Towards Privacy-Preserving Unmanned Aerial Vehicles Shared Logistics via Dynamic Sanitizable Signature with Multiple Sanitizers

Mingwei Zeng, Qingyang Zhang, Jie Cui *Senior Member, IEEE*, Hong Zhong, and Fengqun Wang

***Abstract*—In recent years, unmanned aerial vehicles (UAVs) have shown great potential in logistics delivery due to their ability to bypass traffic congestion and adapt to complex terrains. Their high efficiency, low cost, and wide coverage make them a valuable supplement to last-mile logistics. However, third-party UAV systems operating in open environments are vulnerable to eavesdropping, tampering, and other cyber-attacks, which poses risks of sensitive information leakage. Meanwhile, warehouse nodes for sanitizing private information are widely deployed in logistics systems and need to be dynamically adjusted according to demand, which poses a challenge for the management of sanitization permissions. To address these issues, we propose a dynamic sanitizable signature with multiple sanitizers (DSSMS), enabling each sanitizer to independently sanitize sensitive information in signed messages, thus preserving logistics data privacy. The scheme supports joining and revoking sanitizers while retaining the long-term keys of existing members. Under the stated assumptions, the scheme provides external unforgeability and protects removed content in the final sanitized view. An implementation on AmovLab Prometheus 600 UAVs demonstrates lower computational, communication and energy consumption cost than existing privacy-preserving schemes, confirming its practicality in UAV logistics systems.**

***Index Terms*—Privacy-Preserving, Sanitizable Signature, Unmanned Aerial Vehicle Logistics.**

I. Introduction

In recent years, the low-altitude economy has rapidly developed as a new economic paradigm, with unmanned aerial vehicles emerging as a core enabling technology due to their flexibility and efficiency [1]. UAV-based logistics shows significant potential [2], particularly in addressing last-mile delivery in urban environments. UAV delivery reduces delivery time through optimized aerial routes [3], [4]. Moreover, UAVs can operate autonomously around the clock and reach areas that are inaccessible to traditional couriers [5], [6]. Therefore, many companies, including Amazon [7], DHL [8], Meituan, and SF Express [9] have launched pilot deployments of UAV delivery systems. Notably, Amazon introduced Amazon Prime Air to enhance customer delivery services through UAV technology. Meanwhile, the Federal Aviation Administration (FAA), Google and Walmart are also actively promoting the adoption of UAVs in urban logistics [10].

M. Zeng, Q. Zhang, J. Cui, H. Zhong and F. Wang are with the Key Laboratory of Intelligent Computing and Signal Processing of Ministry of Education, School of Computer Science and Technology, Anhui University, Hefei 230039, China, and the Anhui Engineering Research Center for IoT Security Technologies, Anhui University, Hefei 230039, China (e-mail: qyzhang@ahu.edu.cn).

With the development of the low-altitude economy and continued investment by logistics companies in UAV delivery [11], we have observed the emergence of an independently operated low-altitude last-mile logistics network. To systematically characterize this evolving trend, we abstract a system architecture for future urban UAV logistics based on typical industry practices. This architecture involves five types of entities: a trusted authority, an order initiator, a logistics company, warehouse nodes W_i, and UAVs operated by third-party providers. When a user submits a delivery request via a relevant online platform, the order initiator transmits the order information to the logistics company [12], which is responsible for overall coordination. The logistics company selects the optimal warehouse node for dispatch, delivers the package to a designated UAV launch site near the recipient, and finally, the UAV completes the last-mile delivery task.

To prevent orders from being forged or tampered with, and to ensure that in the event of a dispute (such as the delivered package not matching the order), the UAV can prove that the package indeed originates from the order initiator and that relevant parties can verify its source, the order initiator usually generate a digital signature over the order and transmit it and the order to the logistics company [13]. Because order data typically contains sensitive information such as package details, recipient names, and contact numbers, and UAVs are often operated by third parties in environments vulnerable to eavesdropping or tampering [14], directly transmitting the complete order may result in serious privacy leakage [15]. Before forwarding an order to the UAV, the warehouse removes sensitive fields. The order initiator binds delivery-critical fields in the immutable part of the signed order [16].

On the other hand, logistics companies typically operate numerous warehouse nodes across the country. As business scales or operational strategies evolve, these nodes may be frequently added or removed. While centralized removal of sensitive information by the logistics company may seem more straightforward [17], it may lead to performance bottlenecks and creates a potential single point of failure. To better adapt to such dynamic environments, the system should allow each warehouse node to remove sensitive information in the order independently within the shared editable part. Additionally, it should support a highly flexible permission management mechanism, including the dynamic assignment of permissions to new warehouse nodes and timely revocation of permissions for retired ones [18], thereby ensuring that sensitive information is securely and accurately managed throughout the entire

unmanned aerial vehicles logistics process [19].

Sanitizable signature [20] allows an authorized sanitizer to modify designated fields in a signed message, enabling selective removal of sensitive information. However, existing schemes support only a single sanitizer, while extending them to multiple sanitizers either requires complex group signatures or separate signatures for each sanitizer, resulting in high overhead [21]. To overcome these limitations, we introduce a new sanitizable signature supporting multiple sanitizers with dynamic permission management. Our design leverages the chinese remainder theorem to enable efficient delegation and revocation of sanitization rights [22], ensures signer traceability for accountability, and offers a scalable and practical solution for order privacy protection in logistics networks. Our main contributions are summarized as follows:

- **Algorithm Design.** We construct a dynamic sanitizable signature with multiple sanitizers (DSS-MS) utilizing the chinese remainder theorem and chameleon hash. This scheme allows multiple sanitizers to independently sanitize signatures while supporting efficient permission delegation and revocation. Furthermore, it employs a non-interactive zero-knowledge proof for bidirectional weak accountability between the signer and the sanitizer group.
- **Privacy-Preserving Protocol for UAV Logistics.** Based on practical logistics requirements, we design a privacy-preserving protocol that integrates order signing, warehouse sanitization, and UAV verification into the logistics workflow. The protocol enables authorized warehouses to remove sensitive information from orders while preserving signature validity, allowing UAVs to verify order authenticity using only the necessary delivery information. This ensures both user privacy protection and order authenticity verification. Furthermore, in the event of disputes over order content, the protocol supports the generation and verification of accountability evidence, providing a basis for distinguishing responsibility between the signer and the sanitizer group.
- **Performance Evaluation**. We formally prove that the proposed scheme satisfies a range of security properties. We also implement it on UAV hardware and compare it with existing approaches. Experimental results show that our scheme significantly reduces computational and communication overhead. The code is open-sourced at https://github.com/rookie-papers/DSS-MS.git.

Organization. The remainder of this paper is organized as follows. Section III reviews related work. Section II covers cryptographic preliminaries. Section IV presents the system model, and Section V describes the design of the proposed DSS-MS scheme. Section VI shows its integration into a UAV logistics system. Section VII provides the security analysis, and Section VIII reports experimental results and comparisons.

II. Preliminaries

To facilitate a clear understanding of our DSS-MS, we introduces the foundational concepts employed in our scheme.

Chinese Remainder Theorem (CRT). It provides a way to reconstruct integers from their remainders modulo several pairwise coprime moduli. Let $n_1, n_2, \ldots, n_k$ be pairwise coprime positive integers, and let $M = \prod_{i=1}^{k} n_i$. For any $a_1, \ldots, a_k$, the system of congruences $x \equiv a_i \pmod{n_i}, \quad i = 1, \ldots, k$, has a unique solution modulo M. The solution can be computed as $x \equiv \sum_{i=1}^{k} a_i \cdot M_i \cdot y_i \pmod{M}$, where $M_i = \frac{M}{n_i}$ and y_i is the multiplicative inverse of M_i modulo n_i.

Elliptic Curve Cryptography. An elliptic curve E over a prime field $\mathbb{F}_p$ is typically defined as follow: $E : y^2 = x^3 + ax + b,\ a, b \in \mathbb{F}_p$, where the discriminant $\Delta = 4a^3 + 27b^2 \neq 0$. The set of points on the curve, together with infinity $\mathcal{O}$, forms an abelian group. The group operation is defined as follows:

- **Point Addition:** For distinct points P and Q, $R = P + Q$ is the reflection over the x-axis of the third point where the line through P and Q intersects the curve.
- **Point Doubling:** For $P = Q$, the operation $R = 2P$ is defined as the reflection across the x-axis of the point where the curve intersects the tangent line at P.
- **Scalar Multiplication:** Given P and an integer k, $k \cdot P$ is defined as repeated addition of P with itself k times.

III. Related Work

We first review existing work on sanitizable signatures and privacy-preserving mechanisms for sensitive information.

Sanitizable Signatures. Sanitizable signatures, first introduced by Ateniese et al. [20], allow an authorized sanitizer to modify designated parts of a signed message while preserving signature validity. Brzuska et al. [23] formalized the security properties of such schemes and demonstrated that the original design [20] satisfies key requirements. Later, they extended the model by introducing unlinkability [21], ensuring that different sanitized versions cannot be linked to the same origin. However, their group signature-based construction incurs significant computational overhead. Fleischhacker et al. [24] addressed this limitation by proposing a construction with perfectly re-randomizable keys. Subsequent works explored alternative cryptographic tools to enhance flexibility. Camenisch et al. [25] leveraged chameleon hash with ephemeral trapdoors [26] to give the signer fine-grained sanitization control. Zhan et al. [27] proposed a bounded sanitizable signature scheme that limits the number and duration of allowed modifications, improving privacy granularity but incurring pairing overhead. Yang et al. [28] introduced sanitizable blind signatures and applied them to privacy-preserving key agreement and handover in mobile networks. To eliminate the reliance on complex certificate management and costly pairing operations, Sun et al. [29] proposed a pairing-free certificateless sanitizable signature scheme with a designated verifier. Most recently, Allabwani et al. [30] extended the paradigm to multi-sanitizer environments, introducing the IUT-k-SAN scheme to support independent admissibility policies for distinct sanitizers while ensuring invisibility and unlinkability.

Despite these advances, single-sanitizer schemes typically bind sanitization rights to a designated sanitizer's key. Extending such schemes by generating separate signatures for multiple sanitizers increases signing and transmission costs with the number of sanitizers. While multi-sanitizer constructions, such as IUT-k-SAN, support multiple sanitizers, they rely on expensive bilinear pairings, and efficient dynamic sanitizer addition and revocation remain challenging.

Sensitive Data Privacy Protection. Sanitizable signatures have been applied to various privacy-preserving data sharing scenarios. Shen et al. [31] used block-level signatures to support privacy-preserving integrity verification in cloud storage. Liu et al. [32] proposed an identity-based sanitizable signature scheme to simplify certificate management, though it suffers from pairing overhead. Zhan et al. [27] applied bounded sanitizable signatures to electronic health record (EHR) sharing, enabling fine-grained control over patient data. Beyond sanitizable signatures, Gao et al. [33] introduced Relation Authentication Labels (RAL) to enable privacy-preserving auditing. Li et al. [34] combined signature aggregation and sanitization in a blockchain-based system for industrial IoT, achieving efficient data aggregation with privacy guarantees.

Functionality Comparison. We compare the features of our scheme with those of existing schemes in Table I, including multi-sanitizer support (**F1**), weak accountability or tracing mechanisms that determine signature origin (**F2**), sanitizer addition (**F3**) and revocation (**F4**), retention of existing sanitizers' long-term private keys during membership changes (**F5**), unlinkability (**F6**), the ability to change the set of editable message blocks after signing (**F7**), and proof models (**F8**). Checkmarks indicate supported features, while crosses indicate features that are unsupported or not explicitly specified. For F8, R, C, G, and A denote the random oracle model, proofs based on the security of underlying components, the generic group model, and the algebraic group model, respectively.

TABLE I
COMPARISON WITH OTHER PROTOCOLS

Scheme	F1	F2	F3	F4	F5	F6	F7	F8
IBSDIA [35]	✘	✘	✘	✘	✘	✘	✘	R
BFLS10 [21]	✔	✔	✘	✘	✘	✔	✘	C
DVAS [34]	✘	✔	✘	✘	✘	✘	✘	R
PF-CLSS-DV [29]	✘	✘	✘	✘	✘	✘	✘	R
IUT-k-SAN [30]	✔	✔	✘	✘	✔	✔	✘	R/G
ECMSS [36]	✔	✘	✘	✘	✘	✘	✘	R
LSanS [37]	✔	✔	✔	✔	✔	✘	✘	R
PGUS-SBS [28]	✘	✔	✘	✘	✘	✔	✘	R/C
DSS-MS	✔	✔	✔	✔	✔	✘	✘	R/A

Note: IUT-k-SAN and DSS-MS attribute responsibility to the signer or sanitizer group, not individual sanitizers, but differ in accountability scope.

IV. System Architecture

In this section, we introduce the system model, which is illustrated in Fig. 1 and describes the system architecture.

- **TA (Trusted Authority):** It is a trusted entity, typically operated by government agencies or other authoritative institutions. It sets up the system and sanitizer keys, signs epoch manifests, and certifies entity public keys.
- **OI (Order Initiator):** The order initiator, typically an entity such as an e-commerce platform or a medical institution, receives a user's purchase request and generates an order containing details about the requested package and the user's sensitive information. It signs the order using its private key to ensure authenticity and integrity, and then forwards it to the logistics company.
- **LC (Logistics Company):** Based on the order from the order initiator, the logistics company selects a geographically optimal warehouse W_i for dispatch. The logistics company requests membership changes from the TA.
- **Warehouse Node (W_i):** Warehouse nodes are managed by the LC. Each authorized warehouse node W_i acts as a sanitizer S_i in DSS-MS. To protect user privacy, the selected warehouse sanitizes sensitive order fields before handing the package over to the UAV.
- **UAV (Unmanned Aerial Vehicle):** UAVs are usually operated by third-party providers and are responsible for the final stage of package delivery. Considering that UAVs may be subject to potential security threats, they receive the sanitized order and the fields needed for delivery. Instead, the UAV verifies the order's integrity and authenticity before completing the delivery, ensuring that the data has not been tampered with during transmission.

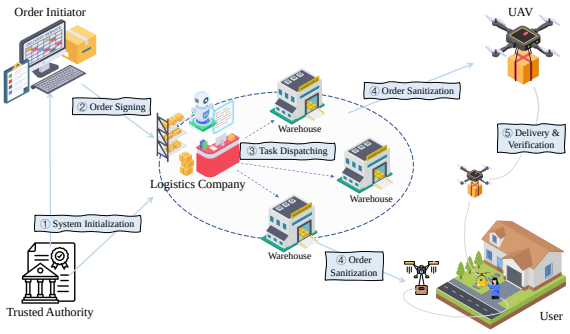

Fig. 1. Our architecture of the proposed UAV logistics system.

V. The Proposed DSS-MS Scheme

In this section, we formalize the syntax of DSS-MS, and define the threat model along with the desired security goals.

A. Dynamic Sanitizable Signature with Multiple Sanitizers

DSS-MS consists of the following PPT algorithms. Section V-E specifies their inputs and checks in detail.

- $pp \leftarrow \mathsf{Setup}(1^\lambda)$ sets the group and the public cryptographic primitives.
- $\mathsf{KGen}(pp, n)$ provisions the sanitizer keys K_i and public moduli p_i and generates the TA authentication pair (ak, vk). The signer locally generates (sk, PK).
- $\mathsf{EpochGen}(pp, A_e, e)$ uses the TA state to issue the certified manifest $param_e$. $\mathsf{Recover}(K_i, param_e)$ returns $sk_{s,e}$ to an authorized member or fails with $\perp$.
- $\mathsf{Sign}(pp, M, sk, param_e)$ returns σ on $M = (m_0, m)$ and retains the original randomness. $\mathsf{Sanitizing}(\sigma, M, m', PK, K_i, param_e)$ returns a valid signature σ' on $M' = (m_0, m')$ if the input is accepted.
- $\mathsf{Verify}(M, \sigma, PK, param_e)$ returns 1 for a valid signature and 0 otherwise.
- Proof uses a stored original record and the disputed pair to generate π. $\mathsf{Judge}(\pi, PK, param_e)$ returns `san` for accepted evidence or $\perp$ if it rejects the evidence.
- $\mathsf{Join}(A_e, J)$ provisions new members and $\mathsf{Revoke}(A_e, I)$ removes members from the authorized set. Both publish $param_{e+1}$ through $\mathsf{EpochGen}$.

B. Threat Model

The sanitized-content privacy guarantee applies to the final sanitized order and its signature under the privacy experiment's conditions. In the proposed logistics workflow, communications carrying original orders or secret key material are assumed to use authenticated encrypted channels, such as TLS 1.3. This assumption covers both the OI-to-LC and LC-to-warehouse links that carry the original order. The TA is fully trusted. We consider two types of adversaries:

$\mathcal{A}_1$**:** Following the Dolev–Yao model [38], $\mathcal{A}_1$ can observe, modify, delay, and replay network traffic. It is probabilistic polynomial-time (PPT), cannot efficiently solve the underlying cryptographic hard problems, and cannot directly access secrets securely stored by trusted entities.

$\mathcal{A}_2$**:** In addition to the capabilities of $\mathcal{A}_1$, $\mathcal{A}_2$ can corrupt UAVs or warehouses, obtain the sanitizer secret keys of corrupted warehouses, and collude with other parties. Each security experiment in Fig. 2 specifies the permitted corruptions and key exposures. Revoked members may retain and pool their old keys and all public epoch parameter manifests.

C. Security goals

To support dynamic sanitizer management and trustworthy message sanitization, the proposed DSS-MS scheme is designed to meet the following security goals.

Correctness. For every epoch e, each legitimately authorized sanitizer can recover the sanitization key using the public parameters and its own long-term secret. All legitimately generated original signatures, sanitized signatures, and attribution proofs pass their corresponding verification procedures.

Dynamic Sanitization-Key Confidentiality. For a prespecified target epoch e^*, recovery of sk_{s,e^*} is infeasible from all public manifests, keys of non-target epochs, and long-term keys of nonmembers of A_{e^*}, provided the TA, target key, and all K_i for $i \in A_{e^*}$ remain unexposed.

Revocation and Join Security. Under the same exposure restrictions, revoked members cannot recover later epoch keys while unauthorized. Newly admitted members cannot recover keys of epochs in which they lacked authorization.

External Unforgeability. No unauthorized entity can generate a valid signature on a new message without a legitimate signing or sanitizing operation. An authorized sanitizer, however, can modify a message and its signature within the permitted scope while preserving signature validity.

Immutability. Even if all sanitizers collude, they cannot produce a valid signature for immutable content m_0 never authorized by the signer. An authorized sanitizer can replace the entire editable part m.

Transparency. For the same final message within a given epoch, directly generated and legitimately sanitized signatures are computationally indistinguishable in distribution. Thus, a verifier cannot determine whether the final signature was directly generated or obtained through legitimate sanitization based on the final message and its signature alone.

Sanitized Content Privacy. Sanitization protects sensitive content replaced in the original message. For distinct original messages with identical immutable content, public information, and final sanitized results, an external entity cannot distinguish which original message was used based solely on the final sanitized message and its signature.

Bidirectional Weak Accountability. An honest signer can attribute a valid new, unsigned message to the sanitizers as a group. A malicious signer without the target trapdoor cannot obtain accepted evidence against that group for a message never output by the sanitizing interface.

Parameter Authentication and Epoch Binding. The system ensures the authenticity and integrity of public parameters for each epoch and binds them to the epoch identifier, authorized sanitizer set, and sanitization public key. A receiver rejects uncertified parameters or those preceding its trusted minimum epoch. Signing, sanitization, and verification use the parameters associated with the message's epoch, preventing cross-epoch parameter mixing and rollback below the receiver's trusted minimum epoch.

D. Security definition

Let $\mathcal{O}ut \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{Sign}}(\cdot),\mathcal{O}_{\mathrm{Sanit}}(\cdot),\mathcal{O}_{\mathrm{Proof}}(\cdot),\cdots}(\mathcal{I})$ denote that an adversary $\mathcal{A}$, given input $\mathcal{I}$ and oracle access, outputs $\mathcal{O}ut$. We first fix the common notation and the adversarial capabilities, and then formalize the security properties through the experiments in Fig. 2.

Notation and model. Let $G = \langle P \rangle$ be an additive group of prime order q with $\log_2 q = \Theta(\lambda)$, and let $H_0, H_1, H_2 : \{0,1\}^* \to \mathbb{Z}_q$ be mutually independent, domain-separated hash functions. All tuple encodings are canonical and unambiguous. Let $F_{K_i}^{(p_i)}$ be a PRF with range $\mathbb{Z}_{p_i}$ and (ak, vk) be an ordinary authentication key pair of the TA.

The sanitizer set is dynamic. Each epoch has a monotonically increasing identifier e and an authorized set A_e. A membership change starts a new epoch. An authorized set may recur in a later epoch. The parameters of that epoch are $param_e = (ctx_e, S_{x,e}, \tau_e)$, where ctx_e binds the system identifier sid, the epoch e, a fresh nonce ν_e, A_e, the public moduli $\{(i, p_i)\}_{i \in A_e}$ and the sanitization public key $PK_{s,e}$. $S_{x,e}$ is the CRT parameter. And $\tau_e = \mathrm{AuthSign}_{ak}(\mathrm{Enc}(\mathcal{P}_e))$ is the TA's signature on $\mathcal{P}_e = (ctx_e, S_{x,e})$. Only one valid $param_e$ is issued per system and epoch. A receiver persistently stores its trusted lowest epoch $e_{\min}$ and the identifier of the manifest already accepted for that epoch. State updates are monotone, atomic and cannot be rolled back.

The signer key pair is $sk \in \mathbb{Z}_q^*$ and $PK = skP$. The long-term PRF key $K_i \leftarrow \{0,1\}^{\ell_K}$ of node i and the public CRT modulus $p_i > q$ are independent objects: the moduli $\{p_i\}$ are pairwise coprime and public, while K_i is held only by the node. The sanitization key pair is generated independently for each epoch as $sk_{s,e} \leftarrow \mathbb{Z}_q^*$ and $PK_{s,e} = sk_{s,e}P$.

Adversarial capabilities. Each experiment fixes the target signer, system, and epoch where applicable. The adversary may interleave the listed interfaces subject to the exposure restrictions of that experiment. The lists start empty. $\mathcal{I}$ denotes the public inputs provided by the experiment. Each execution uses consistent key state and oracle tables. Where no changes are allowed, the challenger initializes one active target epoch.

Exp. $\mathsf{KeyConfidentiality}^{\mathsf{DSS\text{-}MS}}_{\mathcal{A}}(\lambda)$

1: $\mathcal{A}$ fixes a reachable (e^*, A^*). Run Setup, KGen.
2: Generate history through e^* with $A_{e^*} = A^* \neq \varnothing$.
3: Give pp, PK, vk, public moduli, and the issued manifests to the adversary $\mathcal{A}$.
4: Keep the TA secrets, sanitization key sk_{s,e^*}, and all K_i with $i \in A^*$ unexposed.
5: $x^* \leftarrow \mathcal{A}(\mathcal{I})$ with $\mathcal{O}_{\mathrm{Sign}}, \mathcal{O}_{\mathrm{Sanit}}, \mathcal{O}_{\mathrm{Proof}}, \mathcal{O}_H, \mathcal{O}_{\mathrm{Update}}, \mathcal{O}_{\mathrm{Corrupt}}, \mathcal{O}_{\mathrm{EpochReveal}}$
6: Return 1 iff $x^* \in \mathbb{Z}_q^*$ and $x^*P = PK_{s,e^*}$.

Exp. $\mathsf{Unforgeability}^{\mathsf{DSS\text{-}MS}}_{\mathcal{A}}(\lambda)$

1: Fix target epoch e and initialize as in KeyConfidentiality with $e^* = e$.
2: Also keep sk unexposed.
3: $(M^*, \sigma^*) \leftarrow \mathcal{A}(\mathcal{I})$ with $\mathcal{O}_{\mathrm{Sign}}, \mathcal{O}_{\mathrm{Sanit}}, \mathcal{O}_{\mathrm{Proof}}, \mathcal{O}_H, \mathcal{O}_{\mathrm{Update}}, \mathcal{O}_{\mathrm{Corrupt}}$
4: If $(PK, e, M^*) \in L_{\mathrm{out}}$, return 0.
5: If $\mathsf{Verify}(M^*, \sigma^*, PK, param_e) \neq 1$, return 0
6: return 1

Exp. $\mathsf{Immutability}^{\mathsf{DSS\text{-}MS}}_{\mathcal{A}}(\lambda)$

1: Use the same setup at epoch e. Protect sk.
2: Give all K_i and the current $sk_{s,e}$ to $\mathcal{A}$
3: $(M^*, \sigma^*) \leftarrow \mathcal{A}(\mathcal{I})$ with $\mathcal{O}_{\mathrm{Sign}}, \mathcal{O}_{\mathrm{Sanit}}, \mathcal{O}_{\mathrm{Proof}}, \mathcal{O}_H$
4: If m_0^* was ever issued by $\mathcal{O}_{\mathrm{Sign}}$, return 0
5: If $\mathsf{Verify}(M^*, \sigma^*, PK, param_e) \neq 1$, return 0
6: return 1

Exp. $\mathsf{Transparency}^{\mathsf{DSS\text{-}MS}}_{\mathcal{A}}(\lambda)$

1: Initialize as above. Fix epoch e, final message (m_0, m') and public metadata
2: Choose $b \leftarrow \{0, 1\}$
3: $b = 0$: direct signature. $b = 1$: sign-then-sanitize
4: Send one final signature. Reserve its challenge record identifier.
5: Obtain b' with restricted access to the sign, sanitize, proof, and hash interfaces.
6: return 1 if $b' = b$ else return 0

Exp. $\mathsf{Privacy}^{\mathsf{DSS\text{-}MS}}_{\mathcal{A}}(\lambda)$

1: $\mathcal{A}$ chooses $M^{(0)}, M^{(1)}$, $\mathcal{L}(M^{(0)}) = \mathcal{L}(M^{(1)})$
2: Both are sanitized to the same final message $M' = (m_0, m')$
3: Generate $b \leftarrow \{0, 1\}$ and the challenge from $M^{(b)}$
4: Return (M', σ') and reserve its challenge record identifier.
5: Obtain b' with the same query restrictions as for transparency.
6: Return 1 iff $b' = b$.

Exp. $\mathsf{Accountability\text{-}I}^{\mathsf{DSS\text{-}MS}}_{\mathcal{A}}(\lambda)$

1: The signer keeps sk and stores all original message signature records together with the randomness t
2: Give $sk_{s,e}$ and all K_i to $\mathcal{A}$
3: $(M, \sigma) \leftarrow \mathcal{A}(\mathcal{I})$ with $\mathcal{O}_{\mathrm{Sign}}, \mathcal{O}_{\mathrm{Proof}}, \mathcal{O}_H$
4: Return 0 if Verify rejects or $\mathcal{O}_{\mathrm{Sign}}$ returned M.
5: The honest signer finds the originating record and runs Proof and Judge.
6: Return 1 if no record is found or the result is $\perp$. Otherwise return 0.

Exp. $\mathsf{Accountability\text{-}II}^{\mathsf{DSS\text{-}MS}}_{\mathcal{A}}(\lambda)$

1: Give sk to $\mathcal{A}$. Protect the TA secrets, $sk_{s,e}$, and the authorized K_i.
2: Record output messages of $\mathcal{O}_{\mathrm{Sanit}}$ in L^M_{san}.
3: Obtain π from $\mathcal{A}$ with $\mathcal{O}_{\mathrm{Sanit}}, \mathcal{O}_H$.
4: Parse the disputed pair (M, σ) in π. Return 0 on failure or if $M \in L^M_{\mathrm{san}}$.
5: Return 1 iff $\mathsf{Judge}(\pi, PK, param_e) = \mathtt{san}$.

Exp. $\mathsf{ParameterBinding}^{\mathsf{DSS\text{-}MS}}_{\mathcal{A}}(\lambda)$

1: $(ak, vk) \leftarrow$ Auth.KGen. The receiver stores $e_{\min}$
2: Send vk and all historical manifests to the adversary.
3: The receiver verifies τ_e and checks $e \geq e_{\min}$, uniqueness and validity
4: return 1 if an unissued $\mathcal{P}^*$ is accepted
5: Return 1 if a manifest below $e_{\min}$ or with different content for the same accepted epoch is accepted.

Exp. $\mathsf{CollisionResistance}^{\mathsf{DSS\text{-}MS}}_{\mathcal{A}}(\lambda)$

1: Fix a target epoch as in KeyConfidentiality.
2: Give the public inputs and authorized oracle access as specified below.
3: Obtain j and an opening (m', s', T') with its algebraic representation.
4: Return 1 iff it opens the issued CH_j under $m_{0,j}$ and is absent from L_j.

Fig. 2. Security experiments for epoch-key confidentiality, external unforgeability, immutability, transparency, sanitized-content privacy, bidirectional weak accountability, manifest binding, and collision resistance of honest commitments.

- $\mathcal{O}_{\mathrm{Sign}}$: returns an original signature $\sigma = (R, z, s, T)$ on a given (m_0, m), adds (PK, e, M) to L_{out}, and lets the honest signer secretly retain the randomness t.
- $\mathcal{O}_{\mathrm{Sanit}}$: sanitizes only valid signatures within the same certified epoch. On success it records both pairs in L_{san}, adds (PK, e, M') to L_{out} and records the output message M' in L^M_{san} for the corresponding signer and epoch.
- $\mathcal{O}_{\mathrm{Proof}}$: generates evidence only for disputes supported by a stored, honestly generated original record.
- $\mathcal{O}_{H_j}$: returns a stored response if available. Otherwise, samples, records, and returns a fresh random value.
- $\mathcal{O}_{\mathrm{Update}}$: processes a Join or Revoke request, returning the next authenticated $param_e$. It cannot modify issued epochs or reset the game's key-exposure restrictions.
- $\mathcal{O}_{\mathrm{Corrupt}}$: returns the long-term key K_i of any node whose corruption is permitted during the experiment.
- $\mathcal{O}_{\mathrm{EpochReveal}}$: returns the sanitization key $sk_{s,e}$ of any non-challenge epoch e to the adversary.
- $\mathcal{O}_{\mathrm{Publish}}$: returns a legitimate historical parameter manifest and its authentication signature.
- $\mathcal{O}_{\mathrm{Adapt}}$: takes an issued honest commitment index j and a replacement m', returns a legitimate opening of CH_j for m', and appends that opening to L_j. The initial opening of CH_j is entered in L_j on issuance.

In the transparency and privacy games the challenge is freshly generated and exposes only the final output and the agreed public leakage $\mathcal{L}$, which includes any public fields or observable metadata in the chosen view. The challenger reserves the identifier (PK, e, m_0, R, z) of the final signature in both branches and rejects all $\mathcal{O}_{\mathrm{Proof}}$ requests for that identifier, including requests for its derived versions. Challenge source records and their randomness remain private to the challenge procedure. Sign and sanitization queries outside the challenge do not access this private state.

In the collision experiment, the adversary may request honest commitments on chosen (m_0, m) in the target epoch and obtain both each CH_j and its original opening. It has access to $\mathcal{O}_H$ and $\mathcal{O}_{\mathrm{Adapt}}(j, m')$. In the collision, external unforgeability, and second accountability experiments, an algebraic adversary supplies a representation of each group point when submitting it to the challenger, including points in canonically encoded hash queries. The reduction records the representation before issuing the corresponding hash answer [39].

It should be noted that, except for secret keys, all temporary randomness r, s, t, k, ρ is sampled uniformly from $\mathbb{Z}_q$.

Definition 1. If, for any PPT adversary $\mathcal{A}$, the security experiment $\mathrm{Exp}^{\mathsf{KeyConfidentiality}}_{\mathcal{A}}(\lambda)$ defined in Fig. 2 outputs 1 with at most negligible probability, then DSS-MS satisfies the property of dynamic sanitization-key confidentiality.

The adversary pre-specifies a target epoch e^* and nonempty set A^* reachable by a valid membership history. The challenger generates a polynomial-length valid prefix through that epoch and releases its public manifests, public keys, and moduli. Later manifests are generated on $\mathcal{O}_{\mathrm{Update}}$ and may also be observed. No query exposes a long-term key in A^*, the TA secrets, or the target epoch key. Non-target epoch keys and keys of nodes outside A^* may be exposed as specified above. The adversary wins by returning $x^* \in \mathbb{Z}_q^*$ with $x^*P = PK_{s,e^*}$.

The remaining experiments follow a similar structure and

semantics. We therefore omit their detailed descriptions and state only the corresponding definitions.

Definition 2. Let set $\xi \in \{\mathsf{EUF}, \mathsf{Imm}, \mathsf{WAcc\text{-}1}, \mathsf{WAcc\text{-}2}, \mathsf{CR}, \mathsf{ParamBind}\}$. For a PPT adversary $\mathcal{A}$ subject to the algebraic requirement where specified, let $\mathrm{Exp}^{\xi}_{\mathcal{A}}(\lambda)$ denote the corresponding security experiment in Fig. 2, where output 1 indicates an adversarial win. If $\Pr[\mathrm{Exp}^{\xi}_{\mathcal{A}}(\lambda) = 1]$ is negligible for every ξ and every such adversary, then DSS-MS satisfies external unforgeability, immutability, bidirectional weak accountability, and parameter authentication. Moreover, the chameleon hash of honest signing commitments is collision resistant, as required by Lemma 1 in Section VII.

Definition 3. For any PPT adversary $\mathcal{A}$, let $\mathrm{Exp}_{\mathcal{A}}(\lambda)$ denote either $\mathrm{Exp}^{\mathsf{Trans}}$ or $\mathrm{Exp}^{\mathsf{Priv}}$ in Fig. 2. If $\left|\Pr[\mathrm{Exp}_{\mathcal{A}}(\lambda) = 1] - \frac{1}{2}\right|$ is negligible for both experiments, then DSS-MS satisfies transparency and sanitized content privacy.

E. Construction Based on CRT

Table II summarizes the notation used in our scheme. Fig. 3 provides an overview of the algorithms, detailed below.

TABLE II
NOTATIONS USED IN THE DSS-MS SCHEME

Notation	Description
(q, G, P)	Prime-order elliptic curve group and generator
$\mathbb{F}_p, \mathbb{Z}_q, \mathbb{Z}_q^*$	Coordinate field, scalar field, and nonzero scalars
(sk, PK)	Signer's key pair, $PK = skP$
K_i	Secret PRF key of sanitizer S_i
p_i	Public, pairwise-coprime CRT modulus ($p_i > q$)
A_e, e	Authorized sanitizer set and its epoch
$sk_{s,e}, PK_{s,e}$	Epoch-specific sanitization key pair
$ctx_e, S_{x,e}, \tau_e$	Epoch context, CRT value, and TA signature
$param_e$	Epoch parameters $(ctx_e, S_{x,e}, \tau_e)$
(ak, vk)	TA authentication key pair. vk is trusted
$F^{(p_i)}_{K_i}$	PRF with output in $\mathbb{Z}_{p_i}$
H_0, H_1, H_2	Domain-separated hashes into $\mathbb{Z}_q$
m_0, m, m'	Immutable, sanitizable, and sanitized messages
σ, σ', π	Original and sanitized signatures. Attribution proof
$e_{\min}$	Trusted minimum epoch stored by a receiver

*1) **Setup**:* The TA runs $\mathsf{ECC.Setup}(\lambda)$ to generate a group $G = \langle P \rangle$ of prime order q with generator P. It fixes the domain-separated hash functions H_0, H_1, H_2, a range-specific PRF F and an ordinary authentication scheme Auth, and outputs $pp = (q, G, P, H_0, H_1, H_2, F, \mathsf{Auth})$.

*2) **Key Generation**:* Given the public parameters and the number of sanitizers n, the TA samples mutually independent long-term PRF keys $K_i \leftarrow \{0,1\}^{\ell_K}$ for $i \in [n]$ and chooses public, pairwise coprime integers $p_i > q$ that are independent of all secret keys. Each K_i is delivered privately to sanitizer S_i over a secure channel while $\{p_i\}$ is published, so the long-term PRF key and the public CRT modulus are distinct objects. The TA also generates $(ak, vk) \leftarrow \mathsf{Auth.KGen}$ and distributes vk through trusted registration. The signer locally samples $sk \leftarrow \mathbb{Z}_q^*$, sets $PK = skP$, and registers PK with the TA.

*3) **Epoch Generation**:* For each epoch e, if the authorized set A_e is empty, the TA publishes an authenticated inactive-state record bound to $(sid, e, \varnothing)$ and terminates the epoch-generation procedure. Sign, Recover, and Verify reject inactive states. Otherwise it samples a fresh $sk_{s,e} \leftarrow \mathbb{Z}_q^*$, sets $PK_{s,e} = sk_{s,e}P$ and builds the unique context $ctx_e = \mathrm{Enc}(sid, e, \nu_e, A_e, \{(i, p_i)\}_{i \in A_e}, PK_{s,e})$ with a fresh nonce ν_e. The context binds $PK_{s,e}$. For each $i \in A_e$ it computes $b_{i,e} = F^{(p_i)}_{K_i}(ctx_e, i)$ and $a_{i,e} = (sk_{s,e} + b_{i,e}) \bmod p_i$. With $N_e = \prod_{i \in A_e} p_i$, $N_{i,e} = N_e / p_i$ and $y_{i,e} = N_{i,e}^{-1} \bmod p_i$, it computes $S_{x,e} = (\sum_{i \in A_e} a_{i,e} y_{i,e} N_{i,e}) \bmod N_e$. It then sets $\mathcal{P}_e = (ctx_e, S_{x,e})$, signs $\tau_e = \mathsf{Auth.Sign}_{ak}(\mathrm{Enc}(\mathcal{P}_e))$, and publishes the certified manifest $C_e = (\mathcal{P}_e, \tau_e)$, namely $param_e$. Only valid $param_e$ is issued per epoch. Each epoch uses a distinct context and independently samples its key.

*4) **Recovering the Sanitization Key**:* An authorized sanitizer S_i first checks $\mathsf{Auth.Verify}_{vk}(\mathrm{Enc}(\mathcal{P}_e), \tau_e) = 1$, that the epoch is the intended one and that $i \in A_e$. It then computes $\widehat{sk}_{s,e} = ((S_{x,e} \bmod p_i) - F^{(p_i)}_{K_i}(ctx_e, i)) \bmod p_i$ and returns this value only if $1 \le \widehat{sk}_{s,e} < q$ and $\widehat{sk}_{s,e} P = PK_{s,e}$. All sanitizers authorized in epoch e recover the same sanitization key and may independently replace the editable part m while leaving the immutable part m_0 unchanged.

*5) **Computing the Original Signature**:* To sign a message $M = (m_0, m)$, the signer checks the certified epoch parameters and requires m_0 to be encoded unambiguously so that it binds the system identifier, the epoch identifier e, the order identifier and the immutable fields. It samples $r, s, t \leftarrow \mathbb{Z}_q$, computes $R = rP$, $T = tP$ and $h = H_0(m_0, m, T)$, and then sets $CH = T + sP + h\,PK_{s,e}$, $c = H_1(m_0, R, CH)$ and $z = r + c\,sk \pmod q$. The signature is $\sigma = (R, z, s, T)$, and the signer securely retains the original randomness t.

*6) **Sanitizing the Original Signature**:* To sanitize $\sigma = (R, z, s, T)$ on $M = (m_0, m)$ to m', a sanitizer holding K_i accepts only a valid signature in the same certified epoch. All authorized sanitizers share the ability to replace m, with m_0 fixed. It obtains $sk_{s,e} \leftarrow \mathsf{Recover}(K_i, C_e)$, reconstructs $CH = T + sP + H_0(m_0, m, T)PK_{s,e}$, samples $k \leftarrow \mathbb{Z}_q$ and sets $T' = CH - kP$, $h' = H_0(m_0, m', T')$ and $s' = k - h'\,sk_{s,e} \pmod q$. It returns $M' = (m_0, m')$ and $\sigma' = (R, z, s', T')$ and records the valid adaptation for auditing.

*7) **Signature Verification**:* Given $M = (m_0, m)$, $\sigma = (R, z, s, T)$, PK and the certified manifest, the verifier checks the epoch binding, the syntax and group membership, computes $h = H_0(m_0, m, T)$, $CH = T + sP + h\,PK_{s,e}$ and $c = H_1(m_0, R, CH)$, and returns 1 if and only if $zP = R + c\,PK$. The value $PK_{s,e}$ is taken from the manifest of the epoch bound to m_0 and cannot be overridden by the caller.

*8) **Generating the Proof**:* In a dispute, the signer uses a stored honest original record with $T^* = t^*P$ together with two valid same-epoch signatures having identical m_0, R, z and $m^* \neq m$. If these conditions fail, Proof returns $\perp$. It forms the complete claim $\mathsf{stmt} = \mathrm{Encode}(ctx_e, PK, M^*, \sigma^*, M, \sigma)$, samples $\rho \leftarrow \mathbb{Z}_q$, and computes $U = \rho P$, $d = H_2(\mathsf{stmt}, U)$, $v = \rho + d\,t^* \pmod q$ and proof $\pi = (M^*, \sigma^*, M, \sigma, U, v)$.

*9) **Judging the Origin of the Signature**:* Given π, PK, and the certified manifest, Judge validates the certified epoch and verifies $\mathsf{Verify}(M^*, \sigma^*) = 1$ and $\mathsf{Verify}(M, \sigma) = 1$. It checks that $m_0^* = m_0$, $R^* = R$, $z^* = z$, $m^* \neq m$, and both signatures belong to the same epoch. It then recomputes stmt and $d = H_2(\mathsf{stmt}, U)$, returning `san` if $vP = U + dT^*$.

*10) **Join New Sanitizers**:* For a set J of new sanitizers, the TA generates a fresh (K_j, p_j) for each $j \in J$ so that all public moduli remain pairwise coprime, and sets $A_{e+1} = A_e \cup J$. It then runs $\mathsf{EpochGen}(pp, A_{e+1}, e+1)$ and publishes the new certified $param_{e+1}$. The long-term keys of existing sanitizers remain unchanged, while the public parameters and the sanitization key are refreshed for the new epoch.

*11) **Revoke Sanitizers**:* For a set I of sanitizers to be revoked, the TA sets $A_{e+1} = A_e \setminus I$ and runs $\mathsf{EpochGen}(pp, A_{e+1}, e+1)$ with a fresh trapdoor and context, or publishes an authenticated inactive state if $A_{e+1} = \varnothing$. The old K_i, $sk_{s,e}$ and $param_e$ retained by a revoked node cannot recover the sanitization key of the new epoch.

VI. Privacy-protected UAV Logistics System

This section presents the workflow of a privacy-preserving UAV logistics system. The system involves the five types of entities defined in Section IV: the TA, the OI, the LC, warehouse nodes W_i, and UAVs operated by third-party providers. The interaction flow of the entire process is shown in Fig. 4 .

Since warehouse membership changes are relatively infrequent, we assume that participants verify and cache the TA's certified epoch manifest $param_e$ during registration and epoch updates, and reuse it during normal operations. If an order's epoch differs from the cached epoch, they query the TA for the latest epoch and corresponding parameters.

*1) **System Initialization**:* The TA runs *DSS-MS.Setup* to generate system public parameters and establishes the epoch manifest authentication key pair. During registration, it distributes the public parameters, system identifier, and trusted verification key. After warehouse registration, the TA invokes *DSS-MS.EpochGen* to publish a certified epoch manifest for the initial authorized set, enabling participants to execute the corresponding algorithms. The TA archives historical manifests for order verification and dispute resolution.

*2) **Entity Registration**:* Participants register with the TA to obtain identity credentials and system parameters. The TA manages key provisioning and publishes $param_e$. The OI locally generates (sk, PK), submits PK to the TA for identity authentication and certification, and securely stores sk. The LC manages order dispatch, warehouse selection, and membership-change requests, and submits its warehouse information to the TA. Using *KGen*, the TA assigns each warehouse W_i a long-term secret key K_i and public modulus p_i, and securely delivers K_i. Registered UAV operators obtain authorization credentials, public parameters, and the verification key for order authentication and signature verification.

*3) **Signature Generation**:* After a user places an order with the OI (e.g., an e-commerce platform), the OI generates an order containing the user's name, phone number, delivery address, and product details. It divides the order into an immutable part m_0, containing information such as the epoch and unique order identifiers, and a sanitizable part m, containing sensitive fields to be hidden from the UAV. The OI invokes *DSS-MS.Sign* to sign $M = (m_0, m)$ and securely stores the original order, the signature σ, and the secret randomness t. The OI then sends the original order and σ to the LC over an authenticated encrypted channel.

Dynamic Sanitizable Signature with Multiple Sanitizers

Setup(1^λ) :

1: **return** $pp = (q, G, P, \{H_i\}, F, \mathsf{Auth}) \leftarrow \mathsf{ECC.Setup}()$

KGen(pp, n) :

1: Sample $K_i \leftarrow \{0,1\}^{\ell_K}$. Publish pairwise coprime $p_i > q$ independent of all K_i. Generate $(ak, vk) \leftarrow \mathsf{Auth.KGen}$.
2: The signer samples $sk \leftarrow \mathbb{Z}_q^*$ and registers $PK = skP$.
3: The TA provides K_i to S_i privately.

EpochGen(pp, A_e, e) :

1: For $A_e = \varnothing$, publish an authenticated inactive state and stop. Otherwise sample $sk_{s,e} \leftarrow \mathbb{Z}_q^*$ and set $PK_{s,e} = sk_{s,e}P$.
2: Build $ctx_e = \mathrm{Enc}(sid, e, \nu_e, A_e, \{(i, p_i)\}_{i \in A_e}, PK_{s,e})$ and set $b_{i,e} = F_{K_i}^{(p_i)}(ctx_e, i)$ and $a_{i,e} = (sk_{s,e} + b_{i,e}) \bmod p_i$.
3: Compute $N_e = \prod_{i \in A_e} p_i$ and $y_{i,e} = N_{i,e}^{-1} \bmod p_i$, where $N_{i,e} = N_e/p_i$. Then set $S_{x,e} = (\sum_{i \in A_e} a_{i,e} y_{i,e} N_{i,e}) \bmod N_e$, and publish $\mathcal{P}_e = (ctx_e, S_{x,e})$ with signature $\tau_e = \mathsf{Auth.Sign}_{ak}(\mathrm{Enc}(\mathcal{P}_e))$, yielding $param_e = (\mathcal{P}_e, \tau_e)$.

Recover$(K_i, param_e)$:

1: Check the certified manifest, intended epoch and membership. Compute $\widehat{sk}_{s,e} = ((S_{x,e} \bmod p_i) - F_{K_i}^{(p_i)}(ctx_e, i)) \bmod p_i$. Return it if $1 \le \widehat{sk}_{s,e} < q$ and $\widehat{sk}_{s,e}P = PK_{s,e}$, or return $\perp$.

Sign$(pp, M = (m_0, m), sk, param_e)$:

1: Check the certified epoch bound to m_0. Select $r, s, t \leftarrow \mathbb{Z}_q$,
2: Compute $R = rP$, $T = tP$, $h = H_0(m_0, m, T)$, $CH = T + sP + hPK_{s,e}$ and $c = H_1(m_0, R, CH)$, $z = r + csk \pmod q$.
3: **return** $\sigma = (R, z, s, T)$ and retain t.

Sanitizing$(\sigma, M, m', PK, K_i, param_e)$:

1: Verify the input and recover $sk_{s,e}$ for its certified epoch. Reject on failure. Reconstruct $CH = T + sP + H_0(m_0, m, T)PK_{s,e}$.
2: Select $k \leftarrow \mathbb{Z}_q$ and compute $T' = CH - kP$, $h' = H_0(m_0, m', T')$, and $s' = k - h'sk_{s,e} \pmod q$.
3: **return** $M' = (m_0, m')$ and $\sigma' = (R, z, s', T')$.

Verify$(M, \sigma, PK, param_e)$:

1: Check the certified epoch, syntax, and group membership. Compute $h = H_0(m_0, m, T)$ and $CH = T + sP + hPK_{s,e}$.
2: **return** 1 if $zP = R + H_1(m_0, R, CH)PK$, else 0.

Proof$((M^*, \sigma^*, t^*), (M, \sigma), PK, param_e)$:

1: Check the stored original record and the two pairs as specified in the text. Set $\mathsf{stmt} = \mathrm{Encode}(ctx_e, PK, M^*, \sigma^*, M, \sigma)$.
2: Sample $\rho \leftarrow \mathbb{Z}_q$ and compute $U = \rho P$, $d = H_2(\mathsf{stmt}, U)$, and $v = \rho + dt^* \pmod q$. **return** $\pi = (M^*, \sigma^*, M, \sigma, U, v)$.

Judge$(\pi, PK, param_e)$:

1: **return** $\perp$ if either signature is invalid. Check $m_0^* = m_0$, $R^* = R$, $z^* = z$ and $m^* \neq m$ and recompute $d = H_2(\mathsf{stmt}, U)$.
2: **return** `san` if $vP = U + dT^*$, else **return** $\perp$.

Join(A_e, J):

1: Provision fresh (K_j, p_j) for each $j \in J$, ensuring that all p_i are pairwise coprime, and update $A_{e+1} = A_e \cup J$.
2: Run $\mathsf{EpochGen}(pp, A_{e+1}, e+1)$ and publish $param_{e+1}$.

Revoke(A_e, I):

1: Set $A_{e+1} = A_e \setminus I$. Run $\mathsf{EpochGen}(pp, A_{e+1}, e+1)$ with a fresh trapdoor and context.

Fig. 3. The proposed dynamic sanitizable signature with multiple sanitizers.

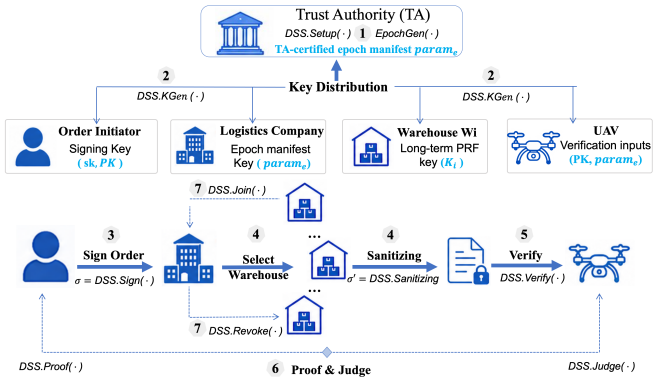

Fig. 4. UAV logistics workflow from TA registration to order signing, warehouse dispatch and sanitization, and verification in the current epoch.

*4) **Sanitizing Sensitive Information**:* The LC selects a suitable warehouse based on package details and forwards the complete order and signature. The warehouse arranges UAV delivery and redacts unnecessary sensitive information, such as names and phone numbers, using predefined masks. It verifies the original signature, invokes *DSS-MS.Recover* with K_i to recover the epoch sanitization key $sk_{s,e}$, and runs *DSS-MS.Sanitizing* to generate the sanitized order M' and signature σ'. It then sends M', σ', and the manifest identifier to the UAV.

*5) **Signature Verification**:* The UAV checks that the order's epoch and manifest identifier match its cached current state, checks order freshness, and authenticates the OI. It then invokes *DSS-MS.Verify* to verify the signature and checks its local acceptance records to reject duplicate order identifiers from the same OI. Once all checks pass, it records acceptance and proceeds with delivery.

*6) **Proof & Judge**:* When order contents are disputed, the OI locates the original record using the signer's identity, signature components (R, z), and other relevant information. It invokes *DSS-MS.Proof* using the stored original order, signature, and secret randomness t to generate dispute evidence π, which is submitted to an authorized adjudicator. The adjudicator obtains the certified manifest for the disputed epoch and invokes *DSS-MS.Judge* to verify the evidence. If the algorithm returns `san`, the valid modification is attributed to the sanitizers as a group, and the decision is recorded. The LC may further consult order dispatch and warehouse handling records to help identify the warehouse that processed the order.

*7) **Sanitizer Authority Management**:* To add warehouses, the LC submits their registration information and a membership-change request to the TA. The TA invokes *DSS-MS.Join* to assign each new warehouse a long-term secret key K_j and public modulus p_j, and securely distributes the secret keys. It then invokes *DSS-MS.EpochGen* to generate an independent sanitization key for the updated authorized set and publish the certified manifest $param_{e+1}$. Existing warehouses retain their long-term keys and recover the new sanitization key using the updated manifest. To revoke warehouse privileges, the LC submits the relevant member information to the TA, which invokes *DSS-MS.Revoke* to update the authorized set and follows the same epoch update procedure to generate a new sanitization key and certified manifest.

VII. Security Proof and Analyses

We analyze the signature and knowledge-proof algorithms in the random oracle model (ROM). Lemma 1, Theorem 4 via that lemma, and the second direction of Theorem 8 also use the algebraic group model (AGM). We assume a prime-order group, independent domain-separated hashes, PRF security, and an EUF-CMA secure TA authentication signature. All query counts are polynomial in the security parameter.

Each epoch has a unique identifier e and authorized set A_e. For an active epoch, $param_e = (ctx_e, S_{x,e}, \tau_e)$ binds its identifier, membership, public moduli, sanitization public key, and CRT value. Membership updates issue a certified manifest and independently sample the new epoch key. Each order uses the certified manifest bound to its immutable part. In fixed-epoch derivations, PK_s and sk_s denote $PK_{s,e}$ and $sk_{s,e}$.

A. Correctness

Theorem 1. *For every active epoch, an authorized member recovers the correct key from a properly generated $param_e$. Original signatures and valid sanitizations verify under their certified epoch. Evidence generated from matching valid records with an actual message change is accepted by Judge.*

By the CRT construction, $S_{x,e} \bmod p_i = a_{i,e} = (sk_{s,e} + b_{i,e}) \bmod p_i$. Since $p_i > q$, each authorized member can correctly recover $sk_{s,e}$ as $(a_{i,e} - b_{i,e}) \bmod p_i = sk_{s,e}$.

For an original signature, $zP = (r + H_1(m_0, R, CH)sk)P = R + H_1(m_0, R, CH)PK$, which satisfies the verification equation and thus ensures correctness.

In sanitizing, let $h' = H_0(m_0, m', T')$. Then $T' + s'P + h'PK_s = CH - kP + (k - h'sk_s)P + h'sk_sP = CH$, so the sanitized signature still satisfies the verification equation.

For a valid attribution proof, $vP = (\rho + dt^*)P = U + dT^*$, so Judge accepts the matching original and disputed records.

Each membership update executes the same construction for a new epoch, so the derivation applies to every active epoch.

B. Security Assumption

Elliptic Curve Discrete Logarithm Problem (ECDLP): Given P and $Q = xP$ in an elliptic curve cyclic group G, the ECDLP assumption states that no probabilistic polynomial-time (PPT) adversary can recover $x \in \mathbb{Z}_q$ with non-negligible probability. Formally, $\Pr[\mathcal{A}(P, Q) = x \mid Q = xP] \leq \epsilon$, where ϵ is negligible in the security parameter.

C. Security Proof

Theorem 2. *Assume the PRF is pseudorandom and the ECDLP is hard. For a pre-specified target epoch e^* determined by an authorized set A^*, if K_i of all sanitizers in A^* remain unexposed, and neither the TA nor the target sanitization key is compromised, then no adversary can recover the sanitization key sk_{s,e^*} of epoch e^* with non-negligible probability.*

Our proof strategy is as follows: if a PPT adversary $\mathcal{A}$ recovers the sanitization key of the target epoch with non-negligible probability, then we can construct an algorithm $\mathcal{B}_2$ that solves the ECDLP with non-negligible probability.

Initialization and Challenge Embedding. The challenger generates the public parameters, public moduli, and signer and TA key pairs. The adversary fixes the target (e^*, A^*) before seeing these values. The challenger generates the valid history through e^* and provides its public manifests.

Hybrid $\mathcal{B}_1$ replaces the PRFs of members of A^* one by one with independent consistent random functions at every epoch in which they participate. A single-user PRF challenger can answer all calls for the replaced key because that key is never revealed. This includes calls whose outputs can later be inferred from non-target epoch-key disclosures.

Let $n^* = |A^*|$ and let ϵ_{PRF} bound the single-user distinguishing advantage over the full query budget. The hybrid loss is at most $n^*\epsilon_{\mathrm{PRF}} + \delta_F$, where δ_F bounds the aggregate statistical bias of any mappings into the residue spaces.

Since the target-epoch context ctx_{e^*} is fresh, each authorized sanitizer's mask $b_{i,e^*} = F_{K_i}^{(p_i)}(ctx_{e^*}, i)$ is independently uniform over $\mathbb{Z}_{p_i}$. Hence, for any target sanitization key sk_{s,e^*}, the CRT residue $a_{i,e^*} = (sk_{s,e^*} + b_{i,e^*}) \bmod p_i$ is also uniform over $\mathbb{Z}_{p_i}$. Since the CRT moduli p_i are pairwise coprime, there is a bijection between the residue vector $(a_{i,e^*})_{i\in A^*}$ and S_{x,e^*}. Consequently, the public CRT parameter S_{x,e^*} contained in $param_{e^*}$ is also uniformly distributed.

Using this property, $\mathcal{B}_2$ receives an ECDLP challenge $(P, X = xP)$, where the scalar $x \in \mathbb{Z}_q^*$ corresponding to X is unknown, and aims to output x. It directly sets the sanitization public key for the target epoch e^* to $PK_{s,e^*} = X = xP$. Since $\mathcal{B}_2$ does not know x, It cannot compute a_{i,e^*} for the target epoch as required by the real algorithm. However, it can sample $a_{i,e^*} \in \mathbb{Z}_{p_i}$ independently for each $i \in A^*$ and compute the public parameter S_{x,e^*} of $param_{e^*}$ by CRT. It then constructs the context ctx_{e^*} binding the epoch identifier e^*, the member set A^*, the public moduli and the sanitization public key, and authenticates the complete $param_{e^*}$ with a TA authentication key it generated itself. Thus, without knowing x, $\mathcal{B}_2$ still generates a $param_{e^*}$ distributed as in the final hybrid experiment, completing the challenge embedding.

Adversary Query Phase. After initialization, $\mathcal{A}$ may adaptively issue various queries, which $\mathcal{B}_2$ must simulate to ensure the experiment is indistinguishable from the real system.

For an update query, $\mathcal{B}_2$ applies the membership change to the current valid state. It independently samples the new active epoch key, evaluates the applicable PRFs or random-function tables, and authenticates the result.

For a corruption query, $\mathcal{B}_2$ returns the long-term key of an eligible node outside A^*. For an epoch-key reveal, $\mathcal{B}_2$ returns the sampled key of any active non-target epoch. Distinct epoch contexts ensure that the resulting historical PRF samples do not expose the target masks in the random-function hybrid.

For a sign query, $\mathcal{B}_2$ can generate signatures normally. Specifically, the chameleon hash in an original signature is $CH = T + sP + H_0(m_0, m, T)PK_{s,e^*}$. This computation only needs the public $PK_{s,e^*} = X$ in $param_{e^*}$. Hence $\mathcal{B}_2$ can compute CH and produce the outer Schnorr signature with the signer private key it holds, while storing the original randomness t for later legitimate Proof queries.

For a sanitizing query, since $\mathcal{B}_2$ does not know x, it simulates sanitization through random-oracle programming. Specifically, $\mathcal{B}_2$ samples $s', h' \leftarrow \mathbb{Z}_q$, computes $T' = CH - s'P - h'X$, and programs $H_0(m_0, m', T') = h'$ if the corresponding oracle entry is undefined. The resulting simulated sanitized signature is valid since $CH' = T' + s'P + h'PK_{s,e^*} = CH$.

Hence the chameleon hash is unchanged by sanitization and the original outer Schnorr signature remains valid. Moreover, since the randomness of the real sanitization algorithm satisfies $k = s' + h'x \pmod q$, the simulated and real sanitization outputs have the same distribution provided scalars are sampled uniformly from the whole of $\mathbb{Z}_q$ and no random-oracle programming collision occurs. For a Proof query, $\mathcal{B}_2$ uses the stored original randomness t to generate a legitimate proof, and again needs no knowledge of the target sanitization key. Overall, apart from negligible events such as random-oracle programming collisions and sampling statistical errors, $\mathcal{B}_2$ correctly simulates every query allowed to the adversary.

Security Reduction. After the queries, $\mathcal{A}$ outputs a candidate sanitization key $x^* \in \mathbb{Z}_q^*$. A successful recovery requires $x^*P = PK_{s,e^*} = X = xP$. Since P belongs to a group of prime order q, this implies $x^* = x \pmod q$. Reduction $\mathcal{B}_2$ then simply outputs x^* to solve the ECDLP. Therefore, if the adversary recovers the sanitization key of the target epoch with non-negligible probability, $\mathcal{B}_2$ solves the ECDLP with non-negligible probability.

Success Probability Analysis. Let Q_S count simulated sanitizations and Q_0 bound the number of other H_0 evaluations, including all internal signing, verification, and proof evaluations. For each programmed point, T' is uniform conditioned on the current table. The reduction aborts if the input already has a value. A union bound gives $\delta_{\mathrm{sim}} \le [Q_S(Q_0 + Q_S)]/q$. Therefore the target-key recovery probability satisfies

$$\Pr[\mathsf{Recover} = 1] \le n^*\epsilon_{\mathrm{PRF}} + \mathrm{Adv}_{\mathrm{DL}}^{G}(\mathcal{B}_2) + \delta_F + \delta_{\mathrm{sim}}.$$

The PRF and discrete logarithm terms are negligible by assumption. Sampling bias and programming error are negligible under the stated bounds. The reduction is polynomial-time.

Theorem 3. *Under the hypotheses of Theorem 2, a removed member cannot recover the key of a later active epoch in which it is unauthorized, and a newly admitted member cannot recover the key of an earlier active epoch in which it had no authorization. This holds for coalitions consisting of nonmembers of the target epoch.*

Revocation security. Take the new epoch e^* defined by the member set after revocation (authorized set A^*) as the target epoch and target set of Theorem 2. Since the revoked member $S_i \notin A^*$, its long-term private key, the sanitization keys of old epochs and all historical $param_e$ are information that Theorem 2 allows the adversary to obtain. Hence, if the adversary could recover the sanitization key sk_{s,e^*} of the new epoch e^* from this information, it would break Theorem 2.

Join separation. Take any active epoch e^* before the new member joins in which it was not authorized, with authorized set A^*, as the target epoch and set in Theorem 2. Since the new member $S_j \notin A^*$, its long-term private key and the sanitization keys obtained after joining are likewise information that Theorem 2 allows to leak. Hence, recovering the sanitization key sk_{s,e^*} of that historical epoch would also violate Theorem 2.

Probability Analysis. In both cases, forwarding the permitted queries and the candidate key gives a key-confidentiality adversary with the same success probability. Recovery of an earlier key by a corrupt member that was authorized in that epoch is possible from its K_i and the published manifest.

Lemma 1. *The proposed chameleon hash provides collision resistance for honestly generated commitments. Specifically, in the ROM and AGM, an adversary wins the honest-commitment experiment in Fig. 2 with negligible probability under the ECDLP assumption and the PRF security and target-epoch key protection assumptions of Theorem 2.*

Assume that a PPT adversary $\mathcal{A}$ can produce a previously unreturned valid chameleon-hash collision with non-negligible probability. We construct a reduction $\mathcal{B}_3$ to solve the ECDLP.

Challenge Embedding. Apply the PRF hybrid from Theorem 2. In this hybrid, $\mathcal{B}_3$ embeds $X = xP$ as $PK_{s,e}$ and authenticates the target manifest using uniform CRT residues. It generates the signer and TA authentication keys itself.

Honest Commitment Query. For input $(m_{0,j}, m_j)$, $\mathcal{B}_3$ samples t_j, s_j, computes $T_j = t_j P$ and $h_j = H_0(m_{0,j}, m_j, T_j)$, and returns $CH_j = (t_j + s_j)P + h_j X$ and the original opening (m_j, s_j, T_j). It initializes L_j with that opening and records $A_j = t_j + s_j$ and $B_j = h_j$, so that $CH_j = A_j P + B_j X$.

Sanitization Query. When $\mathcal{A}$ requests sanitizing the message of commitment CH_j to m', reduction $\mathcal{B}_3$ samples $s', h' \leftarrow \mathbb{Z}_q$ and computes $T' = CH_j - s'P - h'X$. If the input $(m_{0,j}, m', T')$ is not yet in the hash table, the reduction sets $H_0(m_{0,j}, m', T') = h'$ and returns the new chameleon-hash input (m', s', T'). This input satisfies $T' + s'P + h'X = CH_j$.

In real sanitization, k is uniform and $s' = k - h'x \pmod q$, $T' = CH_j - kP$. In the simulation s' is uniform, and setting $k = s' + h'x \pmod q$ gives the same relation. Therefore, except for random-oracle programming collisions on previously queried inputs, the simulated and real outputs are identically distributed. The reduction also records previously returned valid inputs for each commitment to determine whether the adversary's final output is new.

Security Reduction. Suppose $\mathcal{A}$ outputs a previously unreturned valid input (m^*, s^*, T^*) for the j-th commitment. In the AGM the reduction obtains the algebraic representation of this point relative to the available group elements and expands it as $T^* = AP + BX$. Let $h^* = H_0(m_{0,j}, m^*, T^*)$. Validity of the input gives $(A + s^* - A_j)P + (B + h^* - B_j)X = \mathcal{O}$.

If $D = B + h^* - B_j \neq 0$, the reduction solves the ECDLP by computing $x = (A_j - A - s^*)D^{-1} \bmod q$.

If $D = 0$, validity requires $A + s^* = A_j$. For an opening in L_j, the message and point determine a unique s and the game rejects it. A programmed opening of another anchor has coefficient sum $B + h = B_\ell$, so the exceptional event requires $B_\ell = B_j$. For ordinary hash entries, the point coefficient is fixed before answering. For distinct entries, whichever of $B + h$ and B_j is sampled later must hit the earlier value, with probability $1/q$. If a point receives distinct valid representations, their difference yields x.

Probability Analysis. Let Q bound all hash evaluations, commitments, and adaptations, including implicit evaluations at output. Fresh-point table collisions cost $O(Q^2/q)$ by a union bound. Pairwise coefficient coincidences contribute $O(Q^2/q)$ as well. Including the PRF hybrid loss gives

$$\mathrm{Adv}^{\mathrm{anchor}}_{\mathrm{CH}}(\mathcal{A}) \le \mathrm{Adv}^{\mathrm{DL}}_{G}(\mathcal{B}_3) + n^* \epsilon_{\mathrm{PRF}} + \delta_F + O(Q^2/q).$$

The ideal-PRF experiment uses the same extraction without hybrid loss. The reduction performs polynomially many group operations and maintains polynomially many table entries.

Lemma 2. *Given a public point $T^* = t^* P$ bound to the complete claim* stmt*, the protocol defined by $U = \rho P$, $d = H_2(\mathsf{stmt}, U)$, and $v = \rho + dt^* \pmod q$ constitutes a non-interactive proof of knowledge of the secret scalar t^* corresponding to the public point T^*.*

This protocol is derived by applying the Fiat–Shamir transform to Schnorr's proof of knowledge of a discrete logarithm. Its completeness, knowledge soundness, and simulatability in the ROM are well established in the literature [40].

Theorem 4. *In the ROM and AGM, under the key-protection conditions of the external unforgeability experiment and the assumptions of Lemma 1 and Schnorr unforgeability, the adversary has negligible probability of producing a valid signature on a message absent from the target signer and epoch entries of L_{out}.*

Assume a PPT adversary $\mathcal{A}$ produces a valid signature for a new message. Depending on whether the corresponding outer message $\mu = \mathrm{Encode}(m_0, CH)$ was ever signed, we split the forgery into two cases and reduce them to the unforgeability of Schnorr signatures and Lemma 1, respectively.

Query Simulation. In a signing query the simulator generates a chameleon-hash commitment CH, obtains a Schnorr signature (R, z) on the outer message $\mu = \mathrm{Encode}(m_0, CH)$, and records the full message, the outer message and the original randomness. A sanitizing query first verifies the input signature, sanitizes only legitimate inputs and records the output message. A proof query is answered with the stored original randomness. The values CH and m_0 are unchanged by sanitizing, so both correspond to the same outer message.

If, when submitting a sanitizing query, the adversary first presents a valid signature whose outer message μ was never issued by the signing interface, that input already constitutes a Schnorr forgery of case one below. Otherwise all legitimate sanitizing queries concern commitments produced by honest signatures and can be simulated as in Lemma 1. In Case 2, we simulate the target-epoch manifest and sanitization queries as in Theorem 2 and include the resulting simulation error.

Case 1: $\mu = \mathrm{Encode}(m_0, CH)$ was not signed. Let the adversary finally output a new message $M^* = (m_0^*, m^*)$ and a valid signature $\sigma^* = (R^*, z^*, s^*, T^*)$. The reduction computes $CH^* = T^* + s^* P + H_0(m_0^*, m^*, T^*) PK_s$ from the signature and sets $\mu^* = \mathrm{Encode}(m_0^*, CH^*)$. Since verification succeeds, (R^*, z^*) is a valid Schnorr signature on μ^*. If μ^* was never issued through the signing interface, reduction $\mathcal{B}_5$ outputs (μ^*, R^*, z^*) to the Schnorr security experiment, obtaining a valid forgery on a new outer message. $\mathcal{B}_5$ answers outer signing queries using the Schnorr signing interface and generates sanitization keys and original randomness itself, so it can simulate all other legitimate queries.

Case 2: If μ^* was already signed, the output opens an honest $CH_j = CH^*$ under the same m_0^*. Its message is absent from L_{out}, so its opening is absent from L_j. The reduction uses the ideal-PRF simulation of Lemma 1. It generates the signer key and records the original randomness, allowing it to answer the signing and proof queries in the same execution.

Probability Analysis. Let n^* be the number of authorized members in the target epoch e, ϵ_{PRF} the maximal distinguishing advantage of a single PRF, and δ_F and δ_{sim} the output-mapping bias and query-simulation error. From the two cases and the hybrid experiment of Theorem 2,

$$\begin{aligned}\text{Adv}^{\text{EUF}}_{\text{DSS}-\text{MS}} \leq\ & \text{Adv}^{\text{EUF}}_{\text{Schnorr}}(\mathcal{B}_5) + \text{Adv}^{\text{anchor}}_{\text{CH,ideal}} \\ & + n^*\epsilon_{\text{PRF}} + \delta_F + \delta_{\text{sim}}.\end{aligned}$$

The reduction only simulates polynomially many queries and maintains the corresponding records, so its running time is polynomial. By Schnorr security, Lemma 1 and Theorem 2 the bound is negligible, which proves the theorem.

Theorem 5 (Immutability)**.** *Assume the signing key remains secret. Even an adversary holding every sanitizer key cannot output a valid signature on immutable content m_0^* never authorized by the signer. The encoded m_0^* includes the system, epoch, and order identifiers.*

$\mathcal{B}_6$ uses the Schnorr challenge public key as the signing public key and generates sanitization keys itself, so it can answer signing, sanitizing and attribution queries. If the adversary outputs a valid signature whose m_0^* was never signed, the outer message (m_0^*, CH^*) recovered from that signature was never signed either. The output therefore constitutes a Schnorr signature forgery and $\text{Adv}^{\text{imm}}_{\text{DSS}-\text{MS}}(\mathcal{A}) \leq \text{Adv}^{\text{EUF}}_{\text{Schnorr}}(\mathcal{B}_6)$. By Schnorr unforgeability the theorem follows.

Theorem 6 (Transparency)**.** *For a fixed epoch, final message, and public leakage, a fresh direct signature and a fresh sign-then-sanitize output have the same distribution under uniform sampling. The transparency experiment restricts proof queries for the challenge identifier as specified in Section V-D.*

Let the final message be (m_0, m'). The adversary sees the final signature (R, z, s', T') and must decide whether it was generated directly by the signer or obtained by sanitizing an original signature. We compare the distributions of the complete signature in both cases.

For a direct signature, T' and s' are generated from uniform randomness. Fixing T', the term $T' + H_0(m_0, m', T')PK_s$ is a fixed group element while $s'P$ is uniform in G, hence $CH = T' + s'P + H_0(m_0, m', T')PK_s$ is also uniform in G. Therefore (CH, T') is uniform over $G \times G$.

For sign-then-sanitize, the original signature likewise makes CH uniform in G. Sanitizing samples a uniform k and sets $T' = CH - kP$. Even conditioned on CH, T' remains uniform in G. Hence (CH, T') is also uniform over $G \times G$ in this case.

Given the same (CH, T') and (m_0, m'), both constructions yield s' satisfying $s'P = CH - T' - H_0(m_0, m', T')PK_s$. Since the group order is prime, the $s' \in \mathbb{Z}_q$ satisfying this equation is unique. In both constructions the outer signature uses an independent uniform r to compute $R = rP$ and $z = r + H_1(m_0, R, CH)sk \pmod q$, so the joint distribution of the complete signature (R, z, s', T') is the same. Fixing the hash functions in both worlds, subsequent queries have the same distributions because they use the common final view and the same reserved-identifier rule. A fresh R matches one of at most Q independent honest record identifiers with probability at most Q/q. This yields at most $O(Q_C Q/q)$ distinguishing advantage for Q_C challenges, including collisions with other challenge records in Q.

Theorem 7. *For two orders with the same immutable content and the same publicly leaked information, both sanitized to the same final message, if the challenge original order and its signature are not disclosed, the adversary cannot tell which original order was used from the final sanitized signature.*

For either original message, the final signature has the distribution of a fresh direct signature on the common final message by Theorem 6. With equal public leakage and the same challenge-query restrictions, the advantage is at most $O(Q_C Q/q)$ for Q_C challenges and Q related records. Even for low-entropy fields, the signature provides no additional way to validate a guess given the common final view. The channel protection in Section VI is a deployment assumption.

Theorem 8. *Under the two key-exposure experiments in Fig. 2, DSS-MS provides bidirectional weak accountability for the signer and the sanitizers collectively. The first direction uses strong Schnorr unforgeability in the ROM. The second uses the ROM and AGM, ECDLP, and PRF security.*

We consider separately a malicious sanitizer trying to frame the signer and a malicious signer trying to frame the sanitizers.

Case 1: Malicious Sanitizer. The disputed pair must verify and its message must be absent from the signing history. If its outer signature is new for its outer message, it gives a strong Schnorr forgery. To obtain this bound, the Schnorr simulation programs the hash at each signing commitment. Reusing a programmed entry fixes the original response z by the verification equation. A new entry permits the standard forking extraction of the signing key. Otherwise an honest stored record shares the same m_0, R, z, CH. The signer uses the retained t^* to produce evidence. Since the disputed message differs from the recorded message, correctness gives $\mathsf{Judge} = \texttt{san}$.

Case 2: Malicious Signer. $\mathcal{A}$ holds sk and can request sanitization of any valid input in the target epoch. It must submit evidence for a disputed message absent from that epoch's sanitization outputs. The game computes both freshness and acceptance on the disputed pair in the same evidence.

$\mathcal{B}_8$ embeds $X = xP$ as $PK_{s,e}$ and simulates the target manifest and sanitization as in Theorem 2. It records representations $T^* = AP + BX$ and $T = \widetilde{A}P + \widetilde{B}X$. For each well-formed claim queried to H_2, it consistently evaluates the two verification hashes before returning the challenge. Thus the claim and both signature checks are fixed at the fork. For $T^* = \mathcal{O}$, the witness is 0. Otherwise, excluding an unqueried challenge guess of probability at most $1/q$, forking on $H_2(\mathsf{stmt}, U)$ yields t^* by Lemma 2.

If $B \neq 0$, subtracting the equations for T^* yields $x = (t^* - A)B^{-1} \bmod q$. If $B = 0$, then $A = t^*$. Since Judge requires both signatures to share m_0, R, z, they correspond to

the same commitment, except for outer hash collisions. Let $h^* = H_0(m_0, m^*, T^*)$ and $h = H_0(m_0, m, T)$. Equality of the two commitment openings gives

$$(t^* + s^*)P + h^*X = (\widetilde{A} + s)P + (\widetilde{B} + h)X.$$

Let $D = \widetilde{B} + h - h^*$. If $D \neq 0$, the reduction computes $x = (t^* + s^* - \widetilde{A} - s)D^{-1} \bmod q$. If $B = 0$ and the original hash entry was programmed by sanitization, its point coefficient equals the input commitment coefficient minus the fresh programmed hash. The probability it vanishes is at most $1/q$ per programmed point. The disputed hash entry cannot be a previously programmed output, since H_0 binds the complete (m_0, m) and the actual disputed message is fresh. For two ordinary hash entries, $m^* \neq m$ ensures distinct inputs. Their later sample must hit the earlier coefficient sum for $D = 0$, with probability $1/q$ per pair. A conflicting representation of a point instead yields x directly.

Probability Analysis. Let Q count all queries and internal evaluations, and $Q_2 \geq 1$ bound H_2 queries. For the nonidentity case, let η be the successful proof probability after accounting for the PRF hybrid, programming aborts, and unqueried guesses. The general forking lemma gives probability at least $[\eta(\eta/Q_2 - 1/q)]_+$ of two accepting transcripts with distinct challenges. A union bound over the two executions adds $O(Q^2/q)$ for the exceptional coefficient and hash collisions described above. The identity case uses the known witness and the same collision analysis directly. A non-negligible winning probability therefore contradicts ECDLP. The first direction follows from strong Schnorr unforgeability and the successful proof on a matching record.

Theorem 9. *If the TA authentication signature is EUF-CMA secure and the TA issues one manifest content per system and epoch, the adversary cannot with non-negligible probability cause acceptance of unissued content. A receiver rejects any epoch below its trusted, persistent* $e_{\min}$.

$\mathcal{B}_7$ uses the challenge verification key as vk and answers publication requests via signing queries for the canonical manifest contents. Acceptance of unissued content yields a signature on a new authentication message. A substitute with different content for an already accepted epoch is unissued under the uniqueness hypothesis. An issued manifest below the receiver's trusted floor fails the epoch check. Consequently,

$$\Pr[\mathsf{AcceptUnissued}] \leq \mathrm{Adv}^{\text{EUF-CMA}}_{\text{AuthSig}}(\mathcal{B}_7),$$
$$\Pr[\mathsf{AcceptOlderThanTrustedFloor}] = 0.$$

Signature verification selects the certified key for the epoch bound to m_0. The trusted floor controls rollback to an older known epoch. The current-epoch check for a new delivery is specified separately in Section VI.

VIII. Experimental Evaluation

A. Experimental Environment

We implemented both our and comparison schemes in C++. The order initiator and warehouse nodes were deployed on a PC, while UAV experiments were conducted on AmovLab Prometheus 600 UAVs [41].



Fig. 5. Experimental platform showing the AmovLab Prometheus 600 UAV with its Jetson Xavier NX (left) and the host PC (right).

For the network latency experiments, the order initiator and warehouse ran in an Ubuntu virtual machine on the host computer. The virtual machine used the AArch64 architecture and Linux kernel 6.8.0-149-generic. The verifier ran on an NVIDIA Jetson Xavier NX onboard the AmovLab Prometheus 600 UAV. The Jetson has a six-core NVIDIA Carmel Armv8.2 CPU and 8 GB of RAM and ran Ubuntu 18.04.4 LTS. We also used the Jetson for the computation and energy benchmarks.

The DSS-MS implementation uses MIRACL Core [42] for elliptic-curve operations on BLS12-381 and GMP for arbitrary-precision integer arithmetic in the CRT computations. We use SHA-256 for hashing and HMAC-SHA-256 for deriving per-member masks. We evaluated the computational overhead using the Google Benchmark framework. Communication among system components was realized via WebSocket.

B. Computation Cost Analysis

We compare DSS-MS with eight schemes on signing, sanitization, and verification. G_i^m and G_i^a denote scalar multiplication and point addition in G_i, respectively, for $i \in \{1, 2\}$. H denotes hashing, H_p denotes hashing to a curve point, T_p denotes a bilinear pairing, G_T^p denotes exponentiation in G_T, and I_q denotes modular inversion. C_{enc} and C_{dec} denote hybrid PKE encryption and decryption, respectively.

Theoretical cost. We estimate the dominant computational costs of each scheme, considering only expensive operations such as scalar multiplications, and pairings, while neglecting negligible costs. For a fair comparison, the number of message blocks is set to $n = 2$ and the number of sanitizers is set to $k = 1$ across all schemes. For BFLS10, we instantiate its generic sanitizable signature construction using the widely adopted and computationally efficient BBS group signature and Schnorr signature schemes. For PGUS-SBS, we instantiate its hybrid public-key encryption module using the Cramer–Shoup scheme recommended by the authors in the supplementary material, combined with AES. Table III summarizes the computational costs of signing, sanitization, and verification.

The results show that DSS-MS relies only on elliptic-curve scalar multiplications, point additions, and standard hash operations, achieving significantly lower computational costs than the evaluated multi-sanitizer schemes while remaining comparable to lightweight single-sanitizer schemes. However, to grant k sanitizers independent sanitization privileges over

TABLE III
CRYPTOGRAPHIC OPERATION COUNTS FOR SIGNING, SANITIZATION, AND VERIFICATION

Scheme		Sign	Sanitizing	Verify
Single sanitizer	DVAS [34]	$2(4G_1^m + G_1^a + 3H + T_p + G_T^p)$	$4G_1^m + G_1^a + 5H + 4T_p + G_T^p$	$6G_1^m + G_1^a + 6H + 6T_p + G_T^p$
	IBSDIA [35]	$5G_1^m + 4G_1^a + 2H_p$	$6T_p + 2H_p + 2G_1^m + 4G_1^a$	$4T_p + 2G_T^p + 3G_1^m + 2G_1^a + 2H_p$
	PF-CLSS-DV [29]	$3G_1^m + 3H$	$2H$	$6G_1^m + 5G_1^a + 4H$
	LSanS [37]	$G_1^m + H$	$3G_1^m + G_1^a + 2H$	$4G_1^m + 2G_1^a + 2H$
	PGUS-SBS [28]	$64G_1^m + 29G_1^a + 34G_2^m + 8G_2^a + 2H_p + H$	$52G_1^m + 28G_1^a + 36G_2^m + 20G_2^a + 128T_p + 3H_p + H + C_{\text{enc}} + C_{\text{dec}}$	$8G_1^m + 7G_1^a + 4G_2^a + 76T_p + 4H_p + H$
	ECMSS [36]	$2G_1^m + G_1^a + H$	$H + I_q$	$3G_1^m + 2G_1^a + H$
Multiple sanitizers	BFLS10 [21]	$10G_1^m + 3G_1^a + 2H + 3T_p + 3G_T^p$	$9G_1^m + 3G_1^a + H + 3T_p + 3G_T^p$	$10G_1^m + 5G_1^a + H + 5T_p + 4G_T^p$
	IUT-k-SAN [30]	$23G_1^m + 5G_2^m + 6G_1^a + 4H_p + H$	$17G_1^m + 5G_2^m + 2G_1^a + 2H_p + H$	$8G_1^m + 4G_1^a + 18T_p + 4G_T^p + 3H_p + H$
	DSS-MS (Ours)	$4G_1^m + 2G_1^a + 2H$	$G_1^m + G_1^a + H$	$4G_1^m + 3G_1^a + 2H$

the same message, single-sanitizer schemes require the signer to generate and transmit a separate signature for each sanitizer, causing both signing costs and total signature transmission overhead to grow linearly with k. Moreover, the numbers of elliptic-curve operations during signing and sanitization in DSS-MS are independent of the number of message blocks. This $O(1)$ complexity is achieved by hashing the immutable and sanitizable message components as a whole before the core cryptographic operations. Among the compared schemes, BFLS10, LSanS, and ECMSS share this property in our implementation, whereas the others incur $O(n)$ computational complexity with respect to the number of message blocks.

Benchmark results. On the Jetson Xavier NX, each scheme used two 128-byte blocks in its native format. DSS-MS and IUT-k-SAN used two sanitizers. Fig. 6 shows mean times and 95% confidence intervals over 30 repetitions per operation.

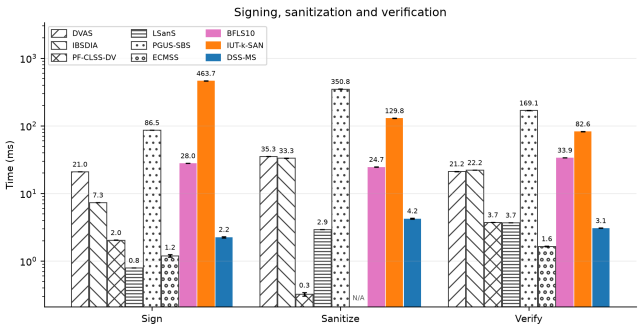

Fig. 6. Mean signing, sanitization, and verification times across nine schemes on the Jetson Xavier NX. Error bars show 95% confidence intervals.

As shown in Fig. 6, DSS-MS achieves signing, sanitization, and verification times of 2.2, 4.2, and 3.1 ms, respectively. With both schemes configured for two sanitizers, IUT-k-SAN incurs approximately (206, 31, 27) times the corresponding costs of DSS-MS. BFLS10, evaluated with one sanitizer, incurs approximately (12, 6, 11) times the corresponding costs of DSS-MS with two sanitizers. Note: IUT-k-SAN supports finer-grained sanitizer-specific admissibility policies, albeit at a higher computational cost.

PF-CLSS-DV and LSanS achieve lower signing and sanitization times but incur higher verification costs than DSS-MS. As shown in Table I, DSS-MS supports a broader range of functionalities. ECMSS achieves lower costs than DSS-MS in all three operations. However, its original signature equations suggest that unauthorized message modifications may still pass signature verification, raising concerns about its unforgeability that should be considered when comparing computational efficiency. Moreover, supporting k sanitizers with these single-sanitizer schemes requires generating and transmitting a separate signature for each sanitizer, causing both signing costs and total signature transmission overhead to grow linearly with k. When $k \geq 2$, DSS-MS achieves a lower signing cost than PF-CLSS-DV. When $k \geq 3$, its signing cost becomes lower than those of both PF-CLSS-DV and LSanS.

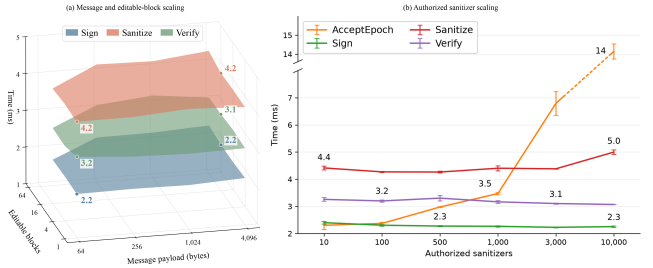

Fig. 7. DSS-MS timing by (a) payload size and editable block count and (b) sanitizer count, with manifest acceptance included in panel (b).

Scalability Analysis. Fig. 7(a) shows the impact of payload size and the number of editable blocks on signing, sanitization, and verification times with 100 authorized sanitizers. The experiments cover 15 configurations, with payload sizes ranging from 64 to 4,096 bytes and editable block counts from 1 to 64. Each configuration was repeated 30 times. Consistent with theoretical analysis, the average times of all three operations remain stable across the tested configurations, exhibiting only minor variations due to system fluctuations and closely matching the benchmark results in Fig. 6.

Fig. 7(b) shows that as the number of authorized sanitizers increases from 10 to 10,000, the average acceptance time (AcceptEpoch) for the current epoch's public parameters param_e increases from 2.30 to 14.14 ms. When accepted public parameters are reused, the average signing, sanitization, and verification times remain nearly independent of the number of authorized sanitizers. Even if the public parameters are revalidated before each operation, the sum of the measured mean acceptance time and the mean time of any operation remains below 20 ms with 10,000 authorized sanitizers. Fig. 7(b) uses a broken time axis to display different time scales, with error bars indicating 95% confidence intervals.

Overall, DSS-MS demonstrates good scalability with re-

spect to payload size, the number of editable blocks, and the number of authorized sanitizers across the evaluated ranges.

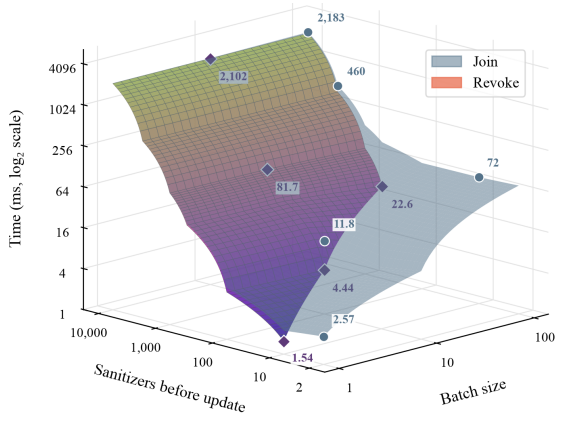

Fig. 8. Mean batch Join and Revoke times by initial sanitizer count and batch size. The time axis uses a base-2 logarithmic scale.

Sanitizer Management Cost. To evaluate the performance of dynamic sanitizer management, we measured batch join and revocation times on the UAV platform, with initial authorized sanitizer populations ranging from 2 to 10,000 and batch sizes of 1, 10, and 100. Fig. 8 shows the mean processing time per batch over 30 repetitions for each configuration.

As the authorized sanitizer population grows, the costs of both batch joins and revocations increase. This is mainly because the authority must recompute the CRT aggregate for the updated authorized set, generate a new authenticated epoch manifest, and issue credentials to newly joined sanitizers. Even with 3,000 existing sanitizers, joining and revoking 100 sanitizers take only 460.50 ms and 372.37 ms, respectively. In logistics scenarios, sanitizers are typically deployed at warehouse nodes, where membership changes are relatively infrequent, making these costs generally acceptable in practice. Moreover, these management costs arise only when the authorized set changes and do not affect the efficiency of routine signing, sanitization, and verification operations.

C. Communication Cost Analysis

Fig. 9 compares the signature sizes of DSS-MS and eight other schemes at the 128-bit security level. The left panel presents theoretical signature sizes as functions of the number of message blocks n and sanitizers k, where applicable, while the right panel compares encoded sizes in bytes for the two-block benchmark. DSS-MS and IUT-k-SAN use $k = 2$, whereas PF-CLSS-DV uses one designated sanitizer.

In the size expressions, ℓ_q, ℓ_1, ℓ_2, and ℓ_{PKE} denote the encoded lengths of a scalar in $\mathbb{Z}_q$, a compressed point in G_1, a compressed point in G_2, and a Paillier ciphertext, respectively. ECMSS counts only the aggregate signature point, while LSanS includes both signature layers for one sanitizer. PGUS-SBS counts the sanitized signature for one sanitizer, including n encoded scalars in the Cramer–Shoup/AES-GCM ciphertext. Its additional 46 bytes account for length and count fields, tag-presence flags, the AES-GCM nonce, and authentication tag.

As shown in Fig. 9, DSS-MS consists of two scalars and two G_1 points, yielding a signature size of $2\ell_q + 2\ell_1 = 162$ bytes with 32-byte scalars and 49-byte points. Padding both scalars to 48 bytes increases the size to 194 bytes. Since its signature size is independent of k, the per-signature transmission overhead remains constant as the authorized set grows. Among the compared schemes, only ECMSS has a smaller signature, although its potential unforgeability risks discussed earlier should also be considered. PF-CLSS-DV also requires 162 bytes per signature. As discussed earlier, the signature transmission overhead of PF-CLSS-DV and IUT-k-SAN grows with k, as indicated by the two arrows in the figure.

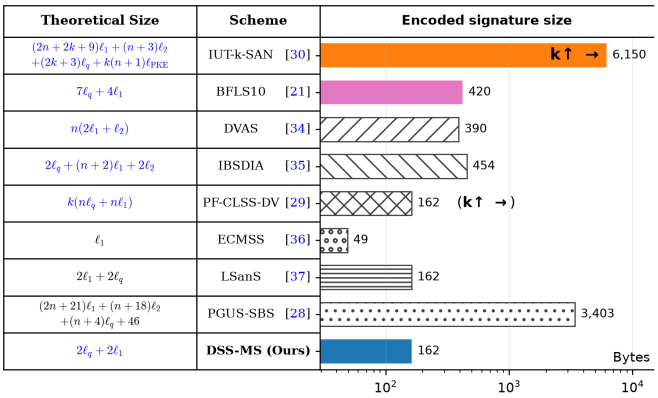

Fig. 9. Theoretical signature sizes and encoded byte counts for nine schemes. Arrows mark payload growth with sanitizer count.

Table IV reports the measured size of $S_{x,e}$ in its shortest encoding and the complete authenticated epoch parameters $(ctx_e, S_{x,e}, \tau_e)$. The second row gives the application-layer payload for one complete parameter transmission to a recipient. From 2 to 10,000 authorized sanitizers, the manifest grows from 343 to 719,647 bytes. The per-order DSS-MS signature remains 162 bytes across these configurations.

TABLE IV
ENCODED EPOCH PARAMETER SIZES (BYTES)

Sanitizers	2	10	100	500	1,000	3,000	10,000
$S_{x,e}$	64	320	3,195	15,972	31,944	95,834	319,448
Full manifest	343	919	7,394	36,171	72,143	216,033	719,647

D. Communication Delay

The order initiator and warehouse ran in an Ubuntu virtual machine on the PC, and the verifier ran on the Jetson Xavier NX aboard the Amov P600. The authentication timer ran from the start of signing until the PC received the Jetson's successful verification response, including both WebSocket connections opened for each order. We applied added delay or rate limits to the PC-to-Jetson traffic and recorded 30 trials per condition after one warm-up. Fig. 10 reports the mean for each condition. The blue curve uses the upper axis for added PC egress delay, and the red curve uses the lower axis for the PC egress bandwidth limit.

With no rate limit, mean authentication latency increases from 40.88 ms with no added delay to 1.95 s with 500 ms of added egress delay. With no added delay, the mean rises from

40.69 ms with no rate limit to 185.27 ms at 64 kbit/s (8 kB/s) and 369.54 ms at 32 kbit/s.

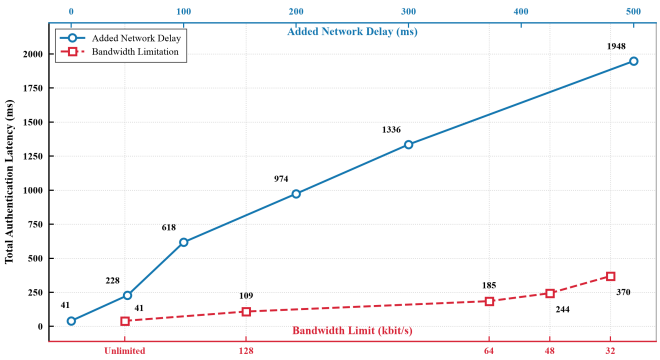

Fig. 10. Mean authentication latency with added PC egress delay (blue, upper axis) or a PC egress bandwidth limit (red, lower axis).

E. Energy Consumption Measurement

We sampled the Jetson Xavier NX module's `VDD_IN` power rail with `tegrastats` on the UAV described in Section VIII-A. For each operation, we collected ten pairs of active and idle windows of about 10 seconds each. We integrated the power samples over each window, scaled the idle energy to the active window duration, and subtracted it from the active energy. We divided this difference by the number of completed calls in the active window to estimate net energy per invocation. We report the mean and a 95% Student t confidence interval over the ten paired estimates.

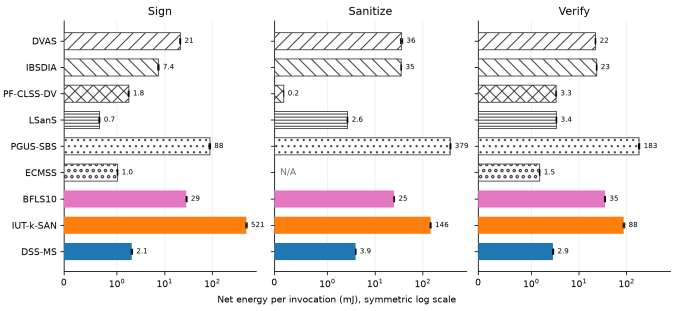

Fig. 11. Mean net energy per invocation for signing, sanitization, and verification across nine schemes. Error bars show 95% confidence intervals.

Fig. 11 compares net energy for signing, sanitization, and verification across nine schemes on the same Jetson module, using a 256-byte payload with two blocks. DSS-MS and IUT-k-SAN both use two authorized sanitizers. DSS-MS consumes 2.05, 3.90, and 2.88 mJ per operation, respectively, compared with 520.52, 146.44, and 87.66 mJ for IUT-k-SAN. The three panels share a symmetric logarithmic axis. Error bars show 95% confidence intervals. ECMSS single-step sanitization is measured separately, and PGUS-SBS signing covers the signer-side stage of blind issuance.

Table V gives the DSS-MS results with 100 initial authorized sanitizers. The per-order operations use a 256-byte payload with four editable blocks. Each Join or Revoke invocation updates ten members, and AcceptEpoch measures local acceptance of the authenticated manifest. Energy is reported in millijoules (mJ), with 95% confidence intervals in brackets.

TABLE V
NET ENERGY PER DSS-MS OPERATION

Sign	Sanitize	Verify	Join	Revoke	AcceptEpoch
2.05 [2.02, 2.08]	3.89 [3.81, 3.96]	2.87 [2.83, 2.91]	10.77 [10.50, 11.04]	3.71 [3.65, 3.78]	2.05 [2.01, 2.09]

F. Practical Deployment Discussion

To ensure the secure and reliable operation of DSS-MS in practical UAV logistics systems, this section briefly discusses several important practical deployment considerations.

First, the TA should be operated by a fully trusted authority, and all participants should securely protect their respective secret keys. According to the secure channel assumptions specified in Section V-B, sensitive communications among the TA, logistics companies, and warehouses should use authenticated encrypted channels (e.g., TLS 1.3) to ensure the confidentiality and integrity of original orders and key materials.

Second, the experiments in Section VIII show that the average signing, sanitization, and verification times are approximately 2.2, 4.2, and 3.1 ms, respectively, with a signature size of 162 bytes. Therefore, DSS-MS is suitable for resource-constrained UAV logistics environments.

Finally, regarding network failures and physical-layer threats, sanitization is performed locally at the warehouse without requiring UAV participation. Since sensitive fields are removed before transmission, even if a UAV is physically captured or wireless communications are intercepted, an adversary cannot recover the removed information solely from the sanitized order and its signature. Thus, although physical-layer attacks may affect UAV communication and operation, they do not directly compromise the privacy guarantee.

IX. CONCLUSION

We propose a DSS-MS to address the challenge of protecting user privacy in typical UAV logistics systems. The scheme ensures that logistics orders remain verifiable, protects removed sensitive data in the final sanitized view, and provides weak accountability between the signer and the sanitizers as a group. Moreover, the flexible sanitizer management mechanism allows the system to adapt to the dynamic needs of logistics and warehouse operations. We provide formal security proofs demonstrating that DSS-MS satisfies a range of security properties, including unforgeability and privacy preservation, under the stated assumptions.

We implemented the scheme on Amov P600 UAV platform. Experimental results show that signature generation, privacy sanitization, verification, and accountability operations can all be completed within 13 milliseconds, demonstrating high efficiency in both computation and communication. Overall, the proposed DSS-MS scheme offers a practical and reliable privacy-preserving solution for UAV logistics systems and has strong potential for broader use in other privacy-sensitive applications involving multiple participants.

REFERENCES

[1] M. A. Khan, I. Ullah, A. Alkhalifah, S. U. Rehman, J. A. Shah, M. I. Uddin, M. H. Alsharif, and F. Algarni, "A provable and privacy-preserving authentication scheme for uav-enabled intelligent transportation systems," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 5, pp. 3416–3425, 2022.

[2] Y. Chen, D. Yang, L. Xiao, F. Wu, and Y. Xu, "Optimal trajectory design for unmanned aerial vehicle cargo pickup and delivery system based on radio map," *IEEE Transactions on Vehicular Technology*, vol. 73, no. 8, pp. 11706–11718, 2024.

[3] K. Ramana, G. Srivastava, M. R. Kumar, T. R. Gadekallu, J. C.-W. Lin, M. Alazab, and C. Iwendi, "A vision transformer approach for traffic congestion prediction in urban areas," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 4, pp. 3922–3934, 2023.

[4] F. Hong, G. Wu, Q. Luo, H. Liu, X. Fang, and W. Pedrycz, "Logistics in the sky: A two-phase optimization approach for the drone package pickup and delivery system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 9, pp. 9175–9190, 2023.

[5] F. Kong, J. Li, B. Jiang, H. Wang, and H. Song, "Trajectory optimization for drone logistics delivery via attention-based pointer network," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 4, pp. 4519–4531, 2023.

[6] S. Wandelt, S. Wang, C. Zheng, and X. Sun, "Aerial: A meta review and discussion of challenges toward unmanned aerial vehicle operations in logistics, mobility, and monitoring," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 7, pp. 6276–6289, 2024.

[7] S. Zhang, H. Zhang, Q. He, K. Bian, and L. Song, "Joint trajectory and power optimization for uav relay networks," *IEEE Communications Letters*, vol. 22, no. 1, pp. 161–164, 2018.

[8] Y. Zeng, Q. Wu, and R. Zhang, "Accessing from the sky: A tutorial on uav communications for 5g and beyond," *Proceedings of the IEEE*, vol. 107, no. 12, pp. 2327–2375, 2019.

[9] Y. Zeng, J. Lyu, and R. Zhang, "Cellular-connected uav: Potential, challenges, and promising technologies," *IEEE Wireless Communications*, vol. 26, no. 1, pp. 120–127, 2019.

[10] "Amazon prime air," Jan. 2025. https://www.aboutamazon.com/news/transportation/amazon-first-drone-delivery-uk-prime-air-location.

[11] A. Lodhi, X. Li, M. A. Saleem, M. A. Lodhi, K. Mahmood, and S. Shamshad, "Dronesec: Efficient and secure communication for resource-constrained drones in iod systems," *IEEE Transactions on Network and Service Management*, vol. 23, pp. 4688–4698, 2026.

[12] S. Iranmanesh, F. S. Abkenar, R. Raad, and A. Jamalipour, "Improving throughput of 5g cellular networks via 3d placement optimization of logistics drones," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 2, pp. 1448–1460, 2021.

[13] H. Guo, W. Gan, M. Zhao, C. Zhang, T. Wu, L. Zhu, and J. Xue, "Prichain: Efficient privacy-preserving fine-grained redactable blockchains in decentralized settings," *Chinese Journal of Electronics*, vol. 34, no. 1, pp. 82–97, 2025.

[14] Y. Yao, P. Dash, and K. Pattabiraman, "Poster: May the swarm be with you: Sensor spoofing attacks against drone swarms," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pp. 3511–3513, 2022.

[15] P. Gope and B. Sikdar, "An efficient privacy-preserving authenticated key agreement scheme for edge-assisted internet of drones," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 13621–13630, 2020.

[16] J. Sun, G. Xu, H. Li, T. Zhang, C. Wu, X. Yang, and R. H. Deng, "Sanitizable cross-domain access control with policy-driven dynamic authorization," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–17, 2025.

[17] R. Alnashwan, P. Gope, and B. Dowling, "Privacy-aware secure region-based handover for small cell networks in 5g-enabled mobile communication," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 1898–1913, 2023.

[18] F. Wang, J. Cui, Q. Zhang, D. He, and H. Zhong, "Blockchain-assisted flexible revocable anonymous authentication in industrial internet of things," *IEEE Transactions on Network Science and Engineering*, vol. 12, no. 1, pp. 518–532, 2025.

[19] M. Gojković and M. Schranz, "Preserving privacy in logistics by using swarm intelligence from the bottom-up," in *2024 IEEE 12th International Conference on Intelligent Systems (IS)*, pp. 1–7, 2024.

[20] G. Ateniese, D. H. Chou, B. de Medeiros, and G. Tsudik, "Sanitizable signatures," in *Computer Security – ESORICS 2005* (S. d. C. di Vimercati, P. Syverson, and D. Gollmann, eds.), (Berlin, Heidelberg), pp. 159–177, Springer Berlin Heidelberg, 2005.

[21] C. Brzuska, M. Fischlin, A. Lehmann, and D. Schröder, "Unlinkability of sanitizable signatures," in *Public Key Cryptography – PKC 2010* (P. Q. Nguyen and D. Pointcheval, eds.), (Berlin, Heidelberg), pp. 444–461, Springer Berlin Heidelberg, 2010.

[22] J. Zhang, J. Cui, H. Zhong, Z. Chen, and L. Liu, "Pa-crt: Chinese remainder theorem based conditional privacy-preserving authentication scheme in vehicular ad-hoc networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 2, pp. 722–735, 2021.

[23] C. Brzuska, M. Fischlin, T. Freudenreich, A. Lehmann, M. Page, J. Schelbert, D. Schröder, and F. Volk, "Security of sanitizable signatures revisited," in *Public Key Cryptography – PKC 2009* (S. Jarecki and G. Tsudik, eds.), (Berlin, Heidelberg), pp. 317–336, Springer Berlin Heidelberg, 2009.

[24] N. Fleischhacker, J. Krupp, G. Malavolta, J. Schneider, D. Schröder, and M. Simkin, "Efficient unlinkable sanitizable signatures from signatures with re-randomizable keys," in *Public-Key Cryptography – PKC 2016* (C.-M. Cheng, K.-M. Chung, G. Persiano, and B.-Y. Yang, eds.), (Berlin, Heidelberg), pp. 301–330, Springer Berlin Heidelberg, 2016.

[25] J. Camenisch, D. Derler, S. Krenn, H. C. Pöhls, K. Samelin, and D. Slamanig, "Chameleon-hashes with ephemeral trapdoors," in *Public-Key Cryptography – PKC 2017* (S. Fehr, ed.), (Berlin, Heidelberg), pp. 152–182, Springer Berlin Heidelberg, 2017.

[26] C. Li, X. Jiao, X. Feng, A. Hu, Q. Shen, and Z. Wu, "Identity-based chameleon hashes in the standard model for mobile devices," *IEEE Transactions on Information Forensics and Security*, 2025.

[27] Y. Zhan, B. Yi, Y. Yang, R. Shi, C. Dong, and M. Huang, "A privilege-constrained sanitizable signature scheme for e-health systems," *Journal of Systems Architecture*, vol. 142, p. 102939, 2023.

[28] Y. Yang, Q. Shi, P. Gope, B. Abdolmaleki, and B. Sikdar, "Pgus: Pretty good user security for thick mvnos with a novel sanitizable blind signature," in *2025 IEEE Symposium on Security and Privacy (SP)*, pp. 1102–1120, IEEE Computer Society, 2025.

[29] Q. Sun, Y. Lu, Y. Sun, and J. Li, "Certificateless sanitizable signature with designated verifier," *IEEE Transactions on Emerging Topics in Computing*, vol. 13, no. 3, pp. 1019–1029, 2025.

[30] O. Allabwani, O. Blazy, P. Lafourcade, C. Olivier-Anclin, and O. Raynaud, "Sanitizable Signatures with Different Admissibility Policies for Multiple Sanitizers," in *ACM Asia Conference on Computer and Communications Security (ASIA CCS '26)*, (Bangalore, India), June 2026. The full version of a publication in AsiaCCS 2026.

[31] W. Shen, J. Qin, J. Yu, R. Hao, and J. Hu, "Enabling identity-based integrity auditing and data sharing with sensitive information hiding for secure cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 2, pp. 331–346, 2019.

[32] Z. Liu, L. Ren, R. Li, Q. Liu, and Y. Zhao, "Id-based sanitizable signature data integrity auditing scheme with privacy-preserving," *Computers & Security*, vol. 121, p. 102858, 2022.

[33] X. Gao, J. Yu, Y. Chang, H. Wang, and J. Fan, "Checking only when it is necessary: Enabling integrity auditing based on the keyword with sensitive information privacy for encrypted cloud data," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 6, pp. 3774–3789, 2022.

[34] T. Li, H. Wang, D. He, and J. Yu, "Designated-verifier aggregate signature scheme with sensitive data privacy protection for permissioned blockchain-assisted iiot," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 4640–4651, 2023.

[35] Z. Liu, L. Ren, R. Li, Q. Liu, and Y. Zhao, "Id-based sanitizable signature data integrity auditing scheme with privacy-preserving," *Computers & Security*, vol. 121, p. 102858, 2022.

[36] H. Yan, H. Hu, Q. Ye, and J. Xu, "Multi-hop sanitizable signature for collaborative edge computing," *Journal of Computer Security*, vol. 33, no. 2, pp. 79–99, 2025.

[37] Y. Liu and Z. Wang, "LSanS: A pairing-free sanitizable signature scheme for accountable VANET certificate derivation," *IEEE Internet of Things Journal*, vol. 13, no. 18, pp. 41929–41942, 2026.

[38] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198–208, 1983.

[39] G. Fuchsbauer, E. Kiltz, and J. Loss, "The algebraic group model and its applications," in *Advances in Cryptology – CRYPTO 2018*, (Cham), pp. 33–62, Springer International Publishing, 2018.

[40] F. Hao, "Schnorr non-interactive zero-knowledge proof," tech. rep., 2017.

[41] Amovlab, "Prometheus autonomous uav open-source project." https://github.com/amov-lab/Prometheus.

[42] "miracl/core," Jan. 2025. https://github.com/miracl/core.