AIGC-CM: An Efficient and Scalable Blockchain Solution for AIGC Copyright Management

Liyuan Ma*†, Dengcheng Hu*†, Xiulong Liu*†, Hao Xu*†, Jianrong Wang*†, Keqiu Li*† *Fellow, IEEE*
*College of Intelligence and Computing, Tianjin University, China
†Tianjin Key Laboratory of Advanced Networking (TANK)
{mly, hdc, xiulong_liu, hao_xu, wjr, keqiu}@tju.edu.cn

IEEE INFOCOM 2025 - IEEE Conference on Computer Communications | 979-8-3315-4305-1/25/$31.00 ©2025 IEEE | DOI: 10.1109/INFOCOM55648.2025.11044774

***Abstract*—With the emergence of distributed Artificial Intelligence Generated Content (AIGC) collaboration, leveraging immutability, transparency, and traceability of blockchain for copyright management has become a key focus. However, existing solutions, such as CopyrightLY and PoAIGC, utilize tokenization and chained ledgers with on-chain hash storage, which exhibit limitations in metadata interpretability, efficient traceability, and formal analysis during distributed collaboration. To this end, we propose AIGC Copyright Management (AIGC-CM), an efficient and scalable blockchain-based framework for AIGC copyright management. Firstly, we propose the blockchain-based MCU concept, integrating feature-based hash and structured metadata to achieve precise copyright authentication and on-chain self-description for AIGC. Next, we design an authentication mechanism based on Hysteresis Signature DAG (HSD) to achieve efficient and secure copyright traceability for MCUs. Furthermore, we employ the UC framework to conduct a formal security analysis of AIGC-CM. To validate the performance and scalability, we implement it on Fabric 2.4 with 4000 LOC and deploy AIGC-CM prototype on 140 nodes. The experimental results show that the average operation time is reduced by 98.86% compared to existing proposals. Compared to the linear increase of chained structure, the average operation time increases by less than 2x when the node scale is expanded 14x.**

***Index Terms*—AIGC, blockchain, and smart contract**

I. Introduction

A. Background and Motivation

The emergence of AIGC along with User Generated Content (UGC) [1], [2] enhances the quality and intricacy of content generation [2]–[4]. It is gradually utilized in multi-medium network environments [5], [6]. According to data from Precedence Research [7], the global AIGC market is expected to reach approximately $73.16 billion by 2030. Without an in-depth exploration of content authentication and its mapping to users, uncontrolled exposure can lead to severe security implications, including tampering, forgery, and unauthorized use [8]. In January 2024, the ICO issued a legal basis analysis report on the AIGC's web data scraping or processing to train AIGC models [9]. Digital watermark [10]–[13], a technique that embeds imperceptible information for unique content identification, faces challenges in real-time tracking and exposure of vulnerabilities under high-compression scenarios.

The first two authors contribute equally to this paper.
Corresponding authors: Xiulong Liu, Hao Xu

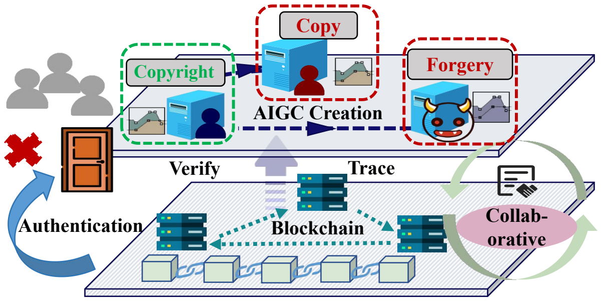

Fig. 1: Blockchain and AIGC copyright management.

Blockchain technology, with decentralized, transparent, and tamper-resistant characteristics, is applied to various scenarios [14]–[17]. Notably, smart contracts offer a versatile tool for tackling AIGC copyright protection issues [18]. The definition and circulation of blockchain-based digital collectibles (*e.g.,* NFTs [19]), inspire the idea of tokenizing AIGC and on-chain management. In 2022, OpenAI designed the AIGC Chain [20] by fostering a collaborative and participatory approach to training distributed AIGC models. However, it lacks AIGC on-chain representation information and ignores the efficiency of traceability. Motivated by existing issues, we investigate strategies for blockchain-based AIGC copyright management, aiming to establish a reliable copyright infrastructure for the authentication and legality of AIGC while preventing tampering, forgery, and unauthorized use.

B. Limitations of Prior Arts

There are two main AIGC copyright management categories: traditional web applications and tokenized methods.

For traditional web applications, techniques like passwords [21], tokens [22], and biometrics [23] are used for copyright authentication. However, they cannot effectively prevent malicious manipulation of AIGC due to their strong reliance on centralization. For tokenized methods, CopyrightLY [24] uses NFTs with semantically rich metadata to establish content ownership. The NFTs maintain the association of the AIGC with the copyright notice and specify restrictions during token transfers. However, the related information of CopyrightLY is stored as transactions in the blockchain ledger, requiring traversing the ledger for traceability. Liu *et al.* [25] introduces PoAIGC to protect copyrights of AIGC. PoAIGC employs

TABLE I: The state-of-the-art for AIGC management.

Solutions	Interpretable Metadata	Efficient Traceability	Formal Analysis
CopyrightLY [24]	×	×	×
PoAIGC [25]	×	×	×
PoAIGC+ [26]	×	×	✓
AIGC-CM	✓	✓	✓

a challenge protocol to detect plagiarism and forgery, canceling the registrations of erroneous contents. The authors also propose a reputation-based strategy for ESP selection. However, these schemes lack AIGC authentication and traceability design. Additionally, both CopyrightLY and PoAIGC lack formal security analysis to evaluate their security. Lin *et al.* [26] presents a semantic communication framework using blockchain to address data exchange and trust-building challenges. They introduce a verification mechanism to ensure digital content authenticity and prevent unpredictable outcomes, focusing on trustworthiness of AIGC. In contrast, zero-knowledge proofs prioritize the trustworthiness of real-time generated content but lack reliable authentication and efficient traceability.

We summarize the state-of-the-art blockchain-based solutions for AIGC copyright management, as shown in TABLE I. Current approaches only focus on utilizing blockchain to achieve hash preservation on-chain with multi-user participation but lack on-chain metadata interpretability, data querying efficiency, and formal security analysis, which discourages user participation in distributed AIGC collaboration.

C. Proposed Framework

We propose AIGC-CM, a comprehensive system that guarantees verifiable authentication and efficient traceability for AIGC. The core idea of AIGC-CM lies in establishing an efficient and scalable framework for blockchain-based AIGC copyright management, which involves the integration of feature extraction method and Directed Acyclic Graph (DAG) data structure. Specifically, AIGC-CM comprises two aspects. On the one hand, we ensure the unique identification and precise verifiability of intellectual property in distributed AIGC collaboration by designing an effective feature extraction method, and achieving one-to-one matching with on-chain AIGC and rights holders. On the other hand, we facilitate AIGC recreations and modifications based on the version iteration relationships to build an efficient traceability structure for managing the AIGC copyright lifecycle.

D. Challenges and Solutions

We need to solve the following two technical challenges during AIGC-CM design and implementation.

The first challenge is how to realize interpretable on-chain content authentication with a secure preview. Existing work hashes AIGC content summaries on-chain, resulting in on-chain metadata lacking interpretability, which reduces content querying efficiency and relevance management, leading to operation delays. To tackle this challenge, we propose the concept of MCU (Minimum Copyright Unit) to establish a one-to-one mapping of AIGC and on-chain copyright metadata. It represents distinct, standalone creative content using SimHash, through weighted feature extraction and dimensionality reduction to facilitate precise and efficient intellectual property management. MCUs distinguish themselves from traditional blockchain tokens (*e.g.* NFTs) that only store unique identifiers such as hash. By upgrading ledger states in a hierarchical format, MCUs can record AIGC and its structured rich-text description on-chain without exposing original content. The copyright attribute content also greatly enhances the interpretation and self-description of AIGC metadata on-chain, and contributes to a more open AIGC creative ecology.

The second challenge is how to build an efficient and secure DAG structure for copyright management. As the volume of AIGC increases, the chain-based management structure needs to be traversed. Likewise, directly using a DAG may result in decreased query efficiency as the number of nodes increases. Basic data structures are inefficient at tracing copyright changes and identifying their pathways. Meanwhile, the current systems are insufficient for achieving rapid identity verification in large-scale content authentication scenarios. To tackle these challenges, we introduce Hysteresis Signatures DAG (HSD), a new data structure for copyright traceability. HSD uses SimHash for similarity or relative MCU organization, with a DAG as the foundational structure. Locality-Sensitive Hashing (LSH) is implemented to build the index for fast identification of relevant sub-DAGs. For each sub-DAG, we utilize hysteresis signatures to build a signatures-based DAG to verify MCU owner identities. Additionally, we integrate an identity verification mechanism with CA certificates to ensure a one-to-one mapping between on-chain accounts and off-chain users. Furthermore, we employ formal proofs to verify the rationality and completeness of AIGC-CM.

E. Novelty and Advantages

This paper for the first time addresses the problem of AIGC copyright management with non-interpretable on-chain data authentication and inefficient traceability. The novelty of this paper is demonstrated by three aspects over the prior arts: (i) proposing the concept of blockchain-based MCU for the first time, innovatively integrating SimHash and structured metadata to achieve precise copyright authentication and on-chain self-description for AIGC; (ii) proposing AIGC-CM, with hysteresis signatures-based DAG to achieve efficient and secure copyright traceability for MCUs, and conduct a formal security analysis of AIGC-CM based on UC framework; (iii) implementing 4000 LOC AIGC-CM prototype based on Fabric 2.4 and deployed it on 140 nodes. Experiment results show that the average latency (*e.g.*, create and verify) is reduced by 98.86% compared to EIP-712. Compared to the linear increase of chained-structure, AIGC-CM latency increases by less than 2x when the node scale expands 14x.

The remainder is organized as follows. Section II provides the system overview. We further describe the detailed design

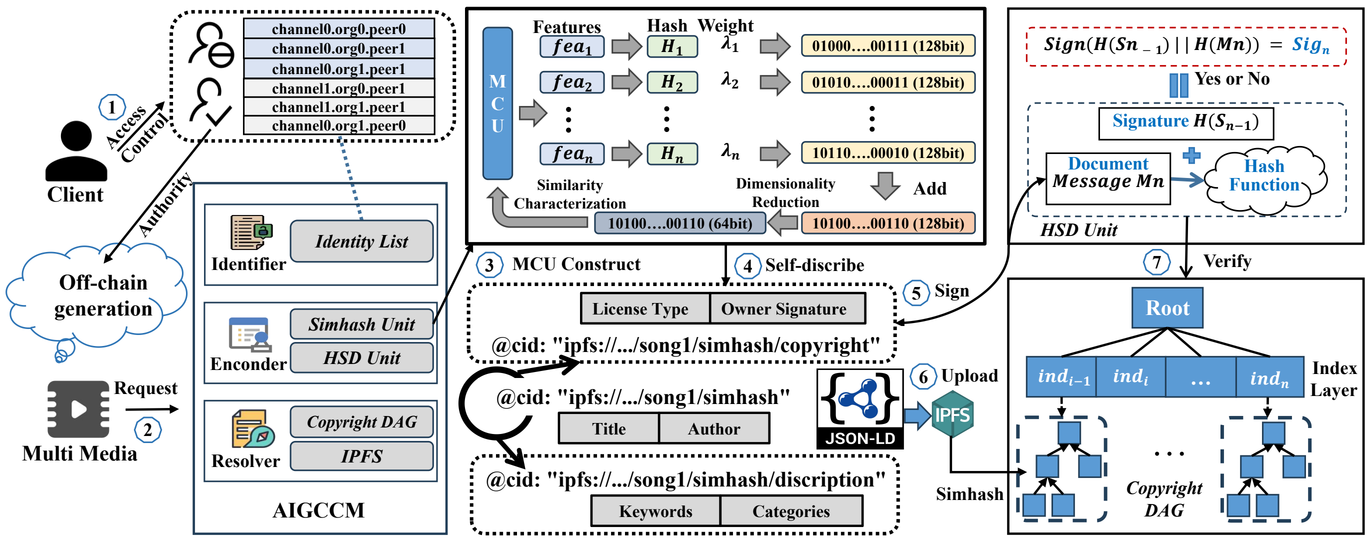

Fig. 2: An illustration of AIGC-CM. After access control, the user initiates an AIGC collaboration request. The encoder constructs the MCU, which includes self-descriptive information in a structured format and links to IPFS. The resolver completes the signature authorization and incorporates it into DAG.

in Section III. Section IV gives UC formal security analysis. Section V focuses on system implementation and evaluation. Lastly, Section VI concludes the paper.

II. System Overview

A. System Module

The system consists of three core modules, as shown in Fig. 2. The **Identifier Module** controls user identity and access within the system. The **Encoding Module** is essential for managing generated content within the system, which focusing on defining the MCU for effective rights management. It provides the foundation for an organized rights management process. The module creates a secure, traceable link between AIGC entities and rights holders using a novel authentication system based on hysteresis signatures and DAGs. The **Resolver Module** efficiently manages content and ownership through DAG construction, segmented by thresholds to control size. It employs locality-sensitive hashing (LSH) for efficient search within sub-DAGs and integrates a deterministic CID-based mapping for AIGC storage in IPFS [27].

B. AIGC-CM Protocol

1) **Access Control**. Users register their identities by sending a request message $(\text{register}, userID, userInfo)$ to the *Identifier Module*. The system completes decentralized identity management by forwarding the request to the *Certificate File System* (CFS) and *Identity Contract* on the blockchain. Access permissions are assigned using the transaction $\mathsf{Tran}_{\mathsf{AIGC}}^{\mathsf{register}}(userEnroll, userID, Certificate)$, which records identity and resource permissions. 2) **AIGC Create**. Registered users create content within their permissions, transforming it into MCU. Users can query the generated content by using $\mathsf{Tran}_{\mathsf{AIGC}}^{\mathsf{query}}(queryType, queryValue)$. Content is created via $\mathsf{Tran}_{\mathsf{AIGC}}^{\mathsf{create}}(Msg)$, where Msg includes encoded content and metadata. The system generates a SimHash from Msg and links the MCU using off-chain storage (*e.g.,* IPFS). 3) **AIGC Recreate**. The resolver contract enables derivative creations and manipulation of generated content. Users update the copyright information and hysteresis signatures of the MCU through $\mathsf{Tran}_{\mathsf{AIGC}}^{\mathsf{reCreate}}(Msg)$. MCU ownership can be transferred using $\mathsf{Tran}_{\mathsf{AIGC}}^{\mathsf{transfer}}(Hash, userID)$. 4) **Regulate and Trace**. In derivative disputes, the hysteresis signatures DAG is essential for tracing and confirming ownership. Plagiarism or infringement in generated content can be detected by $\mathsf{Tran}_{\mathsf{AIGC}}^{\mathsf{verify}}(Hash)$, which compares signatures along the creation lineage. This approach strengthens the copyright protection framework. Contracts maintain a historical record, aligning with the resolver contract's ownership mechanisms. If verification fails, the system initiates $\mathsf{Tran}_{\mathsf{AIGC}}^{\mathsf{delete}}(Hash)$ to remove infringing content from the DAG.

TABLE II: The basic variables and related definitions.

Variables	Definitions	Variables	Definitions
C, C'	Generated contents	$\mathcal{SH}$	SimHash function
$\mathcal{H}$	Basic hash function	M	Information of content
$\mathcal{D}$	DAG structure	S	Signature of content

III. Detailed Design

This section defines the minimum copyright unit and introduces two structures: CopyrightTree for efficient copyright management, and the Hysteresis Signature DAG for rapid verification. TABLE II presents the key variables and definitions.

A. Minimum Copyright Units and SimHash

In AIGC-CM, the MCU is utilized to represent the core intellectual property of generated content, and it also serves as the fundamental unit for copyright management. MCUs encapsulate content into standardized, concise units, reflecting the creative process and aligning with intellectual property

Algorithm 1: SimHash Generation and Verification

```
1  Function generateSimHash(Content $C$):
2      $f_C, W_C \leftarrow C$ ;      // Extract features and weights
3      for each feature $f_C^i$ in $f_C$ do
4          $h_i \leftarrow \mathcal{H}(f_C^i)$ ;      // Hash each feature
5      $\mathcal{V} \leftarrow$ FixedLength$(h_1 * w_1, h_2 * w_2, \ldots, h_n * w_n)$ ;
       // Overlay binary hashes into a fixed-length vector
6      $S \leftarrow$ SimHash$(\mathcal{V})$ ;      // Generate SimHash
7  Function verifySimHash(Content $C$, Modified Content $C'$, SimHash of $C$):
8      $S' \leftarrow$ generateSimHash$(C')$;
9      if $S' \neq S$ then
10         Output: Content inconsistent;
11     else
12         Output: Content consistent;
```

Algorithm 2: CT Construction for AIGC-CM

```
1  Function manageCT(New MCU $C'$):
2      CT $\leftarrow$ InitializeCT();
3      while $C'$ is inserted into CT do
4          $\mathcal{SH}(C') \leftarrow$ generateSimHash$(C')$;
5          $N_I \leftarrow$ findMostSimilarNode$(I, \mathcal{SH}(C'))$;
6          if $N_i$ exists then
7              $proof \leftarrow$ verifyMerkleTree$(N_I, Merkle)$;
8              if $proof$ is valid then
9                  insertContentIntoDAG(CT, $C'$);
10     return CT
```

principles within the AIGC-CM. We design MCU as a hierarchical JSON-LD structure that enhances the URI function of metadata ($mcu.Schema = song$), encompassing basic information (title, author), copyright details (license type, owner signature), and content description (keywords, categories). This improves self-descriptiveness and composability of metadata, facilitating queries and filtering at various levels.

The encoding scheme is crucial for preserving and authenticating generated content in the AIGC-CM. SimHash enhances copyright management by efficiently computing similarities between MCUs, enabling the swift identification of potential plagiarism or infringement. Its efficiency in processing extensive data bolsters intellectual property protection. SimHash assigns a distinct value to each MCU, providing a robust mechanism for verifying ownership and originality.

For a given MCU C, the SimHash construction process is as follows: 1) *Feature Extraction*. The MCU C is initially subjected to a feature extraction process, yielding a high-dimensional feature vector f_C. Each feature f_C^i is represented as a vector with an associated weight w_i, collectively capturing the intrinsic characteristics of the content. 2) *Hash Length Fixing*. Each feature f_C^i is processed through the hash function $\mathcal{H}(\cdot)$, yielding a hash value. The generated hash values are aggregated into a fixed-length vector $\mathcal{V}$, with each bit's contribution weighted by w_i during the combination. The vector $\mathcal{V}$ is updated by incrementing or decrementing each position based on the bit value of the hash, reflecting 1 or 0. 3) *SimHash Generation*. The SimHash $\mathcal{SH}(C)$ is obtained by thresholding vector $\mathcal{V}$. Each bit in $\mathcal{V}$ is compared to a threshold, and the final SimHash is a binary vector where each bit is set to 1 if the corresponding weighted sum in $\mathcal{V}$ is above the threshold, and 0 otherwise.

The $\mathcal{SH}(C)$ serves as a unique identifier for C, leveraging its sensitivity to changes for efficient copyright management and distinguishing between original and derivative works. In AIGC-CM, identifying minor alterations is crucial for safeguarding the intellectual property rights of creators.

We mitigate blockchain storage and query load by using IPFS (Inter Planetary File System) [27] for off-chain URI mapping, storing content data on IPFS and only metadata and URIs on the blockchain. Each MCU's SimHash index maps to its corresponding IPFS URI, enabling seamless access to off-chain content. We employ JSON-LD to structure MCU metadata in a hierarchical manner, encompassing basic information, copyright details, and content descriptions. This approach ensures metadata clarity, ease of querying, and compatibility with semantic web standards, facilitating cross-platform data sharing.

B. Copyright Tree

We introduce the Copyright Tree (CT) to manage copyrights of MCUs, as shown in Fig. 3. The CT integrates three key components: DAG structures, an index table, and a Merkle Tree. The DAG is utilized to effectively manage MCUs. The CT contains multiple DAGs that group related MCUs. A LSH-based index of sub-DAG roots facilitates rapid traceability. The Merkle tree ensures the verifiability of the sub-DAG.

DAG Structure. The DAG facilitates efficient insertion and management of MCUs. When inserting new MCU C into the CT, a unique SimHash $\mathcal{SH}(C)$ is generated. The node most similar to C (based on $\mathcal{SH}(C)$) is identified as the predecessor pre. C is then inserted after pre, and the predecessor's information is specified in C for future verification. When there is a need to confirm the traceability path of MCU C', the predecessor field in the DAG is used. In Fig. 3, if node N_{i+2} contains the predecessor information, pre is quickly confirmed as node N_i, generating the path $N_i \rightarrow N_{i+2}$. The DAG-based structure provides a robust and efficient framework for managing MCUs, offering a potential solution to the complex issue of digital copyright management.

Index Table. The LSH in our system involves representing each element as the f_C. These vectors are the root node of each sub-DAG. When inserting a new MCU, f_C is processed to identify its nearest neighbor in the existing index I, and the nearest index I_{nea} is executed as:

$$I_{nea} = \arg\min_{\mathbf{ind} \in I} |f_C - \mathbf{ind}| \qquad (1)$$

If I_{nea} exists, f_C is inserted into the sub-DAG. Otherwise, f_C is added to I, creating a new DAG for related AIGCs.

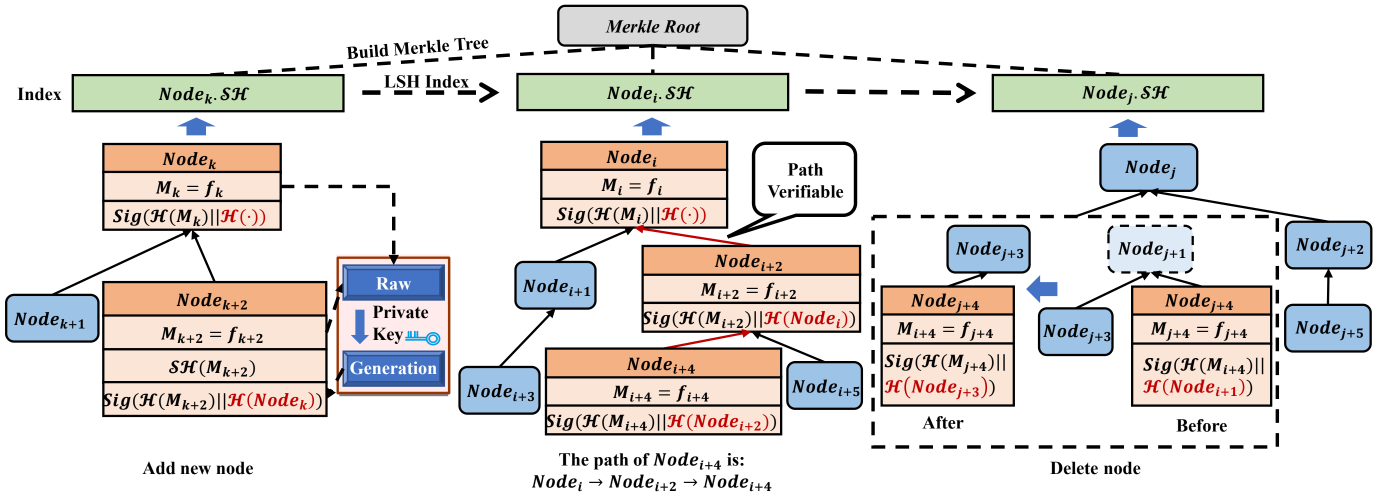

Fig. 3: The basic data structure for copyright management in AIGC-CM. The HSD is composed of multiple DAGs indexed by an index table. This innovative structure offers an efficient solution for copyright traceability. Additionally, HSD incorporates the hysteresis signature to ensure trusted authentication. HSD primarily encompasses functions such as *create*, *getPath*, *delete*. When a user creates a new song, HSD inserts the new content into the appropriate DAG based on the SimHash. Simultaneously, CT generates the complete path of the hysteresis signature, facilitating subsequent authentication. For traceability purposes, HSD can swiftly provide the copyright change path to users, who can then verify the accuracy and security by referring to the included signature. When a user deletes specific content, the system not only updates the DAG but also adjusts the complete signature path accordingly.

Merkle Tree structure. The index table serves as the root of the Merkle Tree. To incorporate a new MCU C' into the ledger, we follow three steps: 1) Identify the most appropriate position for C' in the CT. 2) Evaluate the trustworthiness of the corresponding sub-DAG position using a Merkle proof to validate its completeness. 3) If validation passes, insert C' into the specific sub-DAG and update CT.

CT integrates DAGs, LSH, and Merkle Trees, which provide an efficient and scalable solution for managing copyright in AIGC-CM, enhancing data reliability and trustworthiness.

C. Hysteresis Signature DAG

The HSD brings notable improvements, especially in transitioning from a linear list structure to a more versatile DAG. The protocol of the HSD is illustrated in Fig. 4.

During generation, the system leverages a private key k in tandem with distributed identity management and the whitelist smart contract. For a $\mathsf{Tran}^{\mathsf{create}}_{\mathsf{AIGC}}(M_n)$ request, with the previous signature S_{n-1}, the system calculates the hash $H(S_{n-1})$ and $H(M_n)$ using the CalculateHash function. It then combines these hashes to construct X_n (*i.e.,* $X_n = H(S_{n-1})||H(M_n)$). To maintain consistency with the DAG structure, the system extract a portion of S_{n-1} to retrieve X_{n-1}. The private key k generates the signature Sig for X_n through the (Sign) function. The new signature S_n is then stored, encompassing S_{n-1}, $H(M_n)$, and Sig. The verification phase involves confirming the authenticity of the Hysteresis Signature. Given a message M_n, public key pk, and the computed signature $H(S_{n-1})||H(M_n)$, the system performs a verification process using the VerifySig function. This step ensures that only valid and authorized signatures are accepted, contributing to the overall security of the system.

These hysteresis signatures function as intelligent markers, encapsulating alterations or additions made by the derivative creator while preserving the HSD of the original AIGC creation. The depth of information embedded within these HSDs—including creator attributions, timestamps, generation parameters, and signature hierarchies—anchors derivative works within distributed ledgers. Here we further reduce the storage overhead of the signature logs, $X_n = h(X_{n-1}) \,||\, h(M_n), X_{n-1} = S_{n-1}[0 : (\mathsf{len}(S_{n-1}) - \mathsf{len}(Sig_n))]$. Efficient extraction of validation paths from DAG signature logs requires the development of search algorithms such as DFS (Depth First Search). In general, for a graph with V vertices and E edges, the time complexity is $O(V + E)$. These mechanisms optimize the exploration of signature logs, ensuring rapid identification of validation paths and quick tracing of infringement origins. The utilization of hysteresis signatures, intelligent contract management, and IFPS's decentralized storage not only fortifies the integrity of the copyright chain but also streamlines the validation and resolution process in copyright disputes.

IV. Security Analysis

A. Security Goals

AIGC-CM guarantees the following security goals: 1) *Tamper Resistance:* Given a message M_n, private key k, and signature record $S_{n-1} = (M_{n-1}, \mathsf{Sig}_{n-1})$, it should be computationally infeasible for an adversary to modify S_{n-1} or generate a valid signature for a modified message M'_n without possessing the corresponding private key k. 2) *Forgery Prevention:* Adversaries should not be able to forge a valid signature Sig_n for a message M_n without having access to the legitimate private key k associated with the public key pk. 3) *Illegal Usage Prevention:* Only messages with valid signatures, verified using the corresponding public key pk, should be accepted for accessing or using AIGC works.

A. Generation of Hysteresis Signature DAG
Given message M_n, Private key k, Signature record S_{n-1}, generate $H(S_{n-1})$, $Sign(H(S_{n-1})||H(M_n))$ and signature record S_n with genSignature(M_n, k, S_{n-1}):

```
- $H(S_{n-1}) \leftarrow CalculateHash(S_{n-1})$
- $H(M_n) \leftarrow CalaculateHash(M_n)$
- $X_n \leftarrow H(S_{n-1})||H(M_n)$
- $X_{n-1} \leftarrow S_{n-1}[0 : (len(S_{n-1}) - len(Sig))]$
```

Generate a signature for X_n using the private key pk:

```
- $Sig \leftarrow Sign(X_n, pk)$
- $S_{n-1} \leftarrow S_n$
- $S_n \leftarrow Store(S_{n-1}, H(M_n), Sig)$
```

B. Verification of Hysteresis Signature
Given message M_n, public key pk, signature $H(S_{n-1})$ and $Sign(H(S_{n-1})||H(M_n))$, verify the Hysteresis signature:

```
- $H(M_n) \leftarrow CalculateHash(M_n)$
- $X_n \leftarrow H(S_{n-1})||H(M_n)$
- If $VerifySignature(M_n, H(S_{n-1}),$
  $Sign(H(S_{n-1})||H(M_n)), pk) == True$
    Output: YES
```

C. Verification of Hysteresis Signature DAG
Given signature record S_n, public key pk, through $getPath$ to get signature records $S_1, ..., S_{n-1}$, using verifySignatureDAG$(S_n, pk, S_1, ..., S_{n-1})$ to verify the signature dag:

```
Procedure: GetPath($S_n$):
  - $path \leftarrow []$                          ▷ Initialize empty path
  - Procedure: TraversePath($currentNode, currentPath$)
    – $currentPath$.push($currentNode$)
    – If $currentNode$.hash = $S_n$.hash
          $path \leftarrow currentPath$.slice()  ▷ Copy the path
    – Else
          For each $child$ in $currentNode$.children
              TraversePath($child, currentPath$)
              $currentPath$.pop()                 ▷ Backtrack
  - TraversePath(root, [])      ▷ Start traversal from the root
  - Return $path$
Procedure: VerifySignatureDAG($S_n, pk, S_1, ..., S_{n-1}$)
  - $path \leftarrow GetPath(S_n)$     ▷ Retrieve the creation path
  - For $i$ from 1 to $m$
    – If $VerifySignature(S_i, pk) == False$
          Output: NO
    – Else if $CalculateHash(S_i) \neq S_i.H(S_i)$
          Output: NO
    – Else if $i > 1$
          $VerifySignature(S_{i-1}, S_i, pk)$
  - Output: Verification Success
```

Fig. 4: The protocol of Hysteresis Signature DAG.

B. UC Security Definition

We are now prepared to establish the primary UC framework [28] for AIGC-CM. Let us introduce AIGC-CM as a protocol equipped with the Ledger functionality ($\mathcal{F}_L$) and Contract Functionality ($\mathcal{F}_C$) similar to FairSwap [29].

The Ledger Functionality ($\mathcal{F}_L$): The internal state consists of $\mathcal{L}$, which records all transactions $Tx = \{tx_1, tx_2, \ldots, tx_n\}$ within the blockchain, $\mathcal{L} : \alpha \rightarrow \mathcal{W}$, where transactions address $\alpha \in \{0,1\}^\lambda$ and workID $\mathcal{W} \in \{0,1\}^\lambda$. The interface $\mathsf{AddTrans}(tx)$: $\mathcal{L} \leftarrow \mathcal{L} \cup \{tx\}$ works via sending message : $(\mathsf{AddTrans}, tx, content, signature)$. If tx involves work modification, it updates $\mathcal{W}$ as: $\mathcal{W}(a) \leftarrow \mathcal{W}(a) \cup \{w\}$, where account a is involved in tx with work w modified. It also simulates the irrational parties, more detail is shown in [30].

The Contract Functionality ($\mathcal{F}_C$): $\mathcal{SC}$ maintains the set of active contract instances, each corresponds to an operation $\mathcal{SC} : \beta \rightarrow c$, where $\beta \in \{0,1\}^\lambda$, denotes the contract account of β. The interface $\mathsf{AccessControl}(user, resource)$ ensures that $user$ has the necessary access to $resource$; $\mathsf{VerifySig}(Msg, Sig, pk)$ validates the digital signature Sig for message Msg using pk; $\mathsf{Execute}(contract, params)$ executes $contract$ with specified $params$, $contract \in \{register, createWork, reCreate, changeOwner, delete\}$.

Definition 1: *Let $\lambda \in \mathbb{N}$ be a security parameter and $x \in \{0,1\}^*$ be an auxiliary input. AIGC-CM operates within the ($\mathcal{F}_L$, $\mathcal{F}_C$)-hybrid environment. We define AIGC-CM as the realization of the ideal functionality $\mathcal{F}_H$ if, for every adversary A, there exists a simulator S such that for all polynomial-time computable (PPT) environments Z:*

$$\text{EXEC}^{\mathcal{F}_L,\mathcal{F}_C}_{\text{AIGC},\mathcal{A},\mathcal{Z}}(\lambda, x) \approx \text{IDEAL}^{\mathcal{F}_L;\mathcal{F}_C}_{\mathcal{F}_H,\mathcal{S},\mathcal{Z}}(\lambda, x) \qquad (2)$$

The interaction between the environment $\mathcal{Z}$ with AIGC$\mathcal{A}$, can be represented as $\text{EXEC}^{\mathcal{F}_L,\mathcal{F}_C}_{\text{AIGC},\mathcal{A},\mathcal{Z}}(\lambda, x)$. To denote the interaction with the ideal functionalities $\mathcal{F}_H$ and simulator $\mathcal{S}$, we use $\text{IDEAL}^{\mathcal{F}_L,\mathcal{F}_C}_{\mathcal{F}_H,\mathcal{S},\mathcal{Z}}(\lambda, x)$. The symbol "$\approx$" means computational indistinguishability.

C. Security Proof

Theorem 1. *Protocol of AIGC-CM securely realizes functionality $\mathcal{F}_H$ in the $(\mathcal{F}_L, \mathcal{F}_C)$ hybrid model.*

Proof. Let A be an adversary in the $(\mathcal{F}_L, \mathcal{F}_C)$-hybrid model. We construct an ideal-process adversary $\mathcal{S}$, the simulator, such that the view of any environment $\mathcal{Z}$ from an interaction with A and AIGC is distributed identically to its view of interaction with $\mathcal{S}$ in the ideal process for $\mathcal{F}_H$. For any action, $\mathcal{Z}$ sends a message m to A. The simulator $\mathcal{S}$ intercepts this and sends m to $\mathcal{F}_H$ in the name of A. In the next round, if $\mathcal{C}$ responds to $\mathcal{F}_C$, $\mathcal{S}$ sends the appropriate message to $\mathcal{F}_N$ and outputs the corresponding result (*e.g.*, created). $\mathcal{S}$ ensures that the simulation of these actions is indistinguishable from a real execution by emulating the interactions accurately. If A or $\mathcal{C}$ is corrupt and does not forward or reply to messages, $\mathcal{S}$ terminates the simulation.

For **Access Control**, $\mathcal{Z}$ initiates the process by sending a message $m = (\mathsf{register}, uid, userAttributes, Sig_A)$ to A. Here, uid is the identifier for user A, $userAttributes$ represents the attributes of user A, and Sig_A is the signature generated by user A to authenticate the registration message. $\mathcal{S}$ forwards this request to the contract functionality $\mathcal{F}_C$ as $(\mathsf{register}, uid, userAttributes, Sig_A)$. In response, $\mathcal{F}_C$ executes the contract function $\mathsf{Execute}(contract, params)$, which is simulated by $\mathcal{S}$. $\mathcal{S}$ updates the whitelist ledger

TABLE III: System and Setup Settings

Parameters	Values
Blockchain & Orderer	Hyperledger Fabric v2.4 & Kafka
Organizations & Peers	Org1, Org2 & (5, 10, 30, 50, 70)
Bandwidth	1.5 Gbps Intranet & 10 Mbps Internet
Server Type	AMD EPYC Bergamo 8 core 16 GB
Server Location	Beijing, Jakarta
Designed Codes	4000 LoC JS
Repeat Times	20 (Morning), 20 (Evening)

DAG using $\mathcal{F}_{\mathrm{L}}$'s $\mathsf{AddTrans}(tx)$ method to record the registration transaction. Furthermore, $\mathcal{S}$ updates the hysteresis signature DAG using the $genSignature(userAttributes, k, S_{n-1})$ function as described previously. $\mathcal{S}$ simulates the registration of user A into the system, ensuring that all actions taken by A are indistinguishable from a real execution. $\mathcal{F}_{\mathrm{C}}$ receives a response (registered) and sends it to A. Conversely, if A is corrupt, such as not forwarding or replying to messages, then $\mathcal{S}$ outputs (access denied) and terminates the simulation.

For **Work Creation**, $\mathcal{Z}$ initiates the process by sending a message $m = (\mathsf{createWork}, uid, simHash_A, Msg_A, Sig_A)$ to $\mathcal{F}_{\mathrm{L}}$. Here, uid is the user identifier, $simHash_A$ is the unique identifier for the work created by user A, Msg_A represents the content of the work, $params$ includes additional parameters relevant to the creation process, and Sig_A is the signature generated by user A to authenticate the message. If $\mathcal{F}_{\mathrm{L}}$ forwards this request to the Contract Functionality $\mathcal{F}_{\mathrm{C}}$ in round 1, then $\mathcal{S}$ sends m to the ideal functionality $\mathcal{F}_{\mathrm{H}}$ in the name of A. In the subsequent round, if the Contract Functionality $\mathcal{F}_{\mathrm{C}}$ responds with $(\mathsf{createWork}, uid, simHash_A, Msg_A, params, Sig_A)$, $\mathcal{S}$ sends $(\mathsf{createWork}, uid, simHash_A, Msg_A)$ to $\mathcal{F}_{\mathrm{L}}$ in the name of $\mathcal{F}_{\mathrm{C}}$ and outputs (work created). Additionally, if $\mathsf{VerifySig}(Msg_A, Sig_A, pk_A)$ outputs 1, $\mathcal{S}$ calls the ledger functionality $\mathcal{F}_{\mathrm{L}}$'s $\mathsf{AddTrans}(tx)$ method to record the creation transaction, updating the ledger DAG. Furthermore, $\mathcal{S}$ uses the $genSignature(Msg_A, k, S_{n-1})$ function to update the HSD. It includes calculating $H(S_{n-1}) \leftarrow \mathsf{CalculateHash}(S_{n-1})$, calculating $H(Msg_A) \leftarrow \mathsf{CalculateHash}(Msg_A)$, concatenating $X_n \leftarrow H(S_{n-1})||H(Msg_A)$, generating a signature for X_n using the private key k: $Sig \leftarrow \mathsf{Sign}(X_n, k, S_{n-1})$, and updating the signature record $S_n \leftarrow (S_{n-1}, H(Msg_A), Sig)$. $\mathcal{S}$ simulates the creation, ensuring that all actions A takes are indistinguishable from a real execution. Conversely, if A does not forward or reply to messages, or Sig_A is not correct, the $\mathsf{AddTrans}(tx)$ method won't be triggered, then $\mathcal{S}$ outputs (creation failed) and stops the procedure. $\square$

The proofs for *Work Recreation*, *Ownership Change*, and *Work Modification* are similar. Therefore, AIGC-CM securely realizes functionality $\mathcal{F}_{\mathrm{H}}$ in the $(\mathcal{F}_{\mathrm{L}}, \mathcal{F}_{\mathrm{C}})$-hybrid model.

V. Implementation and Evaluation

A. Implementation and Setup

System Implementation. AIGC-CM is implemented on Hyperledger Fabric [31] release-2.4 branch [32] with 4000 LoC. We use $X.509$ for user authentication, and $secp256k1$ for HSD generation and verification. We encompass a multi-node network consisting of an orderer committee and two distinct organizations in Beijing and Jakarta, each containing 5, 10, 30, 50, and 70 peer nodes respectively, as shown in Fig. 5. The configuration details are outlined in Table III.

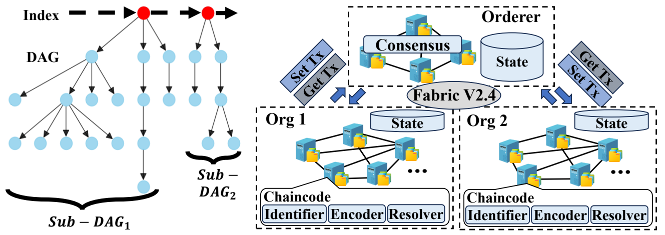

Fig. 5: Visualization of HSD and experiments architecture.

Experiment Setup. In AIGC-CM, a structured data model encapsulates song attributes such as 'style', 'docType', 'maker', and 'record'. We initiate it in a multi-node environment by constructing the HSD shown in Fig. 5. The red nodes in the figure represent the index layer nodes, while the blue nodes represent individual nodes within the sub-DAG. Each node is uniquely indexed using SimHash.

B. Experiment Evaluation

We perform a detailed evaluation of system performance and system scalability. The former refers to the performance comparison of MCUs under different types of operations and *Operation Request Rate* (*OPRR*), including the number of *Operations Processed Per Second* (*OPPS*) and the *Operation Processing Time* (*OPT*), as well as *OPT* under different dataset sizes. The latter mainly refers to the impact of the node scale on *OPT* and *Resource Consumption Rate* (*RCR*) for various operations. We use the ERC-3525, EIP-712 (contract standard that enables divisible ownership and transfer of unique assets on Ethereum, from where we drew design inspiration for MCUs) and chained-structure as baselines. Each group of experiments is repeated 20 times in the morning and evening.

System Performance. As shown in Fig. 6 (a) and Fig. 6 (b), in terms of create operation request rates, AIGC-CM shows a significant advantage, particularly at high operation request rates. *OPPS* remains stable above 20 op/s, while *OPT* belows 300 ms. Even at 500 operations per second, the OPPS of EIP-712 is higher, but its *OPT* is above 17 s, and the red cross marks at specific positions highlight its shortcomings at frequent RPC requests. This result demonstrates that the MCUs not only realize the self-descriptive and interpretable metadata of the AIGC creation ecosystem but also exhibit higher operational performance. Meanwhile, by comparing Fig. 7 (a) and Fig. 7 (b), we can observe that AIGC-CM achieves an average OPT reduction of 39.21%, 28.3%, and 53.7% for $\mathsf{initLedger}$, $\mathsf{createSong}$, and $\mathsf{querySong}$, respectively. This is because the utilizes of index in HSD based on SimHash improves query efficiency. Additionally, the tree-based structure for adding nodes significantly enhances the system's capability to handle operations in parallel. We also compare the performance of AIGC-CM and chain-based structures across varying dataset sizes. As shown in Fig. 6 (b), the data clearly indicate that AIGC-CM consistently achieves lower time costs across all

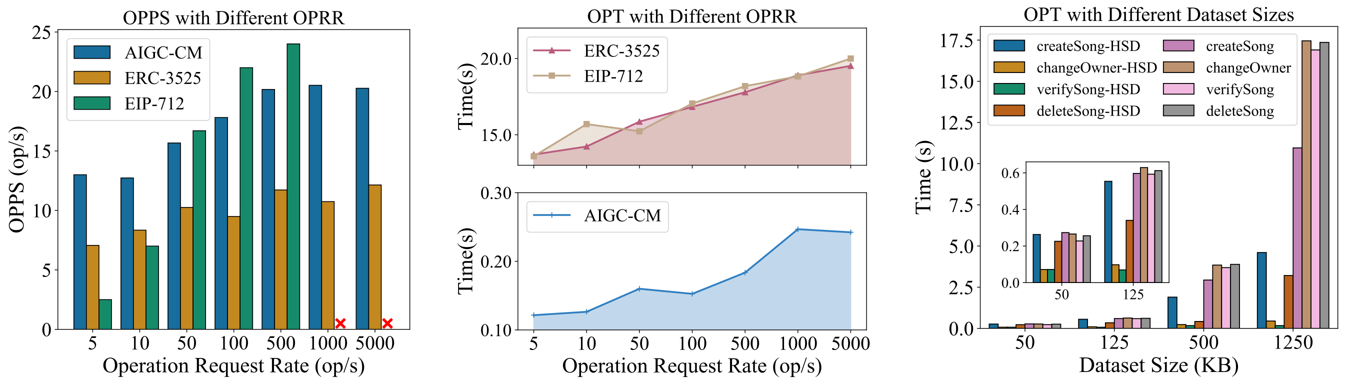

Fig. 6: (a) *OPPS* comparison of AIGC-CM, ERC-3525 and EIP-712 under different operation request rate; (b) *OPT* comparison ofAIGC-CM, ERC-3525 and EIP-712 under different operation request rate; (c) *OPT* of chained-structure (CopyrightLY [24] and PoAIGC [25]) and HSD under different dataset sizes.

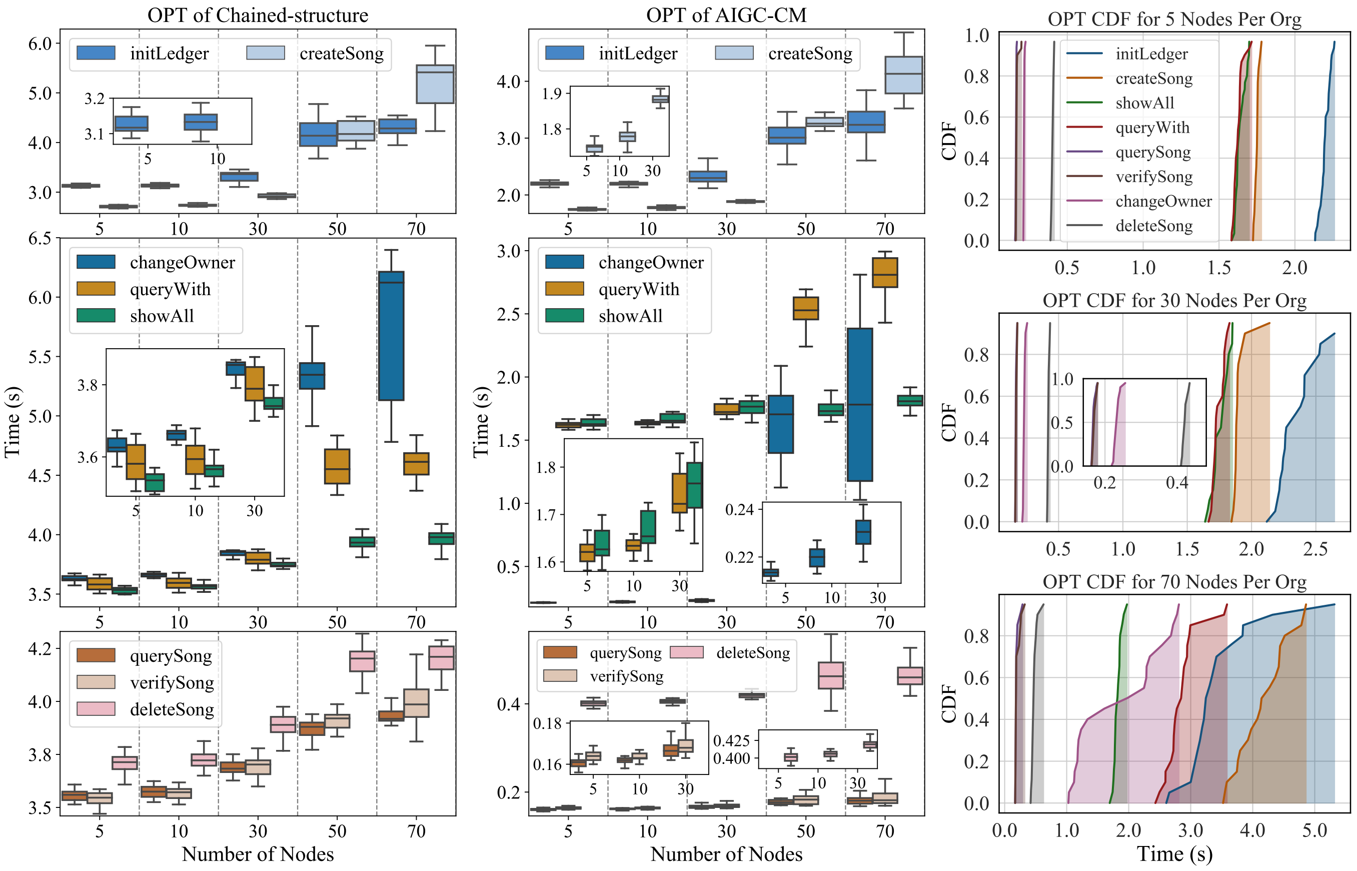

Fig. 7: (a) *OPT* of chained-structure (CopyrightLY [24] and PoAIGC [25]) under different operations and Org scales; (b) *OPT* of AIGC-CM under different operations and Org scales; (c) AIGC-CM *OPT* CDF under different operations and Org scales. We use enlarged images to display areas that are not clear.

operations compared to the chain-based structure. Notably, when dataset size is 1250KB, the createSong-HSD operation takes about 4.6s, representing a 57.98% reduction from the chain-based time cost of 10.9s. Similarly, the changeOwner-HSD operation requires 0.45 seconds, a 97.44% reduction from 17.46s, verifySong-HSD takes 0.17s, a 98.99% reduction from 16.9s, and deleteSong-HSD takes 3.2s, an 81.51% reduction from 17.36s. Furthermore, the results indicate that as the dataset size increases, the performance gap widens, underscoring the scalability and effectiveness of AIGC-CM in handling larger data volumes. The observed results can be attributed to the proposed HSD leveraging SimHash to quickly identify the corresponding sub-DAG from the index. This approach facilitates rapid localization of relevant elements within the HSD with a time complexity of $O(b+k)$, where b is the bucket size of index and k is the complexity within the DAG. HSD eliminates the need for traversal with a time complexity of $O(n)$, thereby reducing the overall time cost.

System Scalability. We investigate the variation trends of *OPT* and resource consumption for different operations in AIGC-CM as the number of nodes increases. As depicted in Fig. 7 (a) and Fig. 7 (b), for simple operations such querySong

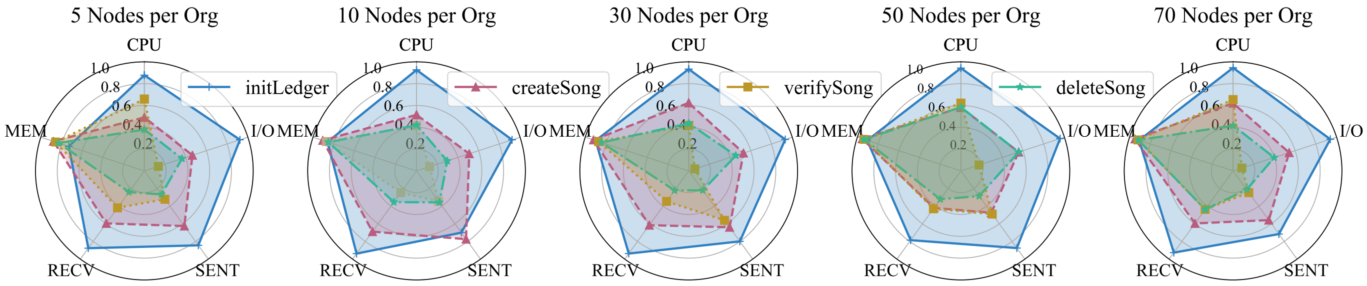

Fig. 8: AIGC-CM resource consumption ratio during initLedger, createSong, verifySong, and deleteSong with the increasing number of nodes.

and verifySong, the median execution times exhibit a slight increase with the growth in the number of nodes, below 10% of *OPT*. However, for complex operations like changeOwner, the *OPT* increases significantly by 10x at larger node scales. This is due to the large delayed signature computations and state I/O, which results in significant delays during multi-node coordination. Even so, when the node scale is increased by 14 times, the *OPT* for most operations (deleteSong, showAll, queryWith, and initLedger) increase only by less than 2 times. This highlights the scalability of AIGC-CM, where optimizing the length of hysteresis signatures is crucial.

We also illustrate *OPT* using CDF. As shown in Fig. 7 (c), we elucidate the performance dynamics of all related operations with different numbers of nodes in each organization. The performance of the verifySong, querySong and showAll operations is efficient, with most cost times clustered within a narrow interval across different nodes, underscoring their consistency and predictability. The CDF curve escalates swiftly, signifying that the bulk of the queries are resolved expeditiously, a testament to the operation's efficiency and the underlying HSD's efficacy. As the node scale varies, initLedger, createSong, changeOwner, deleteSong and queryWith (attributes) all exhibit a wider dispersion in timing. Regardless of the scale, the initLedger operation incurs the highest time cost. This is primarily due to two reasons. Firstly, the nature of initLedger involves inserting all initialization data into the CT, essentially performing multiple createSong operations, resulting in the highest overhead. Secondly, as the node scale increases, the communication overhead between nodes also increases, which is the main reason for the variation in the time range of operations such as createSong and changeOwner. The curve for queryWith (attributes) ascends more gradually with noticeable steps, indicating potential performance bottlenecks or complexities that extend query duration.

As shown in Fig. 8, we observe the resource utilization of different operations across varying node scales using resource *Monitor* of GFBE [33]. We primarily focus on four operations (initLedger, createSong, verifySong and deleteSong). The results reveal variations in the system resource utilization as the number of nodes in an organization increases from 5 to 70. We observe that the utilization rates of the CPU, memory, I/O, and network bandwidth remain similar. The initLedger consumes the most resources, followed by createSong and deleteSong, as these three operations are write-intensive and thus incur higher overhead. verifySong only involve read activities. However, they necessitate communication with nodes, leading to high bandwidth consumption. Overall, the operations performed by AIGC-CM (createSong, verifySong and deleteSong) do not impose a significant resource burden on the nodes, whit the memory utilization fluctuation ratio (the increased ratio of maximum compared to minimum) is less than 8%. The consistent resource consumption trends demonstrate the stability of AIGC-CM. Furthermore, these results also indicate that AIGC-CM possesses strong scalability.

VI. Conclusion

Currently, blockchain for AIGC copyright management has deficiencies in precise authentication and efficient traceability. We introduced AIGC-CM and proposed the blockchain-based MCU concept for the first time, integrating SimHash and structured metadata to achieve precise copyright authentication and on-chain self-description for AIGC. Additionally, we utilized a DAG-based data structure with hysteresis signatures to achieve efficient and secure copyright traceability for MCUs. We deployed AIGC-CM prototype on Fabric 2.4. For system performance, the operation times for create and verify operations on MCU are reduced by 98.86%. Additionally, the operation times for traceability operations like verifySong are recuded by 98.99%. Compared to the linear increase of chained-structure, AIGC-CM latency increases by less than 2x when the node scale is expanded 14x. The memory utilization fluctuation ratio (the increased ratio of maximum compared to minimum over different node numbers) is less than 8%.

VII. Acknowledgement

This work is supported by the National Key Research and Development Program of China under Grant No. 2022YFB2702900, and the National Natural Science Foundation of China under Grant Nos. 62232002, 62072221, and 62032017.

References

[1] W. Sun, X. Min, W. Lu, and G. Zhai, “A deep learning based no-reference quality assessment model for ugc videos,” in *Proc. of ACM MM*. ACM, 2022, p. 856–865.
[2] Y. Cao, S. Li, Y. Liu, Z. Yan, Y. Dai, P. S. Yu, and L. Sun, “A comprehensive survey of ai-generated content (aigc): A history of generative ai from gan to chatgpt,” *arXiv preprint arXiv:2303.04226*, 2023.
[3] J. Wu, W. Gan, Z. Chen, S. Wan, and H. Lin, “Ai-generated content (aigc): A survey,” *arXiv preprint arXiv:2304.06632*, 2023.

[4] M. Xu, H. Du, D. Niyato, J. Kang, Z. Xiong, S. Mao, Z. Han, A. Jamalipour, D. I. Kim, and X. e. Shen, "Unleashing the power of edge-cloud generative ai in mobile networks: A survey of aigc services," *IEEE Communications Surveys & Tutorials*, pp. 1–45, 2024.

[5] H. Du, Z. Li, D. Niyato, J. Kang, Z. Xiong, D. I. Kim *et al.*, "Enabling ai-generated content (aigc) services in wireless edge networks," *arXiv preprint arXiv:2301.03220*, 2023.

[6] H. Du, R. Zhang, D. Niyato, J. Kang, Z. Xiong, D. I. Kim, X. S. Shen, and H. V. Poor, "Exploring collaborative distributed diffusion-based ai-generated content (aigc) in wireless networks," *IEEE Network*, pp. 1–8, 2023.

[7] P. Research. (2023) Global artificial intelligence (ai) market size. Accessed on January 30, 2024. [Online]. Available: https://www.precedenceresearch.com/artificial-intelligence-market

[8] Y. Wang, Y. Pan, M. Yan, Z. Su, and T. H. Luan, "A survey on chatgpt: Ai-generated contents, challenges, and solutions," *arXiv preprint arXiv:2305.18339*, 2023.

[9] U. I. C. Office. (2024) Ico consultation series on generative ai and data protection. Accessed on January 30, 2024. [Online]. Available: https://ico.org.uk/about-the-ico/ico-and-stakeholder-consultations/ico-consultation-series-on-generative-ai-and-data-protection/

[10] W. Peng, J. Yi, F. Wu, S. Wu, B. Zhu, L. Lyu, B. Jiao, T. Xu, G. Sun, and X. Xie, "Are you copying my model? protecting the copyright of large language models for eaas via backdoor watermark," in *In Proc. of ACL*, 2023.

[11] X. He, Q. Xu, Y. Zeng, L. Lyu, F. Wu, J. Li, and R. Jia, "Cater: Intellectual property protection on text generation apis via conditional watermarks," in *In Proc. of NeurIPS*, vol. 35, 2022, pp. 5431–5445.

[12] X. He, Q. Xu, L. Lyu, F. Wu, and C. Wang, "Protecting intellectual property of language generation apis with lexical watermark," in *In Proc. of AAAI*, vol. 36, no. 10, 2022, pp. 10 758–10 766.

[13] J. Kirchenbauer, J. Geiping, Y. Wen, J. Katz, I. Miers, and T. Goldstein, "A watermark for large language models," in *In Proc. of ICML*, 2023.

[14] H. Xu, B. Xiao, X. Liu, L. Wang, S. Jiang, W. Xue, J. Wang, and K. Li, "Empowering authenticated and efficient queries for stk transaction-based blockchains," *IEEE Transactions on Computers*, vol. 72, no. 8, pp. 2209–2223, 2023.

[15] H. Xu, X. Liu, Z. Liang, H. Sun, W. Xue, J. Wang, and K. Li, "A transaction cardinality estimation approach for qos-adjustable intelligent blockchain systems," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 12, pp. 3672–3684, 2022.

[16] Z. Hong, S. Guo, E. Zhou, W. Chen, H. Huang, and A. Zomaya, "Gridb: Scaling blockchain database via sharding and off-chain cross-shard mechanism," *Proceedings of the VLDB Endowment*, vol. 16, no. 7, pp. 1685–1698, 2023.

[17] W. Ni, P. Chen, and L. Chen, "Psfq: A blockchain-based privacy-preserving and verifiable student feedback questionnaire platform," *Proceedings of the VLDB Endowment*, vol. 16, no. 12, pp. 3918–3921, 2023.

[18] C. Chen, Z. Wu, Y. Lai, W. Ou, T. Liao, and Z. Zheng, "Challenges and remedies to privacy and security in aigc: Exploring the potential of privacy computing, blockchain, and beyond," *arXiv prepring arXiv:2306.00419*, 2023.

[19] Q. Wang, R. Li, Q. Wang, and S. Chen, "Non-fungible token (nft): Overview, evaluation, opportunities and challenges," *arXiv preprint arXiv:2105.07447*, 2021.

[20] O. AI. (2022) Aigc chain. Accessed on January 30, 2024. [Online]. Available: https://www.aigcchain.io/about

[21] M. Papathanasaki, L. Maglaras, and N. Ayres, "Modern authentication methods: A comprehensive survey," *AI, Computer Science and Robotics Technology*, 2022.

[22] E. Dauterman, H. Corrigan-Gibbs, D. Mazières, D. Boneh, and D. Rizzo, "True2f: Backdoor-resistant authentication tokens," in *Proc. of IEEE S & P*, 2019, pp. 398–416.

[23] A. C. Weaver, "Biometric authentication," *IEEE Computer*, vol. 39, no. 2, pp. 96–97, 2006.

[24] R. García, A. Cediel, M. Teixidó, and R. Gil, "Semantics and non-fungible tokens for copyright management on the metaverse and beyond," *ACM Transactions on Multimedia Computing, Communications, and Applications*, 2023.

[25] Y. Liu, H. Du, D. Niyato, J. Kang, Z. Xiong, C. Miao, X. S. Shen, and A. Jamalipour, "Blockchain-empowered lifecycle management for ai-generated content products in edge networks," *IEEE Wireless Communications*, pp. 1–9, 2024.

[26] Y. Lin, H. Du, D. Niyato, J. Nie, J. Zhang, Y. Cheng, and Z. Yang, "Blockchain-aided secure semantic communication for ai-generated content in metaverse," *IEEE Open Journal of the Computer Society*, vol. 4, pp. 72–83, 2023.

[27] J. Benet, "Ipfs-content addressed, versioned, p2p file system," *arXiv preprint arXiv:1407.3561*, 2014.

[28] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *In Proc. of IEEE FOCS*, 2001, pp. 136–145.

[29] S. Dziembowski, L. Eckey, and S. Faust, "Fairswap: How to fairly exchange digital goods," in *In Proc. of ACM CCS*, 2018, pp. 967–984.

[30] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *In Proc. of IEEE S&P*, 2016, pp. 839–858.

[31] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, and et.al., "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proc. of EuroSys*, 2018, pp. 1–15.

[32] H. Foundation. (2022) Hyperledger fabric. [Online]. Available: https://github.com/hyperledger/fabric/tree/release-2.4

[33] L. Ma, X. Liu, Y. Li, C. Zhang, G. Shi, and K. Li, "Gfbe: A generalized and fine-grained blockchain evaluation framework," *IEEE Transactions on Computers*, vol. 73, no. 3, pp. 942–955, 2024.