Enabling Privacy-preserving and Drop-out Resilient Federated LLM Fine-tuning for the low-altitude UAV Swarm Networks

Yangyang Bao, Xiaochun Cheng *Senior Member, IEEE*, Liming Nie*, and Junyi Tao

***Abstract*—Advancement of unmanned aerial vehicle (UAV) swarm networks presents transformative opportunities for low-altitude surveillance, disaster response, and distributed sensing, where federated large language models (LLMs) enable collaborative learning while preserving data privacy, enhance swarm-level situational awareness through decentralized knowledge fusion, and support adaptive decision-making across dynamic low-altitude operational environments. However, federated LLM fine-tuning for UAV swarm networks operating in low-altitude settings faces three unresolved security and practical issues: (1) Lack of efficient methods to protect parameter security during uplink/downlink transmission under low-altitude communication constraints; (2) Absence of effective mechanisms to handle frequent UAV dropouts caused by low-altitude dynamics that may compromise the robustness of federated LLM systems; and (3) Constraints in UAVs' computing, storage and communication resources under typical low-altitude mission profiles. To address these challenges, this paper proposes a Secure and privacy-preserving federated fine-tuning (SPFF) scheme for low-altitude UAV swarms that enables: efficient and privacy-preserving one-to-many distribution of global parameters for downlink federated fine-tuning; secure and efficient uplink local parameter uploading adapted to low-altitude network conditions; and encrypted-parameter-based global model fine-tuning. The scheme also incorporates an efficient supervised key update mechanism to address UAV dropout issues common in low-altitude operations. Moreover, we design a delegable extensional SPFF (DE-SPFF) scheme that employs proxy re-encryption to allow UAVs to delegate tasks to other drones before exiting the federated fine-tuning process in volatile low-altitude environments, while providing public verifiability for re-encryption operations performed by semi-trusted edge nodes. Formal security proofs demonstrate the security of the proposed schemes under low-altitude threat models. Theoretical analysis and experimental results confirm their superiority and practicality for low-altitude UAV swarm applications.**

***Index Terms*—Attribute-based encryption, federated fine-tuning, functional encryption, large language model, unmanned aerial vehicle network.**

Manuscript received May 31, 2025, revised September 5, 2025. This work is sponsored by UKRI (Grant Number: EP/W020408/1), Doctoral Training Centre at Swansea University (Grant Number: RS718), National Natural Science Foundation of China (NSFC) (Grant Number: W2412110), and China Postdoctoral Science Foundation (Grant Number: 2024M753597). *(Corresponding author: Liming Nie.)*

Yangyang Bao is with Antai College of Economics and Management, Shanghai Jiao Tong University, Shanghai, and also with the Postdoctoral Station of Credit Reference Center, People's Bank of China, Shanghai, 201201, China (email: byy_sjtu@sjtu.edu.cn).

Xiaochun Cheng is with the Department of Computer Science, Swansea University, Bay Campus, Fabian Way, Swansea, SA1 8EN, Wales, U.K (e-mail: xiaochun.cheng@swansea.ac.uk).

Liming Nie is with the College of Big Data and Internet, Shenzhen Technology University, Shenzhen, Guangdong Province, 518118, China, (e-mail: nieliming@sztu.edu.cn).

Junyi Tao is with the Department of Computer Science, Stony Brook University, Stony Brook, NY 11794 USA (e-mail: jutao@cs.stonybrook.edu).

I. Introduction

THE rapid advancement of unmanned aerial vehicle (UAV) swarm networks [1] has introduced unprecedented opportunities for applications in surveillance, disaster response, and distributed sensing, while simultaneously posing significant challenges in communication, coordination, and autonomous decision-making [2]. These challenges are particularly acute in low-altitude environments, where UAV swarms must operate amidst complex urban canyons, varying terrain, and dynamic atmospheric conditions that create unique constraints and vulnerabilities [3]. Traditional control paradigms often struggle to cope with the dynamic, large-scale, and highly uncertain nature of these environments, necessitating more intelligent and adaptive solutions. In this context, the integration of Large Language Models (LLMs) [4] with federated learning (FL) presents a transformative paradigm for next-generation UAV swarm intelligence. This federated LLM framework is uniquely suited to address the core challenges of low-altitude swarm operations. Unlike centralized AI, it enables collaborative knowledge fusion across the swarm without centralizing raw data, thus preserving privacy and reducing bandwidth overheadła critical advantage for bandwidth-constrained UAV networks. LLMs contribute their remarkable capabilities in contextual reasoning, situational awareness, and complex mission planning, allowing the swarm to interpret complex environmental cues, generate adaptive flight plans, and understand high-level human intent. Federated learning orchestrates this process, allowing each UAV to fine-tune shared LLMs locally on its sensory data and lived experience, after which only the learned model updates (not the sensitive raw data), are securely aggregated to evolve a collective swarm intelligence.

However, while LLMs have demonstrated exceptional performance in general-purpose domains, their direct application to UAV swarm networks operating in low-altitude settings remains constrained by several critical factors, including real-time processing requirements, domain-specific knowledge gaps, and the need for robust operation in mission-critical scenarios. Consequently, federated LLM fine-tuning [5] for UAV swarm networks has become an essential research direction, aiming to bridge the gap between generic language models and the specialized demands of swarm intelligence in low-altitude airspace. This approach involves optimizing LLMs

for multi-agent collaboration, dynamic resource allocation, and emergent behavior modeling under the specific constraints of low-altitude operations, while addressing key challenges such as data efficiency in training, robustness against adversarial conditions, and interpretability in decentralized decision-making systems. Recent studies further suggest that tailored LLM architectures could enhance swarm adaptability in low-altitude environments, enabling more sophisticated human-swarm interaction and autonomous mission planning, thereby opening new frontiers in intelligent swarm control for low-altitude applications.

A. Security and practicality issues

Although federated LLM bring boundless benefits to the performance optimization and application expansion of UAV swarm networks operating in low-altitude environments, due to their openness, dynamism, and unique architectural characteristics, several security and practicality issues need to be addressed for federated LLM systems tailored for low-altitude UAV swarm networks to transition from theoretical blackprints to practical deployment.

(1) Lack of systematic data security protection mechanisms for the upstream/downstream parameters transmission in federated LLM. Federated fine-tuning of LLM in low-altitude UAV swarm networks operates as a data-driven service framework where the core challenge lies in securing the bidirectional parameter exchange between the server and UAV nodes, exposing critical vulnerabilities in data interaction [6]. While cryptographic methods are commonly employed, the unique characteristics of federated LLM parameter transmission-divided into distinct upstream (UAV-to-server, involving the upload of local model updates) and downstream (server-to-UAV, involving the broadcast of global model parameters) flows, introduce unaddressed risks absent in conventional data-sharing scenarios, particularly in low-altitude operations where signal obstructions and interference are prevalent. Downstream data, involving the server broadcasting global parameters to UAVs at each round's outset, requires robust one-to-many access control to prevent unauthorized interception or tampering, yet existing mechanisms fail to ensure parameter integrity and confidentiality across dynamic swarm topologies typical of low-altitude environments. Upstream data, comprising locally fine-tuned parameters uploaded by UAVs, faces dual threats: insecure transmission channels risk exposing sensitive model updates to eavesdropping, while the server's direct access to raw parameters compromises privacy by potentially reconstructing proprietary local data. Crucially, the absence of tailored safeguards for upstream/downstream flows leaves low-altitude UAV swarms susceptible to model inversion attacks, parameter hijacking, or adversarial manipulation, undermining the federated learning paradigm's foundational promise of decentralized privacy.

(2) Lack of effective countermeasures to drop-out issues in dynamic UAV swarm networks. Federated fine-tuning of LLM in low-altitude UAV swarm networks faces a critical challenge due to the lack of effective countermeasures against frequent UAV drop-outs [7], a problem inherently tied to the network's dynamic nature and exacerbated by low-altitude operational constraints. The volatile low-altitude operational environmentłcharacterized by unstable connectivity due to urban obstructions, limited battery endurance, susceptibility to cyber-physical attacks, and mission-induced mobilityłleads to unpredictable node departures that disrupt the federated learning process. Unlike conventional federated systems with relatively stable participants, low-altitude UAV swarms experience constant flux in their active node population, causing severe data skew and model bias as disappearing nodes leave their local updates incomplete or unsynchronized. This instability not only degrades the global model's convergence rate but also risks catastrophic forgetting of knowledge from dropped UAVs, particularly when their data distribution differs significantly from surviving nodes. The absence of dropout-resilient aggregation mechanisms further exacerbates the issue, as standard federated averaging fails to account for the lost contributions, leading to suboptimal or even divergent model performance. Moreover, repeated rejoining of previously dropped UAVs-common in low-altitude swarm scenariosłintroduces additional complications, as their stale parameters may contaminate the global model if improperly reintegrated. Without dedicated strategies to mitigate these effects, the reliability and quality of LLM services in such highly dynamic low-altitude networks remain fundamentally compromised.

(3) Conflict between resources-constraints [8] of the UAV onboard processing unit and heavy computational, storage and communication costs of the secure federated LLM. Federated fine-tuning of LLMs for low-altitude UAV swarm networks faces a critical conflict between resource constraints and security requirements, where the computational, storage, and communication costs of secure federated learning directly challenge the limited capabilities of onboard processing units. Unlike conventional federated learning scenarios with stable infrastructure, low-altitude UAVs operate under strict weight and power limitations, forcing them to rely on resource-constrained edge devices that struggle with the inherent overhead of federated LLM tuningłparticularly the iterative upstream (gradient upload) and downstream (model broadcast) transmissions in each training round. The need for cryptographic protection further exacerbates this tension, as traditional encryption methods impose prohibitive latency and energy consumption on already strained UAV hardware, which is particularly critical in low-altitude operations where rapid response is essential. While lightweight cryptography seems essential, tailoring specialized algorithms for federated LLMs introduces additional technical complexity, potentially creating new bottlenecks in computation or memory usage. These compounded resource demands create an inherent tension where the security mechanisms themselves risk becoming the system's bottleneck, potentially undermining the federated learning process through excessive computation delays, communication congestion, or premature energy depletionłall particularly problematic in low-altitude environments where operational endurance is already limited. The situation is further aggravated by the need to maintain training continuity across intermittent wireless links typical in mobile low-altitude UAV operations.

B. Existing solutions and challenges

Limitations of existing solutions. To the best of our knowledge, there currently exists no systematic cryptographic solution for UAV-aided networks that comprehensively addresses the aforementioned challenges in federated LLM fine-tuning, particularly in low-altitude scenarios.

To elaborate plainly on Problem (1), secure downlink data transmission in federated LLMs could theoretically be achieved through identity-based broadcast encryption (IBBE) [9] or attribute-based encryption (ABE) [10] to enforce one-to-many access control. However, IBBE requires explicit recipient specification and generates multiple ciphertexts, inevitably introducing redundant computational and storage overheadłthereby directly exacerbating Problem (3), especially in resource-constrained low-altitude UAVs. Meanwhile, conventional ABE inherently suffers from high computational and storage costs, which may further escalate with attribute complexity, similarly conflicting with the efficiency demands of Problem (3) for low-altitude operations. In the uplink phase of federated LLM, homomorphic encryption (HE) [11] could theoretically enable aggregation over encrypted local parameters from clients. Yet even the most efficient existing additive HE schemes (e.g., Paillier) impose prohibitive computational burdens, again violating the constraints of Problem (3) for low-altitude UAVs with limited processing capabilities.

For Problem (2), prior work has explored Shamir's secret sharing [12] to split global model parameters into shares distributed across clients, ensuring secret recovery from surviving participants despite device dropout, thus mitigating aggregation failures. However, this approach incurs substantial communication overhead, directly contravening Problem (3), particularly problematic in low-altitude environments with unstable connectivity. Moreover, as an independent component targeting Problem (2), Shamir's secret sharing fails to provide any mechanism to address Problem (1). Additionally, existing research fails to address how to properly handle unfinished tasks after UAVs drop out of the system, a common occurrence in low-altitude operations.

Regarding Problem (3), latest cryptographical techniques like asynchronous encryption [13] and edge-assisted decryption [14] have been proposed to alleviate computational costs in IBE/ABE schemes. However, asynchronous encryption merely adopts a "space-for-time" tradeoff without reducing encryption-phase overhead, while edge-assisted decryption, although offloading client computation, requires generating outsourcing keys, transmitting them to edge devices, and processing converted ciphertexts, thereby introducing prohibitive storage and communication costs unsuitable for low-altitude UAV network scenarios. An alternative approach involves protocol-level cryptographic optimizations, but this would inevitably introduce significant algorithmic design complexity.

Potential solutions and challenges. Intuitively, for secure protection in global parameter distribution, identifying an efficient ABE scheme or optimizing existing ABE schemes in terms of computational efficiency could directly address Problem (1) for low-altitude UAV networks. Regarding local parameter uploads, transitioning to more efficient functional encryption (FE) [15] algorithms appears promising, though FE inherently lacks support for computations across multiple ciphertexts. For Problem (2), implementing a covert monitoring mechanism to periodically assess UAV states before each federated LLM fine-tuning round (e.g., through supervised key updates) would enable timely control over UAV participation eligibility, reminiscent of cryptographic key revocation schemesłparticularly valuable in dynamic low-altitude environments. The natural solution for handling tasks from imminent drop-out UAVs lies in integrating proxy re-encryption (PRE) [16] with enhanced ABE schemes. Concerning Problem (3), optimizing the fundamental ABE construction becomes imperative to avoid additional communication and storage overhead, which is critical for low-altitude UAV operations.

However, seamlessly integrating these technologies to construct a secure parameter transmission framework for low-altitude UAV swarm-based federated LLM systems presents significant challenges beyond mere cryptographic primitive combination. First, the algorithmic foundations of PRE and ABE differ substantially, complicating the reconciliation of their distinct public parameters, key structures, and ciphertext formats. Second, key update-based revocation mechanisms impose additional computational and storage burdens on UAVs, necessitating optimized update protocols to minimize overhead, while re-encryption operations further exacerbate UAVs' resource constraintsłparticularly challenging in low-altitude operations where resources are already limited. Third, the open nature of low-altitude UAV networks demands verifiable re-encryption components to prevent spoofed ciphertexts. Finally, FE cannot be out-of-the-box used; instead, it requires integration with the specific fine-tuning algorithms of LLMs to design a dedicated privacy-preserving fine-tuning mechanism based on encrypted parameters that accounts for the unique constraints of low-altitude environments.

C. Our contributions

In this paper we proposed secure and privacy-preserving federated fine-tuning (SPFF) scheme for low-altitude UAV swarm networks, and designed an extended scheme (DE-SPFF) supporting decryption delegation. The contributions is enumerated as follows.

Efficient and privacy-preserving one-to-many global parameter sharing: To achieve multi-party distribution of global parameters in federated LLM fine-tuning for low-altitude UAV swarm networks, we propose an efficient and privacy-preserving sharing mechanism in the SPFF scheme based on the inner-product ABE scheme. In addition to lightweight storage and computational overhead suitable for low-altitude operations, this mechanism supports full policy privacy protection and flexible wildcard matching.

Secure and efficient local updated parameters upload: To enable efficient and privacy-preserving uploading of local parameters in federated LLM fine-tuning for low-altitude UAV swarm networks, the proposed SPFF scheme leverages functional encryption and customizes an efficient ciphertext-based parameter fine-tuning method tailored for the LoRA fine-tuning strategy. In this approach, UAVs only need to

upload ciphertext parameters, while the LLM server only holds the functional key related to model aggregation operations, enabling efficient and privacy-preserving updates of global parameters suitable for low-altitude constraints.

Efficient maintenance of federated LLM system robustness against UAV drop-out: To mitigate the damage caused by uncontrolled UAV drop-out in low-altitude swarm networks to the federated LLM system, SPFF provides an efficient revocation mechanism: UAVs are periodically granted update keys and can renew their own keys at minimal cost. If a UAV fails to update its key and upload local parameters, it is removed from the system, a crucial feature for maintaining system integrity in dynamic low-altitude environments.

Delegable and verifiable federated fine-Tuning tasks: For benign UAVs that are about to drop out, DE-SPFF delegates federated fine-tuning tasks to another UAV through re-encryption. To avoid additional computational and storage overhead-particularly important for resource-constrained low-altitude UAVs, since the re-encryption process is performed by edge nodes, which cannot access any global parameter information. Considering that edge nodes are not fully trusted, DE-SPFF incorporates a verification algorithm that allows third-party verifiers to check the validity of re-encrypted ciphertexts, ensuring security in open low-altitude environments.

We provide formal security proofs under standard cryptographic assumptions, demonstrating that our schemes achieve IND-CPA security and verifiability while covering various threat models including free-riding attacks, collusion attacks, and malicious edge node behaviors; furthermore, through comprehensive theoretical analysis and experimental validation, we demonstrate the superiority of our schemes in computational efficiency, communication overhead, and storage requirements compared to state-of-the-art alternatives, confirming their practicality for real-world low-altitude UAV swarm applications.

D. Related works

Federated LLM is an emerging field in recent two years. The primitives of federated LLM were initially proposed by Chen *et al.* [5], who planned to integrate Federated Learning (FL) into LLM to meet the demand of LLM for large amounts of high-quality data. For this purpose, they gradually integrated FL with the sub-technologies of LLM respectively until the fusion of FL and LLM was attained. Wu *et al.* [17] proposed an optimized federated LLM fine-tuning method and demonstrated that this method exhibits better performance and lower resource consumption, but it does not take security issues into account. Bian *et al.* [37] specifically analyzed the current LoRA method for LLM fine-tuning and provided a specific technical solution for integrating FL with LLM fine-tuning based on its algorithm structure. Das *et al.* [18] proposed that differential privacy and homomorphic encryption mechanisms can prevent gradient privacy leakage in LLMs. Yao *et al.* [19] highlighted that security concerns in federated LLMs primarily revolve around data and models. Their work examines federated LLMs from various perspectives, including Security, Robustness, Privacy, as well as Attack and Defense strategies. Addressing issues such as the lack of training data and training data extraction attacks in privacy-preserving Federated Instruction Tuning (FedIT) [20], Zhang *et al.* [21] introduced a novel federated algorithm named FewFedPIT. This approach is designed to strengthen privacy protection in federated few-shot learning scenarios. To address the differential privacy amplification effect in privacy-preserving federated fine-tuning LLM, Sun *et al.* [22] proposed an improved LoRA method called FFA-LoRA and alleviated the communication overhead in federated fine-tuning LLM.

To date, there has been no specific work demonstrating how to use cryptographic tools to protect data security and privacy in federated LLMs. Due to the similarity in network structure, we first discuss the related work on data security protection in federated learning and LLM here. Considering the computational overhead of homomorphic encryption, differential privacy is a more practical approach. However, differential privacy techniques inevitably introduce a trade-off between data usability and privacy protection. Nguyen *et al.* [23] for the first time provided a systematic overview of data security and privacy issues in FL, explicitly categorizing them into two aspects: “secure gradient aggregation” and “efficient encryption of local gradients”. In other words, they explained the secure transmission for uplink and downlink data (local gradient) in FL. However, most current research focuses on the security and privacy in data uplink phase. Zhang *et al.* [24] implemented a secure gradient aggregation scheme based on the Paillier homomorphic encryption protocol, where the server cannot obtain the gradient of each client but can only get the result of summing all clients’ gradients. Jin *et al.* [25] proposed a practical federated learning system, FedML-HE, for efficient and secure model aggregation based on homomorphic encryption. It selectively encrypts sensitive parameters, significantly reducing computational and communication overhead during training while providing customizable privacy protection. Xu *et al.* [26] provided a complete solution for parameter transmission during the interactive training process of FL models. This solution uses the double-masking protocol to achieve privacy-preserving local gradient upload and enables verification of the aggregation results (global gradients). Additionally, this scheme ensures that no client drops out in each round of model training through the Shamir secret sharing protocol. Yu *et al.* [27] pointed out that clients are usually resource-constrained, and previously proposed FL schemes with verifiable aggregation would impose additional and heavy computational overhead on clients. Then, they proposed an efficient verifiable gradient aggregation method that can support weighted aggregation. Regarding the security of downlink data (global parameters) in FL, Yin *et al.* [28] proposed a client selection scheme based on ABE, where only clients meeting specific requirements can participate in FL.

Specifically for UAV networks, Wang *et al.* [29] proposed a secure federated learning framework for UAV-assisted mobile crowdsourcing. This framework introduces a blockchain-based collaborative learning architecture that enables UAVs to securely exchange local model updates and verify contributions without a central administrator. Building on this, they designed a privacy-preserving algorithm for federated UAV networks

based on a local differential privacy mechanism. Hou *et al.* [30] show that federated learning for UAV networks can infer original data from shared parameters, whereas previous work has focused on encrypting uploaded parameters. They propose a UAV-supported covert federated learning architecture, in which the UAV not only coordinates the federated learning operations but also generates artificial noise to disrupt eavesdropping by non-target users. Xu *et al.* [31] integrates various cryptographic tools, proposed a verifiable and privacy-preserving collaborative federated learning (VPPFL) scheme for UAV-assisted intelligent connected vehicle networks. Technically, the scheme first generates anonymous identifiers for vehicles. To further protect data privacy, vehicle updates are encrypted using the Paillier homomorphic encryption algorithm, allowing UAVs to directly aggregate the encrypted updates globally. Furthermore, a pseudonymous signature mechanism is proposed, enabling vehicles to generate verifiable signatures, thereby ensuring the authenticity and validity of uploaded local model updates.

II. Preliminaries and building blocks

A. Hardness assumption

Consider two prime-order multiplicative cyclic groups, denoted as $\mathbb{G}_0$ and $\mathbb{G}_1$, with g serving as a generator for $\mathbb{G}_1$. Random elements a, b, and c are selected uniformly from the finite field $\mathbb{Z}_p$. We define a bilinear pairing operation $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$. For the input tuple $(g, g^a, g^b, g^c, \mathcal{Z}) \in \mathbb{G}_0^4 \times \mathbb{G}_1$, determining whether $\mathcal{Z}$ equals $e(g,g)^{abc}$ or represents a random group element in $\mathbb{G}_1$ constitutes the computationally hard Decisional Bilinear Diffie-Hellman (DBDH) challenge.

Definition 1: *The DBDH assumption holds if there exists no probabilistic polynomial-time algorithm capable of solving the DBDH problem with a non-negligible probability.*

Consider an elliptic curve $\mathbb{E}$ defined over a finite field $\mathbb{F}_q$, where $\mathbb{G}$ denotes a base point generating a cyclic subgroup of prime order p. Given two points P and $Q = kP$ on the curve, where k is randomly selected from $\mathbb{Z}_p$, the Elliptic Curve Discrete Logarithm Problem (ECDLP) requires determining the integer k.

Definition 2: *The ECDLP assumption holds if no probabilistic polynomial-time (PPT) adversary can compute $k \in \mathbb{Z}_p$ from $(\mathbb{E}, \mathbb{F}_q, \mathbb{G}, p, P, Q = kP)$ with non-negligible advantage.*

B. Functional encryption supporting basic operations

Functional encryption supporting basic operations (FEBO) was proposed by Xu *et al.* [32] to facilitate encrypted matrix operations on convolutional neural networks in conjunction with Inner-product Functional Encryption. FEBO supports four fundamental arithmetic operations: addition, subtraction, multiplication, and division. The specific definitions of each algorithm are as follows.

• **Setup**(1^κ): Given security parameter κ, outputs master key pair (PK, SK). First generates group parameters $(\mathcal{G}, q, \gamma) \leftarrow \mathsf{GroupGen}(1^\kappa)$ and randomly selects $\alpha \in \mathbb{Z}_q$. Sets $SK = \alpha$ and $PK = (\eta, \gamma)$, where $\eta = \gamma^\alpha$.

• **KeyDerive**$(PK, SK, \sigma, \nabla, z)$: Takes master secret SK, commitment σ, and function input z, outputs functional key sk_{f_∇}. The derivation follows:

$$sk_{f_\nabla} = \begin{cases} \sigma^\alpha \cdot \gamma^{-z} & \text{if } \nabla = + \text{ (addition)} \\ \sigma^\alpha \cdot \gamma^{z} & \text{if } \nabla = - \text{ (subtraction)} \\ (\sigma^\alpha)^z & \text{if } \nabla = \times \text{ (multiplication)} \\ (\sigma^\alpha)^{z^{-1}} & \text{if } \nabla = / \text{ (division)} \end{cases}$$

• **Encrypt**$(PK,\ w)$: Using public key PK and input w, outputs commitment σ and ciphertext ψ. Randomly select $\rho \in \mathbb{Z}_q$, computes $\sigma = \gamma^\rho$, and $\psi = \eta^\rho \cdot \gamma^w$.

• **Decrypt**$(PK, \kappa_{f_\nabla}, \psi, \nabla, z)$: Given public key PK, functional key sk_{f_∇} for operation f_∇ with input z, and ciphertext ψ, recovers:

$$\gamma^{f_\nabla(w,z)} = \begin{cases} \psi/\kappa & \text{for } \nabla \in \{+, -\} \\ \psi^z/\kappa & \text{for } \nabla = \times \\ \psi^{z^{-1}}/\kappa & \text{for } \nabla = / \end{cases}$$

C. Attributes and access policy vectorization

Our scheme implements privacy-preserving attribute-based access control via inner-product operations between policy and attribute vectors. **Algorithm 1** technically realizes the transformation of a symbolic access policy (comprising wildcards "*", positive "+", and negative "-" attributes) into computable numerical vectors. It maps the policy and user attributes to a policy vector v and an attribute vector u, respectively, such that policy satisfaction is equivalent to their inner product being zero: $\langle \mathsf{u}, \mathsf{v} \rangle = 0$. This is achieved by expanding a polynomial constructed from wildcard positions to generate the coefficients of v, while aggregating exponentiated valid attribute positions to form u. The algorithm thus converts semantic access conditions into an algebraic form suitable for inner-product-based cryptographic operations, enabling efficient and privacy-preserving access control.

D. Key update nodes algorithm

The presented cryptographic constructions achieve efficient user revocation through the key update nodes ($\mathsf{KUNodes}$) algorithm [35]. **Algorithm 2** details this procedure, beginning with a binary tree $\mathcal{T}$ containing n leaves and system state σ. For any internal node x, its children are denoted x_L (left) and x_R (right). Each legitimate user is uniquely associated with a leaf node λ, with $\mathcal{P}(\lambda)$ representing all nodes from root to λ. The algorithm processes a revocation list $\mathcal{RL}$ and current time interval τ, initially storing all $\mathcal{P}(\lambda)$ nodes in set $\mathcal{X}$ for each revoked user λ. Subsequently, it constructs set $\mathcal{Y}$ by including sibling nodes of $\mathcal{X}$ members not already in $\mathcal{X}$. The final output $\mathcal{Y} = \mathsf{KUNodes}(\mathcal{RL}, \sigma, \tau)$ identifies the minimal nodes requiring key updates.

For active users, key updates occur for nodes in $\mathcal{P}(\lambda) \cap \mathsf{KUNodes}(\mathcal{RL}, \sigma, \tau)$, while revoked users satisfy $\mathcal{P}(\lambda) \cap \mathsf{KUNodes}(\mathcal{RL}, \sigma, \tau) = \varnothing$.

Algorithm 1 Policy-Attribute Vector Conversion

Input: Access structure containing ℓ wildcards ("*"), ℓ_+ positive ("+") and ℓ_- negative ("-") attributes; Attribute set $\mathcal{A} = \{\mathcal{A}_1, \ldots, \mathcal{A}_N\}$ where $\mathcal{A}_i \in \{\text{“}+\text{”}, \text{“}-\text{”}\}$ for $i \in \{1, \ldots, N\}$.

Output: Policy vector v and attribute vector u.

1) Partition positive attributes and wildcards into position sets $\mathcal{P}$ and $\mathcal{W}$ respectively
2) For each wildcard position $w \in \mathcal{W}$:
   - Expand $\prod_{w \in \mathcal{W}}(i - w) = \sum_{j=0}^{n} c_j i^j$ to obtain coefficients c_j
3) For each $w \in \mathcal{W}$ and $i \in \mathcal{P}$:
   - Compute $\prod_{w \in \mathcal{W}}(i - w)$ and aggregate results
4) Extract positive attribute positions $\mathcal{P}' \subseteq \mathcal{P}$
5) For $i \in \mathcal{P}'$ and $j = 0$ to ℓ:
   - Calculate $u_j = \sum_{i \in \mathcal{P}'} i^j$
6) Construct vectors:
   - $\mathsf{u} = (u_0, u_1, \ldots, u_\ell)$
   - $\mathsf{v} = (c_0, c_1, \ldots, c_n, 0, \ldots, 0)$ (zero-padded to length $\ell + 1$)

Algorithm 2 Key Update Nodes Selection

Input: Binary tree $\mathcal{T}$, revocation list $\mathcal{RL}$, time τ

Output: Node set $\mathcal{Y}$

```
Initialize X ← ∅, Y ← ∅
for each (λ_j, τ_j) ∈ RL do
    if τ_j ≤ τ then
        X ← X ∪ P(λ_j)
for each x ∈ X do
    if x_L ∉ X then
        Y ← Y ∪ {x_L}
    if x_R ∉ X then
        Y ← Y ∪ {x_R}
if RL = ∅ then
    Y ← Y ∪ {root}
return Y
```

E. Message-lock encryption

Message Lock Encryption (MLE) [38] is a cryptographic technique designed to securely encrypt messages based on their unique content. Unlike traditional encryption methods that rely on external keys, MLE derives the encryption key directly from the message itself or a hash of the message. This ensures that identical messages produce the same ciphertext, enabling efficient deduplication in storage systems while maintaining confidentiality. MLE is particularly useful in cloud storage and secure messaging systems, where reducing redundant data is critical without compromising security.

A key feature of MLE is its ability to provide convergent encryption, where the same plaintext always encrypts to the same ciphertext under the same conditions. This approach eliminates the need for key management in certain scenarios but requires careful handling to prevent vulnerabilities, such as brute-force attacks on predictable messages. MLE is often used in applications like secure file sharing or encrypted backup solutions, where both data privacy and storage efficiency are prioritized. However, additional safeguards, like salting or key wrapping, may be needed to enhance security.

F. Federated LLM tuning with LoRA

LoRA-based Fine-tuning: The Low-Rank Adaptation (LoRA) method [36] approximates parameter updates via low-rank decomposition:

$$\mathbf{W}_{\text{new}} = \mathbf{W} + \Delta\mathbf{W} = \mathbf{W} + \mathbf{PQ}$$

where $\mathbf{W} \in \mathbb{R}^{m \times n}$ represents original pre-trained weights (e.g., in attention layers) and $\mathbf{W}_{\text{new}}$ denotes the adapted version. The update is factorized into $\mathbf{P} \in \mathbb{R}^{m \times r}$ and $\mathbf{Q} \in \mathbb{R}^{r \times n}$, with rank $r \ll \min(m, n)$. For the Bert-base model with weight matrix $\mathbf{W} \in \mathbb{R}^{768 \times 768}$, we adopt a low-rank adaptation (LoRA) method [39] by setting the rank $r = 8$, which enables memory-efficient fine-tuning.

FedIT: Federated Homogeneous LoRA: **FedIT** integrates FedAvg and LoRA to provide parameters aggregation in federated LLMs. FedAvg computes global updates through weighted aggregation:

$$\mathbf{W}^{(t+1)} = \mathbf{W}^{(t)} + \sum_{k=1}^{K} p_k \Delta\mathbf{W}_k^{(t)} = \mathbf{W}^{(t)} + \Delta\mathbf{W},$$

where p_k represents client weighting factors.

FedIT [37] applies FedAvg specifically to Homogeneous LoRA modules. Clients locally fine-tune $\mathbf{P}_k$ and $\mathbf{Q}_k$ before the server aggregates, where $p_k = \frac{|\mathcal{D}_k|}{\sum_{k=1}^{K} |\mathcal{D}_k|}$, $|\mathcal{D}_k|$ denotes the size of a local data set:

$$\mathbf{P} = \sum_{k=1}^{K} p_k \mathbf{P}_k, \quad \mathbf{Q} = \sum_{k=1}^{K} p_k \mathbf{Q}_k.$$

This approach introduces approximation errors because:

$$\sum_{k=1}^{K} p_k (\mathbf{P}_k \mathbf{Q}_k) \neq \left(\sum_{k=1}^{K} p_k \mathbf{P}_k \right) \left(\sum_{k=1}^{K} p_k \mathbf{Q}_k \right).$$

For two clients with updates $\Delta\mathbf{W}_0 = \mathbf{P}_0\mathbf{Q}_0$ and $\Delta\mathbf{W}_1 = \mathbf{P}_1\mathbf{Q}_1$, the aggregated update deviates from the true average by cross-client interaction terms.

III. System Architecture and Definitions

A. System model and threat model

The proposed technical solution in this paper can be abstracted as the system model shown in Fig. 1, which involves four types of entities: the Trusted Authority, Unmanned Aerial Vehicles (UAVs), the Large Language Model Server (LLM Server) and the Verifier. The extended solution DE-SPFF additionally involves edge nodes. These entities are described as follows:

- **Trusted Authority**: It is a fully trusted entity responsible for creating the security system, issuing parameters, and UAV secret keys. Additionally, it assists and supervises

UAVs in completing key updates and maintains a revocation list.

- **LLM Server**: It is a semi-trusted (honest-but-curious) entity responsible for establishing the initial LLM and providing UAVs with the initial global parameters. It also periodically collects local parameters from UAVs to fine-tune the global model and returns the final global parameters to the UAVs.
- **UAV**: It is an untrusted entity that receives initial and fine-tuned global parameters from the LLM Server, performs local model training, and sends local parameters to the LLM Server in each round. When the global LLM completes fine-tuning, the UAV obtains the final global parameters and enjoys the LLM service. In the extended scheme DE-SPFF, it also needs to issue re-encryption keys to edge nodes. A UAV may also act as a Delegatee, receiving a re-encrypted ciphertext from an edge node and decrypting it to obtain the global parameters on behalf of a delegator UAV that has dropped out. We assume the Delegatee UAV is also untrusted and is considered a potential adversary that might deviate from the protocol or collude with a compromised edge node to illegally access the global model parameters.
- **Edge Node**: It only appears in the DE-SPFF scheme and is a semi-trusted entity. When a UAV has task delegation requirements, it receives re-encryption keys from the UAV and performs re-encryption operations.
- **Verifier**: It is a new entity introduced in the DE-SPFF scheme to ensure accountability in the open swarm network. The Verifier is a third-party entity that is not trusted with any secret keys or sensitive data. Its sole function is to publicly verify the correctness of re-encryption operations performed by edge nodes, ensuring they have been executed faithfully without tampering.

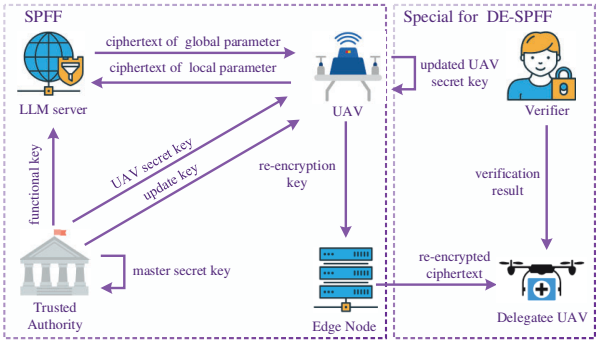


Fig. 1: System model

The data interactions among entities in this paper are shown in Fig. 2. First, the Trusted Authority (TA) runs the **Setup** algorithm to generate the system public parameters params and master secret key msk, then broadcasts params to the system. Subsequently, the TA executes the **SKGen** algorithm to generate a UAV secret key usk for each UAV and function key sk_f for the LLM Server. The LLM Server then executes the **Enc** algorithm to generate ciphertext CT_0 for the initial global LLM parameters $\mathbf{W}_0$. Each UAV runs the **Dec-initial**

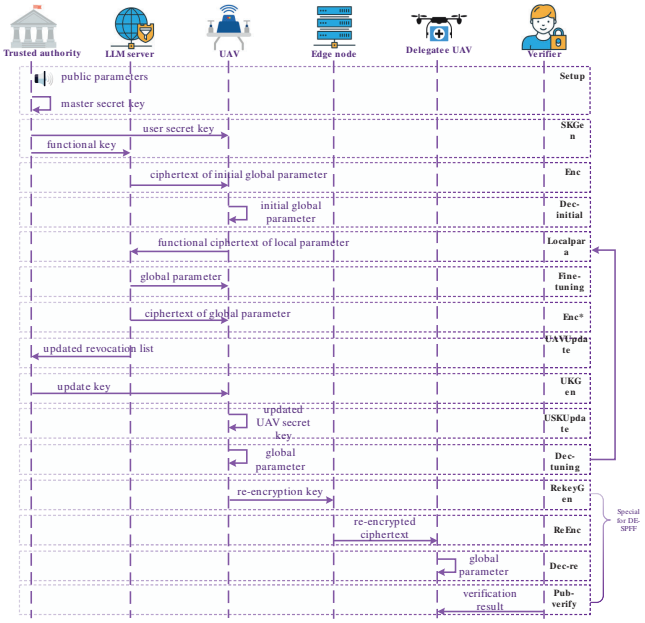


Fig. 2: The interactive workflow

algorithm to decrypt CT_0 using its usk and obtains $\mathbf{W}_0$, upon which it performs local model fine-tuning. The UAV then encrypts and returns its local model parameters $Enc(\mathbf{P}_k)$ and $Enc(\mathbf{Q}_k)$ to the LLM Server via the **Localpara** algorithm. The LLM Server executes the **Fine-tuning** algorithm to aggregate and decrypt the local parameters from all UAVs in the previous round, using them to fine-tune the global model and obtain updated global parameters $\mathbf{W}$. Subsequently, the LLM Server runs the **Enc*** algorithm to generate ciphertext CT for the updated global parameters $\mathbf{W}$ and sends it to the UAVs. Independently, based on UAVs' participation in the previous round of fine-tuning, the LLM Server requests the TA to update the revocation list RL (via the **UAVupdate** algorithm). According to RL, the TA periodically executes the **UKGen** algorithm to issue update keys for UAVs not in RL. If a UAV receives an update key, it runs the **USKUpdate** algorithm to update its usk. Finally, the UAV executes the **Dec-tuning** algorithm to decrypt the global parameters $\mathbf{W}$. These algorithms execute iteratively until the LLM completes fine-tuning, at which point UAVs obtain the final global parameters. Note that in the DE-SPFF scheme, we additionally allow UAVs that cannot continue participating in fine-tuning to delegate their tasks to other UAVs: First, the delegator UAV executes the **RekeyGen** algorithm to generate a re-encryption key rk and sends it to an edge node. The edge node then runs the **ReEnc** algorithm to produce a re-encrypted ciphertext RCT, which is sent to the delegatee UAV. The delegatee UAV can execute the **Dec-re** algorithm to decrypt and obtain the global parameters $\mathbf{W}$. Finally, a third-party verifier can determine the relationship between RCT and CT by executing the **Pub-verify** algorithm.

This system also faces the following potential security threats stemming from its open and decentralized nature: First-

ly, unauthorized UAVs or external adversaries might engage in "free-riding" attacks. These malicious entities aim to illicitly access and exfiltrate the final fine-tuned global parameters without actively participating in the federated learning process or possessing the proper authorization attributes. This undermines the intellectual property and collaborative incentive of the system. Secondly, we consider a sophisticated collusion attack: an edge node, once compromised, could conspire with a malicious delegatee UAV. Their objective is to pool their knowledgełthe edge node's view of the re-encryption process and the delegatee's partial secretsłto decrypt and obtain the global parameters in violation of the access policy dictated by the LLM server. Finally, a malicious or infiltrated edge node could attempt to sabotage the federated learning process by sending deliberately falsified or incorrectly computed re-encrypted ciphertexts to a benign delegatee UAV. This Denial-of-Service (DoS)-style attack aims to prevent the delegatee from correctly obtaining the global parameters, thereby disrupting its local training and compromising the overall robustness of the federated tuning process.

B. Security model

The selective indistinguishability of DE-SPFF under chosen-plaintext attacks is formalized through an interactive game between adversary $\mathcal{A}$ and challenger $\mathcal{C}$:

1. **Initialization**: $\mathcal{A}$ selects target access policy $\mathbb{A}^*$ and challenge timestamp t^*, sending them to $\mathcal{C}$.
2. **Setup**: $\mathcal{C}$ generates public parameters pp using Setup and provides them to $\mathcal{A}$.
3. **Query Phases**: $\mathcal{A}$ can adaptively request:
   - UAV secret keys via SKGen
   - Key updates via UKGen
   - Revocation status via $\mathsf{UAVupdate}$
4. **Challenge**: $\mathcal{A}$ submits messages $\varepsilon_0, \varepsilon_1$. $\mathcal{C}$ randomly chooses $b \leftarrow \{0,1\}$, computes $\mathsf{CT}_b \leftarrow \mathsf{Enc}(\varepsilon_b)$, and returns CT_b.
5. **Guess**: $\mathcal{A}$ outputs guess b'. The advantage is defined as $|\Pr[b'=b] - \frac{1}{2}|$.

Definition 2 (IND-CPA Security): *DE-SPFF achieves IND-CPA security if all PPT $\mathcal{A}$ have negligible advantage $|\Pr[b'=b] - \frac{1}{2}|$, and any such adversary would imply a DBDH problem solver with non-negligible advantage.*

For verifiability, consider the following game:

1. **Initialization**: $\mathcal{A}$ specifies target policy $\mathbb{A}^*$ to $\mathcal{C}$.
2. **Setup**: $\mathcal{C}$ runs Setup and gives pp to $\mathcal{A}$.
3. **Queries**: $\mathcal{A}$ can query:
   - Secret keys via SKGen
   - Re-encryption keys via RKGen
   - Verification results via $\mathsf{Pub\text{-}verify}$
4. **Challenge**: $\mathcal{A}$ requests encryption of ε^* under $\mathbb{A}^*$. $\mathcal{C}$ returns original ciphertext CT^* and re-encrypted $\mathsf{CT}^{*'}$.
5. **Guess**: $\mathcal{A}$ wins if both ciphertexts verify correctly under $\mathsf{Pub\text{-}verify}$.

Definition 3 (Verifiability): *DE-SPFF provides verifiability if no PPT $\mathcal{A}$ can win $\mathsf{Game}_{\mathrm{Verifiability}}$ with non-negligible probability, unless the ECDLP assumption can be broken with comparable advantage.*

IV. Secure and privacy-preserving federated fine-tuning scheme

This section will describe the construction of the secure and privacy-preserving federated fine-tuning (SPFF) scheme for UAV swarm networks. The secure sharing of global parameters is based on Bao *et al.*'s PH-ABE-DS scheme [40], while the process of UAVs returning locally trained parameters in ciphertext that participates in global LLM tuning design follows Xu *et al.*'s FEBO scheme [32]. The proposed scheme consists of three phases: The System initialization phase generates system parameters and distributes keys to entities. The Initial global parameter distribution phase details how the LLM server securely delivers initial global parameters to UAVs. The federated fine-tuning of LLM stage, the core of this scheme, orchestrates UAVs' local model training, secure parameter uploading, and iterative global model refinement until completion. Notably, to address frequent UAV dropout issues in open swarm networks, the scheme incorporates key update and UAV revocation mechanisms to minimize disruption to federated fine-tuning.

A. System initialization

• **Setup** (λ): The trusted authority defines and generates system public parameters and a master key by executing the following steps, which are essential for subsequent procedures.

1). Take the security parameter λ as input, it generates the tuple $\langle \mathbb{G}, \mathbb{G}_T, g, e \rangle$, where $\mathbb{G}, \mathbb{G}_T$ two multiplicative cyclic groups with the prime order p, g is a generator of the group $\mathbb{G}$, and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$.

2). Assume the size of the attribute universe to be n, it randomly selects $\alpha, \tau_1, ..., \tau_n \in \mathbb{Z}_p$, then computes $g_i = g^{\tau_i}$ and $Z = e(g,g)^{\alpha}$, where $i \in [1, n]$.

3). Define an empty revocation list $\mathsf{RL} = \varnothing$ for storing all revoked data users. Besides, initialize a binary tree $\mathbb{BT}$ with q leaf nodes and the state st.

4). Initialize a l-bit timestamp t, then randomly choose $\varrho_0 \in \mathbb{Z}_p$ and $\varrho_1, ..., \varrho_l \in \mathbb{Z}_p$ for each bit of t.

5). Execute the **FEBO.Setup** algorithm to obtain the public parameters mpk and secret key msk for functional encryption.

6). The trusted authority outputs the public parameter $\mathsf{params} = \{\mathbb{G}, \mathbb{G}_T, p, e, g, \{g_i\}_{i\in[1,n]}, Z, \varrho_0, \{\varrho_j\}_{j\in[1,l]}, t, mpk\}$ as well as the binary tree $\mathbb{BT}$ with state st, and the revocation list RL, then secretly holds the master secret key $\mathsf{msk} = \{\alpha, \{\tau_i\}_{i\in[1,n]}, msk\}$.

• **SKGen** $(\mathsf{params}, \mathsf{msk}, S, \mathbb{BT}, \mathsf{st})$: The trusted authority takes the public parameter params, the master secret key msk, the attribute set S of a UAV, the binary tree $\mathbb{BT}$ with state st as input, then generates the initial UAV secret key usk as following steps:

1). Convert the attribute set S to an attribute vector $\mathsf{u} = (u_1, ..., u_n)$ by executing **Algorithm 1**.

2). It first stores the identifier id of a UAV into an undefined leaf node θ_{id} of the binary $\mathbb{BT}$. Then according to the **Algorithm 2**, for each node $x \in \mathsf{Path}(\theta_{id})$, it fetches g_x from the node x if available. Otherwise, if x is undefined,

it randomly selects $g_x \in \mathbb{G}$, then stores g_x into the node x and update the state $\mathsf{st} \leftarrow \mathsf{st} \cup \{x, g_x\}$.

3). Randomly choose $s \in Z_p$, then computes $g'_x = g^{\alpha}/g_x$, $sk_1 = g'_x g^{s \sum_{i=1}^{n} \tau_i u_i}$ and $sk_2 = g^s$.

4). It runs **FEBO.KeyDerive** algorithm to obtain the functional key sk_f for the multiplication operation f.

5). The trusted authority outputs the UAV secret key $\mathsf{usk} = (id, \mathsf{u}, \{sk_1, sk_2\}_{x \in \mathsf{Path}(\theta_{id})})$ and delivers it to the UAV via secure channel. Also, it secretly sends the functional key sk_f to the LLM server.

B. Initial global parameter distribution

In this phase, the LLM server establishes a globally initialized LLM Model_{global}, while each UAV loads an initial local model Model_{UAV}. The LLM server implements the secure distribution of initial global parameters according to the following algorithm.

• **Enc** $(\mathsf{params}, \mathbb{A}, \mathbf{W}_0, t)$: The LLM server takes the public parameters params, the access policy $\mathbb{A}$ and the timestamp t as input, then generates the ciphertext for the initial parameter $\mathbf{W}_0$ by following the steps below.

1). It first creates the access vector $\mathsf{v} = (v_1, ..., v_n)$ according to the access policy $\mathbb{A}$ by invoking the **Algorithm 1**, parses $\mathbf{W}_0$ as a bit string ε_0, then randomly chooses $\delta, \theta \in \mathbb{Z}_p$ and computes $c = \varepsilon_0 \cdot Z^{\delta}$, $\gamma_i = \eta_i + v_i\theta$, where $i \in [1, n]$.

2). For each attribute $i \in [1, n]$, it randomly selects $\{\eta_i\}_{i\in[1,n]} \in \mathbb{Z}_p^{n+2}$, and calculate $c_0 = g^{\delta}, c_1 = g_1^{\delta} g^{-\eta_1}, ..., c_n = g_n^{\delta} g^{-\eta_n}$. And for each bit $j \in [1, l]$ of the timestamp t, it computes $\hat{c}_0 = \varrho_0^{\delta}$, $\hat{c}_j = \varrho_j^{\delta}$.

3). The LLM server finally outputs the ciphertext $\mathsf{CT}_0 = (c, c_0, \hat{c}_0, \{c_i, \gamma_i\}_{i\in[1,n]}, \{\hat{c}_j\}_{\ j\in[1,l]})$.

• **Dec-initial** $(\mathsf{params}, \mathsf{CT}, \mathsf{usk})$: On input the tuple the ciphertext CT and the UAV secret key usk, this algorithm calculates $c'_0 = g^{\sum_{i=1}^{n} \gamma_i u_i} \prod_{i=1}^{n} c_i^{u_i}$, then obtain ε_0 and reconstructs the initial parameter matrix $\mathbf{W}_0$ of the global model by computing $\varepsilon_0 = c \cdot e(c'_0, sk_2)/e(c_0, sk_1)$.

C. Federated fine-tuning of LLM

• **Localpara** $(\mathbf{W}_0, \mathsf{Model}_{UAV})$: After obtaining the initial global parameters $\mathbf{W}_0$, the k-th UAV continues to train its local model Model_{UAV}, resulting in local parameters $\mathbf{P}_k$, $\mathbf{Q}_k$. The UAV runs **FEBO.Encrypt** algorithm over $\mathbf{P}_k$, $\mathbf{Q}_k$ to obtain $Enc(\mathbf{Q}_k)$ and $Enc(\mathbf{P}_k)$, then forwards them to the LLM server.

• **Fine-tuning** $(\{Enc(\mathbf{P}_k)\}, \{Enc(\mathbf{Q}_k)\}, sk_f, \mathsf{Model}_{global})$: The LLM server can obtain the update parameter $\Delta\mathbf{W}$ by inputting the encrypted local parameters $\mathbf{P}_k$, $\mathbf{Q}_k$, then invokes **FEBO.Decrypt** algorithm and executes the following functional decryption, where p_k denotes the corresponding scaling factor [36].

$$\begin{aligned}\Delta\mathbf{W} = \mathbf{Q} \cdot \mathbf{P} &= \sum_{k=0}^{K} p_k \mathbf{Q}_k \cdot \sum_{k=0}^{K} p_k \mathbf{P}_k \\ &= \sum_{k=0}^{K} sk_f(p_k) Enc(\mathbf{Q}_k) \cdot \sum_{k=0}^{K} sk_f(p_k) Enc(\mathbf{P}_k) \\ &= \sum_{k=0}^{K} sk_f(p_k) Enc(\mathbf{Q}_k) \cdot \sum_{k=0}^{K} sk_f(p_k) Enc(\mathbf{P}_k)\end{aligned}$$

The LLM server then updates the global parameter by $\mathbf{W} \leftarrow \mathbf{W} + \Delta\mathbf{W}$.

• **Enc*** $(\mathsf{params}, \mathsf{CT}', \mathbb{A}, \mathbf{W}, t)$: LLM server takes the updated global parameter $\mathbf{W}$, current timestamp t and the access policy $\mathbb{A}$ as input, then generates the ciphertext of $\mathbf{W}$ with the following steps:

1). It first creates the access vector $\mathsf{v} = (v_1, ..., v_n)$ according to the access policy $\mathbb{A}$ by invoking the **Algorithm 1**, parses $\mathbf{W}_0$ as a bit string ε_0, then randomly chooses $\delta, \theta \in \mathbb{Z}_p$ and computes $c = \varepsilon_0 \cdot Z^{\delta}$, $\gamma_i = \eta_i + v_i\theta$, where $i \in [1, n]$.

2). For each attribute $i \in [1, n]$, it randomly selects $\{\eta_i\}_{i\in[1,n]} \in \mathbb{Z}_p^{n+2}$, and calculate $c_0 = g^{\delta}, c_1 = g_1^{\delta} g^{-\eta_1}, ..., c_n = g_n^{\delta} g^{-\eta_n}$. And for each bit $j \in [1, l]$ of the timestamp t, it computes $\hat{c}_0 = \varrho_0^{\delta}$, $\hat{c}_j = \varrho_j^{\delta}$.

3). The LLM server finally outputs the ciphertext $\mathsf{CT}_0 = (c, c_0, \hat{c}_0, \{c_i, \gamma_i\}_{i\in[1,n]}, \{\hat{c}_j\}_{\ j\in[1,l]})$.

• **UAVupdate** $(id^*, t, \mathsf{RL}, \mathbf{st})$: The LLM server determines the UAV identifiers id^* that need to be revoked based on the drop-out status and behaviors (such as sending deceptive parameters, being compromised, etc.) of UAVs during the previous round of fine-tuning in the federated LLM. The trusted authority obtains id^* from the LLM server, then takes the timestamp t, the revocation list RL with state st of the full binary tree $\mathbb{BT}$ as input, updates the revocation list by operating $\mathsf{RL} \leftarrow \mathsf{RL} \cup (x, t)$ for all nodes $x \in \mathsf{Path}(\theta_{id^*})$.

• **UKGen** $(\mathsf{RL}, \mathsf{params}, \mathsf{st}, t)$: The trusted authority takes the public parameter params, the revocation list RL with state st and the current timestamp t as input, then for each $x \in \mathsf{KUNodes}(\mathsf{RL}, \mathsf{st}, t)$, it randomly chooses $\pi \in \mathbb{Z}_p$, and calculates

$$uk_1 = g_x \cdot (\varrho_0 \prod\nolimits_{j\in I} \varrho_j)^{\pi}, uk_2 = g^{\pi}$$

The trusted authority outputs the update key $\mathsf{uk} = (t, \{uk_1, uk_2\}_{\ x\in\mathsf{KUNodes}(\mathsf{RL},\mathsf{st},t)})$ and broadcast it to the UAV network.

• **USKUpdate** $(\mathsf{params}, \mathsf{usk}, \mathsf{uk}_t, t)$: After receiving the updated key, an UAV that has not drop-out executes the following steps to update its own secret key. If the UAV fails to decrypt with the updated UAV secret key, it implies that the UAV was unable to obtain the updated key due to drop-out and thus cannot update its own secret key, or it has voluntarily relinquished its permission to participate in the federated LLM system. Specifically, the UAV takes the public parameter params, the UAV secret key usk, the update key uk as well as the timestamp t as input, then for each node $x \in \mathsf{Path}(\theta_{id}) \cap \mathsf{KUNodes}(\mathsf{RL}, \mathsf{st}, t)$, it updates the UAV secret key as follows

$$\begin{aligned}&sk_1 \leftarrow sk_1 \cdot uk_1 = g^{\alpha} g^{(s+\pi)\sum_{i=1}^{n} \tau_i u_i} \cdot (\varrho_0 \prod\nolimits_{j\in I} \varrho_j)^{\pi} \\ &sk_2 \leftarrow sk_2 \cdot uk_2 = g^{s+\pi} \\ &sk_3 = uk_2 = g^{\pi}\end{aligned}$$

For each $x \in \mathsf{Path}(\theta_{id}) \cap \mathsf{KUNodes}(\mathsf{RL}, \mathsf{st}, t)$, this algorithm outputs the (updated) UAV secret key $\mathsf{usk} = ((id, t), \mathsf{u}, \{sk_1, sk_2, sk_3\})$.

• **Dec-tuning** $(\mathsf{CT}, \mathsf{usk})$: UAV takes the ciphertext CT and the (updated) UAV secret key usk as input, then calculates $c_0' = g^{\sum_{i=1}^{n} \gamma_i u_i} \prod_{i=1}^{n} c_i^{u_i}$ and obtains the global parameter $\mathbf{W}$ by computing $\varepsilon = c \cdot e(c_0', sk_2) e(c', sk_3) / e(c_0, sk_1)$. Now, the UAV can reconstruct and load the current LLM global parameters $\mathbf{W}$ and iterate through the **Localpara** and **Fine-tuning** algorithms until the fine-tuning of the LLM global model Model_{global} is completed.

V. A DELEGABLE EXTENSIONAL SCHEME

Building upon the proposed SPFF scheme, this section presents a more flexible and scalable scheme named DE-SPFF (delegated extensional SPFF) to address the UAV dropout issue in federated fine-tuning of large models. The scheme incorporates the concept of proxy re-encryption, enabling a UAV to securely delegate its tasks to another qualified UAV when it can no longer participate in the federated fine-tuning process. This solution designates edge nodes to perform the re-encryption computations. Considering the open nature of UAV swarm networks and the partial trustworthiness of edge nodes, the scheme provides a publicly verifiable mechanism that allows third parties to validate the correctness of re-encryption operations. The specific algorithmic construction of the DE-SPFF scheme is as follows.

A. System initialization

• **Setup** (λ): The trusted authority defines and generates system public parameters and a master key by executing the following steps, which are essential for subsequent procedures.

1). Take the security parameter λ as input, it generates the tuple $\langle \mathbb{G}, \mathbb{G}_T, g, e \rangle$, where $\mathbb{G}, \mathbb{G}_T$ two multiplicative cyclic groups with the prime order p, g is a generator of the group $\mathbb{G}$, and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$.

2). Assume the size of the attribute/access universe to be n, it randomly selects $\alpha, \tau_1, ..., \tau_n \in \mathbb{Z}_p$, $w, x, y \in \mathbb{G}$, then calculates $g_i = g^{\tau_i}$ and $Z = e(g, g)^{\alpha}$, where $i \in [1, n]$.

3). Define an empty revocation list $\mathsf{RL} = \varnothing$ for storing all revoked data users. Besides, initialize a binary tree $\mathbb{BT}$ with q leaf nodes and the state st.

4). Initialize a l-bit timestamp t, then randomly choose $\varrho_0 \in \mathbb{Z}_p$ and $\varrho_1, ..., \varrho_l \in \mathbb{Z}_p$ for each bit of t.

5). Define two collision-resistant hash functions $\mathsf{H}_1 : \mathbb{G}_T \to \mathbb{G}$ and $\mathsf{H}_2 : \{0,1\}^* \to \mathbb{Z}_p^*$.

6). Execute the **FEBO.Setup** algorithm to obtain the public parameters mpk and secret key msk for functional encryption. Also, predefine a message lock encryption algorithm MLE [42].

7). The trusted authority outputs the public parameter $\mathsf{params} = \{\mathbb{G}, \mathbb{G}_T, p, e, g, x, y, w, \{g_i\}_{i\in[1,n]}, Z, \mathsf{H}_1, \mathsf{H}_2, \varrho_0, \{\varrho_j\}_{j\in[1,l]}, t, mpk, \mathsf{MLE}\}$ as well as the binary tree $\mathbb{BT}$ with state st, and the revocation list RL, then secretly holds the master secret key $\mathsf{msk} = \{\alpha, \{\tau_i\}_{i\in[1,n]}, msk\}$.

• **SKGen** $(\mathsf{params}, \mathsf{msk}, S, \mathbb{BT}, \mathsf{st}, \mathsf{upk})$: The trusted authority takes the public parameter params, the master secret key msk, the attribute set S of a UAV, the binary tree $\mathbb{BT}$ with state st as input, then generates the initial UAV secret key usk as following steps:

1). Convert the attribute set S to an attribute vector $\mathsf{u} = (u_1, ..., u_n)$ by executing **Algorithm 1**.

2). It first stores the identifier id of a UAV into an undefined leaf node θ_{id} of the binary $\mathbb{BT}$. Then according to the **Algorithm 2**, for each node $x \in \mathsf{Path}(\theta_{id})$, it fetches g_x from the node x if available. Otherwise, if x is undefined, it randomly selects $g_x \in \mathbb{G}$, then stores g_x into the node x and update the state $\mathsf{st} \leftarrow \mathsf{st} \cup \{x, g_x\}$.

3). Randomly choose $s \in Z_p$, then computes $sk_1 = g_x' g^s \sum_{i=1}^{n} \tau_i u_i$ and $sk_2 = g^s$.

4). It runs **FEBO.KeyDerive** algorithm to obtain the functional key sk_f for the multiplication operation f.

5). The trusted authority outputs the UAV secret key $\mathsf{usk} = (id, \mathsf{u}, \{sk_1, sk_2\}_{x\in\mathsf{Path}(\theta_{id})})$ and delivers it to the UAV via secure channel. Also, it secretly sends the functional key sk_f to the LLM server.

B. Initial global parameter distribution

This phase is the same as that of the SPFF scheme.

C. Federated fine-tuning of LLM

• **Localpara** $(\mathbf{W}_0, \mathsf{Model}_{UAV})$: This algorithm is the same as that of the SPFF scheme.

• **Fine-tuning** $(\{Enc(\mathbf{P}_k)\}, \{Enc(\mathbf{Q}_k)\}, sk_f, \mathsf{Model}_{global})$: This algorithm is the same as that of the SPFF scheme.

• **Enc*** $(\mathsf{params}, \mathsf{CT}', \mathbb{A}, \mathbf{W}, t)$: LLM server takes the updated global parameter $\mathbf{W}$, current timestamp t and the access policy $\mathbb{A}$ as input, then generates the ciphertext of $\mathbf{W}$ with the following steps:

1). It first obtains the access vector $\mathsf{v} = (v_1, ..., v_n)$ from the access policy $\mathbb{A}'$, disassembles the parameter matrix $\mathbf{W}$ into a bit string ε, then randomly chooses $R \in \{0,1\}^{|\varepsilon|}, \theta \in \mathbb{Z}_p$, computes $\varphi = \mathsf{H}_2(\mathsf{MLE}(R||\varepsilon))$, $c = (R||\varepsilon) \oplus \mathsf{H}_1(Z^{\varphi})$, $\gamma_i = \eta_i + v_i\theta$, where $i \in [1, n]$, $c_1 = x^{\mathsf{H}_2(\varepsilon)} y^{\mathsf{H}_2(R)}$, $c_2 = g^{\varphi}$, $c_3 = w^{\varphi}$.

2). For each attribute $i \in [1, n]$, it randomly selects $\{\eta_i\}_{i\in[1,n]} \in \mathbb{Z}_p^{n+2}$, and calculate $c_0 = g^{\varphi}, c_1 = g_1^{\varphi} g^{-\eta_1}, ..., c_n = g_n^{\varphi} g^{-\eta_n}$. And for each bit $j \in [1, l]$ of the timestamp t, it computes $\hat{c}_0 = \varrho_0^{\varphi}$, $\hat{c}_j = \varrho_j^{\varphi}$ and $c' = \hat{c}_0 \cdot \prod_{j\in I} \hat{c}_j = (\varrho_0 \prod_{j\in I} \varrho_j)^{\varphi}$ for current timestamp t.

3). The LLM server finally outputs the ciphertext $\mathsf{CT} = (c, c', c_0, \hat{c}_0, \{c_i, \gamma_i\}_{i\in[1,n]}, \{\hat{c}_j\}_{j\in[1,l]}, c_1, c_2, c_3)$.

• **UAVupdate** $(id^*, t, \mathsf{RL}, \mathbf{st})$: This algorithm is the same as that of the SPFF scheme.

• **UKGen** $(\mathsf{RL}, \mathsf{params}, \mathsf{st}, t)$: This algorithm is the same as that of the SPFF scheme.

• **SKUpdate** $(\mathsf{params}, \mathsf{usk}, \mathsf{uk}, t)$: This algorithm is the same as that of the SPFF scheme.

• **Dec-tuning** $(\mathsf{CT}, \mathsf{usk})$: The UAV takes the ciphertext CT and its (updated) UAV secret key usk as input, then calculates $c_0' = g^{\sum_{i=1}^{n} \gamma_i u_i} \prod_{i=1}^{n} c_i^{u_i}$ calculates

$$\Phi = \frac{e(c_0, ek_1)}{e(c_0', ek_2) e(c', ek_3)}$$

$$\varepsilon || R = \bar{c} \oplus \mathsf{H}_2(\Phi)$$

The UAV accepts ε and retrieves $\mathbf{W}$ if $c_1 = x^{\mathsf{H}_2(\varepsilon)} y^{\mathsf{H}_1(R)}$. Now, the UAV can load the current LLM global parameters $\mathbf{W}$ and iterate through the **Localpara** and **Fine-tuning** algorithms until the fine-tuning of the LLM global model Model_{global} is completed.

D. Task delegation

If a UAV is unable to continue participating in the fine-tuning of a federated LLM, for instance, due to impending battery depletion or being compromised, it can request its neighboring edge node to re-encrypt the ciphertext. Subsequently, the edge node will send the re-encrypted ciphertext to another UAV. It should be noted that the re-encryption key and the ciphertext need to be meticulously designed to ensure that the delegated UAV can comply with the access policies of both the LLM server and the original UAV. Additionally, the algorithm structure should be designed to be sufficiently streamlined to accommodate the processing capabilities of the lightweight computational units on the UAV.

• **RekeyGen** $(\mathsf{params}, \mathsf{ek})$: The UAV takes the public parameters params and the edge key ek as input, converts the attributes of $\mathbb{A}'$ into $\mathsf{v}' = (\bar{v}_1, ..., \bar{v}_n)$ by invoking **Algorithm 1**, and randomly chooses $\chi \in \mathbb{G}_T$, $\varsigma, r' \in \mathbb{Z}_p$, then calculates the re-encryption key components $rk_0 = Z^{\mathsf{H}_2(\chi)}$, $rk_1 = sk_1^{\mathsf{H}_2(\chi)} \cdot w^{\varsigma}$, $rk_2 = sk_2^{\mathsf{H}_2(\chi)}$, $rk_3 = sk_3^{\mathsf{H}_2(\chi)}$, $rk_4 = g^{\varsigma}$, $rk_5 = \chi \cdot Z^{r'}$, $rk_6 = g^{r'}$, $rk_{7,i} = \bar{\gamma}_i = \eta_i + \bar{v}_i\theta$. The UAV assembles the re-encryption key $\mathsf{rk} = (rk_1, rk_2, rk_3, rk_4, rk_5, rk_6, \{rk_{7,i}\})$, and forwards it to the edge node.

• **ReEnc** $(\mathsf{params}, \mathsf{CT}, \mathsf{rk}, t)$: The edge node takes the public parameters params, ciphertext CT, and the re-encryption key rk as input, then it operates as follows, where $c_0' = g^{\sum_{i=1}^n \gamma_i u_i} \prod_{i=1}^n c_i^{u_i}$.

$$\bar{c}_0 = rk_6, \bar{c} = c, \bar{\gamma}_i = \eta_i + \bar{v}_i\theta, \bar{c}_1 = c_1, \bar{c}_i = c_i$$
$$\bar{c}_2 = c_2, \bar{c}_3 = c_3, \bar{c}_4 = rk_5$$
$$\bar{c}_5 = \frac{e(c_0, rk_1)}{e(c_0', rk_2)e(c', rk_3)e(c_3, rk_4)}, \bar{c}_6 = rk_0$$
$$\bar{c}' = \bar{\tilde{c}}_0 \cdot \prod_{j\in I} \bar{\tilde{c}}_j = (\varrho_0 \prod_{j\in I} \varrho_j)^{r'}$$

For each bit $j \in [1, l]$ of the timestamp t, it computes $\bar{\tilde{c}}_0 = \varrho_0^{r'}$, $\bar{\tilde{c}}_j = \varrho_j^{r'}$. The edge node outputs the re-encrypted ciphertext $\mathsf{RCT} = (\bar{c}, \bar{c}', \bar{c}_0, \bar{c}_1, \bar{c}_2, \bar{c}_3, \bar{c}_4, \bar{c}_5, \bar{c}_6, \{\bar{\gamma}_i\}_{i\in[1,n]}, \bar{\tilde{c}}_0, \{\bar{c}_i\}_{i\in[1,n]}, \{\bar{\tilde{c}}_j\}_{j\in[1,l]})$, and forwards it to the delegated UAV.

• **Dec-re** $(\mathsf{CT}, \mathsf{RCT}, \mathsf{usk})$: The delegatee UAV takes the ciphertext CT, the re-encryption ciphertext RCT and the (updated) UAV secret key usk as input, then calculates $c_0' = g^{\sum_{i=1}^n \gamma_i u_i} \prod_{i=1}^n c_i^{u_i}$ and obtains the global parameter $\mathbf{W}$ by computing

$$\chi = \bar{c}_4 \cdot \frac{e(\bar{c'}_0, sk_2)e(\bar{c'}, sk_3)}{e(\bar{c}_0, sk_1)}$$

where $\bar{c}_0' = g^{\sum_{i=1}^n \gamma_i u_i} \prod_{i=1}^n \bar{c}_i^{u_i}$. The UAV computes $\varepsilon || R = \bar{c} \oplus \mathsf{H}_1(\bar{c}_5^{\frac{1}{\mathsf{H}_2(\chi)}})$, then accepts ε and retrieves $\mathbf{W}$ if $c_1 = x^{\mathsf{H}_2(\varepsilon)} y^{\mathsf{H}_2(R)}$. Now, the UAV can load the current LLM global parameters $\mathbf{W}$ and iterate through the **Localpara** and **Fine-tuning** algorithms until the fine-tuning of the LLM global model Model_{global} is completed.

• **Pub-verify** $(\mathsf{CT}, \mathsf{RCT}, \mathsf{rk}, (\varepsilon, R))$: The verifier takes the ciphertext CT, the re-encrypted ciphertext RCT, the re-encryption key rk and the proof (ε, R) of a shared UAV, then the validity of RCT can be verified as following steps:

1). It verifies whether the following equations hold: $\bar{c}_0 = c_0, \bar{c} = c, \bar{c}_1 = c_1, \bar{c}_2 = c_2, \bar{c}_3 = c_3, \bar{c}' = c'$. If any of the aforementioned equations fails to hold true, the re-encrypted ciphertext RCT is deemed incorrect.

2). It calculates $\varphi = \mathsf{H}_2(\mathsf{MLE}(R||\varepsilon))$, and checks whether there exists $\bar{c}_1 = c_1 = x^{\mathsf{H}_2(\varepsilon)} y^{\mathsf{H}_2(R)}$. Please note that the purpose of checking this equation is to verify whether the original ciphertext CT is related to the parameter $\mathbf{W}$. It then tests whether $\bar{c}_6 = \bar{c}_7^{\varphi}$ holds, if this formula holds, it indicates that the re-encrypted ciphertext is related to the parameters $\mathbf{W}$, in other words, the re-encrypted ciphertext RCT is correct.

VI. Security analysis

Theorem 1 (Correctness): *The correctness of the proposed SPFF and DE-SPFF schemes guarantees that if the ciphertext, re-encrypted ciphertext, and UAV secret key are valid, then both the original and re-encrypted ciphertexts can be decrypted properly.*

Proof: In SPFF, there are two forms of ciphertext: the ciphertext of the initial global parameters ε_0 and the ciphertext of global parameter during federated fine-tuning ε. Due to the similarity in the structure of decryption algorithms, here we only demonstrate the decryption process for obtaining ε.

$$\begin{aligned}
&c \cdot \frac{e(c_0', sk_2)e(c', sk_3)}{e(c_0, sk_1)} \\
=&\frac{\varepsilon \cdot Z^{\delta} \cdot e(g^{\sum_{i=1}^n \gamma_i u_i} \prod_{i=1}^n c_i^{u_i}, g^{s+\pi}) e((\varrho_0 \prod_{j\in I} \varrho_j)^{\delta}, g^{\pi})}{e(g^{\delta}, g^{\alpha} g^{(s+\pi)\sum_{i=1}^n \tau_i u_i} \cdot (\varrho_0 \prod_{j\in I} \varrho_j)^{\pi})} \\
=&\frac{\varepsilon \cdot Z^{\delta} \cdot e(g^{\sum_{i=1}^n \gamma_i u_i} \prod_{i=1}^n c_i^{u_i}, g^{s+\pi}) e((\varrho_0 \prod_{j\in I} \varrho_j)^{\delta}, g^{\pi})}{e(g^{\delta}, g^{\alpha} g^{(s+\pi)\sum_{i=1}^n \tau_i u_i}) e(g^{\delta}, (\varrho_0 \prod_{j\in I} \varrho_j)^{\pi})} \\
=&\frac{\varepsilon \cdot Z^{\delta} \cdot e(g^{\sum_{i=1}^n \gamma_i u_i} \prod_{i=1}^n c_i^{u_i}, g^{s+\pi})}{e(g^{\delta}, g^{\alpha}) e(g^{\delta}, g^{(s+\pi)\sum_{i=1}^n \tau_i u_i})} \\
=&\frac{\varepsilon \cdot Z^{\delta} \cdot e(g^{\sum_{i=1}^n v_i u_i \theta} g^{\sum_{i=1}^n u_i \delta \tau_i}, g^{(s+\pi)})}{e(g^{\delta}, g^{\alpha}) e(g^{\delta}, g^{(s+\pi)\sum_{i=1}^n \tau_i u_i})} \\
=&\frac{\varepsilon \cdot Z^{\delta} \cdot e(g^{\sum_{i=1}^n u_i \delta \tau_i}, g^{(s+\pi)})}{e(g^{\delta}, g^{\alpha}) e(g^{\delta}, g^{(s+\pi)\sum_{i=1}^n \tau_i u_i})} = \varepsilon
\end{aligned}$$

For the DE-SPFF scheme, owing to the structural similarity with SPFF, we only provide the proof of correctness for the decryption process during the *Task delegation* phase.

$$\begin{aligned}
&\bar{c}_4 \cdot \frac{e(\bar{c'}_0, sk_2)e(\bar{c'}, sk_3)}{e(\bar{c}_0, sk_1)} \\
=&\frac{\chi \cdot e(g,g)^{\alpha r'} e(g^{\sum_{i=1}^n r' \gamma_i u_i} \prod_{i=1}^n c_i^{r' u_i}, g^{s+\pi}) e((\varrho_0 \prod_{j\in I} \varrho_j)^{r'}, g^{\pi})}{e(g^{r'}, g^{\alpha} g^{(s+\pi)\sum_{i=1}^n \tau_i u_i} \cdot (\varrho_0 \prod_{j\in I} \varrho_j)^{\pi})} \\
=&\frac{\chi \cdot e(g,g)^{\alpha r'}}{e(g,g)^{\alpha r'}} = \chi
\end{aligned}$$

$$\bar{c}_5 = \frac{e(c_0, rk_1)}{e(c'_0, rk_2)e(c', rk_3)e(c_3, rk_4)}$$
$$= \frac{e(g^{\varphi}, g^{\alpha \mathsf{H}_2(\chi)} g^{\mathsf{H}_2(\chi)(s+\pi)\sum_{i=1}^{n}\tau_i u_i} \cdot (\varrho_0 \prod_{j\in I}\varrho_j)^{\pi \mathsf{H}_2(\chi)} \cdot w^{\varsigma})}{e(g^{\sum_{i=1}^{n}\gamma_i u_i}\prod_{i=1}^{n} c_i^{u_i}, g^{\mathsf{H}_2(\chi)(s+\pi)})e((\varrho_0\prod_{j\in I}\varrho_j)^{\varphi}, g^{\pi \mathsf{H}_2(\chi)})e(w^{\varphi}, g^{\varsigma})}$$
$$= \frac{e(g^{\varphi}, g^{\alpha \mathsf{H}_2(\chi)} g^{\mathsf{H}_2(\chi)(s+\pi)\sum_{i=1}^{n}\tau_i u_i})}{e(g^{\sum_{i=1}^{n}\gamma_i u_i}\prod_{i=1}^{n} c_i^{u_i}, g^{\mathsf{H}_2(\chi)(s+\pi)})}$$
$$= \frac{e(g^{\varphi}, g^{\alpha \mathsf{H}_2(\chi)} g^{\mathsf{H}_2(\chi)(s+\pi)\sum_{i=1}^{n}\tau_i u_i})}{e(g^{\sum_{i=1}^{n}(\eta_i+v_i\theta)u_i}\prod_{i=1}^{n}(g^{\tau_i\varphi}g^{-\eta_i})^{u_i}, g^{\mathsf{H}_2(\chi)(s+\pi)})}$$
$$= e(g,g)^{\alpha\varphi \mathsf{H}_2(\chi)}$$

On this basis, we have

$$\bar{c} \oplus \mathsf{H}_1(\bar{c}_5^{\frac{1}{\mathsf{H}_2(\chi)}})$$
$$= (R||\varepsilon) \oplus \mathsf{H}_1(e(g,g)^{\alpha\varphi}) \oplus \mathsf{H}_1(e(g,g)^{\alpha\varphi \mathsf{H}_2(\chi)\cdot\frac{1}{\mathsf{H}_2(\chi)}}) = R||\varepsilon$$

Theorem 2*: The proposed SPFF and DE-SPFF schemes are indistinguishable under the selectively chosen plaintext attack (IND-CPA)* if that the DBDH problem is intractable.

Proof: Suppose there is a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ can break the IND-CPA security of SPFF with a non-negligible advantage, then a challenger $\mathcal{C}$ can be constructed that solves the DBDH problem with a non-negligible probability by invoking the challenger $\mathcal{C}_{\mathrm{IBE}}$ in [43]. Given the parameters $(g, g^a, g^b, g^c, \mathcal{R})$, $\mathcal{A}$ is required to determine whether $\mathcal{R} = e(g,g)^{abc}$ or $\mathcal{R}$ is a random element in $\mathbb{G}_T$. The interactive game between the adversary $\mathcal{A}$ and the challenger $\mathcal{C}$ is described as follows.

Initialize: $\mathcal{A}$ specifies a challenge access policy $\mathbb{A}^*$, then $\mathcal{C}$ samples $\sigma \in \{0,1\}$. When $\sigma = 0$, $\mathcal{A}$ is a non-revoked user, $\mathcal{C}$ then aborts if the attribute set S of an identity id contents the challenge access policy $\mathbb{A}^*$. When $\sigma = 1$, $\mathcal{A}$ is a revoked user, then $\mathcal{C}$ aborts if the attribute set S contents the target access policy $\mathbb{A}^*$ and the identity id is not revoked before the challenge timestamp t^* of the challenge ciphertext CT^*.

Setup: Given the security parameter λ, $\mathcal{C}$ sets the bilinear parameters tuple $\langle g, \mathbb{G}, \mathbb{G}_T, e, \rangle$ as the **Setup** algorithm in the DE-SPFF scheme. In addition, $\mathcal{C}$ defines the revocation list RL, the full binary tree $\mathbb{BT}$ with state st. When $\sigma = 0$, we sample $\varrho_0 \in \mathbb{Z}_p$ and $\varrho_1, ..., \varrho_l \in \mathbb{Z}_p$ for each bit of t. For each element in the attribute universe U, $\mathcal{C}$ randomly chooses $\psi, \beta'_1, ..., \beta'_n \in \mathbb{Z}_p$, it then simulates $g_1 = (g^a)^{-\psi v_1} g^{\beta'_1} = g^{-a\psi v_1 + \beta'_1}, ..., g_n = (g^a)^{-\psi v_n} g^{\beta'_n} = g^{-a\psi v_n + \beta'_n}, Z = e(g^a, g^b)$ to implicitly set $\alpha = ab, \tau_1 = -a\psi v_1 + \beta'_1, ..., \tau_n = -a\psi v_n + \beta'_n$. $\mathcal{C}$ publishes $\mathsf{params} = \{G, G_T, p, e, g, \{g_i\}_{i\in[1,n]}, Z, \varrho_0, \{\varrho_j\}_{j\in[1,l]}, t\}$, $\mathbb{BT}$ with state st and RL. And when $\sigma = 1$, $\mathcal{C}$ calls $\mathcal{C}_{\mathrm{IBE}}$ for $params_{\mathrm{IBE}}$, then it runs as the simulation for $\sigma = 0$. $\mathcal{C}$ outputs the public parameter $\mathsf{params} = \{\mathsf{params}_{\mathrm{IBE}}, g, \{g_i\}_{i\in[1,n]}, t\}$, the full binary tree $\mathbb{BT}$ with state st and the revocation list RL.

Phase 1: $\mathcal{A}$ issues a sequence of queries to the challenger $\mathcal{C}$, then $\mathcal{C}$ interacts with $\mathcal{A}$ in the following way.

UAV secret key query. $\mathcal{A}$ queries on the UAV secret key of an identity id and an attribute set S, $\mathcal{C}$ first invokes the **Algorithm 3** to obtain the attribute vector u then answers as follows: When $\sigma = 0$ and $\langle \mathsf{u}, \mathsf{v} \rangle = 0$, $\mathcal{C}$ aborts to tell that id is revoked even if S contents $\mathbb{A}$. When $\sigma = 0$ and $\langle \mathsf{u}, \mathsf{v} \rangle \neq 0$, $\mathcal{C}$ samples $g_x \in \mathbb{G}, \epsilon \in \mathbb{Z}_p$ then creates the UAV secret key for each $x \in \mathrm{Path}(\theta_{\mathrm{id}})$:

$$sk_1 = g_x^{-1} \prod_{i=1}^{n} (((g^a)^{-\psi u_i} g^{\beta'_i})^{v_i \epsilon}) \cdot (g^b)^{\frac{\beta'_i u_i}{\psi \cdot \langle \mathsf{u}, \mathsf{v} \rangle}}$$
$$sk_2 = g^{\epsilon} (g^b)^{\frac{1}{\psi \cdot \langle \mathsf{u}, \mathsf{v} \rangle}}$$

Notice that the UAV secret key can be parsed as below if we set $s = \epsilon + \frac{b}{\psi \cdot \langle \mathsf{u}, \mathsf{v} \rangle}$:

$$\begin{aligned}
sk_1 =& g_x^{-1} \prod_{i=1}^{n} ((g^a)^{-\psi u_i} g^{\beta'_i})^{v_i \epsilon} \cdot (g^b)^{\frac{\beta'_i u_i}{\psi \cdot \langle \mathsf{u}, \mathsf{v} \rangle}} \\
=& g_x^{-1} \prod_{i=1}^{n} g^{-a u_i v_i \psi \epsilon} g^{-ab\psi u_i v_i \frac{1}{\psi < u_i, v_i >}} \\
& \cdot g^{ab\psi u_i v_i \frac{1}{\psi < u_i, v_i >}} g^{\beta'_i v_i \epsilon} g^{\frac{b\beta'_i u_i}{\psi \cdot \langle \mathsf{u}, \mathsf{v} \rangle}} \\
=& g_x^{-1} \prod_{i=1}^{n} (g^{-a\psi v_i})^{u_i(\epsilon + \frac{b}{\psi \langle \mathsf{u}, \mathsf{v} \rangle})} (g^{\beta'_i})^{u_i(\epsilon + \frac{b}{\psi \langle \mathsf{u}, \mathsf{v} \rangle})} \\
& \cdot g^{ab\psi u_i v_i \frac{1}{\psi \langle \mathsf{u}, \mathsf{v} \rangle}} \\
=& g_x^{-1} g^{ab} \prod_{i=1}^{n} (g^{-a\psi v_i} g^{\beta'_i})^{u_i(\epsilon + \frac{b}{\psi \langle \mathsf{u}, \mathsf{v} \rangle})} \\
=& g^{\alpha} / g_x \prod_{i=1}^{n} (g_i)^{u_i s} \\
sk_2 =& g^{\epsilon} (g^b)^{\frac{1}{\psi \cdot \langle \mathsf{u}, \mathsf{v} \rangle}} = g^{\epsilon + \frac{b}{\psi \cdot \langle \mathsf{u}, \mathsf{v} \rangle}} = g^s
\end{aligned}$$

$\mathcal{C}$ returns the UAV secret key $\mathsf{usk} = (id, \vec{u}, \{sk_1, sk_2\}_{x\in \mathsf{Path}(\theta_{id})})$ with state $\mathsf{st} \leftarrow \mathsf{st} \cup \{x, g_x\}$.

When $\sigma = 1$ and $\langle \mathsf{u}, \mathsf{v} \rangle = 0$, $\mathcal{C}$ specifies an unoccupied leaf node θ_{id} for an identity id. For each $x \in \mathsf{Path}(\theta_{id})$, it defines g_x if available, otherwise randomly selects $g_x, s \in \mathbb{Z}_p$ and calculates

$$sk_1 = g_x g^{s \sum_{i=1}^{n} \tau_i u_i}, sk_2 = g^s$$

$\mathcal{C}$ returns $\mathsf{usk} = (id, \vec{u}, \{sk_1, sk_2\}_{x\in \mathsf{Path}(\theta_{id})})$.

Update key query. $\mathcal{A}$ queries for the update key of a timestamp t, then $\mathcal{C}$ defines a collection $I = \{j | t[j] = 0, j \in [1, l]\}$. When $\sigma = 0$, $\mathcal{C}$ randomly chooses $\pi \in \mathbb{Z}_p$, then calculates

$$uk_1 = g_x \cdot (\varrho_0 \prod\nolimits_{j\in I} \varrho_j)^{\pi}, uk_2 = g^{\pi}$$

$\mathcal{C}$ returns the update key $\mathsf{uk} = (t, \{uk_1, uk_2\}_{x\in \mathsf{KUNodes}(\mathsf{RL},\mathsf{st},t)})$.

When $\sigma = 1$, $\mathcal{C}$ calls the secret key $\mathsf{sk}_{\mathrm{IBE}} = (sk_{1,\mathrm{IBE}}, sk_{2,\mathrm{IBE}})$ from $\mathcal{C}_{\mathrm{IBE}}$ about t, then selects $\bar{r} \in \mathbb{Z}_p$ randomly and computes

$$\begin{aligned}
uk_1 &= sk_{1,\mathrm{IBE}} / g_x \cdot (\varrho_0 \prod\nolimits_{j\in I} \varrho_j)^{\bar{r}} \\
&= (g^{\alpha} / g_x) \cdot (\varrho_0 \prod\nolimits_{j\in I} \varrho_j)^{\pi + \bar{r}} \\
uk_2 &= sk_{2,\mathrm{IBE}} \cdot g^{\bar{r}} = g^{\pi + \bar{r}}
\end{aligned}$$

$\mathcal{C}$ returns the update key $\mathsf{uk} = (t, \{uk_1, uk_2\}_{x\in \mathsf{KUNodes}(\mathsf{RL},\mathsf{st},t)})$ to $\mathcal{A}$.

UAV secret key update query. $\mathcal{A}$ requests for the update on its UAV secret key usk. When $\sigma = 0$ and $\langle \mathsf{u}, \mathsf{v} \rangle = 0$, $\mathcal{C}$ aborts. And when $\sigma = 0$, $\langle \mathsf{u}, \mathsf{v} \rangle = 1$, $\mathcal{C}$ retrieves the UAV secret key usk and the update key uk if available, otherwise $\mathcal{C}$ performs the *UAV secret key query* and the *update key query* to obtain usk and uk. Then $\mathcal{C}$ calculates

$$\begin{aligned} sk_1 =& sk_1 \cdot uk_1 = \prod_{i=1}^{n}(((g^a)^{-\psi u_i} g^{\beta'_i})^{v_i \epsilon}) \cdot (g^b)^{\frac{\beta'_i u_i}{\psi \cdot \langle \mathsf{u}, \mathsf{v} \rangle}} \\ & \cdot (\varrho_0 \prod\nolimits_{j \in I} \varrho_j)^{\pi} \\ sk_2 =& sk_1 \cdot uk_2 = g^{\epsilon+\pi} (g^b)^{\frac{\beta'}{\psi \cdot \langle \mathsf{u}, \mathsf{v} \rangle}} \\ sk_3 =& uk_2 = g^{\pi} \end{aligned}$$

$\mathcal{C}$ returns the updated UAV secret key $\mathsf{usk} = ((id, t), \mathsf{u}, \{sk_1, sk_2, sk_3\})$.

When $\sigma = 1$, $\mathcal{C}$ also retrieves usk and uk if available, otherwise $\mathcal{C}$ performs the *UAV secret key query* and the *update key query* to obtain usk and uk. $\mathcal{C}$ then calculates

$$\begin{aligned} sk_1 =& sk_1 \cdot uk_1 = g^{\alpha} \cdot g^{s \sum_{i=1}^{n} \tau_i u_i} \cdot (\varrho_0 \prod\nolimits_{j \in I} \varrho_j)^{\pi + \bar{r}} \\ sk_2 =& sk_2 \cdot uk_2 = g^{\pi + \bar{r}} \\ sk_3 =& uk_2 = g^{\bar{r}} \end{aligned}$$

$\mathcal{C}$ returns the updated UAV secret key $\mathsf{usk} = ((id, t), \mathsf{u}, \{sk_1, sk_2, sk_3\})$.

Revocation query. $\mathcal{A}$ issues the revocation query on an identity id and a timestamp t, then $\mathcal{C}$ updates the revocation list by operating $\mathsf{RL} \leftarrow \mathsf{RL} \cup (x, t)$ for all nodes $x \in \mathsf{Path}(\theta_{id})$.

Challenge: $\mathcal{A}$ specifies two equal-length messages ε_0, ε_1. Then, for $\sigma = 0$, $\mathcal{C}$ picks $b \in \{0, 1\}$ and calculates $c^* = \varepsilon_b \cdot \mathcal{R}$ and $(g^c, (g^c)^{\beta'_1}, ..., (g^c)^{\beta'_n})$. They can also be parsed as $(c_0^*, \{c_i^*\}_{i \in [1,n]}) = (g^{\delta}, g^{v_1 \delta}(g_1)^{\delta}, ..., (g^{v_n \delta}(g_n)^{\delta})$ if it defines $\delta = c, \theta = ac\psi$, where

$$\begin{aligned} c_i^* &= (g^c)^{\beta'_i} = (g^c)^{a\psi v_i - a\psi v_i + \beta'_i} \\ &= (g^c)^{a\psi v_i} (g^c)^{-a\psi v_i + \beta'_i} = g^{v_i \theta} (g_i)^t \end{aligned}$$

$\mathcal{C}$ also picks $\eta'_1, ..., \eta'_n \in \mathbb{Z}_p$ then calculates:

$$c_i^* \cdot g^{-\eta'_i} = g^{v_i \theta} (g_i)^{\delta} g^{-\eta'_i} = g^{v_i \theta - \eta'_i} (g_i)^{\delta} = g_i^{\delta} g^{\eta_i} = c_i^*$$

where $\eta_i = -v_i \theta + \eta'_i$, so we further obtain $\eta'_i = v_i \theta + \eta_i = \gamma_i$.

$\mathcal{C}$ then calculates $\hat{c}_0^* = \varrho_0^{\delta}$, $\hat{c}_j^* = \varrho_j^{\delta}$, and $c'^* = \hat{c}_0^* \cdot \prod_{j \in I} \hat{c}_j^* = (\varrho_0 \prod_{j \in I} \varrho_j)^{\delta}$, where I is the index collection of all zero bits $t[j] = 0$ in t.

When $\sigma = 1$ and $t \leq t^*$, the user with identity id^* and $\langle \mathsf{u}, \mathsf{v}^* \rangle = 0$ has been revoked, that is, $(id^*, t) \in \mathsf{RL}$. Then $\mathcal{C}$ aborts because id^* was revoked before the time period t^*. When $\sigma = 1$ and $t > t^*$, the user with id^* and $\langle \mathsf{u}, \mathsf{v}^* \rangle = 0$ has been revoked, that is $(id^*, t) \in \mathsf{RL}$. $\mathcal{C}$ sends $(\varepsilon_0, \varepsilon_1, t^*)$ to $\mathcal{C}_{\mathrm{IBE}}$, then $\mathcal{C}_{\mathrm{IBE}}$ returns $c_{\mathrm{IBE}}^* = (c^* = \varepsilon_b \cdot e(g, g)^{\alpha\delta}, c_0^* = g^{\delta}, c'^* = (\varrho_0 \prod_{j \in I} \varrho_j)^{\delta})$. Then, $\mathcal{C}$ picks $\theta, \{\eta_i\}_{i \in [1,n]} \in \mathbb{Z}_p^{n+1}$ and calculates $\gamma_i^* = \eta_i + v_i \theta$, $c_i^* = g_i^{\delta} g^{-\eta_i}$ and $\hat{c}_0^* = \varrho_0^{\delta}$ and $\hat{c}_j^* = \varrho_j^{\delta}$. $\mathcal{C}$ returns $\mathsf{CT}^* = (c^*, c'^*, c_0^*, \hat{c}_0^*, \{c_i^*, \gamma_i^*\}_{i \in [1,n]}, \{\hat{c}_j^*\}_{j \in [1,l]})$ about the message M_b and $\mathbb{A}^*$.

Phase 2: This phase is identical to **Phase 1**.

Guess: $\mathcal{A}$ outputs its guess $b' \in \{0, 1\}$ on b. When $\sigma = 0$, if $b' = b$, we say $\mathcal{A}$ wins the game $\mathsf{Game}_{\mathrm{IND-CPA}}^{\mathrm{PH-ABE-DS}}$, then $\mathcal{C}$ outputs 1 to guess $\mathcal{R} = e(g, g)^{abc}$. Otherwise, $\mathcal{A}$ fails and outputs 0 to indicate that $\mathcal{R}$ is a random element in the group G. When $\sigma = 1$, $\mathcal{C}$ solves the DBDH problem by resorting to $\mathcal{C}_{\mathrm{IBE}}$.

A comprehensive analysis, considering the number of queries q_{sk} made by $\mathcal{A}$, shows that the probability that $\mathcal{C}$ successfully completes the simulation without aborting is at least $1/(2 \cdot \xi(q_{sk}))$, where $\xi(\cdot)$ is a polynomial function representing the worst-case number of possible abort conditions per query.

If $\mathcal{C}$ does not abort, the simulation for $\mathcal{A}$ is perfect. Therefore, if $\mathcal{A}$ wins the IND-CPA game (i.e., guesses $b' = b$ correctly), so does $\mathcal{C}$ in solving the DBDH instance. Conversely, if $\mathcal{A}$ fails, $\mathcal{C}$ guesses randomly.

Thus, the advantage of $\mathcal{C}$ in solving the DBDH problem is directly related to the advantage of $\mathcal{A}$:

$$\mathbf{Adv}_{\mathcal{C}}^{\mathrm{DBDH}}(\kappa) \geq \frac{1}{2 \cdot \xi(q_{sk})} \cdot \mathbf{Adv}_{\mathcal{A}}^{\text{IND-CPA}}(\kappa) - \mathsf{negl}(\kappa)$$

Since $\mathbf{Adv}\mathcal{A}^{\mathrm{IND-CPA}}(\kappa)$ is assumed to be non-negligible, and $1/(2 \cdot \xi(qsk))$ is a non-negligible fraction (as q_{sk} is polynomial in κ), it follows that $\mathbf{Adv}_{\mathcal{C}}^{\mathrm{DBDH}}(\kappa)$ is also non-negligible. This contradicts the DBDH hardness assumption, proving the theorem.

□

Theorem 3*: The re-encryption process of the proposed DE-SPFF scheme is verifiable under the discrete logarithm (DL) assumption.*

Proof: Assume an adversary $\mathcal{A}$ can break the verifiability of the proposed DE-SPFF scheme with a non-negligible probability , then we can construct a probabilistic polynomial time challenger algorithm $\mathcal{C}$ that solves the DL problem with a non-negligible probability. More specifically, given the tuple $(e, \mathbb{G}, \mathbb{G}_T, p, g, g^{\kappa})$, the goal of $\mathcal{C}$ is to solve the value κ. The verifiability game is described as follows.

Initialize: The adversary $\mathcal{A}$ claims the challenge access policy $\mathbb{A}^*$.

Setup: $\mathcal{C}$ randomly selects $\alpha, \kappa, \psi, \omega, \tau_1, ..., \tau_n \in \mathbb{Z}_p$, then calculates $g_i = g^{\tau_i}$, $Z = e(g, g)^{\alpha}$, $x = g^{\kappa}$, $y = g^{\psi}$, $w = g^{\omega}$, where $i \in [1, n]$. It also defines two collision-resistant hash functions $\mathsf{H}_1 : \mathbb{G}_T \rightarrow \mathbb{G}$ and $\mathsf{H}_2 : \{0, 1\}^* \rightarrow \mathbb{Z}_p^*$, and a message lock encryption algorithm MLE. The challenger $\mathcal{C}$ outputs and forwards the public parameter $\mathsf{params} = \{\mathbb{G}, \mathbb{G}_T, p, e, g, x, y, w, \{g_i\}_{i \in [1,n]}, Z, \mathsf{H}_1, \mathsf{H}_2, \varrho_0, \{\varrho_j\}_{j \in [1,l]}, \mathsf{MLE}\}$, then secretly holds the master secret key $\mathsf{msk} = \{\alpha, \{\tau_i\}_{i \in [1,n]}\}$.

Phase 1: The adversary $\mathcal{A}$ issues a series of queries on the UAV secret key usk, and the re-encryption key rk and re-encrypted ciphertext verification. Then, the challenger $\mathcal{C}$ answers these queries by running the **SKGen**, **RKGen** and **Pub-verify** algorithms.

Challenge: The adversary $\mathcal{A}$ queries a ciphertext by sending a message ε^* and $\mathbb{A}^*$. The challenger $\mathcal{C}$ invokes the **Enc*** algorithm and returns the ciphertext $\mathsf{CT}^* = (c^*, c'^*, c_0^*, \hat{c}_0^*, \{c_i^*, \gamma_i^*\}_{i \in [1,n]}, \{\hat{c}_j^*\}_{j \in [1,l]}, c_1^*, c_2^*, c_3^*)$.

Phase 2: In this phase, the adversary $\mathcal{A}$ issues the same queries to $\mathcal{C}$ as **Phase 1**.

Guess: The adversary $\mathcal{A}$ sends a ciphertext $\mathsf{CT}'^{*} = (c^{*'}, c'^{*'}, c_0^{*'}, \hat{c}_0^{*'}, \{c_i^{*'}, \gamma_i^{*'}\}_{i\in[1,n]}, \{\hat{c}_j^{*'}\}_{j\in[1,l]}, c_1^{*'}, c_2^{*'}, c_3^{*'})$ to $\mathcal{C}$. The challenger $\mathcal{C}$ obtains a UAV secret key usk, then decrypts the ciphertext CT^{*}. If the ciphertext CT'^{*} can be correctly decrypted, which implies $c_1^{*} = c_1^{*'}$, that is, $x^{\mathsf{H}_2(\varepsilon^{*})} y^{\mathsf{H}_2(R^{*})} = x^{\mathsf{H}_2(\varepsilon^{*'})} y^{\mathsf{H}_2(R^{*'})}$. It can be parsed as $g^{\kappa(\mathsf{H}_2(\varepsilon^{*}))} g^{\psi(\mathsf{H}_2(R^{*}))} = g^{\kappa(\mathsf{H}_2(\varepsilon^{*'}))} g^{\psi(\mathsf{H}_2(R^{*'}))}$, then we have $\kappa(\mathsf{H}_2(\varepsilon^{*'}) - \mathsf{H}_2(\varepsilon^{*})) = \psi(\mathsf{H}_2(R^{*'}) - \mathsf{H}_2(R^{*}))$, and κ can be calculated by $\kappa = \frac{\psi(\mathsf{H}_2(R^{*'}) - \mathsf{H}_2(R^{*}))}{\mathsf{H}_2(\varepsilon^{*'}) - \mathsf{H}_2(\varepsilon^{*})}$.

If the forgery is valid but RCT^{*} is incorrect, it implies that $\mathcal{A}$ has somehow manipulated the components related to the secret κ. By examining the difference between a correctly formed RCT and the forged RCT^{*}, and by leveraging the answers $\mathcal{A}$ made to the random oracles H_2, $\mathcal{C}$ can extract a solution to the DL problem.

Using the Forking Lemma [44], $\mathcal{C}$ can rewind $\mathcal{A}$ and run it again with a different response for the H_2 query on the message ε^{*} (or R^{*}). From the two forged signatures obtained from these two runs with different hash outputs, $\mathcal{C}$ can solve for the unknown exponent κ.

The probability that $\mathcal{C}$ successfully extracts the discrete logarithm κ is polynomially related to the probability that $\mathcal{A}$ produces a forgery. Specifically, there exists a polynomial function $\zeta(\kappa)$ such that:

$$\mathbf{Adv}_{\mathcal{C}}^{\mathrm{DL}}(\kappa) \geq \frac{1}{\zeta(\kappa)} \cdot \left(\mathbf{Adv}_{\mathcal{A}}^{\mathrm{Verif}}(\kappa) - \mathsf{negl}(\kappa)\right)^2$$

The square term and the polynomial fraction $1/\zeta(\kappa)$ are typical results obtained through the application of the Forking Lemma. Since $\mathbf{Adv}_{\mathcal{A}}^{\mathrm{Verif}}(\kappa)$ is non-negligible by assumption, it follows that $\mathbf{Adv}_{\mathcal{C}}^{\mathrm{DL}}(\kappa)$ is also non-negligible. This contradicts the DL hardness assumption, proving the theorem. □

Theorem 4*: The proposed DE-SPFF scheme is secure against free-riding attack, collusion attack, and falsified re-encryption.*

Proof: Resilience against free-riding attack: The threat of unauthorized entities "free-riding" to steal global parameters is mitigated by the core design of our attribute-based encryption scheme. Access to the global parameters $\mathbf{W}$ is strictly controlled by the ciphertext policy $\mathbb{A}$ embedded in $\mathbf{CT}$ by the LLM server. An unauthorized UAV or external adversary possesses a secret key usk linked to an attribute set S that does not satisfy $\mathbb{A}$. As formally proven in Theorem 2 under the DBDH assumption, such an adversary cannot decrypt the ciphertext $\mathbf{CT}$ to recover $\mathbf{W}$. The scheme does not rely on network-level security to prevent access; rather, cryptographic access control ensures that only authorized participants (those with valid, non-revoked credentials whose attributes satisfy the policy) can decrypt and benefit from the fine-tuned model, effectively eliminating the free-riding threat.

Resilience against collusion between a compromised edge node and a delegatee UAV: This is a sophisticated attack where a malicious delegatee UAV (the recipient of a re-encrypted ciphertext RCT) colludes with a compromised edge node to try and extract the global parameters. The security of the proxy re-encryption process in DE-SPFF is specifically designed to prevent this. The edge node performs its re-encryption operation using a re-encryption key rk provided by the delegator UAV. Crucially, this rk is generated from the delegator's secret key usk and is bound to the access policy $\mathbb{A}'$ of the delegatee. While the colluding parties have access to rk and RCT, they do not possess the private key of the delegatee. Decryption of RCT (Dec-re algorithm) requires the delegatee's own secret key to compute the final symmetric key. The design ensures that the re-encryption key rk alone is insufficient for decryption. The confidentiality of the underlying plaintext is maintained under the hardness of the DBDH problem, as the re-encryption process does not create a new vulnerability. The edge node and the delegatee cannot combine their knowledge to learn anything beyond what the delegatee is already authorized to learn, which is the plaintext $\mathbf{W}$ if its attributes satisfy the policy, an event that is not considered a breach. If the delegatee's attributes do not satisfy the policy, the collusion still cannot decrypt the ciphertext.

Resilience against falsified re-encryption by a malicious edge node: The threat of an edge node returning a falsified or incorrectly computed RCT to disrupt the delegation process is neutralized by the public verifiability feature of the DE-SPFF scheme. As proven in Theorem 3 under the DL assumption, any deviation from the correct ReEnc algorithm will be detected with overwhelming probability by the Pub-verify algorithm. A third-party verifier (which could be the delegatee UAV itself, the delegator, or a ground control station) can check the correctness of RCT relative to the original CT and the public re-encryption key rk without needing any secret information. The verification process checks the structural integrity and cryptographic consistency of the re-encrypted ciphertext. Therefore, a malicious edge node cannot successfully sabotage the learning process of a benign delegatee UAV by sending garbage data, as such action would be detected and the faulty RCT would be rejected. □

VII. Performance analysis

This section presents the analysis of the proposed DE-SPFF scheme in terms of theoretical comparison, experimental comparison, and simulation evaluation. Please note that we are the first to apply ABE to the scenario of federated LLM fine-tuning for UAV swarm networks. Therefore, we only compared DE-SPFF with Ge *et al.*'s VF-ABPRE scheme [41], Zhang *et al.*'s EFPR scheme [45], and Li *et al.*'s DAPRE scheme [46] at the algorithmic level.

A. Theoretical comparisons

A functional comparison of the aforementioned schemes is presented in Table I. Among them, the proposed DE-SPFF only implements the AND-Gate access structure, whereas the other schemes provide the more expressive LSSS access structure. Only DE-SPFF supports user revocation, which means it can effectively address scenarios involving UAV drop-out during the execution of federated LLM fine-tuning tasks. Only DE-SPFF provides access policy hiding, indicating its

capability to protect users' attribute privacy information while providing data access control. Similarly, only DE-SPFF supports wildcard matching, allowing for flexibility in attribute-based access control. All four schemes are delegatable, which enhances their scalability and flexibility. Compared to EFPR [45] and DAPRE [46], both VF-ABPRE [41] and DE-SPFF enable verifiable delegation, thereby preventing third parties from generating fraudulent re-encrypted ciphertexts.

TABLE I: Functional comparisons

Schemes	Access structure	User revocable	Policy hiding	Wildcard matching	Cross domain	Delegation verifiable
VF-ABPRE [41]	LSSS	×	×	×	✓	✓
EFPR [45]	LSSS	×	×	×	✓	×
DAPRE [46]	LSSS	×	×	×	✓	×
DE-SPFF	AND	✓	✓	✓	✓	✓

A comparison of the aforementioned schemes in terms of storage and computational complexity is presented in Table II and Table III, respectively. As can be readily observed from Table II, the proposed DE-SPFF scheme features a fixed length for user keys. In terms of ciphertext, EFPR [45] and DAPRE [46] exhibit fixed complexity, which is superior to that of VF-ABPRE [41] and DE-SPFF. Regarding the storage complexity of re-encryption keys, DE-SPFF demonstrates a significant advantage over VF-ABPRE [41], EFPR [45], and DAPRE [46].

Table III illustrates the comparison of the aforementioned schemes in terms of computational complexity. It is evident that the proposed DE-SPFF scheme incurs only constant overheads in the key generation, key update, re-encryption key generation, and verification phases. Although the other schemes do not involve key update and verification phases, DE-SPFF still exhibits significant advantages over them in the key generation and re-encryption key generation phases. Among the other algorithms, the computational complexity of SPFF is related to the number of attributes or the bit length of the timestamp.

B. Experimental comparison

We conducted experimental evaluations of the target schemes on a laptop equipped with an Intel Core Ultra 9 185H processor (2.30 GHz), 32 GB of RAM, and a 64-bit Windows 10 OS. The primary aim was to assess and contrast the real-world performance of these schemes. The implementation leverages the PBC library [47]. These schemes rely on a supersingular elliptic curve $E/\mathbb{F}_p : y^2 = x^3 + x$ defined over the finite field $\mathbb{F}_p$, with an embedding degree of 2. Here, p represents the prime order of the group $\mathbb{G}_1$. Consequently, the bit-lengths of elements in $\mathbb{G}_1$ and $\mathbb{G}_2$ are both 128 Bytes, while the bit-length of elements in $\mathbb{Z}_p$ is 20 Bytes. The dataset used is the Semantic Drone Dataset, released by ETH Zurich, which is an aerial image semantic segmentation dataset. It consists of high-resolution urban scene images captured by drones (700 images in total) with pixel-level annotations, covering 20 object categories such as buildings, roads, and vehicles. We randomly selected a subset of these images and converted them into bit strings using the OpenCV library to evaluate the performance of our algorithm on bit strings.

The storage overhead comparison between our proposed DE-SPFF scheme and the VF-ABPRE [41], EFPR [45], and DAPRE [46] schemes across different phases is illustrated in Fig. 3, where we varied the number of attributes from 10 to 50, and set the bit length of timestamp as 10. Fig. 3-(a) shows that while the user secret key storage overhead of VF-ABPRE [41], EFPR [45], and DAPRE [46] increases with attribute count, DE-SPFF maintains a constant and significantly lower overhead due to our optimized key structure. In Fig. 3-(b), both VF-ABPRE [41] and DE-SPFF exhibit growing ciphertext storage overhead with increasing attributes, with DE-SPFF outperforming VF-ABPRE [41] while remaining below 10 KB at 50 attributes, though slightly higher than EFPR [45] and DAPRE [46]. The re-encryption key storage comparison in Fig. 3-(c) reveals that all schemes scale with attribute count, but DE-SPFF maintains the lowest overhead (under 10 KB at 50 attributes). Similarly, Fig. 3-(d) demonstrates that while re-encrypted ciphertext storage grows with attributes in all schemes, DE-SPFF consistently achieves superior efficiency compared to the alternatives.

Fig. 4 presents the computational overhead comparison across different phases, where we varied the number of attributes from 5 to 50 in increments of 5. As shown in Fig. 4-(a), DE-SPFF maintains stable computational time for UAV secret key generation with an average of 15.27ms, while other schemes exhibit near-linear growth that significantly exceeds DE-SPFF's performance (VF-ABPRE requires nearly 1400ms for user secret key generation at 50 attributes). For encryption time (Fig. 4-(b)), EFPR remains stable while other schemes including DE-SPFF show near-linear growth, with DE-SPFF requiring 926.10ms at 50 attributes - though higher than EFPR [45], it demonstrates clear advantages over VF-ABPRE [41] and DAPRE. Regarding re-encryption key generation (Fig. 4-(c)), VF-ABPRE [41] and EFPR display near-linear growth, whereas DAPRE [46] and DE-SPFF maintain stability with DE-SPFF showing particularly superior efficiency. During re-encryption (Fig. 4-(d)), EFPR [45] and DAPRE [46] remain lightweight (¡100ms), while DE-SPFF (539.61ms at 50 attributes) significantly outperforms VF-ABPRE [41] despite its attribute-dependent growth. Decryption performance (Fig. 4-(e)) reveals that while all schemes scale with attribute count, DE-SPFF demonstrates marked superiority (under 500ms at 50 attributes). Fig. 4-(f) compares verification time between VF-ABPRE [41] and DE-SPFF (the only schemes supporting public verification), showing consistently low and stable overhead for both. Crucially, DE-SPFF innovatively achieves wildcard matching, user revocation, policy privacy preservation, and publicly verifiable re-encryption without introducing complex algorithmic structures, while significantly outperforming other schemes in user/UAV secret key generation, re-encryption key generation, and decryption phases, while maintaining competitive advantages in encryption, re-encryption, and public verification phases.

We also designed an experiment to simulate and evaluate the additional communication and computational time overhead incurred by passing upstream and downstream parameters to

TABLE II: Storage complexity comparisons

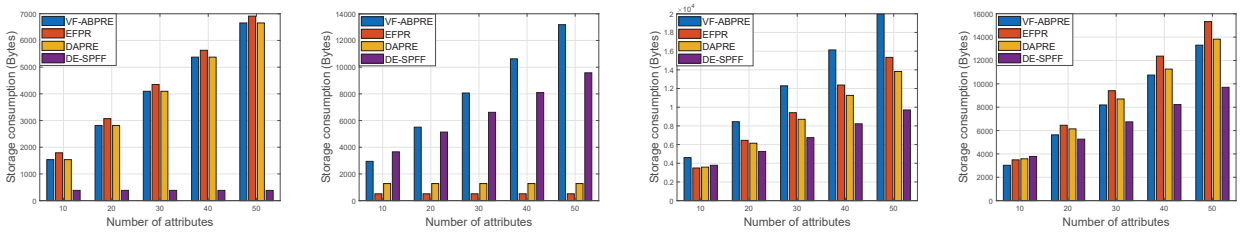
Schemes	**user secret key**	**ciphertext**	**re-encryption key**	**re-encrypted ciphertext**
VF-ABPRE [41]	$(n+2)\|\mathbb{G}\|$	$(2n+3)\|\mathbb{G}\|$	$(3n+4)\|\mathbb{G}\|+2\|\mathbb{G}_T\|$	$(2n+2)\|\mathbb{G}\|+2\|\mathbb{G}_T\|$
EFPR [45]	$(n+4)\|\mathbb{G}\|$	$3\|\mathbb{G}\|+\|\mathbb{G}_T\|$	$(2n+4)\|\mathbb{G}\|+(2n+1)\|\mathbb{Z}_p\|$	$(2n+3)\|\mathbb{G}\|+\|\mathbb{G}_T\|+(2n+1)\|\mathbb{Z}_p\|$
DAPRE [46]	$(n+2)\|\mathbb{G}\|$	$9\|\mathbb{G}\|+\|\mathbb{G}_T\|$	$(2n+6)\|\mathbb{G}\|+\|\mathbb{G}_T\|$	$(2n+6)\|\mathbb{G}\|+2\|\mathbb{G}_T\|$
DE-SPFF	$3\|\mathbb{G}\|$	$(n+l+7)\|\mathbb{G}\|+n\|\mathbb{Z}_p\|$	$4\|\mathbb{G}\|+2\|\mathbb{G}_T\|+n\|\mathbb{Z}_p\|$	$(n+l+6)\|\mathbb{G}\|+2\|\mathbb{G}_T\|+n\|\mathbb{Z}_p\|$

Notations: n: number of attributes; l: bit length of the timestamp; $|\mathbb{Z}_p|$: an element in $\mathbb{Z}_p$; $|\mathbb{G}|$: an element in the group $\mathbb{G}$; $|\mathbb{G}_T|$: an element in $\mathbb{G}_T$.

TABLE III: Computational complexity comparisons

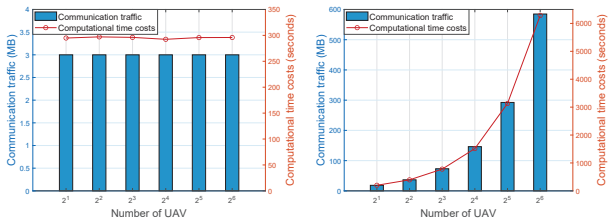
Schemes	**key generation**	**encryption**	**rk genration**	**re-encryption**	**decryption**	**key update**	**verification**
VF-ABPRE [41]	$(n+3)e_G$	$(3n+4)e_G+e_{G_T}$	$(4n+5)e_G+2e_{G_T}$	$(2n+2)P$	$(2n+1)P+2e_G+e_{G_T}$	N/A	$2e_G$
EFPR [45]	$(n+7)e_G$	$4e_G+e_{G_T}$	$(3n+2)e_G+e_{G_T}$	$2P$	$(2n+2)P+(3n+2)e_G$	N/A	N/A
DAPRE [46]	$(2n+15)e_G+(2n+6)H$	$(4n+7)e_G+2nH$	$14e_G+3H$	$6P$	$(n+4)P$	N/A	N/A
DE-SPFF	$2e_G$	$(2n+l+6)e_G+e_{G_T}+H$	$6e_G+2e_{G_T}$	$4P+(n+l+2)e_G$	$3P+(n+3)e_G+e_{G_T}+H$	$2e_G$	$2e_G$

Notations: n: number of user's attributes; l: bit length of the timestamp; e_G: exponential operation over the group $\mathbb{G}$; e_{G_T}: exponential operation over the group $\mathbb{G}_T$; P: bilinear pairig; H: map-to-point hash function; N/A: Not applicable.



(a) User/UAV secret key (b) Ciphertext (c) Re-encryption key (d) Re-encrypted ciphertext

Fig. 3: Comparisons of storage costs



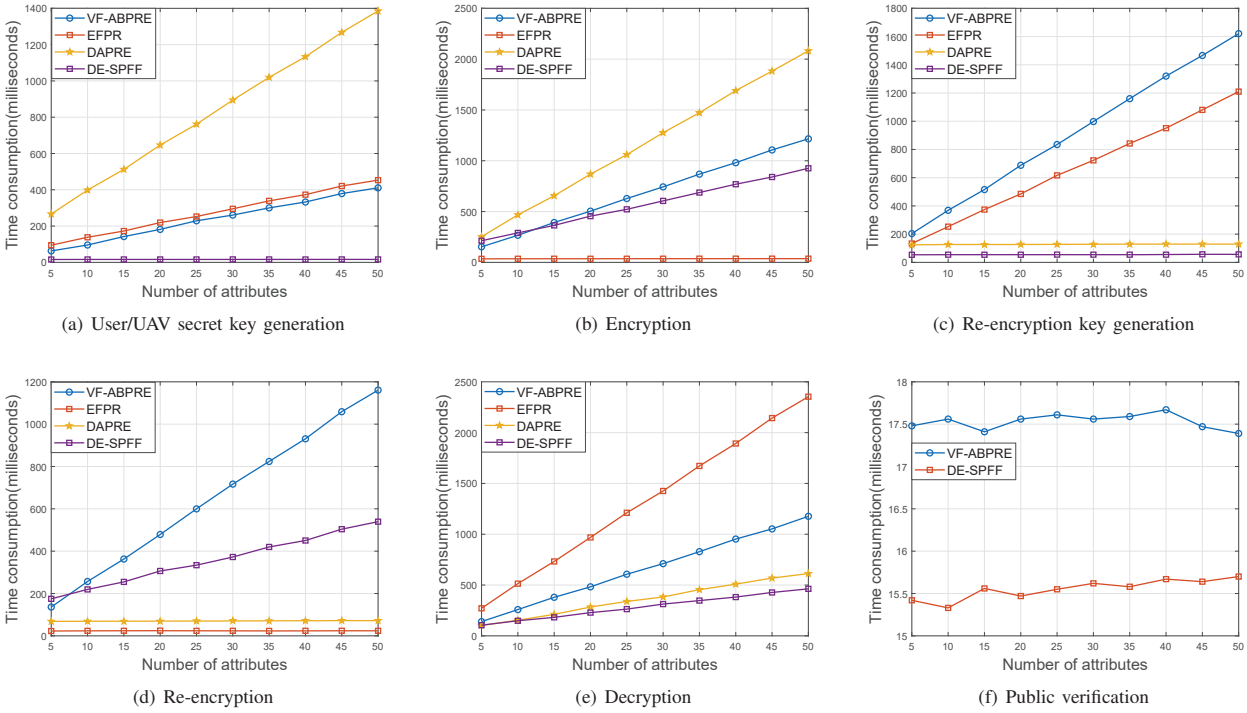
(a) UAV side (b) Server side

Fig. 5: Communication traffic and computational costs of the UAV and the server

UAVs and the server, respectively, during federated fine-tuning enabled by function encryption. We set the number of UAVs to increase exponentially from 2^1 to 2^6, and recorded the results for each round of federated fine-tuning, as shown in Fig. 5. Fig. 5-(a) shows the communication and computational time overhead on the UAV side. It shows that the overhead per UAV is nearly constant, with a communication overhead of 3MB and an average (encrypted) computational time of 297.36 seconds for one round of federated fine-tuning. Given the UAV hardware configuration and task execution cycle, these storage and computational time overheads are reasonable. Fig. 5-(b) shows the communication and computational time overhead on the server side. It shows that both server-side computational and communication overheads show a significant upward trend with the number of UAVs. When the number of UAVs reaches 64, the communication overhead for security and privacy protection in a round of global model fine-tuning on the server is 584.96 MB, and the computational time is 6291.2 seconds. These results are simulated on our personal computer. For the server in real scenarios, the above communication overhead is reasonable and the computing time will be significantly reduced.

VIII. Conclusion

This paper provides systematic solutions to several critical issues urgently needing resolution in federated LLM fine-tuning for UAV swarm networks. Specifically, we propose the SPFF scheme, which is based on an improved inner-product ABE algorithm. This algorithm can provide efficient and privacy-protected one-to-many sharing of downlink global parameters for federated LLM fine-tuning in UAV swarm networks. A secure upload mechanism for uplink local update parameters and a fine-tuning mechanism over encrypted parameters are constructed based on functional encryption. Aiming at the potential impact of UAV drop-out on the robustness of the federated LLM system, the scheme judges



(a) User/UAV secret key generation (b) Encryption (c) Re-encryption key generation

(d) Re-encryption (e) Decryption (f) Public verification

Fig. 4: Comparisons of computational costs

and processes the UAVs that have left the federated fine-tuning by designing an efficient revocation mechanism. On this basis, the DE-SPFF scheme designs an efficient and verifiable proxy re-encryption mechanism. This mechanism is used to provide task delegation for UAVs that can no longer participate in tasks and allows public verification of the validity of re-encrypted ciphertexts. Strict formal security proofs and performance comparison analyses show that the proposed schemes are secure, efficient, and practical. Despite the promising results, our proposed schemes have certain limitations that warrant further investigation. First, the current access control structure in DE-SPFF is limited to AND-gate policies, which may not fully capture complex real-world access requirements. Future work could explore more expressive policy structures, such as LSSS, while maintaining efficiency. Second, the reliance on a trusted authority for key management introduces a single point of failure and may not be fully aligned with the decentralized ethos of UAV swarms. Investigating decentralized key management mechanisms, such as blockchain-based approaches, could enhance system resilience and trust distribution. Third, while the schemes are designed for resource-constrained environments, the computational overhead of cryptographic operations remains non-negligible for extremely lightweight UAVs. Further optimizations, including hardware acceleration or more lightweight cryptographic primitives, could be explored to better suit ultra-low-power devices. Finally, the current security model assumes semi-honest edge nodes; extending the scheme to withstand fully malicious adversaries would strengthen its applicability in more hostile environments. Future research will also focus on large-scale real-world deployments and interoperability with existing UAV communication standards.

REFERENCES

[1] Wang X, Zhao Z, Yi L, et al. “A Survey on Security of UAV Swarm Networks: Attacks and Countermeasures,” *ACM Computing Surveys,* vol. 53, no. 3, pp. 1-37, 2024.

[2] Yu K, Zhou H, Xu Y, et al. “Large Sequence Model for MIMO Equalization in Fully Decoupled Radio Access Network,” *IEEE Open Journal of the Communications Society,* vol. 6, pp. 4491-4504, 2025.

[3] Yu K, Yu Q, Tang Z, et al. “Fully-decoupled radio access networks: A flexible downlink multi-connectivity and dynamic resource cooperation framework,” *IEEE Transactions on Wireless Communications,* vol. 22, no. 6, pp. 4202-4214, 2022.

[4] Chang Y, Wang X, Wang J, et al. “A survey on evaluation of large language models,” *ACM Transactions on Intelligent Systems and Technology,* vol. 15, no. 3, pp. 1-45, 2024.

[5] Chen C, Feng X, Zhou J, et al. “Federated large language model: A position paper,” *arXiv e-prints,* 2023: arXiv: 2307.08925.

[6] Rezazadeh A, Li Z, Lou A, et al. “Collaborative Memory: Multi-User Memory Sharing in LLM Agents with Dynamic Access Control,” *arXiv preprint arXiv*:2505.18279, 2025.

[7] Wang H, Xu J. “Friends to help: Saving federated learning from client dropout,” *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, pp. 8896-8900, 2024.

[8] Dhuheir M A, Baccour E, Erbad A, et al. “Deep reinforcement learning for trajectory path planning and distributed inference in resource-constrained UAV swarms,” *IEEE Internet of Things Journal,* vol. 10, no. 9, pp. 8185-8201, 2022.

[9] Sun J, Xu G, Li H, et al. “Sanitizable Cross-Domain Access Control With Policy-Driven Dynamic Authorization,” *IEEE Transactions on Dependable and Secure Computing,* 2025. DOI: 10.1109/TDSC.2025.3541819

[10] Sun J, Bao Y, Qiu W, et al. "Privacy-preserving fine-grained data sharing with dynamic service for the cloud-edge IoT," *IEEE Transactions on Dependable and Secure Computing,* vol. 22, no. 2, pp. 1329-1346, 2024.

[11] Qu B, Wang Z, Shen B, et al. "Secure particle filtering with Paillier encryptionCdecryption scheme: Application to multi-machine power grids," *IEEE Transactions on Smart Grid,* vol. 15, no. 1, pp. 863-873, 2023.

[12] Klein O, Komargodski I. "New bounds on the local leakage resilience of Shamirs secret sharing scheme," *Annual International Cryptology Conference. Cham: Springer Nature Switzerland,* LNCS, vol. 14081, pp. 139-170, 2023.

[13] Yang C, Jiang P, Zhu L. "Accelerating decentralized and partial-privacy data access for VANET via online/offline functional encryption," *IEEE Transactions on Vehicular Technology,* vol. 71, no. 8, pp. 8944-8954, 2022.

[14] Miao Y, Li F, Li X, et al. "Verifiable outsourced attribute-based encryption scheme for cloud-assisted mobile e-health system," *IEEE Transactions on Dependable and Secure Computing,* vol. 21, no. 4, pp. 1845-1862, 2023.

[15] Chang Y, Zhang K, Gong J, et al. "Privacy-preserving federated learning via functional encryption, revisited," *IEEE Transactions on Information Forensics and Security,* vol. 18, pp. 1855-1869, 2023.

[16] Luo F, Wang H, Susilo W, et al. "Public trace-and-revoke proxy re-encryption for secure data sharing in clouds," *IEEE Transactions on Information Forensics and Security,* vol. 19, no. 2919-2934, 2024.

[17] Wu F, Li Z, Li Y, et al. "Fedbiot: Llm local fine-tuning in federated learning without full model, *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining.* pp. 3345-3355, 2024.

[18] Das B C, Amini M H, Wu Y. "Security and privacy challenges of large language models: A survey," *ACM Computing Surveys,* vol. 57, no. 6, pp. 1-39, 2025.

[19] Yao Y, Zhang J, Wu J, et al. "Federated large language models: Current progress and future directions," arXiv preprint arXiv:2409.15723, 2024.

[20] Zhang J, Vahidian S, Kuo M, et al. "Towards building the federatedgpt: Federated instruction tuning," *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE,* pp. 6915-6919, 2024.

[21] Zhang Z, Zhang J, Huang J, et al. "FewFedPIT: Towards Privacy-preserving and Few-shot Federated Instruction Tuning," arXiv preprint arXiv:2403.06131, 2024.

[22] Sun Y, Li Z, Li Y, et al. "Improving lora in privacy-preserving federated learning," arXiv preprint arXiv:2403.12313, 2024.

[23] Nguyen T, Thai M T. "Preserving privacy and security in federated learning," *IEEE/ACM Transactions on Networking,* vol. 32, no. 1, pp. 833-843, 2023.

[24] Zhang X, Fu A, Wang H, et al. "A privacy-preserving and verifiable federated learning scheme," *ICC 2020-2020 IEEE International Conference on Communications (ICC).* IEEE, pp. 1-6, 2020.

[25] Jin W, Yao Y, Han S, et al. "FedML-HE: An efficient homomorphic-encryption-based privacy-preserving federated learning system," arXiv preprint arXiv:2303.10837, 2023.

[26] Xu G, Li H, Liu S, et al. "VerifyNet: Secure and verifiable federated learning," *IEEE Transactions on Information Forensics and Security,* vol. 15, pp. 911-926, 2019.

[27] Yu H, Xu R, Zhang H, et al. "EV-FL: Efficient verifiable federated learning with weighted aggregation for industrial IoT networks," *IEEE/ACM Transactions on Networking,* vol. 32, no. 2, pp. 1723-1737, 2023.

[28] Yin X, Qiu H, Wu X, et al. "An Efficient Attribute-Based Participant Selecting Scheme with Blockchain for Federated Learning in Smart Cities," *Computers,* vol. 13, no. 5, pp. 118, 2024.

[29] Wang Y, Su Z, Zhang N, et al. "Learning in the Air: Secure Federated Learning for UAV-assisted Crowdsensing," *IEEE Transactions on Network Science and Engineering,* vol. 8, no. 2, pp. 1055-1069, 2020.

[30] Hou X, Wang J, Jiang C, et al. "UAV-enabled covert federated learning," *IEEE Transactions on Wireless Communications,* vol. 22, no. 10, pp. 6793-6809, 2023.

[31] Xu Q, Lan Y, Su Z, et al. "Verifiable and privacy-preserving cooperative federated learning in UAV-assisted vehicular networks," *ICC 2023-IEEE International Conference on Communications.* IEEE, 2023: 2288-2293.

[32] Xu R, Joshi J B D, Li C. "Cryptonn: Training neural networks over encrypted data," *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS).* IEEE, pp. 1199-1209, 2019.

[33] Phuong T V X, Yang G, Susilo W. "Hidden ciphertext policy attribute-based encryption under standard assumptions," *IEEE Transactions on Information Forensics and Security,* vol. 11, no. 1, pp. 35-45, 2015.

[34] Sun J, Xiong H, Liu X, et al. "Lightweight and privacy-aware fine-grained access control for IoT-oriented smart health," *IEEE Internet of Things Journal,* vol. 7, no. 7, pp. 6566-6575, 2020.

[35] Xu G, Xu S, Ma J, et al. "An adaptively secure and efficient data sharing system for dynamic user groups in cloud," *IEEE Transactions on Information Forensics and Security,* vol. 18, pp. 5171-5185, 2023.

[36] Wang Z, Shen Z, He Y, et al. "Flora: Federated fine-tuning large language models with heterogeneous low-rank adaptations". *arXiv preprint* arXiv:2409.05976, 2024.

[37] Bian J, Wang L, Zhang L, et al. "LoRA-FAIR: Federated LoRA Fine-Tuning with Aggregation and Initialization Refinement," *arXiv preprint arXiv:2411.14961,* 2024.

[38] Bellare M, Keelveedhi S. "Interactive message-locked encryption and secure deduplication," *Public-Key Cryptography–PKC 2015: 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30–April 1, 2015, Proceedings 18.* Springer Berlin Heidelberg, LNSC, vol. 9020, pp. 516-538, 2015.

[39] Yan Y, Yang Q, Tang S, et al. "Federa: Efficient fine-tuning of language models in federated learning leveraging weight decomposition," arXiv preprint arXiv:2404.18848, 2024.

[40] Bao Y, Qiu W, Cheng X, et al. "Fine-grained data sharing with enhanced privacy protection and dynamic users group service for the IoV," *IEEE Transactions on Intelligent Transportation Systems,* vol. 24, no. 11, pp. 13035-13049, 2022.

[41] Ge C, Susilo W, Baek J, et al. "A verifiable and fair attribute-based proxy re-encryption scheme for data sharing in clouds," *IEEE Transactions on Dependable and Secure Computing,* vol. 19, no. 5, pp. 2907-2919, 2021.

[42] Bellare M, Keelveedhi S, Ristenpart T. "Message-locked encryption and secure deduplication," *Annual international conference on the theory and applications of cryptographic techniques. Berlin, Heidelberg: Springer Berlin Heidelberg,* LNSC, vol. 7881, pp. 296-312, 2013.

[43] Waters B. "Efficient identity-based encryption without random oracles," *Annual international conference on the theory and applications of cryptographic techniques. Berlin, Heidelberg: Springer Berlin Heidelberg,* LNSC, vol. 3494, pp. 114-127, 2005.

[44] Bagherzandi A, Cheon J H, Jarecki S. "Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma," *Proceedings of the 15th ACM conference on Computer and communications security.* pp. 449-458, 2008.

[45] Zhang Q, Fu Y, Cui J, et al. "Efficient Fine-grained Data Sharing Based on Proxy Re-encryption in IIoT," *IEEE Transactions on Dependable and Secure Computing,* vol. 21, no. 6, pp. 5797-5809, 2024.

[46] Li X, Xie Y, Wang H, et al. "dAPRE: Efficient and Reliable Attribute-Based Proxy Re-Encryption Using DAG for Data Sharing in IoT," *IEEE Transactions on Consumer Electronics,* vol. 70, no. 1, pp. 584-596, 2023.

[47] Lynn B. "Pbc library-pairing-based cryptography," http://crypto. stanford. edu/pbc/, 2007.