setting

This section presents simulations and tests to evaluate the performance of the proposed multiagent reinforcement learning algorithm and the zero knowledge proof system. The results aim to demonstrate the effectiveness and efficiency of both approaches in ensuring secure model verification and optimal UAV coordination. A 200 × 200 simulation area is established, with the cloud server positioned at the center of the area. Four edge servers are randomly distributed across four different locations within the area. In the simulation, three UAVs are deployed, with each UAV's take-off point randomly set. This random initialization helps ensure better exploration of the environment during the algorithm's training phase, allowing the UAVs to effectively learn optimal strategies for verification and coordination. Additionally, following established paper [39], [40], the settings for other parameters are detailed in Table I. This table provides a comprehensive overview of the simulation environment, including parameters relevant to the UAVs, edge servers, and the verification process. When training and testing the reinforcement learning algorithms, the computer used was equipped with 16 GB of memory and an NVIDIA GeForce RTX 4070 Ti GPU. When testing the performance of the zero knowledge proof system, we used a VMware virtual machine running Ubuntu 20.04, equipped with 12 processor cores and 16 GB of memory.

To demonstrate the superiority of the proposed EMAPPO algorithm, three algorithms—MAPPO, MADDPG, and DQN—are used as benchmarks for comparison.

MAPPO is an extension of the proximal policy optimization (PPO) algorithm designed to handle both continuous and discrete action spaces. This algorithm stabilizes the learning process by constraining the step size of policy updates, ensuring smoother and more reliable training. MAPPO is well-suited for scenarios involving multirobot collaborative operations and multiagent adversarial settings, where multiple agents or robots need to coordinate or compete effectively.

MADDPG is a multiagent reinforcement learning algorithm designed to handle continuous action spaces effectively. This algorithm uses a deterministic policy gradient method to directly output a specific action rather than a probability distribution of an action. The algorithm is also applicable to cooperative or confrontation scenarios among multiple agents.

DQN is a classic reinforcement learning algorithm that has been widely used by researchers to address UAV path planning and resource allocation problems. By combining Q-learning with deep neural networks, DQN can handle complex decision-making tasks, making it a popular choice for solving challenges in UAV operations and other domains.

B. Simulation Results

Fig. 3 illustrates the differences in rewards among the four reinforcement learning algorithms: 1) EMAPPO; 2) MAPPO; 3) MADDPG; and 4) DQN. During the training process, the learning rate of the four algorithms is set to 0.001. It can be observed that the convergence speed of the proposed EMAPPO algorithm is relatively fast, with its reward value being approximately 10% higher than that of MAPPO. This demonstrates the efficiency and effectiveness of EMAPPO in optimizing the task compared to the standard MAPPO algorithm. The MADDPG algorithm shows almost no change in reward, which may be due to the fact that our action space contains both discrete and continuous actions. This mixed action space makes it challenging for MADDPG. The DQN algorithm remains in an oscillating state throughout the training process. This is likely because the multiagent

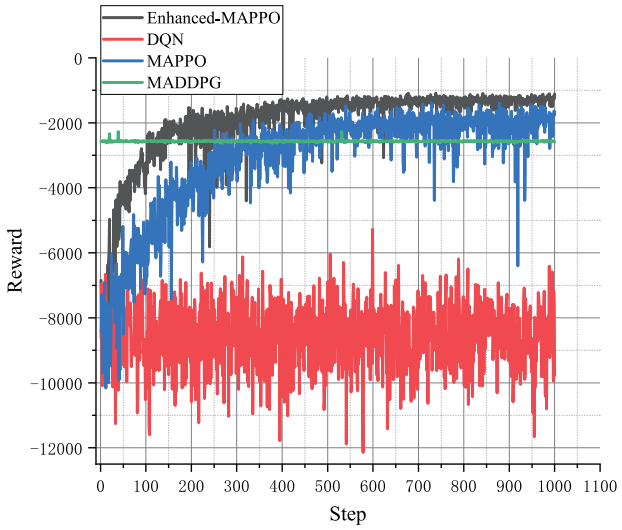

Fig. 3. Comparison of reward among four algorithms.

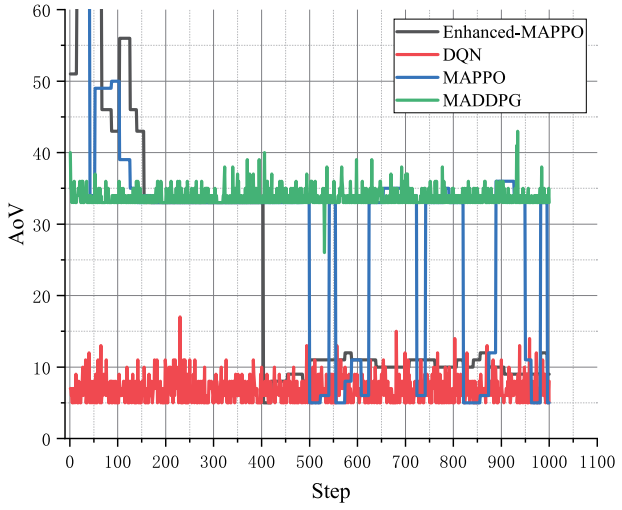

Fig. 4. Comparison of AoV among four algorithms.

optimization problem is too complex for DQN to handle effectively.

Fig. 4 illustrates the changing trends of the AoV during the training process of the four algorithm. It can be observed that the AoV of the proposed EMAPPO algorithm decreases rapidly as training progresses and then stabilizes. This indicates that the EMAPPO algorithm is effective in quickly optimizing the verification process, ensuring timely checks while maintaining stability in the long run. In contrast, the AoV of the MAPPO algorithm decreases more slowly and exhibits instability, highlighting the effectiveness of the reward normalization operation in the EMAPPO algorithm. Similar to the results shown in Fig. 3, the AoV of the MADDPG algorithm still has difficulty converging. Additionally, to investigate whether DQN can achieve convergence, Fig. 4 presents the AoV trend after 20 000 steps of DQN training. It can be observed that, with extended training, DQN's AoV becomes comparable to that of EMAPPO, but it still lacks stability. This suggests that, although DQN can improve with more

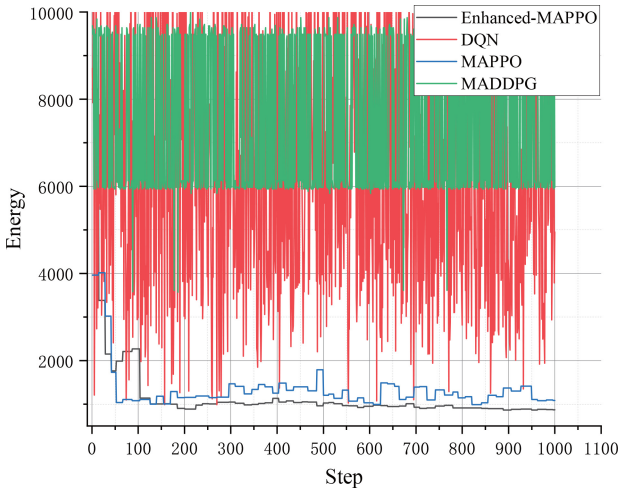

Fig. 5. Comparison of energy consumption among four algorithms.

training, it struggles to achieve the same level of consistency and robustness as EMAPPO.

This study treats AoV and verification energy consumption as a tradeoff problem. Fig. 5 illustrates the optimization effects of the four algorithms on energy consumption. It can be observed that the verification energy consumption achieved by the EMAPPO algorithm is the lowest among the four algorithms. When combined with its strong performance on AoV, this fully demonstrates the superiority of our proposed algorithm, as it effectively balances energy efficiency with timely and accurate verification. The verification energy consumption of the MAPPO algorithm is less stable compared to the algorithm we proposed, and is approximately 10%–100% higher. As with previous results, the MADDPG algorithm fails to converge, further demonstrating its difficulty in optimizing the verification energy consumption and adapting to the UAV environment in this scenario. Additionally, we present the verification energy consumption performance of the DQN algorithm for the same training steps as shown in Fig. 5. It can be seen that the verification energy consumption of the DQN algorithm fluctuates significantly, indicating that DQN struggles to balance both AoV and verification energy consumption simultaneously. This highlights the algorithm's difficulty in achieving stable optimization for both metrics.

In addition to demonstrating the superiority of the proposed EMAPPO algorithm, we also tested the latency performance of the zero knowledge proof Groth-16 algorithm during the model verification process. This evaluation provides insights into the computational efficiency and practicality of applying Groth-16 for secure verification in real-world scenarios. For the GAN model with an output image size of 64×64, the latency performance of each stage using the Groth-16 algorithm is presented in Fig. 6. It can be observed that the latency for the three operations—transposed convolution, BatchNorm, and ReLU—in the setup phase and the proof generation phase does not differ significantly. However, the latency for transposed convolution in these two phases is notably higher than for the other two operations. This indicates that, within zero knowledge proof, transposed convolution involves the most

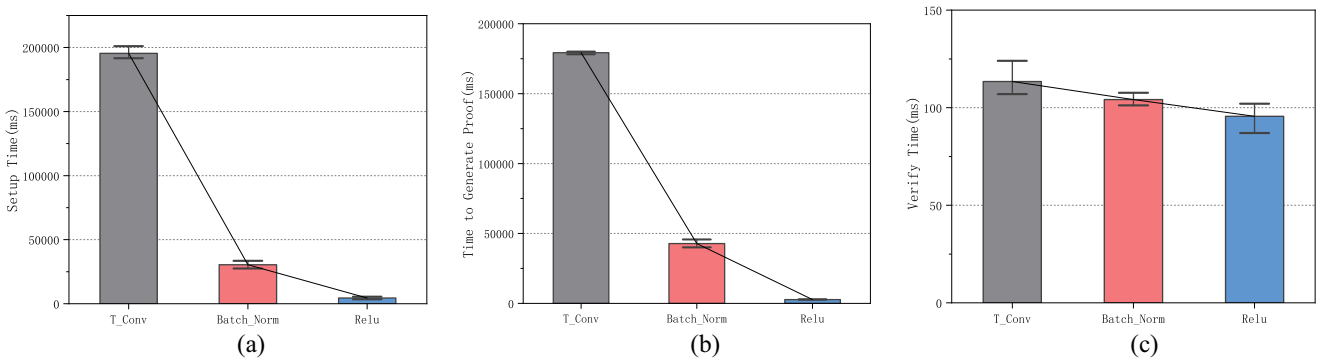

Fig. 6. Zero knowledge proof latency for a GAN model with an output image size of 64 × 64. (a) Delay of setup phase. (b) Delay of proof generation. (c) Delay of verification phase.

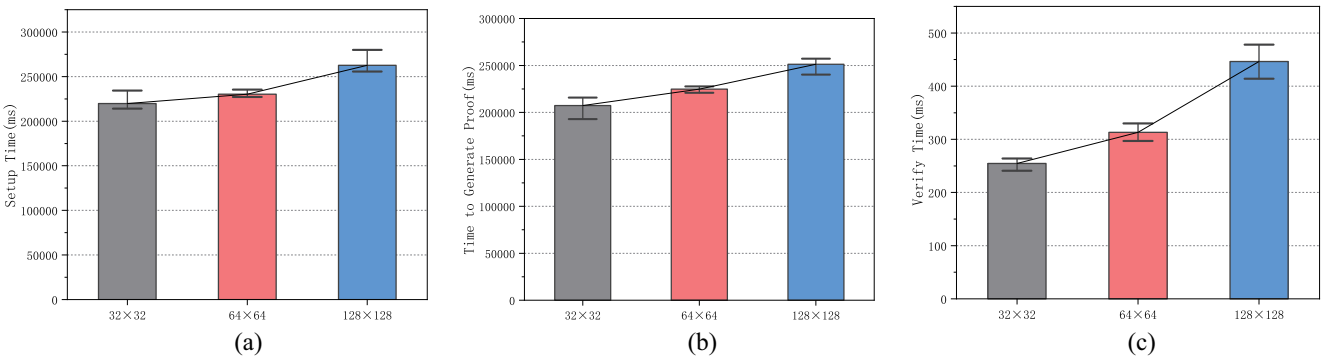

Fig. 7. Comparison of zero knowledge proof latency for three GAN models with different outputs. (a) Delay of setup phase. (b) Delay of proof generation. (c) Delay of verification phase.

computation, which aligns with its computational demand during model inference. Compared to the setup and proof generation stages, the vertical axis of Fig. 6(c) shows that the latency in the verification stage is significantly lower than in the first two stages. Additionally, the latency differences among the three GAN operations in the verification stage are minimal. This indicates that the computational workload during the verification stage is quite small, and the computing power of the UAV is sufficient to meet the verification requirements efficiently.

Fig. 7 shows the latency performance of different GAN models in the three stages of zero knowledge proof. The settings of the GAN model are shown in Table II. In general, as the size of the generated image increases, the complexity of the model also grows, leading to higher latency in the zero knowledge proof process across all three stages. At the same time, it can be seen that in the verification phase, the verification delays of the three GAN models are all within 500 ms and the difference between them is not large. The above results further illustrate the practicality of UAV as a zero knowledge proof verifier.

VII. Conclusion

This study aims to solve the verification problem of the GAI model. We propose a UAV-based zero knowledge proof architecture to ensure both the quality of the model and the security of its parameters. To enhance the timeliness of verification, we introduce a new evaluation metric AoV to measure the freshness of model verification. We also design an enhanced MAPPO algorithm to balance the energy consumption of the verifiers (UAV) and the AoV of the edge server. Simulation results demonstrate that the architecture and algorithm we proposed are superior, offering improved performance in verification efficiency and energy management. In the future, we will further explore additional zero knowledge proof methods for GAI models and delve deeper into the data security challenges associated with GAI models.

References

[1] J. Wang et al., "Generative AI for integrated sensing and communication: Insights from the physical layer perspective," *IEEE Wireless Commun.*, vol. 31, no. 5, pp. 246–255, Oct. 2024.

[2] J. Wang et al., "Generative AI based secure wireless sensing for ISAC networks," 2024, *arXiv:2408.11398*.

[3] R. Cheng, Y. Sun, D. Niyato, L. Zhang, L. Zhang, and M. A. Imran, "A wireless AI-generated content (AIGC) provisioning framework empowered by semantic communication," *IEEE Trans. Mobile Comput.*, vol. 24, no. 3, pp. 2137–2150, Mar. 2025.

[4] Y. Liu et al., "Optimizing mobile-edge AI-generated everything (AIGX) services by prompt engineering: Fundamental, framework, and case study," *IEEE Networks*, vol. 38, no. 5, pp. 220–228, Sep. 2024.

[5] M. Xu et al., "When large language model agents meet 6G networks: Perception, grounding, and alignment," *IEEE Wireless Commun.*, vol. 31, no. 6, pp. 63–71, Dec. 2024.

[6] X. Li et al., "Transformer-based visual segmentation: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 12, pp. 10138–10163, Dec. 2024.

[7] J. Wang et al., "Optimizing 6G integrated sensing and communications (ISAC) via expert networks," 2024, *arXiv:2406.00408*.

[8] P. Wu, Q. Liu, Y. Dong, Z. Wang, and F. Wang, "LMaaS: Exploring pricing strategy of large model as a service for communication," *IEEE Trans. Mobile Comput.*, vol. 23, no. 12, pp. 12748–12760, Dec. 2024.
[9] V. Liagkou, E. Filiopoulou, G. Fragiadakis, M. Nikolaidou, and C. Michalakelis, "The cost perspective of adopting large language model-as-a-service," in *Proc. IEEE Int. Conf. Joint Cloud Comput. (JCC)*, 2024, pp. 80–83.
[10] P. Han, S. Wang, Y. Jiao, and J. Huang, "Federated learning while providing model as a service: Joint training and inference optimization," in *Proc. IEEE Conf. Comput. Commun.*, 2024, pp. 631–640.
[11] H. Zhou et al., "Large language model (LLM) for telecommunications: A comprehensive survey on principles, key techniques, and opportunities," *IEEE Commun. Surveys Tuts.*, early access, Sep. 23, 2024, doi: 10.1109/COMST.2024.3465447.
[12] B. Feng, C. Feng, D. Feng, Y. Wu, and X.-G. Xia, "Proactive content caching scheme in urban vehicular networks," *IEEE Trans. Commun.*, vol. 71, no. 7, pp. 4165–4180, Jul. 2023.
[13] L. U. Khan, I. Yaqoob, N. H. Tran, S. M. A. Kazmi, T. N. Dang, and C. S. Hong, "Edge-computing-enabled smart cities: A comprehensive survey," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 10200–10232, Oct. 2020.
[14] P. Li et al., "FAST: Fidelity-adjustable semantic transmission over heterogeneous wireless networks," in *Proc. IEEE Int. Conf. Commun.*, 2023, pp. 4689–4694.
[15] S. Rong, W. Zhong, X. Huang, J. Kang, S. Xie, and C. Yuen, "Joint path selection, energy trading and task offloading in electric vehicle charging and computing network," *IEEE Internet Things J.*, vol. 11, no. 10, pp. 17067–17081, May 2024.
[16] C. Shang, D. T. Hoang, M. Hao, D. Niyato, and J. Yu, "Energy-efficient Decentralized federated learning for UAV swarm with spiking neural networks and leader election mechanism," *IEEE Wireless Commun. Lett.*, vol. 13, no. 10, pp. 2742–2746, Oct. 2024.
[17] G. Sun et al., "Multi-objective optimization for multi-UAV-assisted mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 23, no. 12, pp. 14803–14820, Dec. 2024.
[18] S. Liu et al., "UAV-enabled collaborative beamforming via multi-agent deep reinforcement learning," *IEEE Trans. Mobile Comput.*, vol. 23, no. 12, pp. 13015–13032, Dec. 2024.
[19] H. Pan, Y. Liu, G. Sun, J. Fan, S. Liang, and C. Yuen, "Joint power and 3D trajectory optimization for UAV-enabled wireless powered communication networks with obstacles," *IEEE Trans. Commun.*, vol. 71, no. 4, pp. 2364–2380, Apr. 2023.
[20] Y. Fan, B. Xu, L. Zhang, J. Song, A. Zomaya, and K.-C. Li, "Validating the integrity of convolutional neural network predictions based on zero-knowledge proof," *Inf. Sci.*, vol. 625, pp. 125–140, May 2023.
[21] S. Lee, H. Ko, J. Kim, and H. Oh, "vCNN: Verifiable convolutional neural network based on zk-SNARKs," *IEEE Trans. Dependable Secure Comput.*, vol. 21, no. 4, pp. 4254–4270, Jul./Aug. 2024.
[22] B.-M. Ganescu and J. Passerat-Palmbach, "Trust the process: Zero-knowledge machine learning to enhance trust in generative AI interactions," 2024, *arXiv:2402.06414*.
[23] S. Wellington, "BasedAI: A decentralized P2P network for zero knowledge large language models (ZK-LLMs)," 2024, *arXiv:2403.01008*.
[24] Y. Lin et al., "Blockchain-aided secure semantic communication for AI-generated content in metaverse," *IEEE Open J. Comput. Soc.*, vol. 4, pp. 72–83, Mar. 2023, doi: 10.1109/OJCS.2023.3260732.
[25] H. Tan, W. Zheng, and P. Vijayakumar, "Secure and efficient authenticated key management scheme for UAV-assisted infrastructure-less IoVs," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 6, pp. 6389–6400, Jun. 2023.
[26] Y. Wang, Z. Su, T. H. Luan, J. Li, Q. Xu, and R. Li, "SEAL: A strategy-proof and privacy-preserving UAV computation offloading framework," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 5213–5228, 2023.
[27] Y. Zheng, C.-H. Chang, S.-H. Huang, P.-Y. Chen, and S. Picek, "An overview of trustworthy AI: Advances in IP protection, privacy-preserving federated learning, security verification, and GAI safety alignment," *IEEE J. Emerg. Select. Topics Circuits Syst.*, vol. 14, no. 4, pp. 582–607, Dec. 2024.
[28] B. Wang et al., "DecodingTrust: A comprehensive assessment of trustworthiness in GPT models," in *Proc. 37th NeurIPS*, 2023, pp. 1–110.
[29] X. Zhang et al., "PrivacyAsst: Safeguarding user privacy in tool-using large language model agents," *IEEE Trans. Dependable Secure Comput.*, vol. 21, no. 6, pp. 5242–5258, Nov./Dec. 2024.
[30] J. Zhao, K. Chen, X. Yuan, Y. Qi, W. Zhang, and N. Yu, "Silent guardian: Protecting text from malicious exploitation by large language models," *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 8600–8615, 2024.
[31] G. Sun et al., "Joint task offloading and resource allocation in aerial-terrestrial UAV networks with edge and fog computing for post-disaster rescue," *IEEE Trans. Mobile Comput.*, vol. 23, no. 9, pp. 8582–8600, Sep. 2024.
[32] U. Fiege, A. Fiat, and A. Shamir, "Zero knowledge proofs of identity," in *Proc. 19th Annu. ACM Symp. Theory Comput.*, 1987, pp. 210–217.
[33] K. Baghery, M. Kohlweiss, J. Siim, and M. Volkhov, "Another look at extraction and randomization of Groth's Zk-SNARK," in *Proc. Int. Conf. Financ. Cryptogr. Data Security*, 2021, pp. 457–475.
[34] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2015, *arXiv:1511.06434*.
[35] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1125–1134.
[36] F. Sun, Z. Zhang, X. Chang, and K. Zhu, "Toward heterogeneous environment: Lyapunov-orientated imphetero reinforcement learning for task offloading," *IEEE Trans. Netw. Service Manag.*, vol. 20, no. 2, pp. 1572–1586, Jun. 2023.
[37] C. Shang, D. T. Hoang, J. Yu, M. Hao, and D. Niyato, "Constructing the metaverse with a new perspective: UAV FoV-assisted low-latency imaging," *IEEE Wireless Commun. Lett.*, vol. 14, no. 1, pp. 158–162, Jan. 2025.
[38] J. Li et al., "Collaborative ground-space communications via evolutionary multi-objective deep reinforcement learning," *IEEE J. Sel. Areas Commun.*, vol. 42, no. 12, pp. 3395–3411, Dec. 2024.
[39] Y. Yu, J. Tang, J. Huang, X. Zhang, D. K. C. So, and K.-K. Wong, "Multi-objective optimization for UAV-assisted wireless powered IoT networks based on extended DDPG algorithm," *IEEE Trans. Commun.*, vol. 69, no. 9, pp. 6361–6374, Sep. 2021.
[40] M. Wu, D. Ye, J. Ding, Y. Guo, R. Yu, and M. Pan, "Incentivizing differentially private federated learning: A multidimensional contract approach," *IEEE Internet Things J.*, vol. 8, no. 13, pp. 10639–10651, Jul. 2021.

Min Hao received the Ph.D. degree from the School of Automation, Guangdong University of Technology, Guangzhou, China, in 2023.

He is currently a Research Fellow with the Pillar of Information Systems Technology and Design, Singapore University of Technology and Design, Singapore. His research interests include vehicular networks, semantic communications, and metaverse.

Chen Shang received the M.S. degree from the College of Big Data and Information Engineering, Guizhou University, Guizhou, China, in 2023.

He is currently a Research Assistant with the Pillar of Information Systems Technology and Design, Singapore University of Technology and Design, Singapore. His research interests include machine learning, wireless communication, and metaverse.

Siming Wang received the Ph.D. degree from the School of Automation, Guangdong University of Technology, Guangzhou, China, in 2023.

He is currently a Research Fellow with the Pillar of Information Systems Technology and Design, Singapore University of Technology and Design, Singapore. His research interests include blockchain, edge intelligence, and vehicular networks.

Wenchao Jiang (Member, IEEE) received the Ph.D. degree from the Department of Computer Science and Engineering, University of Minnesota Twin Cities, Minneapolis, MN, USA, in 2019.

He is currently an Assistant Professor with the Pillar of Information System Technology and Design, Singapore University of Technology and Design, Singapore. His research interests include the Internet of Things, wireless and low-power embedded networks, and mobile computing.

Jiangtian Nie received the Ph.D. degree from ERI@N, Interdisciplinary Graduate School, Nanyang Technological University (NTU), Singapore, in 2021.

She was a visiting student with Princeton University, Princeton, NJ, USA, and the University of Waterloo, Waterloo, ON, Canada. She is currently with NTU. She has published over 30 peer-reviewed papers in leading journals and flagship conferences. Her research interests include network economics, game theory, wireless blockchain, crowd sensing, and learning.